



UNIVERSIDAD INTERNACIONAL DE LA RIOJA

PROGRAMA DE DOCTORADO EN CIENCIAS DE
LA COMPUTACIÓN

TESIS DOCTORAL

**Evaluación y Propuesta de una Arquitectura de
Correo Electrónico Seguro Basado en Blockchain**

Memoria presentada por:

Diego Piedrahita Castillo

para optar al grado de Doctor por la
Universidad Internacional de La Rioja

Dirigida por los Doctores:

Dr. Francisco Machío Regidor

Dr. Javier Bermejo Higuera

Madrid, 2022

Dedicatoria

A todos los usuarios del correo electrónico, a los profesionales de la ciberseguridad que se dedican cada día por velar por la seguridad de nuestros datos.

A usted, que cada día escribe un correo electrónico sin pensar que sus datos son vulnerables en el mundo de Internet.

A todos, que de una u otra manera hacen posible que cada día se recopile información y poder acceder a ella. A todas las personas que alimentan la red en estos tiempos de la información, sin ustedes sería imposible escribir esta tesis.

A usted, que en estos momentos se dispone a consultar esta memoria, con el ánimo de poder obtener información sobre los avances del correo electrónico.

Licencia

Esta tesis doctoral, titulada “Evaluación y Propuesta de un Protocolo de Seguridad de Correo Electrónico Avanzado Basado en Blockchain”, realizada por Don Diego Piedrahita Castillo, está sujeta a la Licencia Internacional Pública de Atribución/Reconocimiento-NoComercial-SinDerivadas 4.0 de Creative Commons.



Por tanto, esta obra se puede compartir y comunicar públicamente, siempre que se reconozca su autoría y no se persigan fines comerciales. Así mismo, no se permite la distribución de obras derivadas.

Resumen

El principal objetivo de esta investigación es el desarrollo de un protocolo de correo electrónico seguro aprovechando el potencial que nos brinda *blockchain* en aspectos de seguridad. En este trabajo se analiza el estado del arte de los problemas y las soluciones disponibles para proteger las comunicaciones de correo electrónico. Como resultado de ese estudio, se obtienen los requisitos ideales que una solución de correo electrónico debe brindar para proporcionar seguridad a sus usuarios. Se identifica que una solución de correo electrónico segura, además de garantizar la confidencialidad, integridad y disponibilidad, debe proporcionar otros servicios o características como son el no repudio, autenticación, control, auditoría, usabilidad e integración.

Se comparan las propuestas analizadas para ver si garantizaban los aspectos de seguridad identificados. Se encuentra que la mayoría de las soluciones se enfocan en solventar algunos aspectos específicos, dejando fuera de su alcance los demás aspectos. Muchas de las soluciones no tienen en cuenta las necesidades de uso por parte de los usuarios, lo que dificulta su difusión y su adopción.

Se realiza un estudio de la tecnología *blockchain* con el objetivo de analizar sus características de seguridad y ver como pueden ser aprovechadas en la presente propuesta. Se analizan las propuestas basadas en *blockchain* que van en dirección de proporcionarle seguridad al correo electrónico. Se contrastan estas propuestas con los aspectos de seguridad identificados y se encuentra que dichas soluciones no cumplen con todos los aspectos de seguridad requeridos. Se identifica cómo se puede utilizar la tecnología *blockchain* adecuadamente para proporcionar los servicios de seguridad considerados necesarios para alcanzar los objetivos del presente trabajo.

Fue conveniente definir una serie de requisitos de usuario que sirvieron de base para el diseño de una arquitectura sobre la que se desarrolló la propuesta. Esta propuesta la componen una serie de procesos y protocolos basados en una arquitectura que hace uso de la tecnología *blockchain*. Se plantea una solución distribuida y descentralizada donde los protocolos operan sobre *blockchain* proporcionando seguridad al correo electrónico, que es almacenado de forma no inmutable. A diferencia de las criptomonedas, donde toda la información tiene que estar disponible para poder verificar las transacciones, con el correo electrónico no es necesario almacenar toda la información, ya que existe poca relación entre los correos, y no es necesaria la verificación de un estado anterior para generar un nuevo correo. Debido a esa baja cohesión entre los correos, en esta tesis se propone una *blockchain* privada en la que los participantes interesados almacenen los correos y se encarguen de mantenerla sin perder confidencialidad, integridad y disponibilidad.

Se concluye que esta propuesta ofrece una solución integral al problema de seguridad del correo electrónico, proporcionando solución a todos los requisitos descritos en este trabajo. Finalmente, se desarrolla una prueba de concepto basada en la arquitectura propuesta para validar su factibilidad.

Palabras clave: *blockchain*, correo seguro, ciberseguridad, correo no deseado, protocolos distribuidos.

Abstract

The main objective of this research is the development of a secure email protocol taking advantage of the potential that the blockchain offers us in security aspects. In this work, we have analyzed the state-of-the-art of problems and solutions available to protect email communications. As a result of this study, we obtained the ideal requirements for an email solution to provide security to its users. We found that a secure email solution, in addition to guaranteeing confidentiality, integrity and availability, must provide other services or features such as non-repudiation, authentication, control, auditing, usability, and integration.

We compared the analyzed proposals to see if they guaranteed the security aspects previously identified. We found that most solutions focused on solving some specific aspects, leaving the other aspects out. Many solutions do not consider the needs of use by users, which makes the dissemination of such tools and their adoption difficult.

We performed a study of blockchain technology to analyze its security features and check if we could use them in our proposal. We analyzed the blockchain-based proposals that go in the direction of providing email security. We contrasted these proposals with the security aspects previously identified, and we found that these solutions do not comply with all the required security aspects. We identified how to use blockchain technology appropriately to provide the security services that we believe are necessary to achieve the objectives of this work.

It was necessary to define a series of user requirements that served as the basis for the design of the architecture on which we developed the proposal. Our proposal consists of a series of processes and protocols based on an architecture that uses blockchain technology. We propose a distributed and decentralized solution, where the protocols operate on the blockchain providing security to email, which is stored in a non-immutable way. Unlike cryptocurrencies, where all the information must be available to be able to verify the transactions, in the email, it is not necessary to store all the information, since there is little relationship between the emails, and verification of a previous state is not necessary to generate a new mail. Due to this low cohesion between emails, we proposed a private blockchain in which the interested participants store the emails and are responsible for maintaining it without losing confidentiality, integrity, and availability.

We concluded that our proposal offers a comprehensive solution to the email security problem, providing a solution to all the requirements described in this work. Finally, we developed a proof of concept based on the proposed architecture to validate its feasibility.

Keywords: *blockchain*, secure email, cybersecurity, spam, distributed protocols.

ÍNDICE

CAPÍTULO 1 – INTRODUCCIÓN

I.	MOTIVACIÓN Y DEFINICIÓN DEL PROBLEMA	3
II.	HIPÓTESIS DE LA INVESTIGACIÓN	9
III.	OBJETIVO DE LA INVESTIGACIÓN	10
IV.	METODOLOGÍA	11
V.	ESTRUCTURA DEL TRABAJO	14

CAPÍTULO 2 – MARCO TEÓRICO

I.	EVOLUCIÓN DE LOS PROTOCOLOS DE CORREO ELECTRÓNICO DESDE 1971 HASTA 2020	17
1.	PROPUESTA INICIAL	17
2.	EL CORREO FTP	18
3.	MAIL PROTOCOL (MP)	19
4.	PROTOCOLO DE TRANSFERENCIA DE CORREO (MTP)	20
5.	PROTOCOLO DE TRANSFERENCIA DE CORREO SIMPLE (SMTP)	21
6.	PROTOCOLO DE OFICINA POSTAL (POP) Y PROTOCOLO DE ACCESO A MENSAJES DE INTERNET (IMAP)	23
II.	ARQUITECTURA DE LOS PROTOCOLOS DE CORREO ELECTRÓNICO EN LA ACTUALIDAD	24
1.	ACTORES PRINCIPALES	24
1.1.	Actores usuarios	26
1.2.	Actores del MHS	29
1.3.	Actores administrativos	32
2.	IDENTIDADES	34
2.1.	Buzón	34
2.2.	Nombre de dominio	35
2.3.	Identificador del mensaje	35
3.	SERVICIOS Y ESTÁNDARES	36
3.1.	Mensaje	38
3.2.	Servicios a nivel de usuario	41
3.3.	Servicios a nivel de MHS	43
4.	CONSIDERACIONES ADICIONALES DE LA ARQUITECTURA	45
4.1.	Modos de transición	45
4.2.	Implementación y operación	46
4.3.	Mediadores	46
III.	ESTADO DEL ARTE EN EL CONTEXTO DE LA SEGURIDAD DEL CORREO ELECTRÓNICO	49
1.	CONSIDERACIONES DE SEGURIDAD DE LA ARQUITECTURA DE CORREO ELECTRÓNICO	49
2.	ESTUDIO DE LA SEGURIDAD EN EL CONTEXTO DEL CORREO ELECTRÓNICO	50
2.1.	Usabilidad	51
2.2.	Phishing	52
2.3.	Correo no deseado	56
2.4.	Soluciones de seguridad	60

IV. TECNOLOGÍA BLOCKCHAIN	85
1. INTRODUCCIÓN	85
2. ESTRUCTURA Y ARQUITECTURA DE <i>BLOCKCHAIN</i>	87
2.1. Tipología	87
2.2. Ciclo de vida de una transacción en <i>blockchain</i>	92
2.3. Conceptos criptográficos utilizados en <i>blockchain</i>	94
2.4. Algoritmos de consenso	102
2.5. Evolución de <i>blockchain</i>	120
2.6. Arquitectura de <i>blockchain</i>	127
3. PROPIEDADES DE SEGURIDAD Y PRIVACIDAD DE <i>BLOCKCHAIN</i>	132
4. ESTADO DEL ARTE DE SOLUCIONES DE CORREO ELECTRÓNICO SOBRE <i>BLOCKCHAIN</i>	135
V. CONCLUSIONES SOBRE LAS SOLUCIONES DE CORREO ELECTRÓNICO Y JUSTIFICACIÓN DEL ESTUDIO	140

CAPÍTULO 3 – DESARROLLO DE LA CONTRIBUCIÓN Y DISCUSIONES

I. INTRODUCCIÓN	145
II. CONCEPTOS Y DEFINICIONES PRELIMINARES	146
III. LISTA DE REQUISITOS PARA LA GESTIÓN SEGURA DE CORREO ELECTRÓNICO	154
IV. ARQUITECTURA DE BBM (BLOCKCHAIN BASED MAIL)	187
1. INTRODUCCIÓN	187
2. COMPONENTES DE LA ARQUITECTURA BBM	188
2.1. Red de Proveedores de Almacenamiento (RPA)	188
2.2. Servidor	188
2.3. Blockchain de BBM (BBMChain)	189
2.4. Generador de claves (KG)	190
2.5. Cliente BBM	190
2.6. Almacenamiento BBM (BBMStorage)	190
2.7. Integrador BBM (BBMI)	191
3. ESPECIFICACIÓN Y DESCRIPCIÓN DE LA ARQUITECTURA BBM	191
3.1. Diagrama de contexto del sistema	191
3.2. Diagrama de contenedores	192
3.3. Diagramas de componentes	195
4. ASPECTOS DINÁMICOS DE LA ARQUITECTURA BBM	202
4.1. Protocolo de Generación de Claves (PGC)	203
4.2. Protocolo de Registro de Servidor (PRS)	207
4.3. Protocolo de Consulta de datos de registro de Servidor (PCS)	214
4.4. Protocolo de Modificación de datos de Servidor (PMS)	219
4.5. Protocolo de Registro de Usuario (PRU)	223
4.6. Protocolo de Modificación de datos de Usuario (PMU)	234
4.7. Protocolo de Creación de Cuentas (PCC)	239
4.8. Protocolo de Modificación de Cuentas (PMC)	241
4.9. Protocolo de Creación de Buzones (PCB)	244

4.10. Protocolo de Modificación de Buzón (PMB)	247
4.11. Protocolo de Registro de Cuenta en el Servidor (PRCS)	249
4.12. Protocolo de Solicitud y Autorización de Contacto (PSAC)	252
4.13. Protocolo de Intercambio de Localización entre Servidores (PILS)	263
4.14. Protocolo de Envío de Correo (PEC)	266
4.15. Protocolo de Recepción de Correo en Servidor (PRECS)	279
4.16. Protocolo de Autenticación de Servidor (PAS)	283
4.17. Protocolo de Recepción de Correo (PRC)	285
5. OTROS PROTOCOLOS	288
5.1. Protocolo de Certificación de Identidad (PCI)	288
5.2. Protocolo de Recuperación y Bloqueo de Cuenta (PRBC)	289
5.3. Protocolo de Acceso a Terceros (PAT)	289
5.4. Protocolo de Autorización Grupal (PAG)	290
5.5. Protocolo de Supervisión y Autorización de Cuentas (PSATC)	290
5.6. Protocolo de Borrado de Correo (PBC)	291
5.7. Protocolo de Reenvío de Correo (PREC)	292
5.8. Protocolo de Correo Certificado (PCCR)	293
5.9. Protocolo de Envío de Correo SMTP (PECSMTP)	294
V. ASPECTOS CLAVE DE LOS PROTOCOLOS DE BBM	295
VI. BBMAIL: PRUEBA DE CONCEPTO BASADA EN BBM	300
VII. DISCUSIONES ADICIONALES SOBRE LOS RESULTADOS	322
VIII. ASPECTOS DESTACABLES DE LA INVESTIGACIÓN	324
IX. TRABAJOS FUTUROS	326
CAPÍTULO 4 - CONCLUSIONES	329
CAPÍTULO 5 - REFERENCIAS BIBLIOGRÁFICAS	335
CAPÍTULO 6 - ANEXO	
ANEXO. CONTRIBUCIONES CIENTÍFICAS	357

Lista de abreviaciones

- ACD----- Algoritmo de cifrado de datos.
- ACDC----- Algoritmo de cifrado de datos compartidos.
- ACK----- Acknowledge – Confirmación.
- ADDC----- Algoritmo de descifrado para datos compartidos.
- ADMD----- Administrative Management Domain – Dominio de gestión administrativa.
- Admin----- Usuario administrador.
- AES----- Advanced Encryption Standard – Estándar de cifrado avanzado.
- AGCG----- Algoritmo de generación de claves de gestión.
- aMS----- Author Message Store – Almacén de mensajes del autor.
- aMSA----- Author Message Submission Agent – Funciones de MSA centradas en el autor.
- aMUA----- Message User Agent – MUA del autor.
- API----- Application Programming Interfaces – Interface de programación de aplicaciones.
- ARPAnet----- Advanced Research Projects Agency Network – Red de la Agencia de Proyectos de Investigación Avanzada.
- BBM----- Blockchain Based Mail – Correo electrónico basado en cadena de bloques.
- BBMChain----- BBM Blockchain – Cadena de bloques de BBM.
- BBMI----- Integrador BBM.
- BBMStorage----- Almacenamiento BBM.
- BEC----- Business Email Compromise – Compromiso de correo electrónico comercial.
- BIP----- Bitcoin Improvement Proposal – Propuesta de mejora de Bitcoin.
- BOINC ----- Berkeley Open Infrastructure for Network Computing – Infraestructura Abierta de Berkeley para la Computación en Red.
- CA----- Certification Authority – Autoridad de certificación.
- CKD----- Child Key Derivation – Derivación de claves secundarias o hijas.
- CVN----- Cooperatively Validated Nodes – Nodos validados cooperativamente.
- DANE----- DNS-based Authentication of Named Entities – Autenticación basada en DNS de entidades nombradas.

- dApp-----Decentralized application – Aplicación descentralizada.
- dBFT-----Delegated Byzantine Fault – Falla bizantina delegada.
- DDoS-----Distributed Denial of Service – Denegación de servicio distribuida.
- DEA----- Disposable Email address – Dirección de correo electrónico desechable.
- DHT-----Distributed Hash Table – Tabla de *hash* distribuida.
- DIME -----Dark Internet Mail Environment – Entorno de correo de internet oscuro.
- DKIM----- DomainKeys Identified Mail – Correo electrónico identificado con clave de dominio.
- DMAP-----Dark Mail Access Protocol – Protocolo de acceso a correo oscuro.
- DMARC-----Domain-based Message Authentication, Reporting and Conformance – Autenticación, Informes y Conformidad de Mensajes basada en Dominios.
- DMTP-----Dark Mail Transfer Protocol – Protocolo de transferencia de correo oscuro.
- DNS----- Domain Name System – Sistema de nombres de dominio.
- DNSSEC----- Domain Name System Security Extensions – Extensiones de seguridad del sistema de nombres de dominio.
- DOS----- Denial of Service – Denegación de servicio.
- DPoS-----Delegated Proof of Stake – Prueba de participación delegada.
- DSN----- Delivery Status Notification – Notificación de estado de entrega.
- EAC----- Email Account Compromise – Compromiso de cuenta de correo electrónico.
- EBIA----- Email-based identification and authentication – Autenticación e identificación basada en correo electrónico.
- ECC----- Elliptic Curve Cryptography – Criptografía de curva elíptica.
- ECDSA-----Elliptic Curve Digital Signature – Firma digital de curva elíptica.
- ECIES----- Elliptic Curve Integrated Encryption Scheme – Esquema de cifrado integrado de curva elíptica.
- EdDSA-----Edward curve digital signature algorithm – Algoritmo de firma digital de la curva de Edwards.
- ENVID-----Envelope Identifier – Identificador del sobre.
- FTP-----File Transfer Protocol – Protocolo de transferencia de fichero.
- HMAC-----Hash-based Message Authentication Code – Código de autenticación de mensajes basado en resumen.

- hMDA-----MHS MDA – Funciones MDA centradas en MHS.
- hMSA----- MHS MSA – Funciones de MSA centradas en MHS.
- HTTP----- Hypertext Transfer Protocol – Protocolo de transferencia de hipertexto.
- IBE----- Identity-Based Encryption – Cifrado basado en identidad.
- IC3----- Internet Crime Complaint Center – Centro de Quejas de Delitos en Internet
- IMAP-----Internet Message Access Protocol – Protocolo de acceso a mensajes de internet.
- IMF-----Internet Message Format – Formato de mensajes de Internet.
- IPFS----- Interplanetary File System – Sistema de archivos interplanetario.
- ISP----- Internet Service Provider – Proveedor de servicios de Internet.
- JSON----- JavaScript Object Notation – Notación de objetos Javascript.
- KDF----- Key derivation Function – Función de derivación de claves.
- KG----- KeysGenerator – Generador de claves.
- LMTP----- Local Mail Transfer Protocol – Protocolo de transporte local de correo.
- LPoS----- Leased Proof of Stake – Prueba de participación arrendada.
- MAC-----Message Authentication Code – Código de autenticación de mensaje.
- MDA----- Message Delivery Agent – Agente de entrega de mensajes.
- MDN----- Message Disposition Notification – Notificación de disposición de mensaje.
- MFA-----Multi-factor Authentication – Autenticación multifactor.
- MHS-----Mail Handling Service – Servicio de manejo de correo.
- MIME----- Multipurpose Internet Mail Extensions – Extensiones multipropósito de correo de Internet.
- MITM----- Man-in-the-Middle – Ataque de hombre en medio.
- MP-----Mail Protocol – Protocolo de correo.
- MS-----Message Store – Almacén de mensaje.
- MSA----- Message Submission Agent – Agente de envío de mensajes.
- MTA-----Message Transfer Agents – Agente de transferencia de mensajes.
- MTA STS----- MTA Strict Transport Security – Seguridad de transporte estricta del agente de transferencia de mensajes.
- MTP-----Mail Transfer Protocol – Protocolo de transferencia de correo.

- MUA----- Message User Agents – Agentes de usuario de mensajes.
- MX----- Mail Exchange – Intercambio de correo.
- NCP----- Network Control Protocol – Protocolo de control de red.
- NEM----- New Economy Movement – Movimiento de nueva economía.
- ODMR----- On-Demand Mail Relay – Retransmisión de correo bajo demanda.
- OpenPGP-----Open Pretty Good Privacy – Privacidad bastante buena abierta.
- P2P-----Peer To Peer – Punto a punto.
- PAG----- Protocolo de Autorización Grupal.
- PAS-----Protocolo de Autenticación de Servidor.
- PAT----- Protocolo de Acceso a Terceros.
- PBC----- Protocolo de Borrado de Correo.
- PBFT----- Practical Byzantine Fault Tolerance – Tolerancia práctica a falla bizantina.
- PCB----- Protocolo de Creación de Buzones.
- PCC-----Protocolo de Creación de Cuentas.
- PCCR-----Protocolo de Correo Certificado.
- PCEC-----Protocolo de Cambio de Estado de Contacto.
- PCI----- Protocolo de Certificación de Identidad.
- PCS-----Protocolo de Consulta de datos de registro de Servidor.
- PEC----- Protocolo de Envío de Correo.
- PECSMTP----- Protocolo de envío de correo SMTP.
- PEM-----Privacy Enhanced Mail – Correo con privacidad mejorada.
- PGC----- Protocolo de Generación de Claves.
- PGP----- Pretty Good Privacy – Privacidad Bastante Buena.
- PILS----- Protocolo de Intercambio de Localización entre Servidores.
- pk----- Clave pública.
- PKI----- Public Key Infrastructure – Infraestructura de clave pública.
- PMB-----Protocolo de Modificación de Buzón.
- PMC-----Protocolo de Modificación de Cuentas.
- PMS----- Protocolo de Modificación de datos de Servidor.

- PMU-----Protocolo de Modificación de datos de Usuario.
- PNG----- Pseudorandom Number Generator – Generador de números pseudoaleatorios.
- PoA----- Proof-Of-Activity – Prueba de actividad.
- PoAT----- Proof-Of-Authority – Prueba de autoridad.
- PoB-----Proof-of-Burn – Prueba de quemado o grabación.
- PoC-----Proof of Cooperation – Prueba de cooperación.
- PoET----- Proof of elapsed time – Prueba de tiempo transcurrido.
- PoI----- Proof-of-Importance – Prueba de importancia.
- POP----- Post Office Protocol – Protocolo de Oficina Postal.
- PoP-----Proof-Of-Publication – Prueba de publicación.
- PoR----- Proof of Research – Prueba de investigación.
- PoS----- Proof-of-Stake – Prueba de participación.
- PoSV----- Proof Of Stake-Velocity – Prueba de velocidad de participación.
- PoW-----Proof-of-Work – Prueba de trabajo.
- PRBC-----Protocolo de Recuperación y Bloqueo de Cuenta.
- PRC----- Protocolo de Recepción de Correo.
- PRCS----- Protocolo de Registro de Cuenta en el Servidor.
- PRCZ-----Protocolo de Rechazo de Contacto.
- PREC-----Protocolo de Reenvió de Correo.
- PRECS----- Protocolo de Recepción de Correo en Servidor.
- PRS-----Protocolo de Registro de Servidor.
- PRU----- Protocolo de Registro de Usuario.
- PSAC----- Protocolo de Solicitud y Autorización de Contacto.
- PSATC----- Protocolo de Supervisión y Autorización de Cuentas.
- PSE-----Personal Security Environment – Entorno de seguridad personal.
- RC----- Relación de comunicación.
- REAP----- Rolling Email Addresses Protocols – Protocolos de direcciones de correo electrónico móviles.

- REST----- Representational State Transfer – Transferencia de estado representacional.
- rMDA----- Receptor MDA – Funciones MDA centradas en el receptor.
- rMS----- Receptor MS – MS del receptor.
- rMUA----- Receptor MUA – MUA del receptor.
- RPA----- Red de Proveedores de Almacenamiento.
- RPC----- Remote Procedure Call – Llamada de procedimiento remoto.
- S/MIME----- Secure/Multipurpose Internet Mail Extensions – Extensiones de correo de Internet de propósitos múltiples / seguro.
- SBFT ----- Simplified Byzantine Fault – Falla bizantina simplificada.
- SEFAP----- Signed Email for Anti-Phishing – Correo electrónico firmado para *antiphishing*.
- SGX----- Software Guard Extensions.
- sk----- Clave secreta o privada.
- SMTP----- Simple Mail Transfer Protocol – Protocolo de transferencia de correo simple.
- SPF----- Sender Policy Framework – Marco de políticas del remitente.
- TCP/IP----- Transmission Control Protocol/Internet Protocol – Protocolo de control de transmisión/Protocolo de Internet.
- Telnet----- Teletype Network – Red de Teletipo.
- TLS----- Transport Layer Security – Seguridad de la capa de transporte.
- TLS-RPT----- SMTP TLS Reporting – Informes de problemas de conectividad con TLS.
- TME----- Targeted Malicious Email – Correos maliciosos dirigidos.
- TTP----- Trusted Third Party – Tercero de confianza.
- UE----- Usuario corporativo.
- UP----- Usuario personal.
- usuinf----- Usuario de nivel inferior.
- VPIM----- Voice Profile for Internet Mail – Perfiles de voz para correo de Internet.
- ZKP----- Zero Knowledge Proof – Pruebas de conocimiento cero.

Lista de figuras

Figura 1. Volumen de correo no deseado con tema “coronavirus” en marzo de 2020.....	6
Figura 2. Países más atacados por correos no deseados maliciosos con el asunto “coronavirus”.	7
Figura 3. Protocolo de correo electrónico de 4 pasos.	17
Figura 4. Modelo para el uso de MTP.	21
Figura 5. Formato del mensaje.	22
Figura 6. Arquitectura básica del servicio de correo en Internet.	25
Figura 7. Relaciones entre los actores usuarios.....	28
Figura 8. Relaciones entre los actores de MHS.....	29
Figura 9. Ejemplo de ADMD.	32
Figura 10. Protocolos y servicios dentro de la arquitectura de correo de Internet.	37
Figura 11. Técnicas utilizadas para llevar a cabo un ataque de <i>phishing</i>	53
Figura 12. Evolución de los métodos <i>anti-phishing</i> desde el año 2000.	55
Figura 13. Interconexión entre las partes de un correo electrónico y las diferentes técnicas de detección de correo no deseado.	58
Figura 14. Modelo conceptual de la arquitectura de correo electrónico.	60
Figura 15. Aspectos de seguridad de la arquitectura de correo electrónico y puntos clave de interés de las propuestas actuales de seguridad.	61
Figura 16. Sesión STARTTLS de SMTP entre cliente y servidor.	69
Figura 17. Evaluación de seguridad de STARTTLS.	69
Figura 18. Evaluación de seguridad de MTA STS.....	71
Figura 19. Evaluación de seguridad de DANE.	73
Figura 20. Evaluación de seguridad de DARKMAIL.....	74
Figura 21. Evaluación de seguridad de Protonmail.	75
Figura 22. Evaluación de seguridad de Bitmessage.....	77
Figura 23. Evaluación de seguridad de S/MIME.	78
Figura 24. Evaluación de seguridad de OpenPGP.....	79
Figura 25. Evaluación de seguridad de SPF.....	80
Figura 26. Evaluación de seguridad de DKIM.....	81
Figura 27. Evaluación de seguridad de DMARC.....	83
Figura 28. Esquema simple de una cadena de bloques.	85
Figura 29. Arquitectura cliente-servidor y P2P.....	87
Figura 30. Blockchain pública.....	89
Figura 31. Blockchain privada.	90
Figura 32. Blockchain de consorcio.....	91
Figura 33. Ciclo de vida de una transacción en blockchain.	92

Figura 34. Funciones de cifrado/descifrado y funciones hash.....	94
Figura 35. Estructura de bloque simplificada.....	97
Figura 36. Árbol de Merkle.....	98
Figura 37. Esquema simplificado de firma electrónica.....	101
Figura 38. Propiedades estructurales de los algoritmos de consenso.	104
Figura 39. Propiedades de recompensa y bloque de los algoritmos de consenso.....	105
Figura 40. Arquitectura de cadena de bloques pública, privada e híbrida basada en un libro mayor único.....	129
Figura 41. Arquitectura de <i>blockchain</i> basada en múltiples libros mayores.....	130
Figura 42. Arquitectura de <i>blockchain</i> basada en interoperabilidad.....	132
Figura 43. Arquitectura de BBM.....	187
Figura 44. Diagrama de contexto BBM.	192
Figura 45. Diagrama de contenedores de BBM.....	193
Figura 46. Diagrama de componentes de cliente BBM.	196
Figura 47. Diagrama de componentes del integrador.....	197
Figura 48. Flujo de correo desde SMTP y BBM.	198
Figura 49. Utilización de BBM como un servicio adicional al correo electrónico SMTP.....	199
Figura 50. Diagrama de componentes de BBMChain.	200
Figura 51. Diagrama de componentes del servidor.	201
Figura 52. Proceso de generación de dirección Ethereum.....	203
Figura 53. Diagrama del protocolo PGC.....	205
Figura 54. Diagrama de clases del servidor.....	207
Figura 55. Algoritmo de cifrado de datos (AES + Scrypt + HMAC).	210
Figura 56. Diagrama del protocolo PRS.	212
Figura 57. Diagrama del protocolo PCS.....	215
Figura 58. Cifrado en BBM utilizando la clave pública del destinatario basado en cifrado híbrido ECIES.....	218
Figura 59. Descifrado en BBM utilizando la clave privada del destinatario basado en cifrado híbrido ECIES.....	219
Figura 60. Máquina de estados del servidor.	220
Figura 61. Diagrama del protocolo PMS.....	222
Figura 62. Generación de las claves.....	224
Figura 63. Estructura de claves en BBM.	225
Figura 64. Diagrama de clases del usuario.....	227
Figura 65. Estructura jerárquica de los usuarios corporativos en BBM.....	230
Figura 66. Diagrama del protocolo PRU.....	233
Figura 67. Diagrama de estados del usuario.	234
Figura 68. Diagrama del protocolo PMU.....	236

Figura 69. Diagrama del protocolo PCC.....	241
Figura 70. Diagrama de máquina de estados de la cuenta.....	242
Figura 71. Diagrama del protocolo PMC.....	243
Figura 72. Diagrama del protocolo PCB.....	246
Figura 73. Diagrama de estados del buzón.....	248
Figura 74. Diagrama del protocolo PMB.....	249
Figura 75. Diagrama del protocolo PRCS.....	251
Figura 76. Esquema de solicitud de contacto.....	256
Figura 77. Diagrama de máquina de estados del contacto.....	257
Figura 78. Diagrama de protocolo PSAC (rol originador).....	259
Figura 79. Diagrama de protocolo PSAC (rol destinatario).....	260
Figura 80. Diagrama de protocolo PILS.....	264
Figura 81. Diagrama de máquina de estados del correo.....	271
Figura 82. Diagrama del protocolo PEC.....	276
Figura 83. Diagrama del protocolo PRECS.....	281
Figura 84. Diagrama del protocolo PAS.....	284
Figura 85. Diagrama del protocolo PRC.....	286
Figura 86. Arquitectura de BBMail.....	300

Lista de tablas

Tabla 1. Tráfico diario de emails diarios en el mundo entre 2020 y 2024.....	3
Tabla 2. Número de usuarios con cuentas de correo electrónico entre 2020 y 2024.	4
Tabla 3. Fuentes de infección por virus informáticos, desde 1996 a 2003.....	4
Tabla 4. Número de denuncias por ciber ataques y pérdidas económicas agregadas del 2017 al 2021.....	5
Tabla 5. Actores usuarios y sus responsabilidades.....	27
Tabla 6. Actores MHS y sus responsabilidades.....	31
Tabla 7. Tipos básicos de ADMD.....	33
Tabla 8. Identificadores en las diferentes partes del mensaje.....	40
Tabla 9. Explicación de las partes de un correo electrónico.....	57
Tabla 10. Nomenclatura de colores utilizada para la evaluación de los aspectos de seguridad.	66
Tabla 11. Aspectos de seguridad de las soluciones de seguridad más extendidas.....	84
Tabla 12. Comparativa de propuestas de correo electrónico basado en blockchain.....	139
Tabla 13. Lista de requisitos identificados con los aspectos de seguridad considerados en cada uno de ellos.....	155
Tabla 14. Registro de seguridad de usuario.....	238
Tabla 15. Aspectos clave de los Protocolos de BBM.....	295
Tabla 16: Funcionalidad BBMail.....	302
Tabla 17. Aspectos de seguridad de los datos sensibles manejados en BBMail.....	304
Tabla 18. Pruebas requisitos buzones.....	307
Tabla 19. Pruebas requisitos usuarios.....	311
Tabla 20. Pruebas requisitos cuentas.....	314
Tabla 21. Pruebas requisitos correos.....	321
Tabla 22. Resumen soluciones propuestas a los aspectos de seguridad.....	330

Capítulo 1

Introducción

I. MOTIVACIÓN Y DEFINICIÓN DEL PROBLEMA

La integridad, la confidencialidad y la disponibilidad de los recursos de información son el principal objetivo de la seguridad de la información. El correo electrónico (*email*), definido como un mensaje digital que es transmitido entre un remitente y un receptor, contiene información y, por lo tanto, es necesario garantizar su seguridad. Dentro del contexto de la seguridad de los sistemas de correo electrónico, cualquier debilidad en esas tres áreas claves dejará expuesto el sistema a cualquier explotación (Cocca, 2004).

A largo del manuscrito, se utilizarán los términos correo y correo electrónico indistintamente para referirnos a lo mismo, a menos que se necesite diferenciar por alguna razón del correo postal.

El correo electrónico sigue siendo el método de comunicación más utilizado en Internet por muchos usuarios, ya que se requiere una cuenta de correo electrónico (una dirección de correo electrónico) para registrarse en cualquier tipo de actividad en línea, incluidas redes sociales, mensajería instantánea y cualquier otra forma de presencia en Internet. Las compras en línea también implican una dirección de correo electrónico para confirmaciones y notificaciones, entre otros (The Radicati Group, 2020).

Según el grupo Radicati (2020), el número total de correos electrónicos de empresas y consumidores enviados y recibidos por día superará los 347,3 mil millones en 2023, y se prevé que aumente a más de 361 mil millones para fines de 2024 (Tabla 1).

Tabla 1. *Tráfico diario de emails diarios en el mundo entre 2020 y 2024.*

Tráfico de email diario	2020	2021	2022	2023	2024
Total de emails enviados y recibidos por día en el mundo (B)	306,4	319,6	333,2	347,3	361,6
% Crecimiento	4,40%	4,30%	4,30%	4,20%	4,10%

Fuente: Elaboración propia a partir de datos de Radicati (2020).

Más de la mitad de la población mundial ya usaba el correo electrónico en 2020. La cantidad de usuarios de correo electrónico en todo el mundo superará los 4,3 billones en 2022 y se espera que aumente a más de 4,400 millones para finales de 2024 (Tabla 2).

Tabla 2. Número de usuarios con cuentas de correo electrónico entre 2020 y 2024.

	2020	2021	2022	2023	2024
Usuarios de Email en el mundo (M)	4.037	4.147	4.258	4.371	4.481
% Crecimiento	3%	3%	3%	3%	3%

Fuente: Elaboración propia a partir de datos de Radicati (2020).

En el 2004, Pam Cooca de SANS Institute (Cocca, 2004) realizó un estudio donde reflejaba cuáles eran en ese momento las amenazas de seguridad del correo electrónico. Los virus, el correo no deseado (*spam*) y el *phishing* eran los principales problemas y, como muestra la Tabla 3, la mayor parte de los virus estaban ocultos en los ficheros adjuntos de los mensajes de correo.

Tabla 3. Fuentes de infección por virus informáticos, desde 1996 a 2003.

Fuente del Virus	1996	1997	1998	1999	2000	2001	2002	2003
Adjunto en Email	9	26	32	56	87	83	86	88
Descarga de Internet	10	16	9	11	1	13	11	16
Navegación Web	0	5	2	3	0	7	4	4
Desconocido	15	7	5	9	2	1	1	3
Otro Vector	0	5	1	1	1	2	3	11
Distribución Software	0	3	3	0	1	2	0	0
Diskette	71	84	64	27	7	1	0	0

Fuente: Modificado de Cocca (2004).

Después de pasados muchos años desde el informe de Cocca (2004), los mismos problemas persisten. Ya en el año 2019, un estudio mostró que el número de enlaces maliciosos había superado el número de ficheros adjuntos maliciosos en los correos electrónicos. En el mismo informe, se puede ver que el 61% de los programas maliciosos fueron Emotet, una red de robots (*botnet*) que puede cargar muchos módulos adicionales y puede realizar muchos tipos de ataques, desde robos de información, correo no deseado, hasta convertir el sistema anfitrión en parte de la red para seguir distribuyendo programas maliciosos (Proofpoint, 2018).

Hay que tener en cuenta que estos tipos de ataques generan cuantiosas pérdidas económicas a las organizaciones y los individuos. Estafas como BEC (*Business Email Compromise*, compromiso de correo electrónico comercial) y EAC (*Email Account Compromise*, compromiso de cuenta de correo electrónico) son estafas muy sofisticadas dirigidas tanto a empresas como

a personas que realizan transferencias de fondos. La estafa se lleva a cabo cuando las cuentas de correo electrónico comerciales son comprometidas a través de ingeniería social o técnicas de intrusión informática para realizar transferencias de fondos no autorizadas.

En 2021, el IC3 (*Internet Crime Complaint Center*, Centro de Quejas de Delitos en Internet del FBI) recibió 847.376 quejas de BEC y EAC con pérdidas de más de \$ 6,9 mil millones (*FBI's Internet Crime Complaint Center*, 2021). La Tabla 4 muestra la cantidad de denuncias por ciber ataques y pérdidas económicas sufridas por las organizaciones entre el 2017 y el 2021 debido a los diferentes ciber ataques ejecutados con éxito durante esas fechas.

Tabla 4. Número de denuncias por ciber ataques y pérdidas económicas agregadas del 2017 al 2021.

Año	Denuncias	Pérdidas (Billones)
2017	301.580	\$1,40
2018	351.937	\$2,70
2019	467.361	\$3,50
2020	791.790	\$4,20
2021	847.376	\$6,90
Totales	2.760.044	\$18,70

Fuente: Elaboración propia a partir de datos de FBI's Internet Crime Complaint Center (2021).

El análisis semestral de la División de Inteligencia Cibernética de Agari (ACID) del año 2021 sobre fraudes de correo reflejó la evolución continua del BEC, el *phishing* dirigido (*spearphishing*), las estafas de suplantación de marca dirigidas al consumidor y otras amenazas avanzadas de correo electrónico. Esta evolución está adoptando nuevas permutaciones y nuevos caminos, de tal forma que las amenazas basadas en el correo electrónico se están volviendo más peligrosas e impredecibles día a día (Inc. Agari Data, 2021). SophosLabs (2021) menciona esta sofisticación en uno de los apartados de su informe, cuando analiza BEC. Comenta que esos mensajes no se parecen en nada a un correo no deseado malicioso y contienen información muy convincente para engañar a las víctimas.

Muchas técnicas de ingeniería social están siendo utilizadas hoy en día. El correo electrónico es un medio que utilizan los ciberdelincuentes para engañar y llevar al usuario a que acceda a un sitio malicioso. Se tiene constancia de que el 2% de los empleados acceden a un enlace de phishing (Verizon, 2020).

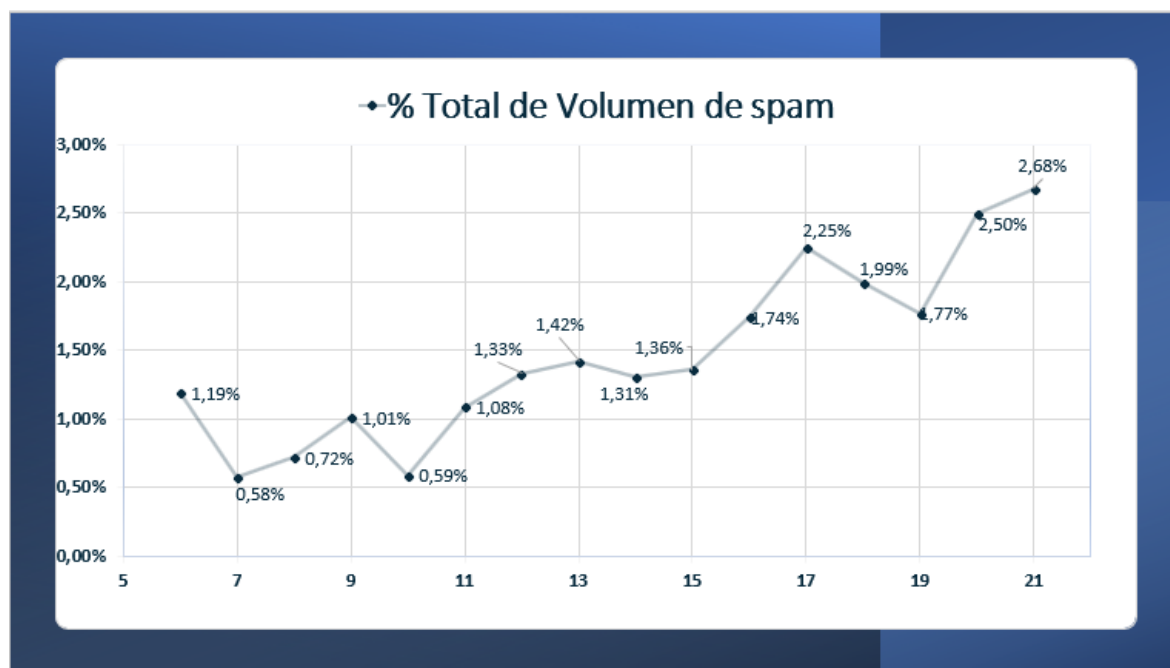
Las técnicas de ingeniería social son utilizadas con frecuencia por los actores de amenazas con motivaciones criminales para personalizar campañas de *phishing*, correos no deseados y

estafas fraudulentas. La psicología detrás de muchas de estas técnicas es aprovecharse de las emociones y el comportamiento humanos, los más explotables de los cuales son la codicia, la curiosidad, el miedo y el deseo de ayudar.

La pandemia de COVID-19 brindó a los actores criminales una oportunidad única de utilizar contenido de señuelos y técnicas de ingeniería social capaces de atacar cada uno de estos componentes del comportamiento humano. Como tema, COVID-19 tiene un impacto global, ha tenido una amplia cobertura de noticias desde 2020 y es un término todavía en auge (CrowdStrike, 2021).

Según el informe de CrowdStrike (2021), uno de los métodos de distribución utilizado por los ciber criminales sigue siendo el correo electrónico. Este comportamiento criminal se ve reflejado en el informe de SophosLabs (2021). En la Figura 1, se puede observar el crecimiento del correo no deseado relacionado con el coronavirus en el mes de marzo de 2020, cuando los trabajadores y estudiantes tuvieron que trabajar en casa debido al confinamiento.

Figura 1. Volumen de correo no deseado con tema "coronavirus" en marzo de 2020.

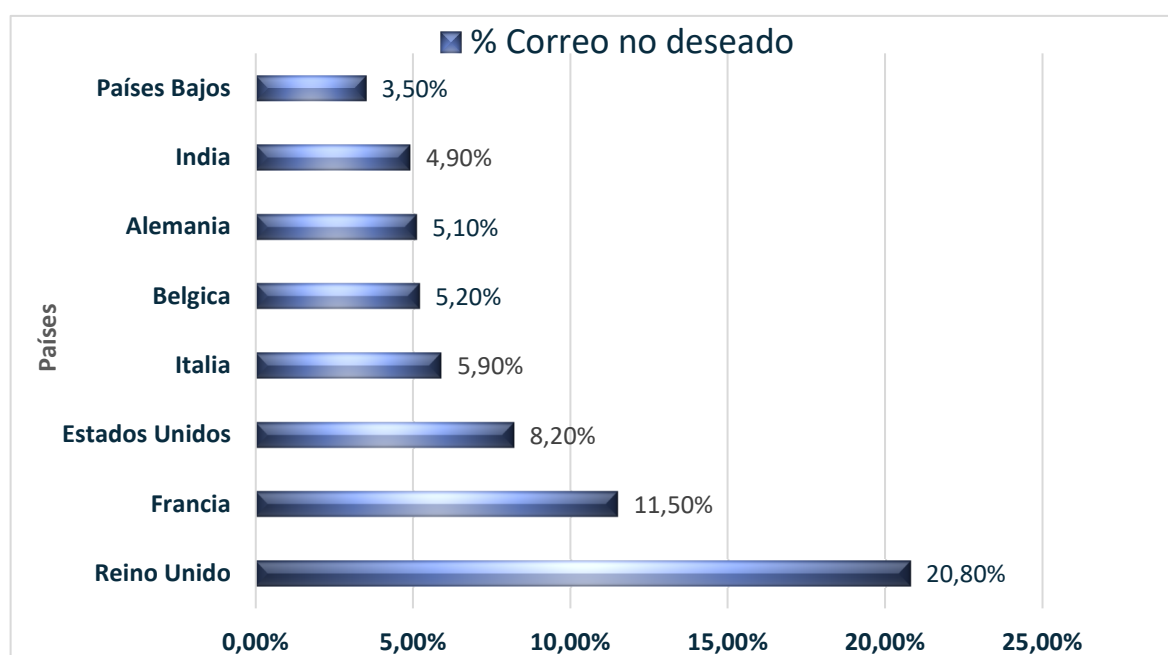


Fuente: Modificado de SophosLabs (2021).

También se vieron reflejados esos datos en otro informe en el que se confirmaron más ataques en todos los países del mundo utilizando la temática de COVID-19, pasando de un 17% en enero del 2020 a un 131% en marzo (Delloite, 2020). La Figura 2 muestra los países con más ataques de correos no deseados maliciosos con el asunto “coronavirus”.

En el año 2020 hubo un aumento significativo en muchas categorías de amenazas, en la actualidad muchos de los incidentes de programas de secuestro y *phishing* se inician o distribuyen a partir de un enlace o descarga de un correo electrónico (McAfee, 2020). Se ha constatado que los archivos adjuntos de correo electrónico y los enlaces no verificados de *phishing*, son puertas de entrada para la activación y distribución de distintos tipos de programas maliciosos, programas de secuestro (*ransomware*) y muchos otros ataques que aprovechan vulnerabilidades presentes en el sistema anfitrión.

Figura 2. Países más atacados por correos no deseados maliciosos con el asunto “coronavirus”.



Fuente: Elaboración propia a partir de datos de Delloite (2020).

Según otros estudios realizados entre 2019 y 2021 (Acronis, 2021; ACSC, 2019; Anti-Phishing Working Group, 2020; Australian Cyber Security Centre, 2020; Check Point, 2021; Cofense, 2020; European Union Agency for Network and Information Security, 2019; Proofpoint, 2020; Verizon, 2020), la tendencia de los ataques utilizando programas maliciosos, programas de secuestro, correo no deseado y *phishing* se encuentra en crecimiento.

Según el informe de DarkTrace (2020) sobre amenazas de seguridad sobre correo electrónico, más del 90% de los ciber ataques se origina dentro del entorno del correo electrónico,

enfaticando que el 94% de los programas maliciosos hoy se origina en la bandeja de entrada. También añade que más de 60.000 sitios de *phishing* fueron reportado solo en marzo del 2020.

En conclusión, nos enfrentamos a un problema que genera un coste económico muy elevado a las empresas, añadiendo a esto los diversos problemas asociados por violaciones de la confidencialidad, la integridad y la disponibilidad de la información a los que está expuesta la población en general. Es necesario analizar la seguridad en el contexto del correo electrónico desde el origen y plantear una solución integral que haga frente a los ciberataques comentados en esta sección. Ante la situación expuesta anteriormente, surgen las siguientes cuestiones:

¿Es posible crear una solución cuya arquitectura dé solución al problema de la seguridad del correo electrónico y que se acople con facilidad a las infraestructuras tecnológicas existentes hoy en día para su rápida adopción?

¿Están preparados los protocolos de correo electrónico utilizados en la actualidad para adaptarse a los nuevos requisitos de seguridad expuestos por la sociedad moderna en el contexto del correo electrónico o es necesaria la adopción de nuevos protocolos?

II. HIPÓTESIS DE LA INVESTIGACIÓN

Se formula la hipótesis general del trabajo de la investigación de la forma siguiente:

Hipótesis general: El correo electrónico hoy en día presenta innumerables vulnerabilidades de seguridad que pueden ser resueltas utilizando un sistema de correo alternativo distribuido y seguro basado en un nuevo protocolo, acorde a los avances tecnológicos de los últimos tiempos en seguridad informática y que cumpla con: los principios básicos de la seguridad de la información y los requisitos de los diferentes participantes que hacen uso de la solución.

Se plantean cuatro hipótesis específicas:

Hipótesis específica 1: La determinación de los requisitos de seguridad de cada uno de los participantes facilita el diseño de una solución que se adapte a sus necesidades.

Hipótesis específica 2: La creación de una arquitectura flexible permite la configuración de la solución para diferentes casos de uso.

Hipótesis específica 3: La definición de los protocolos de seguridad facilitará su adopción y estandarización por los organismos interesados.

Hipótesis específica 4: La realización de una prueba de concepto facilitará el análisis y definición de los trabajos futuros.

III. OBJETIVO DE LA INVESTIGACIÓN

Una vez presentada la motivación y la definición del problema, el objetivo principal de la investigación es:

Objetivo principal: Diseñar una solución de correo electrónico segura basada en una arquitectura flexible, integrable y de fácil adopción por parte de los usuarios para dar solución al problema de la seguridad de este tipo de sistemas proporcionando servicios de confidencialidad, integridad, disponibilidad, no repudio, autenticación, control, auditoría, usabilidad e integración.

Los siguientes objetivos específicos deben ser cumplidos para alcanzar el objetivo general expuesto.

Objetivo específico 1: Analizar y definir los requisitos de seguridad deseables por los diferentes actores o participantes en una comunicación de correo electrónico.

Objetivo específico 2: Diseñar una arquitectura de *software* flexible que permita la gestión del correo electrónico.

Objetivo específico 3: Diseñar los protocolos para enviar, recibir y gestionar el correo electrónico.

Objetivo específico 4: Desarrollar una prueba de concepto basada en la arquitectura desarrollada.

IV. METODOLOGÍA

El desarrollo de esta tesis se basó en una metodología de mejora continua de las actividades de investigación según la *Guide Expérimental pour la Qualité en Recherche* (Miguel, 2005). Este ciclo de mejora continua basado en el ciclo de Deming clásico, permitió la valoración y evaluación rápida de los criterios elegidos mediante la aplicación directa de las soluciones en herramientas que podían ser implantadas, dando una gran efectividad a todo el proceso de investigación. Las fases de investigación giraron en torno a tres actividades: definición del objetivo, realización de la investigación y valoración de los resultados.

En una primera fase de objetivos se realizó lo siguiente:

1. Evaluar los objetivos de acuerdo con los conocimientos actuales. Para ello se utilizaron las siguientes herramientas: revisiones bibliográficas, contactos con empresas (empresa colaboradora si fuese necesario) y utilización de redes de expertos.
2. Evaluar los recursos necesarios y analizar los riesgos asociados a la investigación. Para ello, se realizó una planificación detallada del anteproyecto en el primer año.
3. Más específicamente se desarrollaron las siguientes actividades:
 -  Estudio del Protocolo de Transferencia de Correo Simple (SMTP), el Protocolo de Oficina Postal (POP) y el Protocolo de Acceso a Mensajes de Internet (IMAP) con el objetivo de conocer en profundidad los protocolos de correo más utilizados.
 -  Estudio de las principales amenazas y ataques que sufre hoy en día el correo electrónico y los costes para los usuarios.
 -  Estudio del estado del arte sobre la seguridad del correo electrónico. En este estudio se hizo un análisis de los principales aportes de soluciones de seguridad, comerciales y académicos, y sus resultados comparativos.
 -  Identificación de requisitos para el desarrollo de un correo electrónico seguro y los aspectos de seguridad considerados.
 -  Estudio de la cadena de bloques y sus principales características de seguridad.

En una segunda fase de desarrollo se realizó lo siguiente:

4. Se diseñó y desarrolló la contribución asegurando la trazabilidad de los resultados y haciendo revisiones de seguimiento.
5. Se realizaron revisiones periódicas con el director y tutor de la tesis y se pusieron en marcha acciones correctivas y preventivas.
6. Las actividades realizadas en esta fase fueron las siguientes:
 -  Diseño de una arquitectura denominada BBM con una serie de protocolos que explotan las características de seguridad de *blockchain*.
 -  Implementación de una solución basada en BBM que garantizó los aspectos de seguridad descritos en este trabajo.
 -  Desarrollo de pruebas para la validación de los requisitos identificados.

Para el desarrollo de esta investigación ha sido necesaria la utilización de diferentes tecnologías:

- **Visual Studio Code:** Editor de código fuente desarrollado por Microsoft.
- **JavaScript:** Lenguaje de programación interpretado del estándar ECMAScript.
- **Java:** Lenguaje de programación orientado a objetos. Existen muchas librerías criptográficas disponibles.
- **Python:** Lenguaje de programación de alto nivel interpretado.
- **Enterprise Architect:** Herramienta de modelado que facilita el diseño de los protocolos. Se utilizó para el modelado de la arquitectura, empleando la notación C4 para el modelado de arquitecturas de software.
- **Ganache:** Blockchain para uso personal y liviana basada en Ethereum que facilita el desarrollo y programación de contratos inteligentes. Facilita la programación y pruebas de aplicaciones descentralizadas (dApps).
- **Truffle:** Entorno de desarrollo que utiliza la máquina virtual de Ethereum (EVM) como base. Facilita enormemente el desarrollo y despliegue de contratos inteligentes sobre Ethereum.
- **Solidity:** Lenguaje de programación orientado a objetos utilizado para la creación de contratos inteligentes en distintas plataformas de *blockchain*.

- **Remix IDE:** Aplicación de escritorio y *web* para el desarrollo, despliegue y ejecución de contratos inteligentes sobre Ethereum.
- **MongoDB:** Sistema de bases de datos NoSQL orientado a documentos y que utiliza el formato de Notación de objetos Javascript (JSON) para el almacenamiento de información.
- **SQLite:** Base de datos relacional compatible con SQL-92 contenida en una biblioteca de librerías muy pequeña e ideal para programas ligeros.
- **Node.js:** Entorno multiplataforma de código abierto basado en JavaScript que facilita el desarrollo de aplicación *backend*.

V. ESTRUCTURA DEL TRABAJO

La estructura del trabajo se divide en 6 capítulos:

 **Capítulo 1:** Establece la motivación, objetivos, hipótesis, metodología y estructura de la tesis que ha llevado al desarrollo de esta memoria.

 **Capítulo 2:** Comprende el marco teórico, donde se realiza un estudio de la evolución de los protocolos de correo electrónico, se presenta su arquitectura y el estado del arte de las principales propuestas de seguridad de correo electrónico existentes. Se introducen las tecnologías *blockchain* y se describen sus propiedades de seguridad, los diferentes protocolos de consenso y su arquitectura.

 **Capítulo 3:** Se presentan los resultados de esta investigación. Inicialmente se definen una serie de conceptos base que sirven como introducción de la propuesta. Se listan los requisitos que deben ser considerados para el diseño y desarrollo de una propuesta de correo electrónico seguro. En base a esos requisitos, se diseñan los aspectos estáticos y dinámicos de la arquitectura propuesta denominada BBM. De los aspectos estáticos se muestra los principales componentes y contenedores de la arquitectura.

En cuanto a la parte dinámica, se desarrollan todos los protocolos que garantizan los aspectos de seguridad considerados. En la Tabla 15 se presenta un resumen de todos los protocolos de BBM y los aspectos claves considerados para comprender fácilmente la filosofía y enfoque de seguridad en BBM. Se detalla la funcionalidad de una aplicación distribuida desarrollada como prueba de concepto, denominada BBMail, que implementa los protocolos BBM, y se precisa técnicamente como fueron alcanzados los aspectos de seguridad.

Por último, se añaden algunas consideraciones adicionales, las limitaciones y fortalezas de este trabajo y el trabajo futuro para darle continuidad a este trabajo.

 **Capítulo 4:** Conclusiones sobre la hipótesis del trabajo y cómo se han alcanzado cada uno de los objetivos planteados.

 **Capítulo 5:** Bibliografía utilizada para desarrollar este trabajo.

 **Capítulo 6:** Anexo. Contribuciones científicas.

Capítulo 2

Marco Teórico

I. EVOLUCIÓN DE LOS PROTOCOLOS DE CORREO ELECTRÓNICO DESDE 1971 HASTA 2020

1. Propuesta inicial

Desde el mismo momento en que la tecnología facilitó el uso compartido de información, los usuarios empezaron a intercambiar mensajes entre ellos (Vleck, 2001). Dada esa situación fue necesario implementar protocolos de comunicación que permitieran de forma estándar realizar dicho intercambio de mensajes (Watson, 1971b). Estos primeros protocolos hicieron uso de tecnologías que aparecieron en paralelo y que fueron concebidas para otros propósitos, como el Protocolo de Transferencia de Fichero (FTP) (Bhushan, 1971). Dicha propuesta sirvió como base para una primera especificación (Watson, 1971a) y, basado en un modelo simple, un fichero (el correo) era enviado bajo el protocolo FTP de cuatro pasos, como se muestra en la Figura 3.

Figura 3. Protocolo de correo electrónico de 4 pasos.



Fuente: Modificado de Watson (1971a).

Con esta especificación, basada en un único paso de confirmación (ACK), se dio inicio al proceso de definición e implementación de los protocolos de correo electrónico como se conocen hoy en día. La actualización y creación de nuevos protocolos de correo electrónico fue realizándose conforme a los avances tecnológicos en telecomunicaciones, a la aparición de nuevos formatos

de recursos digitales, al incremento de usuarios de correo y a las necesidades de implementar mecanismos de seguridad eficaces en dicho intercambio de información.

Hemos publicado un estudio sobre la evolución del correo electrónico en el que se describe como se ha ido adaptando al paso del tiempo y a las nuevas necesidades que iban surgiendo con la evolución de las distintas tecnologías (Piedrahita Castillo, Machío Regidor, Bermejo Higuera, Bermejo Higuera, & Sicilia Montalvo, 2020).

2. El correo FTP

En el año 1973, ya había muchos sistemas funcionando que enviaban y recibían correo (Kudlick, 1973a). En ese entonces, empieza a surgir la preocupación por tener un control sobre el diálogo entre dos usuarios. Con ello aparece la necesidad de poder trazar el correo y centralizar la identificación y autenticación de los usuarios. Como inicialmente el correo electrónico circulaba por FTP, los primeros pasos se centraron en actualizar dicho protocolo para adaptarlo a las necesidades del correo electrónico (Bressler & Thomas, 1973).

Bhusman (1973) diferenció entre buzón (*mailbox*) y la dirección del buzón (*mailbox addresses*). Un buzón es definido como un lugar donde el correo es almacenado y el usuario puede seleccionarlo, pudiendo ser un fichero en el directorio del usuario o un binario para un fichero impreso. Por otra parte, la dirección del buzón o dirección del correo es la dirección que requiere el emisor para enviar el correo a un buzón de destino. Inicialmente la dirección era la del servidor anfitrión (*host*) y la identificación del buzón era el usuario en el *host*, que es el usuario creado en el servidor FTP para poder enviar correo.

Los requisitos del contenido del correo FTP fueron descritos específicamente por Kudlick (1973b) y posteriormente implementados dentro de los comandos de FTP. Los siguientes campos fueron propuestos:

-  **TO:** Es requerido y puede contener muchas direcciones. La sintaxis estándar es `user@host`.
-  **FROM:** Proporciona la dirección de retorno para notificaciones en caso de que el correo no pueda ser entregado y es la identificación del emisor.
-  **Author:** El autor del correo, que puede ser uno o más. Esto diferencia al autor del emisor.
-  **Acknowledgment:** Es la confirmación para indicar éxito o fallo en el envío.
-  **Recorded:** Indica si se almacena o no el correo.
-  **Type:** Indica al servidor qué hacer cuando almacena el correo.

📄 **Text/File/Citation:** Se propuso el campo *Text* para el texto del correo. *File* no estaba claro, pero fue el precursor del adjunto. *Citation* es un puntero a un fichero que el emisor quiere que se lea.

Una primera aplicación de este protocolo fue el uso de este por las oficinas postales. Se propuso el envío de correo a un servidor de la oficina postal para que desde ahí se le hiciera llegar una copia impresa a las personas que no tenían acceso a terminales online (Braden, 1973). Otra de las aplicaciones fue como servicio de envío de artículos periodísticos (White, 1973b).

3. Mail Protocol (MP)

En junio de 1973, James White propone un protocolo de correo nuevo denominado Mail Protocol (MP) con la intención de que fuera más rico en expresividad que el antecesor, aunque continuó funcionando como un subconjunto de los comandos de FTP (White, 1973a).

Este protocolo se basó en los trabajos anteriores y definió un modelo que tenía las siguientes características:

- ✉ Soportaba todas las formas de comunicación entre usuarios.
- ✉ Las piezas de correo podían ser reenviadas de un lugar a otro.
- ✉ Las piezas de correo podían ser contestadas.
- ✉ La identidad del autor podía ser verificada, evitando la falsificación y tergiversación.
- ✉ Las piezas de email podían ser catalogadas permanentemente.
- ✉ Algunos hosts aceptaban correo de destinatarios que residían en cualquier lugar de la red y asumían la responsabilidad de distribuirle el correo. Si los hosts estaban caídos, otro servidor podía recibir el correo.
- ✉ Listas de acceso. Un conjunto de personas que podían acceder a la pieza de correo almacenada.
- ✉ Lista de distribución. Entre los datos de la lista de usuarios que debían recibir el correo, ya incluían el campo firma.

A medida que se fueron madurando los requisitos de los sistemas de correo, las necesidades fueron acusando las limitaciones de FTP. Una de esas limitaciones era el uso de las cabeceras del correo (Bhushan, Pogran, Tomlinson, & White, 1973), ya que no podían especificarse explícitamente, lo que llevo a proponer soluciones para estandarizar el formato de las cabeceras. Otros códigos de error se añadieron sobre FTP para que el servicio de correo fuera más confiable (Sussman, 1974).

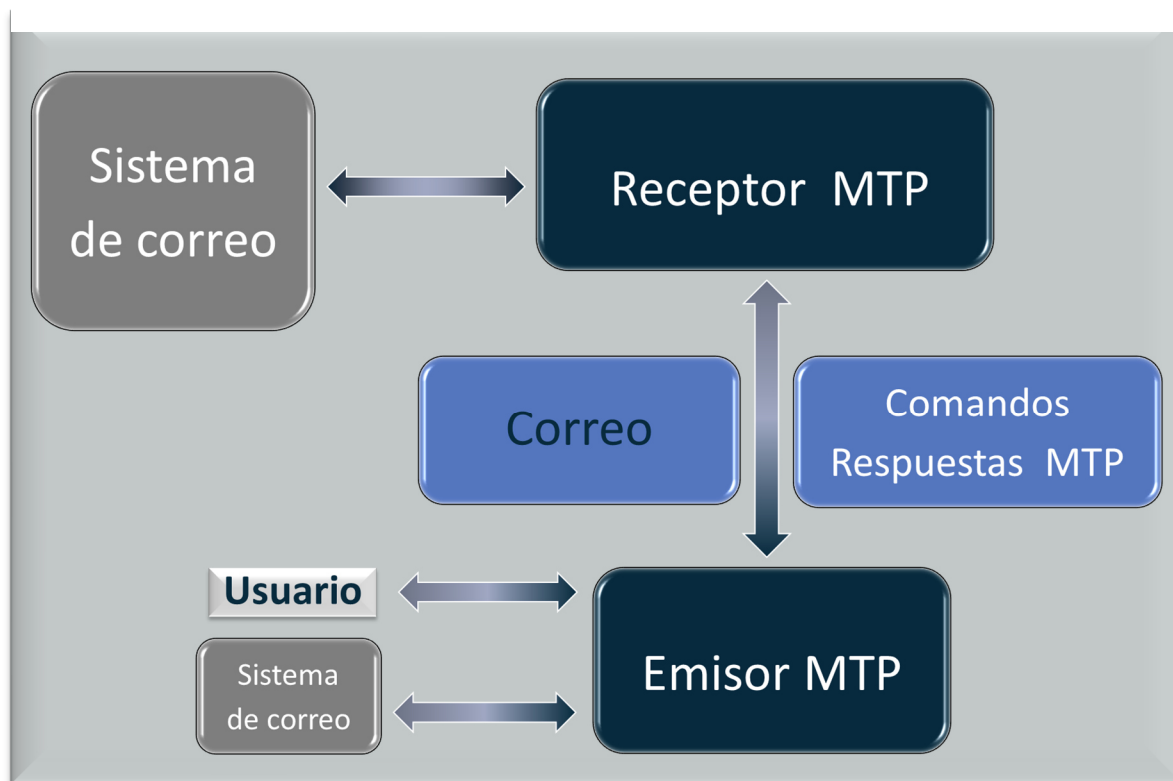
4. Protocolo de Transferencia de Correo (MTP)

A finales de los años 70, el interés en extender el sistema de correo a otros campos aumentó. La idea era permitir también voz, fax, gráficos y texto entre usuarios de diferentes redes, como, por ejemplo, la red de la Agencia de Proyectos de Investigación Avanzada (ARPAnet) (Postel, 1979). Con la aparición y maduración de los nuevos protocolos de redes en 1976, y en especial del Protocolo de control de transmisión/Protocolo de Internet (TCP/IP), a principios de los años 80, surgió la necesidad de elaborar un plan de transición del correo de los viejos protocolos de ARPAnet a los nuevos protocolos de Internet.

El entorno de Internet utilizaba el protocolo TCP como protocolo de transporte host a host y ARPAnet, el Protocolo de Control de Red (NCP). Había que crear un puente entre los dos protocolos para poder realizar envíos de mensajes entre las dos redes, afectando a los usuarios en lo más mínimo. Esta transición dio lugar a la aparición al Protocolo de Transferencia de Correo (MTP), cuya especificación estuvo a cargo de J. Postel en el RFC772 (Sluizer & Postel, 1980). El objetivo, como se comenta en el documento, fue transferir correo de forma segura y eficientemente.

La Figura 4 muestra el modelo para el uso de MTP, en el que se representan los actores del protocolo y sus interacciones. La conexión TCP/NCP se inicia sobre la base del protocolo de la Red de Teletipo (Telnet). Sobre esta conexión, se envían los comandos MTP entre emisor y receptor, incluyendo el correo. Esta comunicación se basa en el estándar puerto/conector (*port/socket*) correspondiente a la conexión TCP/NCP.

En el año 1981, Sluizer & Postel (1981) especificaron más detalles sobre el protocolo MTP. Destaca la propuesta de un protocolo independiente de la capa de transmisión, que hasta esa fecha utilizaban FTP, NCP o TCP. MTP permitía también la capacidad de reenvío de correo de un entorno de transporte a otro. Este protocolo no fue implementado comercialmente, aunque sirvió de base para los siguientes trabajos.

Figura 4. Modelo para el uso de MTP.

Fuente: Modificado de Sluizer & Postel (1980).

5. Protocolo de Transferencia de Correo Simple (SMTP)

En noviembre de 1981, nace SMTP (Postel, 1981b), que reemplaza totalmente a MTP. SMTP es un protocolo a nivel de aplicación dentro de la pila de protocolo TCP que permite transferir correo de un servidor a otro. Dentro de esta primera especificación del protocolo SMTP, vale la pena subrayar algunos aspectos. El modelo SMTP presentado en la propuesta inicial se basaba completamente en el modelo MTP, por lo tanto, tenía las mismas fortalezas y debilidades que su predecesor.

Una de las características de la propuesta es que un servidor SMTP permitía que cualquier usuario lo pudiera utilizar para reenviar correo electrónico. Esta característica se denomina relé (*relay*). De esta forma, un correo enviado a un destinatario que no se encontraba en el servidor, era reenviado al servidor correcto, esto es uno de los problemas que facilita el envío de correo no deseado.

También con SMTP surgieron otros problemas relacionados con el direccionamiento (Postel, 1982a), y entre las soluciones propuestas, la creación de los dominios de email tenía especial relevancia. La propuesta modificaba el modelo típico de direccionamiento user@host en un

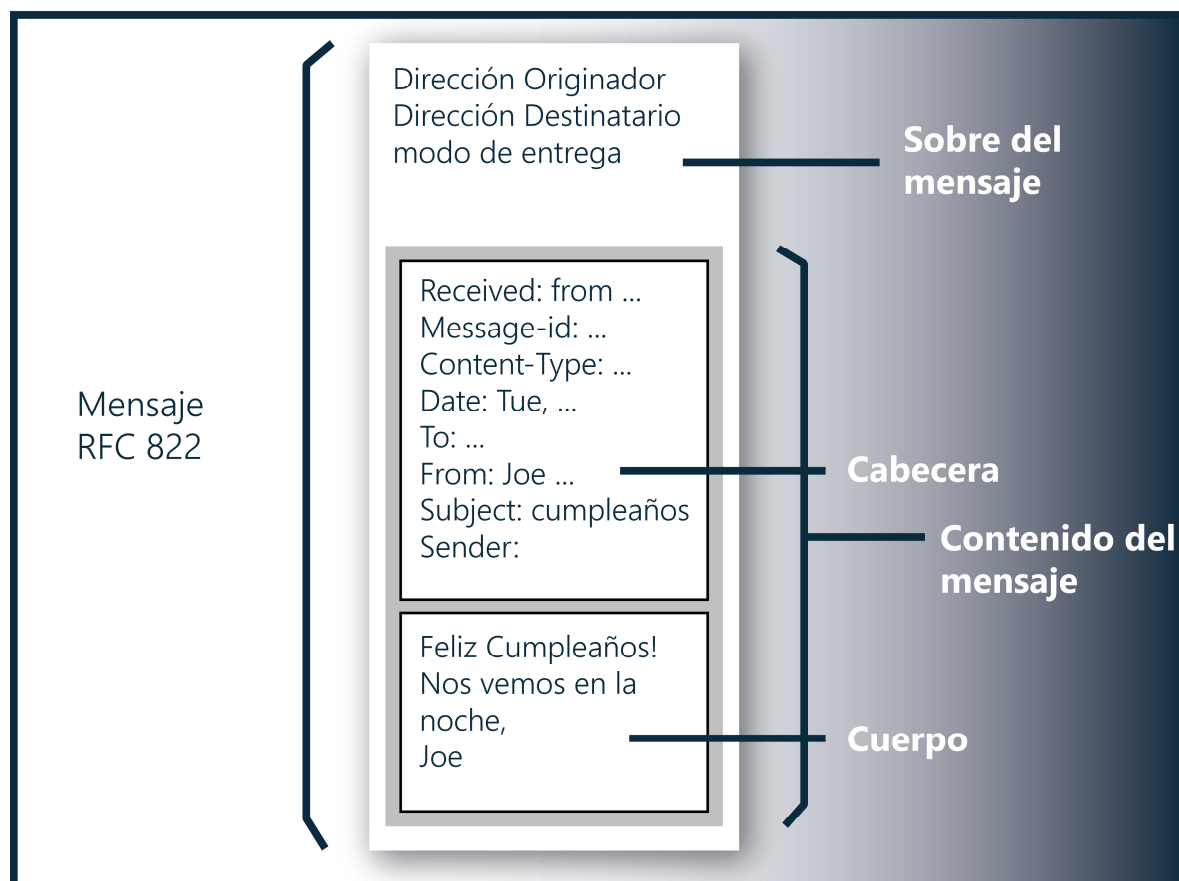
nuevo modelo user@location-id donde el id debería ser único e independiente de la localización.

En menos de un año, la propuesta quedó obsoleta (Postel, 1982b) y a partir de ahí, posteriores actualizaciones fueron dándole más consistencia al protocolo, hasta convertirse en el protocolo que es utilizado actualmente (Durand & Dupont, 1995; Klensin, 2001, 2008, 2015).

El protocolo SMTP no describe el formato del mensaje. Para poder conocerlo hay que ir al RFC822, donde se especifica detalladamente el formato utilizado (Crocker, 1982). En la Figura 5, se puede observar como el mensaje RFC822 consta de un sobre y contenido.

Hay que diferenciar entre el formato definido en el RFC822 y SMTP. Cuando se envía un mensaje utilizando SMTP, todos los campos definidos en la cabecera del contenido del mensaje RFC822 (Figura 5) no tienen ningún efecto durante el transporte SMTP. Este detalle es la base para comprender dónde se encuentran las debilidades del protocolo y cómo es explotado para realizar muchos tipos de ataques.

Figura 5. Formato del mensaje.



Fuente: Modificado de Microsoft (2008).

6. Protocolo de Oficina Postal (POP) y Protocolo de Acceso a Mensajes de Internet (IMAP)

No solo fue necesario formalizar la forma de enviar correo, también fue necesario acceder de forma común a la gran cantidad de correos que llegaba a los buzones. Aunque existían formas no estandarizadas de hacerlo, Reynolds (1984) sugirió un método estandarizado sencillo de recuperar los correos de los servidores, el protocolo POP. Su simplicidad era muy llamativa, ya que, con un esquema simple de autenticación por medio de un usuario y contraseña, el servidor le entregaba al usuario todos los correos y procedía al borrado.

Había pasado menos de cuatro meses cuando se publicó la versión 2 del protocolo POP (M. Butler, Postel, Chase, Goldberger, & Reynolds, 1985). Ya no era una propuesta simple, sino un documento muy detallado en el que se presentaba una filosofía muy clara de solución al problema de la recuperación del correo. Se partía del hecho de que el buzón siempre estaba activo y que la estación de trabajo que utilizaba el usuario algunas veces estaba apagada. De esta forma, el buzón nunca estaba en el equipo del usuario y era el servidor quien lo alojaba, proporcionándole el correo cuando se necesitara. POP2 proporcionaba la separación de trabajos entre el cliente y el servidor, el cliente hacía la descarga del correo y podía trabajar *offline* y el servidor simplemente quedaba a la espera de más correo. Otras actualizaciones fueron apareciendo y, en la actualidad, la versión del protocolo es conocida como POP3 (Myers & Rose, 1996).

Aunque POP estaba popularizado, IMAP hizo su aparición en el año 1988 (M. R. Crispin, 1988). La necesidad de poder conservar el correo en el servidor por muchas razones y el hecho de poder eliminarlo facilitaban la gestión completa del buzón por parte del usuario. IMAP es utilizada actualmente y está especificada en el RFC 3501 bajo la versión 4 (M. Crispin, 2003).

II. ARQUITECTURA DE LOS PROTOCOLOS DE CORREO ELECTRÓNICO EN LA ACTUALIDAD

En el apartado anterior se explicó cómo ha ido evolucionando el correo electrónico y la aparición de los principales protocolos o especificaciones que forman el núcleo de todo el sistema (Freed & Borenstein, 1996b; Klensin, 2008; Resnick, 2008). En este apartado, se presentará la arquitectura del correo electrónico presentada por Crocker (2009) y los actores principales que participan o la componen, así como las relaciones entre ellos. Tener una vista general de todo el sistema facilita su entendimiento y comprensión, y sirve como antesala para describir dónde se presentan los fallos y de qué forma los investigadores participan en sus soluciones.

Un servicio que se ejecuta sobre Internet está integrado por una serie de capacidades relacionadas entre todos los nodos que participan. Dichas capacidades se logran por medio de los protocolos. Lo que conecta un protocolo con un servicio es lo que se denomina una arquitectura. Una arquitectura de correo electrónico especifica cómo los diferentes protocolos que exponen sus capacidades implementan el servicio definiendo sus componentes lógicos y sus relaciones. Desde un punto de vista lógico, un servicio define lo que se está haciendo, una arquitectura define dónde están las piezas (en relación entre sí) y un protocolo define cómo se realizan las capacidades particulares.

La arquitectura de correo electrónico expuesta en este apartado define los componentes lógicos del correo electrónico de Internet, tratando de capturar los principios arquitectónicos que incorporan los protocolos de correo electrónico actuales. Esta arquitectura es el mejor intento de un modelo lógico de correo electrónico de Internet y, en la medida en que el desarrollo de un nuevo protocolo difiera de esta arquitectura, es necesario comprender esas diferencias y explicarlas adecuadamente.

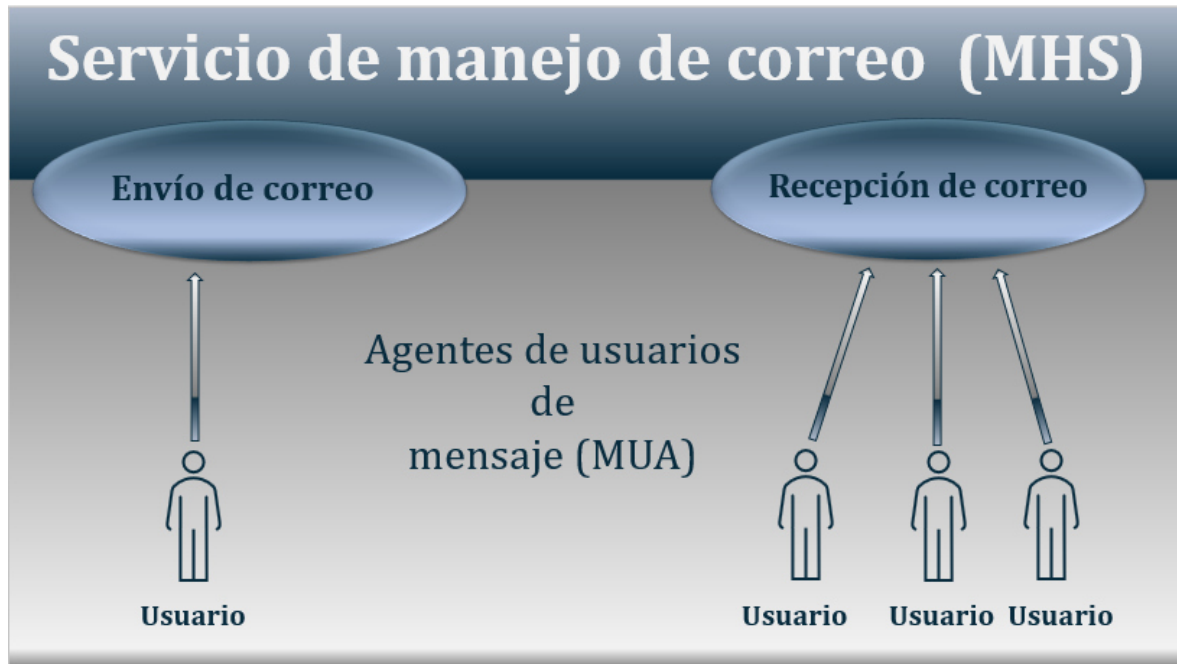
1. Actores Principales

La primera arquitectura especificada para el correo electrónico presentaba dos actores principales: los agentes de usuario de mensajes (MUA) y el servicio de manejo de correo (MHS), donde se realizaban las transferencias de los mensajes y que estaba compuesto por los agentes de transferencia de mensajes (MTA) (Houttuin, 1993).

La Figura 6 muestra de forma esquemática dicha arquitectura, pudiéndose observar como el MHS crea un entorno virtual de intercambio de mensajes entre los diferentes MUA. En esta arquitectura se presentan dos capas lógicas muy diferenciadas. Una de ellas es la capa de

usuarios, en la que hay interacción entre ellos. La otra se encuentra entre los componentes del sistema de transferencia a lo largo de todo el camino que realiza un mensaje para llegar a su destino. Se observan los dos tipos de interacción que hay entre esas dos capas: una se produce cuando se envía un mensaje a través del MHS y la otra, cuando se recibe.

Figura 6. *Arquitectura básica del servicio de correo en Internet.*



Fuente: Modificado de Crocker (2009).

Hoy en día este esquema permanece constante, ya que el servicio no ha cambiado, el direccionamiento de los buzones de correo y el formato base son los mismos. Solamente han cambiado muchos aspectos internos, haciéndolos más especializados.

Por otra parte, han aparecido muchos servicios adicionales, quedando el correo electrónico subordinado a ellos, como, por ejemplo, el correo de voz (Vaudreuil & Parsons, 2004), el intercambio electrónico de datos (EDI) por correo electrónico (Crocker, 1995) y el envío de fax por correo electrónico (Crocker & Klyne, 2005).

Todos estos componentes y sus interacciones se basan en las siguientes reglas:

- ❑ La existencia de un objeto al que se denominará correo electrónico.
- ❑ Un esquema de direccionamiento global para poder localizar los buzones donde se almacenarán o enviarán dichos objetos.
- ❑ Un sistema que permite la transferencia asíncrona de mensajes punto a punto.

- ☐ No hay ningún acuerdo previo entre los diferentes MTA o entre los autores y los destinatarios.
- ☐ No hay ningún acuerdo entre los servicios de transferencia de punto a punto a través de Internet.
- ☐ Los autores y los destinatarios no necesitan estar en línea al mismo tiempo.

Esta falta de acuerdo entre los participantes ha sido una de las características que más ha beneficiado el crecimiento de este servicio y es uno de los requisitos fundamentales del correo electrónico hoy en día.

Justo en los límites del sistema, se requiere algunas veces algún tipo de gestión previa como el control de acceso, las restricciones de enrutamiento y la configuración del servicio de consulta de información. La autenticación del destinatario siempre ha sido requerida para el acceso a los mensajes, pero en los últimos años también es necesaria para su envío. La idea central es validar la identidad del cliente para identificar los participantes.

El correo electrónico en Internet es una red distribuida y, como tal, hay muchos actores con diferentes roles participando en ella. Los actores principales pueden ser clasificados en tres categorías muy diferenciadas:

- ☐ Usuario.
- ☐ MHS.
- ☐ Dominio de gestión administrativa (ADMD).

1.1. Actores usuarios

Los usuarios son las fuentes y receptores de los mensajes. Además, pueden ser personas, procesos u organizaciones. Dentro de un intercambio de mensajes, la cantidad de usuarios puede variar a medida que se realiza dicho intercambio. Para cualquier usuario del sistema, todas las actividades de transferencia de mensajes son realizadas por el MHS, aunque en realidad ese servicio puede ser proporcionado por muchas empresas independientes. De igual forma, los usuarios son clientes de ese servicio. Siempre que un actor responda a otro actor u originador de mensajes, ese actor como tal será un usuario.

La Tabla 5 muestra los diferentes tipos de actores usuarios que existen y sus responsabilidades. Como característica primordial, los actores usuarios son los que pueden generar, modificar y leer el mensaje completo. Se puede ver cuatro tipos de usuarios y su rol correlacionado, lo que significa que en algún momento toman ese rol dentro del proceso de intercambio de mensajes.

Tabla 5. Actores usuarios y sus responsabilidades.

Actores Usuario	Responsabilidad	Rol Correlacionado
Autor	 Crea el mensaje, contenido y su lista de direcciones de destinatarios.	Originador
Destinatario	 Consume el mensaje.  Puede responder al autor.	Receptor
Gestor de devolución	 Gestiona las notificaciones a medida que se transfiere el mensaje.  Los avisos pueden ser sobre fallos o finalizaciones, se envían a una dirección especificada por el originador.  La dirección de devolución no siempre es posible.	Destinatario
Mediador	 Recibe, agrega, reformula y redistribuye mensajes entre autores y destinatarios.  Puede recibir respuestas.  Autor que actúa como agente de uno o más autores.  Usuario de pleno derecho.  Intenta preservar la información del autor original en el mensaje que reformula, pero se le permite realizar cambios significativos en el contenido o sobre del mensaje.  Conecta a otros participantes.  Responsable del nuevo mensaje.  Regula qué usuarios pueden participar y cuándo.  Ejemplo: lista de distribución grupal.	Autor/Puesta de enlace

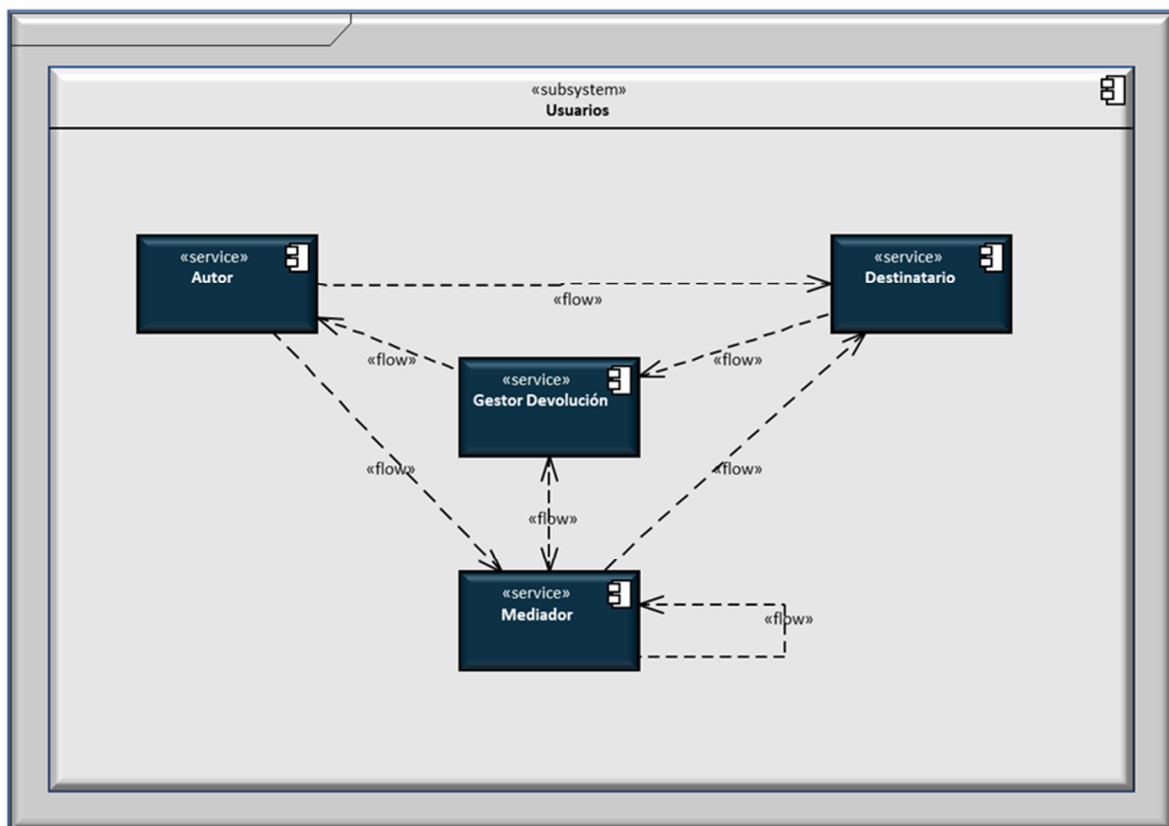
Fuente: Elaboración propia.

Aunque los roles de los diferentes actores usuarios son auto explicativos, el “mediador” necesita una explicación adicional, ya que, desde la perspectiva de los usuarios usuales de correo electrónico, ese rol permanece oculto. La actividad del “mediador” se confunde fácilmente con los intercambios realizado en el MHS. Sin embargo, cada uno tiene propósitos muy diferentes y opera de formas muy diferentes.

Cuando el correo se entrega al “mediador” especificado en el comando RcptTo (Klensin, 2008) en el mensaje original, el MHS lo maneja como cualquier otro destinatario, considerando que cada actividad de publicación y entrega entre fuentes y destinatarios es independiente. No ve la reasignación como una continuación de un proceso.

Un “mediador” intenta preservar la información del autor original en el mensaje que reorganiza, pero puede realizar cambios significativos en el contenido o sobre del mensaje. Un uso más complejo de un “mediador” puede ser realizando una secuencia de pasos formales, como revisar, modificar y aprobar una solicitud de compra. Algunas veces un “mediador” actúa como una puerta de enlace (*gateway*), ya que es un híbrido de usuario y retransmisor que conecta servicios de correo heterogéneos. La Figura 7 muestra los flujos principales y secundarios de los mensajes entre esos actores.

Figura 7. Relaciones entre los actores usuarios.



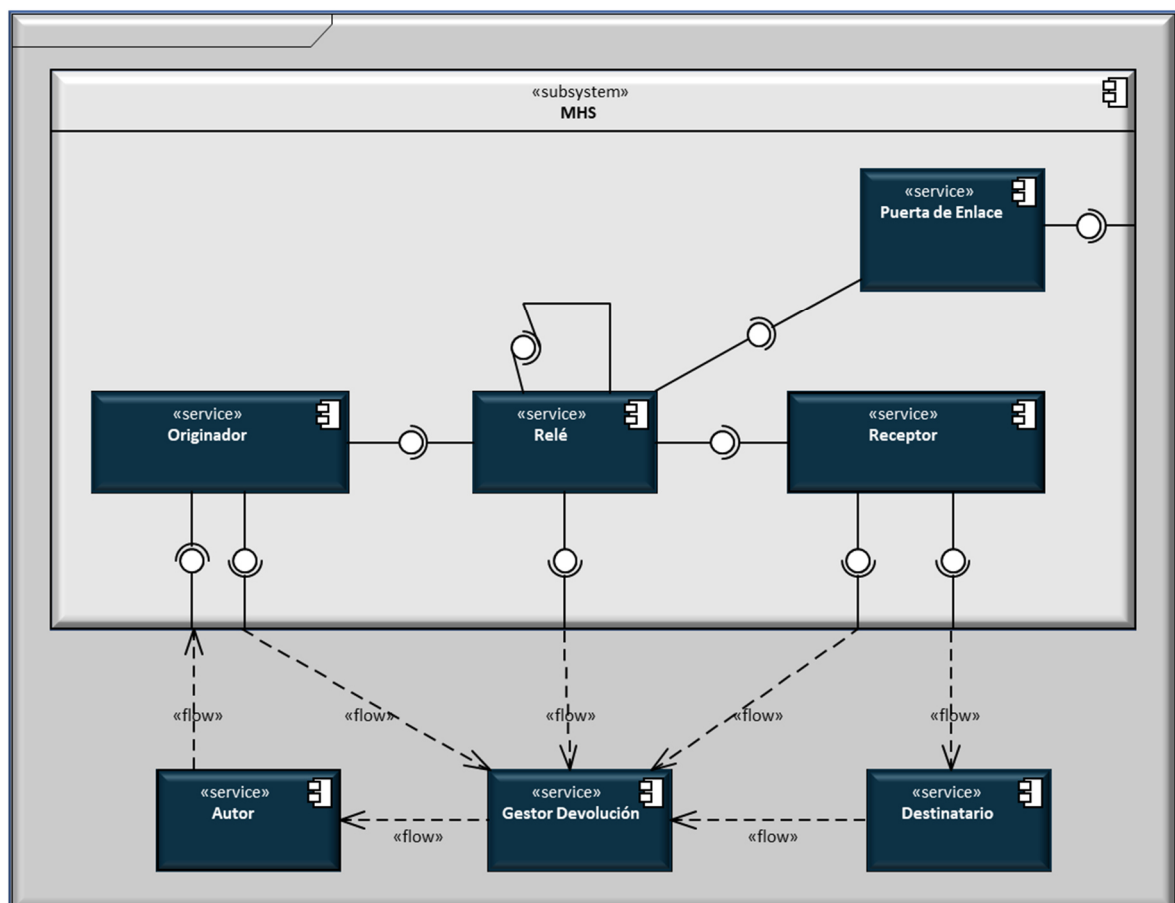
Fuente: Elaboración propia.

1.2. Actores del MHS

MHS transfiere de extremo a extremo los diferentes mensajes con el objetivo de alcanzar las direcciones de destinatario especificadas en los comandos RcptTo originales. Los intercambios que son mediados, como los que se utilizan para la colaboración a lo largo del tiempo, son gestionados por los actores del usuario, no por los actores de MHS. Los actores del MHS generan, modifican o miran solo los datos de transferencia, en lugar de todo el mensaje.

La Figura 8 muestra las relaciones entre los actores de MHS. Aunque en la gráfica se muestran diferenciados el rol “originador” y el de “autor” y el “receptor” del “destinatario”, estos pares de roles generalmente tienen el mismo “actor”. Se puede apreciar en la gráfica que las transferencias pueden hacer uso de uno o más relés (*relays*), es posible la entrega directa desde el “originador” al “receptor”. Los servicios de correo dentro de la organización suelen tener un solo relé.

Figura 8. Relaciones entre los actores de MHS.



Fuente: Elaboración propia.

En la Tabla 6 están listadas las principales responsabilidades de los actores MHS. El “originador” es realmente un validador que simplemente puede revisar el mensaje para verificar su conformidad y rechazarlo si encuentra errores. Además de la validación, el “originador” también realiza las tareas administrativas posteriores al envío.

Una vez el “originador” valida el mensaje, el relé realiza el enrutamiento del servicio de transferencia a nivel del MHS, agregando información de seguimiento (Lindberg, 1999), pero no modifica la información del sobre ni la semántica del contenido del mensaje. Un conjunto de relés forma una red del MHS, pero esa red está por encima de cualquier red de conmutación de paquetes subyacente que pueda utilizarse, y por debajo de cualquier puerta de enlace u otros mediadores, dando lugar a tres capas arquitectónicas distintas, cada una de las cuales proporciona su propio tipo de datos de servicio de almacenamiento y reenvío:

- ☒ Mediadores de usuarios.
- ☒ Relés MHS.
- ☒ Conmutadores de paquetes (servicio IP de Internet).

Un actor importante es la puerta de enlace, ya que es quien permite la conexión entre servicios de correo heterogéneos, funciona como un relé y también con un usuario cuando necesita modificar el contenido del mensaje. Es un actor complejo porque hay muchas diferencias, tanto sintácticas como semánticas, entre los diferentes sistemas de correo electrónico, así como problemas de compatibilidad con el contenido (texto, multimedia, etc.). Al final, el objetivo es que dos usuarios puedan mantener un diálogo de forma transparente entre dos sistemas de correo con características semánticas o sintácticas muy distintas.

Tabla 6. Actores MHS y sus responsabilidades.

Actores Usuario	Responsabilidad	Rol Correlacionado
Autor	 Crea el mensaje, contenido y su lista de direcciones de destinatarios.	Originador
Destinatario	 Consume el mensaje.  Puede responder al autor.	Receptor
Gestor de devolución	 Gestiona las notificaciones a medida que se transfiere el mensaje.  Los avisos pueden ser sobre fallos o finalizaciones, se envían a una dirección especificada por el Originador.  La dirección de devolución no siempre es posible.	Destinatario
Mediador	 Recibe, agrega, reformula y redistribuye mensajes entre autores y destinatarios.  Puede recibir respuestas.  Autor que actúa como agente de uno o más Autores.  Usuario de pleno derecho.  Intenta preservar la información del autor original en el mensaje que reformula, pero se le permite realizar cambios significativos en el contenido o sobre del mensaje.  Conecta a otros participantes.  Responsable del nuevo mensaje.  Regula qué usuarios pueden participar y cuándo.  Ejemplo: lista de distribución grupal.	Autor/Puesta de enlace

Fuente: Elaboración propia.

1.3. Actores administrativos

La necesidad de que diferentes grupos interactúen entre sí y la independencia operativa entre las diferentes organizaciones, donde cada una de ellas maneja sus propias políticas operativas y decisiones basadas en la confianza, da lugar a la distinción de los diferentes ADMD. Un ejemplo claro es la gran diferencia de reglas que puede tener la gestión del correo dentro de una organización y la gestión entre las diferentes organizaciones. Dada la diferencia, es imperativo definir los límites de cada uno, y esto requiere el uso de un ADMD.

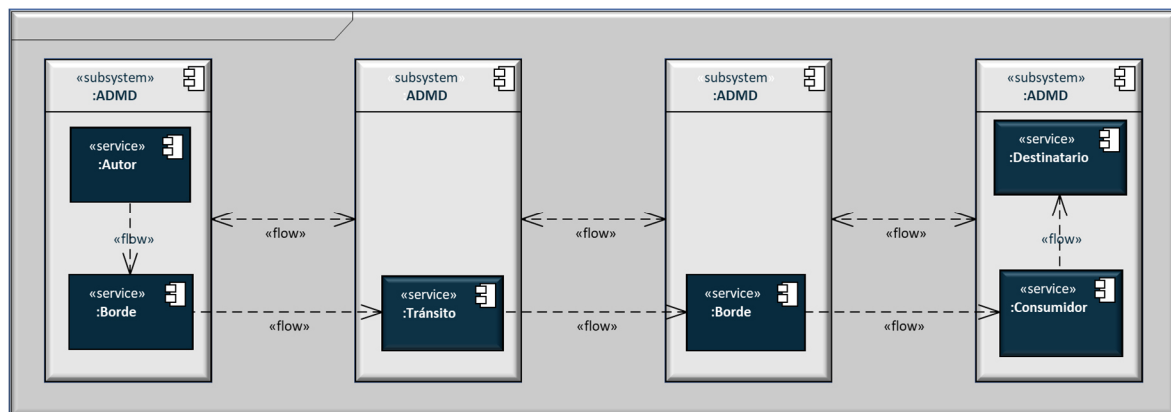
El correo de Internet es usualmente gestionado por diferentes proveedores, cada uno de ellos puede ser perfectamente un ADMD independiente:

- ‡ Un departamento que opera un relé local.
- ‡ Un departamento de sistemas que opera un relé empresarial.
- ‡ Un proveedor de servicios de Internet (ISP) que opera un servicio de correo electrónico compartido público.

Se pueden configurar de manera diferente, dando lugar a disposiciones complejas de componentes funcionales.

La Figura 9 muestra un ejemplo de relaciones entre ADMD independientes. Esta disposición facilita el diseño, las definiciones de políticas y las operaciones para poder distinguir los problemas internos y externos que puedan surgir (D. D. Clark, Wroclawski, Sollins, & Braden, 2005). Todas las entidades que se comunican a través de los límites de ADMD generalmente tienen que hacer cumplir las políticas organizacionales relacionadas con las comunicaciones externas, de tal forma que el enrutamiento del correo entre ADMD es muy problemático, así como la necesidad de enrutar el correo entre socios organizacionales a través de rutas especialmente confiables.

Figura 9. Ejemplo de ADMD.



Fuente: Elaboración propia.

La Tabla 7 muestra los tres tipos básicos de ADMD. No hay que confundir el servicio de tránsito a nivel de correo con la conmutación a nivel de paquetes. En la conmutación de paquetes de un extremo a otro, los mensajes pasan por enrutadores intermedios, mientras que el intercambio de correo electrónico en Internet puede realizarse directamente entre los MTA en los límites de los ADMD de borde.

Tabla 7. *Tipos básicos de ADMD.*

Tipos de ADMD	Descripción	Ejemplo
Bordes (<i>Edge</i>)	Servicios de transferencia independientes en redes en el borde del servicio de correo de Internet.	Proveedores de servicios empresariales.
Consumidor	En algunas ocasiones actúa como servicio de borde, por ejemplo, en el acceso al correo electrónico basado en la web.	Proveedores de servicios de Internet.
Tránsito	Ofrece capacidades de valor agregado para los ADMD de borde, como agregación y filtrado.	Proveedores de servicios de correo.

Fuente: Elaboración propia.

Los actores administrativos son claves a la hora de analizar la seguridad en el contexto del correo electrónico y las interacciones de los componentes de ADMD normalmente están sujetas a diferentes políticas de ese dominio, que cubren cuestiones como:

- ✎ Fiabilidad.
- ✎ Control de acceso.
- ✎ Responsabilidad.
- ✎ Evaluación y modificación de contenido.

Estas políticas se pueden implementar en diferentes componentes funcionales (Hutzler, Crocker, Resnick, Allman, & Finch, 2007), según las necesidades de cada ADMD. Los proveedores que operan servicios de componentes o conjuntos de servicios pueden ofrecer servicios de consumidor, perimetral y de tránsito. Además, es posible que un ADMD aloje servicios para otros ADMD.

2. Identidades

Dentro de la arquitectura de correo electrónico algunos componentes se deben diferenciar de forma universal o global con diferentes objetivos, como pueden ser la trazabilidad o aspectos de seguridad relacionados con un mensaje. Las formas de identidad únicas y globales que utiliza el correo electrónico en Internet son:

- 👉 Buzón.
- 👉 Nombre de dominio.
- 👉 ID de mensaje.
- 👉 Identificador del sobre (ENVID).

2.1. Buzón

EL buzón, denominado comúnmente *mailbox*, es quizás la parte más conocida de la arquitectura del correo electrónico. Es una entidad conceptual donde usualmente se recibe correo y no pertenece necesariamente al almacenamiento de archivos (Resnick, 2008).

Se especifica como una dirección de correo de Internet que tiene dos partes distintas, separadas por el símbolo de arroba (@). El componente situado al lado derecho de esta signo se denomina nombre de dominio y está asociado con un ADMD (Mockapetris, 1987b). La parte localizada a la izquierda del símbolo @ contiene una cadena que es globalmente opaca, se denomina parte local y es interpretado solamente por la entidad asociada al nombre de dominio de la dirección. Casi todas las entidades de la arquitectura del correo no interpretan la parte local y conservan sus detalles originales. La parte local suele ser interpretada por su creador y utilizada para contactar personal de la organización, como es el caso en la utilización de operaciones o funciones comerciales (Crocker, 1997).

Aunque existen convenciones para distinguir partes de una organización, eso solo es utilizado en el ámbito privado por la parte derecha de la dirección. Las excepciones son los servicios especializados que se ajustan a las convenciones públicas estandarizadas, como se indica a continuación. Hay convenciones que le dan un significado especial a la parte local, como, por ejemplo, los formatos de numeración de teléfonos utilizados en los perfiles de voz para correo de Internet (VPIM) (Vaudreuil & Parsons, 2004) y en el fax por Internet (iFax) (Allocchio, 2001; Toyoda & Crocker, 2005), pero estos son casos excepcionales.

2.2. Nombre de dominio

Un recurso dentro de la Internet pública, como un servidor, un servicio o una red, suele ser referenciado de forma global y única por medio de un nombre de dominio. Puede utilizarse el nombre de una organización, pero no necesariamente, también puede utilizarse en el ámbito individual un nombre de un usuario, aunque no es común.

El nombre está estructurado haciendo referencia a una jerarquía cuyo nivel superior se encuentra en el lado derecho de una secuencia de etiquetas separadas por puntos (.). La secuencia de etiquetas al final se ve como una secuencia de nombres, como, por ejemplo, Microsoft.com. En este caso, “.com” es la parte más alta de la jerarquía bajo la que se encuentra el nombre Microsoft.

El Sistema de Nombres de Dominio (DNS) es donde se operan y definen todas esas etiquetas (Elz & Bush, 1997; Mockapetris, 1987a, 1987b). Un nombre de dominio es utilizado para hacer referencia al ADMD o al servidor que gestionó el mensaje, cuando dentro del contexto del correo electrónico no forma parte del nombre del buzón.

2.3. Identificador del mensaje

Los mensajes tienen dos formas de identificarse dentro de la arquitectura del correo electrónico, a través de dos etiquetas y con dos objetivos muy diferentes:

 **Message-ID:** La etiqueta denominada identificador de mensaje se refiere al contenido del mensaje.

 **ENVID:** La etiqueta denominada identificador del sobre se refiere a la transferencia del mensaje.

a. Identificador del mensaje (Message-ID)

Según el formato de mensajes de Internet (IMF), se proporciona una etiqueta IMF a nivel de usuario para un único ID de mensaje (Resnick, 2008). La etiqueta puede ser utilizada de muchas formas, incluidos los hilos de mensajes, la identificación de duplicados y el seguimiento de la notificación de estado de entrega (DSN).

El originador es quien asigna el ID de mensaje y el ADMD del destinatario lo consume, aunque cualquier actor a lo largo de la ruta de transferencia podría utilizarlo. El ID de mensaje es único a nivel mundial y su formato es muy parecido al de un buzón, con dos partes separadas por un símbolo de arroba. A diferencia del buzón, el lado derecho del identificador de mensaje especifica el ADMD o el servidor que asignó el identificador, y el lado izquierdo contiene una

cadena para identificar de forma única el mensaje dentro del dominio al que se hace referencia en el lado derecho.

La asignación de un nuevo ID al mensaje es una tarea que requiere cierta evaluación de pautas y criterios, ya que durante la transferencia de éste pueden pasar muchas cosas y es necesario considerar muchos aspectos como reenvíos, traducción, redirecciones, modificaciones, etc. Debido a una ausencia de criterios objetivos y precisos para la regeneración de un ID de mensaje, éste no está diseñado para usarse para ninguna función que tenga implicaciones de seguridad.

b. Identificador del sobre (ENVID)

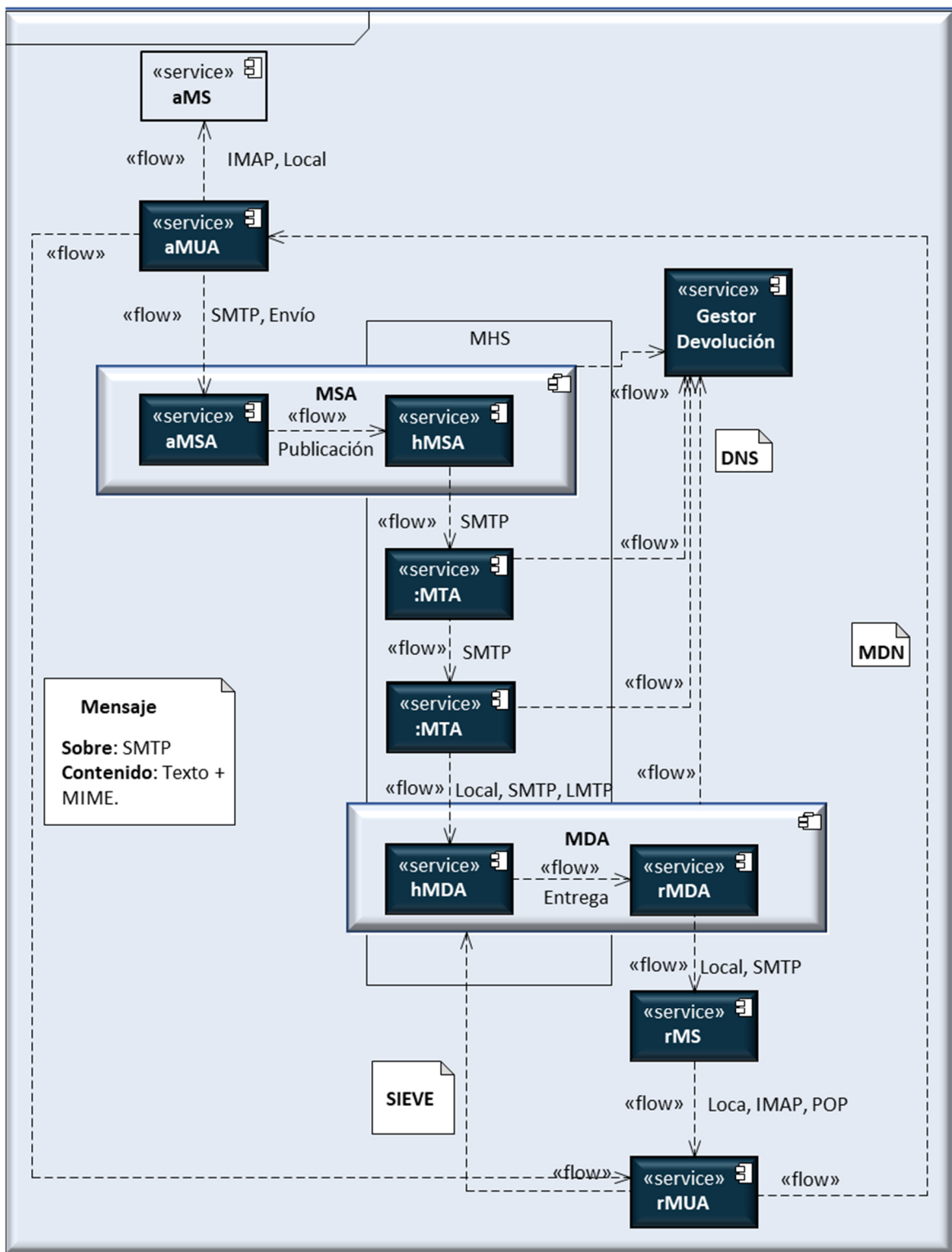
Este identificador se suele utilizar con fines de seguimiento del mensaje a través de toda la ruta de transferencia. El ENVID se utiliza en una sola transferencia del MHS para la publicación de un mensaje hasta que se entrega. Una nueva publicación del mensaje, como, por ejemplo, por parte de un “mediador”, no reutiliza ese ENVID, sino que genera uno nuevo, aunque el mensaje puede conservar su ID de mensaje original.

A diferencia del Identificador del mensaje, el formato de un ENVID es libre. El alcance del identificador sí está determinado por el nombre de dominio de la dirección de retorno (Allman & Hansen, 2004; Moore & Vaudreuil, 2003).

3. Servicios y estándares

Existen seis tipos básicos de funciones dentro de la arquitectura de correo de Internet, que están organizadas con el propósito de prestar un servicio de distribución de correo electrónico. Así mismo, cada una de esas funciones pueden tener diferentes instancias representando roles con funciones más específicas. La Figura 10 muestra esquemáticamente las múltiples instancias y las actividades y relaciones entre los diferentes componentes y los estándares de correo de Internet que se utilizan.

Figura 10. Protocolos y servicios dentro de la arquitectura de correo de Internet.



Fuente: Elaboración propia.

En este apartado, se explicará más detalladamente cada una de las funcionalidades y cómo interactúan entre sí. Se detallarán los seis tipos de funciones y sus especializaciones siguientes:

👤 Mensaje.

👤 Agente de usuario de mensajes (MUA).

☞ MUA del autor (aMUA).

☞ MUA del receptor (rMUA).

👤 Agente de envío de mensajes (MSA).

☞ Funciones de MSA centradas en el autor (aMSA).

☞ Funciones de MSA centradas en MHS (hMSA).

👤 Agente de transferencia de mensajes (MTA).

👤 Agente de entrega de mensajes (MDA).

☞ Funciones MDA centradas en el receptor (rMDA).

☞ Funciones MDA centradas en MHS (hMDA).

👤 Almacén de mensajes (MS).

☞ MS del autor (aMS).

☞ MS del receptor (rMS).

3.1. Mensaje

El objetivo del MHS es intercambiar un mensaje entre los diferentes participantes de una arquitectura de correo electrónico en Internet. Todos los mecanismos internos del sistema deben trabajar coordinadamente para transmitir ese mensaje de su autor a sus destinatarios. Un mensaje puede ser catalogado de acuerdo a su naturaleza y objetivo para el cual se creó (Burger, Candell, Eliot, & Klyne, 2003).

Como se observó en la Figura 5, un mensaje lo componen dos partes, un sobre y el contenido propio del mensaje. El contenido a su vez está dividido en un encabezado bien estructurado y el cuerpo. El encabezado contiene información de seguimiento para la gestión de su tránsito y algunos campos que son parte del mismo contenido del mensaje del “autor” (Resnick, 2008). El cuerpo, que es en sí el contenido del mensaje, puede ser simple como líneas de texto o muy estructurado como un árbol de objetos multimedia, que son conocidos como “adjuntos”, pero que se denominan técnicamente “partes del cuerpo” (Freed & Borenstein, 1996b, 1996c, 1996a; Freed & Klensin, 2005a, 2005b; Moore, 1996).

En la Figura 10, se puede ver las etiquetas DSN, MDN y SIEVE. Estos son controles especiales que se realizan a los datos y que se definen de la siguiente forma:

- ◉ **Notificación de estado de entrega (DSN):** Es un tipo de mensaje que puede ser generado por el MHS (MSA, MTA o MDA) y enviado a la dirección asignada en el comando MailFrom de SMTP (Klensin, 2008). En la Figura 10, se puede ver como MDA y MTA son fuentes de este tipo de mensaje y el destino es una devolución. La información que proporciona este tipo de mensajes suele ser sobre el tránsito de algún mensaje, como los errores de transferencia o una confirmación de entrega (Moore, 2003).
- ◉ **Notificación de disposición de mensajes (MDN):** Este tipo de mensajes son generados justo después de que se ha realizado la entrega y suelen indicar que ya se ha mostrado al receptor algún tipo de contenido que puede ser admitido (Klyne, Iwazaki, & Crocker, 2002). Se puede observar en la Figura 10 que la etiqueta MDN es fuente de los rMUA. El destino de este mensaje son las direcciones Disposition-Notification-To (Hansen & Vaudreuil, 2004). Además, el buzón aparece con la etiqueta "Disp".
- ◉ **Lenguaje de filtrado de mensajes SIEVE:** Es un lenguaje de secuencias de comandos que se utiliza para especificar condiciones de filtro para la gestión del correo en el momento de la entrega (Guenther & Showalter, 2008). La Figura 10 muestra un script de SIEVE que va desde rMUA al MDA. No obstante, el filtrado se puede realizar en cualquier parte de la ruta de tránsito del mensaje, pero en la figura se muestra una sola ruta por simplicidad.

A un mensaje a lo largo del recorrido desde su creación hasta su destino se le asignan una serie de identificadores que son establecidos por diferentes actores dentro del recorrido. La Tabla 8 muestra la capa donde se establece el identificador y el actor que la realiza, así como una descripción que aclara su uso. En ella, se pueden ver las tres capas donde se establecen los identificadores, quién es el actor responsable, una descripción del identificador con su uso y la referencia del documento donde se puede ampliar más información.

Tabla 8. *Identificadores en las diferentes partes del mensaje.*

Capa I. Cuerpo del mensaje			
Campo	Establecido por	Descripción del campo	Referencia
Header MIME	Autor	Cabecera MIME	

Capa II. Campos de la cabecera del mensaje			
Campo	Establecido por	Descripción del campo	Referencia
From:	Autor	Contiene: nombres y direcciones de los autores del contenido del mensaje.	(Resnick, 2008)
Sender:	Originador	- Dirección responsable de enviar el mensaje al servicio de transferencia. - Se puede omitir si contiene la misma dirección que From:. - Suele coincidir con la dirección de devolución de las notificaciones del campo MailFrom (SMTP).	(Klensin, 2008), (Resnick, 2008)
Reply-To:	Autor	Si es establecido, este campo anula el campo From: para las respuestas de los destinatarios.	(Resnick, 2008)
To:, CC:	Autor	- Especifican las direcciones de los destinatarios MUA. - Pueden no estar presentes en el comando RcptTo. - La distinción entre To: y CC: es subjetiva. - Por regla general, un destinatario en To: se considera principal y se espera que actúe en respuesta al mensaje, mientras que un destinatario CC: solo recibe la copia por cortesía.	(Klensin, 2008), (Resnick, 2008)
BCC:	Autor	Indica un destinatario que no será revelado a los demás destinatarios (To:, CC:, BCC).	(Resnick, 2008)
Message-ID:	Originador	Identificador único del mensaje.	
Received:	Originador Relé Mediador Receptor	Este campo contiene información de seguimiento: servidor huésped origen, relés, mediadores, nombres de dominio de los servidores de MSA y direcciones IP.	(Resnick, 2008)
Return-Path:	MDA, de MailFrom	La MDA registra la dirección MailFrom en el campo Return-Path.	(Resnick, 2008)
Resent-*:	Mediador	Utilizado para los reenvíos.	
List-Id:	Mediador Autor	- Contiene las listas de correo con una nomenclatura estándar. - Usa el formato de los nombres de dominio, pero la cadena se construye combinando las dos partes de una dirección de correo electrónico.	(Chandhok & Wenger, 2001)
List-*:	Mediador Autor	Colección de campos de encabezado de mensajes utilizados por las listas de correo.	(Neufeld & Baer, 1998)

Fuente: Elaboración propia.

Tabla 8. *Identificadores en las diferentes partes del mensaje.*

Capa III. Protocolo SMTP			
Campo	Establecido por	Descripción del campo	Referencia
HELO/EHLO	Originador MSA MTA	Comando utilizado por un cliente SMTP (incluido Originador, MSA o MTA), para especificar la identidad de dominio de alojamiento.	(Klensin, 2008)
ENVID	Originador	Identificador del sobre.	(Moore, 2003)
MailFrom	Originador	- Dirección de correo electrónico para recibir información de control de devolución, como mensajes devueltos. - Debe indicar qué dirección debe ser informada sobre los problemas/éxitos a nivel de transferencia.	(Klensin, 2008)
RcptTo	Autor MTA final MDA	- Especifica la dirección de buzón MUA de un destinatario, no siempre es visible en el encabezado. - Puede especificar un miembro de la lista, mientras que To: por lo general especifica una lista de correo.	(Klensin, 2008)
ORCPT	Originador	- Parámetro opcional del comando RCPT. Indica la dirección original a la que corresponde la dirección RcptTo actual, después de la realización de un mapeo durante el tránsito. - Única forma confiable de correlacionar un DNS de una transferencia de mensajes de múltiples destinatarios con el destinatario previsto.	(Klensin, 2008)
Sourcer Address	El último cliente relé	- Establecido por el servidor de envío SMTP del cliente que precede inmediatamente al servidor SMTP del receptor actual. - Dirección IP de origen del servidor desde el que se envió el datagrama. Información establecida y proporcionada por la capa IP. - Frecuentemente es autorizado, aunque puede ser una dirección falsa.	(Postel, 1981a)

Fuente: Elaboración propia.

3.2. Servicios a nivel de usuario

Dentro de la capa MHS de la arquitectura del correo electrónico en Internet, se producen interacciones con protocolos muy diferentes a los que se utilizan dentro de la capa de transporte y sobre la capa de MHS existen otros protocolos que son los utilizados para las interacciones entre los usuarios, que son de los que se comentarán en esta sección.

Debido a que el correo es y fue creado para los usuarios, estos protocolos son muy importantes, pero dada la gran variabilidad de la comunicación y de las necesidades de los usuarios, las reglas no son estrictas y por eso hay muchos aspectos de estos protocolos que no están estandarizados, como, por ejemplo, el uso de listas de correo. Dentro de los actores que participan a este nivel están los MUA y MS.

a. Los Agentes de Usuario de Mensajes (MUA)

Un MUA realiza sus funciones para el actor “usuario” y sus aplicaciones, representa al usuario dentro de la arquitectura del correo electrónico. El autor MUA, representado en la como aMUA, es quien crea el mensaje y realiza el envío inicial, utilizando la infraestructura de transferencia a través de un MSA. Igualmente, puede almacenar los mensajes en el momento de su creación y envío, publicándolos en un almacén, que en la Figura 10 aparece con la etiqueta aMS. Un aMS puede gestionar los mensajes de diferentes formas. Un modelo muy utilizado es el uso de carpetas, que en IMAP, por ejemplo, se denominan buzones de correo (M. Crispin, 2003). Con el tiempo, este modelo de organización se ha popularizado y por esa razón se observan carpetas con diferentes denominaciones como “en cola”, “enviados” o “borradores”, pero en realidad puede ser cualquier tipo de organización y no son necesarias ninguna de esas carpetas.

En el otro lado de la comunicación está el receptor MUA, cuya función principal es procesar el correo recibido por el destinatario del correo. Entre las funciones básicas están la generación de mensajes de control para disposición del correo, la eliminación de mensajes y la generación de respuestas y reenvío de nuevos mensajes. Un MUA puede tener una configuración distribuida, como ocurre cuando se utiliza un dispositivo móvil para acceder al correo, en cuyo caso se utiliza una interface sencilla, pero la mayor parte de la funcionalidad del MUA es ejecutada remotamente (Maes & Melnikov, 2006).

Aunque un “mediador” es un “actor” que hace las veces de MUA, algunas veces pueden confundirse. Por ejemplo, un servicio de masivo programado no debe confundirse con un “mediador” de listas, ya que no existe mensaje que active el servicio programado. Los MUA se pueden automatizar cuando el usuario no está presente, como los correos automáticos de fuera de la oficina. De todos modos, estas respuestas se puede regular, como en el caso de los mensajes que responden a las listas de correo (Moore, 2004). Los campos de identificación relevantes para un MUA son: From:, Reply-To:, Sender:, To:, CC: y BCC: (Resnick, 2008).

b. Almacén de Mensajes (MS)

En la Figura 10, se ve el flujo del mensaje. El almacenamiento es gestionado por el MS y un MUA puede emplear un MS para almacenamiento permanente. Aunque en la imagen se ha representado un aMS y un rMS independientes, el MS puede ser remoto o local al MUA. Cuando un MS adquiere un mensaje de un MDA, lo puede hacer por medio de un proceso local, utilizando un proceso estandarizado como SMTP o mediante el uso de los protocolos POP (Myers & Rose, 1996) o IMAP (M. Crispin, 2003). Con este esquema, queda claro dónde se utilizan este tipo de protocolos.

3.3. Servicios a nivel de MHS

Los agentes MSA, MTA y MDA son los principales actores que dan servicio en la capa de MHS. En este apartado, se describirán sus funciones y como interactúan entre sí para cumplir el objetivo del MHS.

a. Agente de Envío de Mensajes (MSA)

Este agente acepta los mensajes enviados por el aMUA y vela por el cumplimiento de las políticas y estándares del alojamiento ADMD. Está dividido en dos funciones bien diferenciadas ya que, por una parte, es la interfaz con el aMUA y, por otra parte, es la interfaz con el MHS a través del hMSA. Cuando un “autor” envía un mensaje, la responsabilidad pasa al MHS, lo cual se denomina publicación. Por el lado del MHS, el hMSA tiene la responsabilidad de transmitir el mensaje bajo las reglas del MHS e Internet. Tiene la capacidad de rechazar un mensaje cuando no las cumple. Por lo tanto, prepara el mensaje, normalmente agregando campos de encabezado, como la fecha, el ID del mensaje y modifica algunas partes del mensaje, como, por ejemplo, expandiendo la dirección para que cumpla con la notación formal del FMI. Los identificadores relevantes para este agente son: HELO/.EHLO, MailFrom, RcptTO, Received (Klensin, 2008), ENVID (Moore, 2003), y SourceAddr (Postel, 1981a).

b. Agente de Transferencia de Mensajes (MTA)

Un MTA retransmite el correo como un conmutador de paquetes o un enrutador IP, con el objetivo de realizar evaluaciones de enrutamiento y acercar el mensaje a los destinatarios. Ya que un correo electrónico normalmente es mucho más grande que la carga útil de un paquete, es de esperar que haya más latencia de extremo a extremo. La retransmisión se realiza por medio de una secuencia de MTA hasta llegar a un MDA destino. Aunque un MTA tiene un comportamiento de MTA de cliente y servidor; no realiza ningún cambio del sobre o contenido, solamente agrega información de seguimiento (Lindberg, 1999). Los mensajes pueden ser modificados dentro de un ADMD destino, justo en los módulos de retransmisión que actúan como puertas de enlace en lugar de MTA.

Los mecanismos de transferencia punto a punto entre pares de MTA suele utilizar SMTP (Klensin, 2001, 2008; Postel, 1982b), aunque existen otros métodos, como Batch SMTP (Freed, Newman, Belissent, & Hoy, 1998) o la retransmisión de correo bajo demanda (ODMR) SMTP (Gellens, 1999). Como cualquier conmutador, la confiabilidad debería de estar asegurada en alguna medida, siendo algunos conmutadores más robustos que otros dependiendo del grado de confiabilidad que proporcionen. En este aspecto, se espera que un MTA proporcione algún

mecanismo de almacenamiento temporal en caso de ocurrir algún desastre como una caída del servicio, de forma que la recuperación esté asegurada. El protocolo SMTP proporciona códigos de respuesta en su especificación para casos de desastre, pero se han propuesto algunas extensiones a este modelo (Hansen & Klensin, 2008)

El mecanismo de enrutamiento más utilizado es el registro MX (*Mail Exchange*) del DNS (Mockapetris, 1987b), el cual especifica un MTA a través del cual se puede acceder al dominio que se consulta. El registro MX indica cómo se deben enrutar los mensajes de correo electrónico de acuerdo con SMTP. Por lo tanto, un registro MX siempre debe apuntar a otro dominio. Este mecanismo proporciona una red común que permite que cualquier MTA adjunto se conecte a cualquier otro MTA.

No todos los **MTA** son iguales, de tal forma que pueden desempeñar **roles** bien diferenciados:

- ✍ **MTA de límite (*Boundary MTA* o *Border MTA*):** Este MTA forma parte de un ADMD e interactúa con los MTA de otros ADMD. Puede haber diferentes MTA de límite, según la dirección que tome el flujo de correo.
- ✍ **MTA saliente (*Outbound MTA*):** Este MTA transmite mensajes a otros ADMD.
- ✍ **MTA entrante (*Inbound MTA*):** Este MTA recibe mensajes SMTP entrantes de retransmisores MTA en otros ADMD, es el caso de MTA configurado como el destino de un registro MX.
- ✍ **MTA final:** Este MTA transfiere el mensaje al MDA.

Los **identificadores** relevantes para un **MTA** son **HELO/.EHLO**, **MailFrom** y **RcptTo** (Klensin, 2008), **ENVID** (Moore, 2003), **Received**: asignado por el servidor relé (Resnick, 2008) y **SourceAddr** (Postel, 1981a).

c. Agente de Entrega de Mensajes (MDA)

Cuando el mensaje pasa de un MHS a un buzón destino, a esta transferencia se le denomina “entrega”. En la Figura 10, se puede observar como la entrega tiene lugar dentro de un MDA o agente de entrega de correo y se muestra como la transición de hMDA a rMDA. Es necesario que un MDA acceda a la información relacionada con las direcciones, ya que es quien debe saber dónde entregar el mensaje. Al igual que un MSA, un MDA cumple dos funciones, como se muestra en dicha imagen. La hMDA funciona como un motor de un servidor SMTP con una función adicional, la redirección de un mensaje a una dirección alternativa, según lo especificado dentro del mensaje. En la rMDA, se deben realizar las funciones de entrega que especifique el destinatario. La transferencia desde el MHS al MDA es realizada de forma normal

como entre MTA, pero la transferencia desde el MDA al MS utiliza un protocolo de acceso como es POP o IMAP.

Hay dos identidades relevantes para un MDA. La primera de ellas es Return-Path:, que es registrada por el MDA utilizando la dirección MailFrom. El otro campo es Received:, el cual puede ser registrado por un MDA para indicar información de seguimiento, incluidos el servidor de origen, los nombres de dominio del servidor receptor y las direcciones IP.

4. Consideraciones adicionales de la arquitectura

En el apartado anterior se han presentado todos los actores y servicios disponibles en una arquitectura estándar de correo electrónico. Hay algunos aspectos que se explicarán en este apartado para completar todo el engranaje del sistema, como los modos de transición, la implementación y los mediadores más comunes presentes en esta arquitectura.

4.1. Modos de transición

Desde que se genera un mensaje por medio de un “actor” hasta que se entrega a su destino, el modelo utilizado por el correo electrónico en Internet es un modelo de inserción (*push*), lo que significa que cada actor que tiene el mensaje inicia una transferencia al siguiente sitio destino utilizando SMTP (Klensin, 2008) o el Protocolo de transferencia de correo local (LMTP) (Myers, 1996), y un modelo de extracción (*pull*), en el que el actor que tiene el mensaje espera a que el actor del siguiente sitio destino inicie una solicitud de transferencia. Dentro de MHS, los mecanismos utilizados para la extracción son ETRN (Winter, 1996) y ODMR (Gellens, 1999).

ETRN es una extensión de SMTP mediante el cual un cliente (remitente-SMTP) puede solicitar que el servidor (receptor-SMTP) inicie el procesamiento de sus colas de correo para mensajes que están esperando en el servidor de la máquina cliente. Si hay algún mensaje en el servidor para el cliente, entonces el servidor debe crear una nueva sesión SMTP y enviar los mensajes en ese momento.

ODMR es una extensión del protocolo SMTP y fue diseñado para permitir el intercambio de correo entre dos máquinas con conexión no permanente y el servicio de correo SMTP proporcionado por un ISP. En este caso, los roles del cliente y servidor se invierten y la máquina cliente se convierte en la máquina servidora y, a su vez, la máquina servidora en el cliente, después que el cliente se haya autenticado. De esta manera, el servidor remoto envía los correos que el cliente está esperando a través de la conexión segura creada. ODMR es más seguro que ETRN debido a la autenticación requerida cuando se inicia el proceso. Al finalizar

el proceso de entrega, el MUA o MS del destinatario puede acceder a los mensajes utilizando POP e IMAP para extraer los mensajes.

4.2. Implementación y operación

Dentro del marco teórico de la arquitectura de correo electrónico, no debe confundirse la arquitectura con la implementación. La arquitectura como tal define todas las funciones conceptuales del servicio y la implementación puede combinar o separar cada uno de los componentes de la arquitectura según las necesidades de un entorno en particular. Mientras que algún componente como un MTA puede ser implementado por un sistema de *software* para realizar la retransmisión, puede incluir otras funcionalidades, como, por ejemplo, la de un MDA, o puede interactuar con sistemas heterogéneos y hacer las funciones de puerta de enlace.

Otro ejemplo es un MS distribuido, que podría estar dividido en una parte local y otra remota. Desde el punto de vista de un usuario, ese MS podría estar en diferentes estados (Crocker, 2009):

- * **En línea:** El MS es remoto y los mensajes se acceden cuando el MUA y el MS están conectados.
- * **Fuera de línea:** El MS es local y los mensajes se mueven completamente desde cualquier localización remota, en lugar de permanecer en ese sitio.
- * **Desconectado:** Un MS remoto y un MS local se mantienen sincronizados mientras están conectados, en caso contrario, el correo puede llegar al MS remoto y el usuario puede realizar cambios en el MS local.

Los dos almacenes se vuelven a sincronizar cuando se vuelven a conectar.

4.3. Mediadores

Aunque se explican las funciones de un “mediador” en secciones anteriores, en este apartado se desarrollarán un poco más las tareas del “mediador” para diferenciarlo de los MTA con los que suele confundirse y que fueron explicados en secciones separadas. En primer lugar, la transferencia de un mensaje hace uso de una infraestructura asincrónica de envío y recepción de una cadena de MTA, y la secuencia de publicación y entregas se realiza a través de “mediadores”. Aunque el MTA y el “mediador” comparte funciones de retransmisión, el “mediador” tiene mayor flexibilidad con el contenido que el MTA. Además, un “mediador” reenvía un mensaje a través de un proceso de reenvío, no de retransmisión.

Un “mediador” puede registrar la información que recibe y modificar campos de seguimiento hasta la entrega final, pero un aspecto de un “mediador” que lo distingue de cualquier otro MUA que crea mensajes, es la preservación del mensaje original, aunque el “mediador” podría agregar comentarios. Un “mediador” no podrá crear un nuevo mensaje que reenvía un mensaje existente, ya que se considera que el nuevo mensaje proviene del actor que realiza el reenvío, no del autor original. Otro mensaje que no se puede crear es un mensaje de “responder”, debido a que como en el caso anterior, ya no es un reenvío del original, como, por ejemplo, los avisos de vacaciones o fuera de la oficina. Un último tipo de mensaje que un “mediador” no puede crear es un mensaje de anotación, puesto que el nuevo autor está proporcionando comentarios y ya no es el autor original el que participa en dicho mensaje.

a. Alias

Un alias es una forma de redireccionamiento que proporciona una o más direcciones de envío para que un MDA pueda localizar los buzones correctos para realizar la entrega. De esta forma, el mensaje continúa a través del servicio de transferencia para su entrega a una o más direcciones alternativas. El mensaje se vuelve a enviar, conservando toda la información excepto la dirección del destinatario del sobre RcptTo, pero la dirección de devolución MailFrom no se modifica. Aunque es un mecanismo muy parecido al que ejecuta un MTA, la única diferencia es que un MTA no cambia el valor del RcptTo.

Uno de los problemas que presenta este mecanismo es que si ocurre algún inconveniente con la entrega, el informe del problema llega al usuario y no al administrador del alias, lo que hace complicado resolver el problema, ya que el usuario desconoce la configuración del alias (Klensin, 2008).

b. ReSender

Este tipo de “mediador”, también llamado ReDirector realiza funciones muy diferentes a las de reenvío porque el “mediador” une la dirección del “autor” original del mensaje con el “destinatario” del nuevo mensaje. Esta conexión deja claro el rol del “mediador”, ya que registra toda la información del “destinatario” que ha servido como “mediador” sin que el nuevo “destinatario” tenga dudas sobre quien es el “autor” original, incluso si el “mediador” agrega algún comentario (Resnick, 2008).

c. Listas de correo

Este tipo de “mediador” recibe los mensajes como un “destinatario” y vuelve a publicar los mensajes a todos los miembros suscritos a la lista. Esta tarea es muy parecida a la realizada por el ReSender, pero además de poder enviar a muchos destinatarios, puede modificar el contenido, por ejemplo, eliminando archivos adjuntos, cambiando el formato y añadiendo comentarios específicos de la lista. Las listas de correo también archivan los mensajes de los autores conservando las características del “autor” original (Chandhok & Wenger, 2001; Neufeld & Baer, 1998).

d. Puertas de enlace (*Gateways*)

Este “mediador” realiza un trabajo de enrutamiento y transferencia de mensajes a un entorno de mensajería que opera bajo diferentes estándares o políticas potencialmente incompatibles con el sistema utilizado por el “autor” original. Para poder llevar a cabo esa tarea, debe poder modificar el contenido, la estructura, la dirección o los atributos que hagan falta para enviar el mensaje.

Su funcionalidad puede verse como un MTA cuando los sistemas que conecta son compatibles a nivel de estándares técnicos. La distinción fundamental entre un MTA y una puerta de enlace es que una puerta de enlace puede realizar cambios importantes en un mensaje para compatibilizar los diferentes estándares. El objetivo de una puerta de enlace es minimizar la pérdida de semántica cuando se realiza ese mapeo entre los diferentes estándares.

Las puertas de enlace estandarizadas para el correo son el fax (Toyoda & Crocker, 2005), el correo de voz (Vaudreuil & Parsons, 2004) y el Servicio de mensajería multimedia (MMS) (Gellens, 2006).

e. Filtros de límites

Los filtros configurados en los límites de entrega de correo actúan como “mediadores” y suelen configurarse para verificar su conformidad con sus políticas de seguridad. El ejemplo más común son los filtros de *spam* o virus que proliferan en casi todas las implementaciones de correo existentes. Como “mediador”, puede alterar el contenido de un mensaje para convertirlo en un mensaje seguro, por ejemplo, eliminando el contenido inaceptable.

III. ESTADO DEL ARTE EN EL CONTEXTO DE LA SEGURIDAD DEL CORREO ELECTRÓNICO

Una vez presentada la arquitectura del correo actual, es necesario analizar los aspectos de seguridad presentes en la misma. En esta sección, se presentan primero las consideraciones de seguridad inherentes al correo electrónico. A continuación, se describen los problemas de seguridad y las principales soluciones en forma de trabajos científicos o tecnologías de software ya implementadas. Al final del apartado, se realiza una comparativa de las principales soluciones tecnológicas presentadas y se describen los requerimientos que debería de tener una solución de correo electrónico integral.

1. Consideraciones de seguridad de la arquitectura de correo electrónico

Existen muchas especificaciones técnicas que discuten las consideraciones de seguridad de la arquitectura de correo electrónico. Estas especificaciones cubren temas básicos de seguridad, como la autenticación y la privacidad (Callas, Donnerhacke, Finney, Shaw, & Thayer, 2007; Hoffman, 2002; Ramsdell, 2004; Siemborski & Melnikov, 2007). Hay que destacar que el núcleo de la arquitectura del correo electrónico no propone requisitos de seguridad de forma explícita. Por ejemplo, no se requiere la autenticación de los participantes ni se intenta evitar que la información contenida en los mensajes sea divulgada.

Existen amenazas explícitas que han sido aprovechadas por los atacantes, como las que exponen los campos que aparecen en los mensajes. Este es el caso de la copia oculta, que facilita la divulgación de información (Klensin, 2008; Resnick, 2008). Otras consideraciones de seguridad analizadas por algunos autores están relacionadas con el contenido (Freed & Borenstein, 1996b, 1996c; Freed & Klensin, 2005a; Resnick, 2008). Los actores con respuestas automáticas pueden estar expuestos a muchos ataques de denegación de servicio (DoS), transmisión de contenido malicioso y otros problemas de seguridad (Moore, 2004).

Las puertas de enlace afectan los servicios de seguridad de un extremo a otro, ya que un servicio de seguridad que no se aplique desde el punto inicial al final, puede tener regiones expuestas a diferentes ataques (Freed, 1999). Los filtros de límites pueden presentar problemas de seguridad, ya que un script de SIEVE podría ser modificado por un atacante exponiendo información confidencial u ocasionando que el correo sea descartado o enviado a un destinatario no autorizado (Guenther & Showalter, 2008).

El correo basado en el protocolo SMTP es por defecto inseguro, ya cualquier usuario puede interactuar directamente con un servidor SMTP de recepción y retransmisión y crear mensajes que engañarían a un destinatario ingenuo. Para un experto es más fácil detectarlo y entre más

conocimiento tenga el usuario sobre los problemas de seguridad, menos probable es que se vea afectado por este tipo de mensajes.

El correo SMTP en su especificación básica no puede autenticarse o proporcionar integridad a nivel de transporte. Por lo tanto, la seguridad del correo está basada en métodos de extremo a extremo en la que participa el cuerpo del mensaje, como las firmas digitales (Galvin, Murphy, Crocker, & Freed, 1995), tales como Pretty Good Privacy (PGP) (Callas et al., 2007) o las extensiones de correo de Internet multipropósito seguro (S/MIME) (Ramsdell, 2004).

La autenticación a nivel de transporte, como, por ejemplo, la realizada entre los servidores SMTP, puede mejorar la situación, pero el problema es que esa autenticación se realiza entre dos servidores y no entre toda la cadena de retransmisiones a las que puede estar expuesta el correo. Tampoco autentican los usuarios o las máquinas (Gellens & Klensin, 2006; Siemborski & Melnikov, 2007). Otras consideraciones de seguridad, como la falsificación del campo From:, las direcciones que no aparecen en los campos del mensaje pero si en los comandos SMTP, la divulgación de direcciones con el objetivo de enviar correos masivos, deben ser tratadas utilizando extensiones o propuestas adicionales a la especificación de los protocolos básicos utilizados en el correo electrónico (Gellens & Klensin, 2006; Klensin, 2008; Lindberg, 1999).

2. Estudio de la seguridad en el contexto del correo electrónico

Son muchas las publicaciones relacionadas con temas o aspectos de seguridad del correo electrónico. Después de realizar la búsqueda en las bases de datos bibliográfica, se identificaron más de 7.000 publicaciones disponibles. Fue necesario realizar un cribado para eliminar los trabajos considerados como no relevantes y dejar las propuestas cuyo enfoque estaba relacionado de forma directa con la temática del presente trabajo. Posteriormente, se clasificaron las publicaciones seleccionadas en cuatro apartados, que se corresponden con aspectos que más preocupan a nivel de seguridad y que serán expuestos en esta sección:

- ⊕ Usabilidad.
- ⊕ *Phishing*.
- ⊕ Correo no deseado.
- ⊕ Soluciones de seguridad.

2.1. Usabilidad

Payne y Edwards (2008) hacen un estudio de los sistemas de seguridad desde el punto de vista de la usabilidad, aunque se enfocan en dos problemas: la autenticación y el cifrado. Estos autores presentan cómo los métodos de autenticación han ido cambiando, desde la simple contraseña de ocho caracteres hasta las complicadas contraseñas generadas por selecciones de píxeles en las imágenes denominadas Passpoints.

En ese estudio ilustran los problemas de usabilidad que han tenido las soluciones de cifrado para el correo electrónico, como, por ejemplo, los problemas de usabilidad organizacional con las Autoridades de Certificación (CA) del Correo con Privacidad Mejorada (PEM) en 1985; los problemas de escalabilidad de PGP, que hacen que solo pueda ser utilizado para pequeñas comunidades; o los problemas de uso práctico de S/MIME, que es una solución intermedia entre PEM y PGP. Aunque existen soluciones de cifrado de correo disponibles y hay conciencia de la seguridad, muchos usuarios no las utilizan por la dificultad de uso.

Siguiendo la misma línea, Stephen Farrell (2009) hizo una revisión de las características de seguridad presentes en los MUA como Thunderbird y Outlook, con el objetivo de obtener las características de seguridad que necesitan los usuarios para el tratamiento del correo electrónico. En primer lugar, plantea las razones por las cuales los usuarios no utilizan la seguridad cuando gestionan su correo electrónico y coinciden con los otros trabajos expuestos anteriormente, la facilidad de uso es el principal problema y el otro, el desconocimiento. La forma como el usuario percibe la seguridad y la forma como los expertos le proponen la seguridad tienen grandes diferencias.

Según este autor, y en este trabajo se coincide con él, son muchos los años que se han invertido intentando acercar las técnicas más sofisticadas al usuario, pero se ha errado en la forma de acercarlo. Para un usuario es complicado tener que decidir si un buzón destino tiene una clave pública y para la mayoría de los usuarios el uso de certificados es dificultoso. Como conclusión del artículo, el uso de las técnicas de seguridad no es nada intuitivo.

Por otra parte, Moecke y Volkamer (2013) evalúan los criterios de interoperabilidad y seguridad deseables en cualquier sistema de correo electrónico. Concluyen que ni las técnicas de seguridad complementarias ni los proveedores de correo disponibles aseguran todos los requisitos. Asimismo, comentan que las soluciones basadas en complementos de seguridad para los servicios de correo electrónico estándar son decisiones de confianza para el usuario y lleva tiempo configurarlos. Añaden que se requiere confianza en las técnicas de seguridad y

estas no son aptas para un usuario medio. Es muy importante tener en cuenta la usabilidad de la solución para que sea adoptada por la mayoría de los usuarios.

2.2. *Phishing*

Uno de los ataques más peligrosos para los usuarios de Internet es el *phishing*. El *phishing* es una estafa de Internet utilizada por un atacante para engañar a sus usuarios con el objetivo de robar información. Este tipo de ataque puede ser perpetuado de muchas formas, siendo el correo electrónico el medio más utilizado para ello. Se han desarrollado numerosas soluciones para intentar prevenir esta actividad, pero a pesar de la variedad de sistemas *anti-phishing* existentes, su efectividad y su usabilidad se han visto comprometidas, como se ha visto en el apartado anterior. El problema con este tipo de amenaza es que la víctima proporciona información sensible al atacante por medio de engaños perpetrados por técnicas de ingeniería social o falsificación.

El *phishing* no es un ataque llevado a cabo exclusivamente a través del correo electrónico, como lo muestra la Figura 11. En esta imagen, se puede ver como el *phishing* por medio de correo electrónico está clasificado como una técnica de ingeniería social.

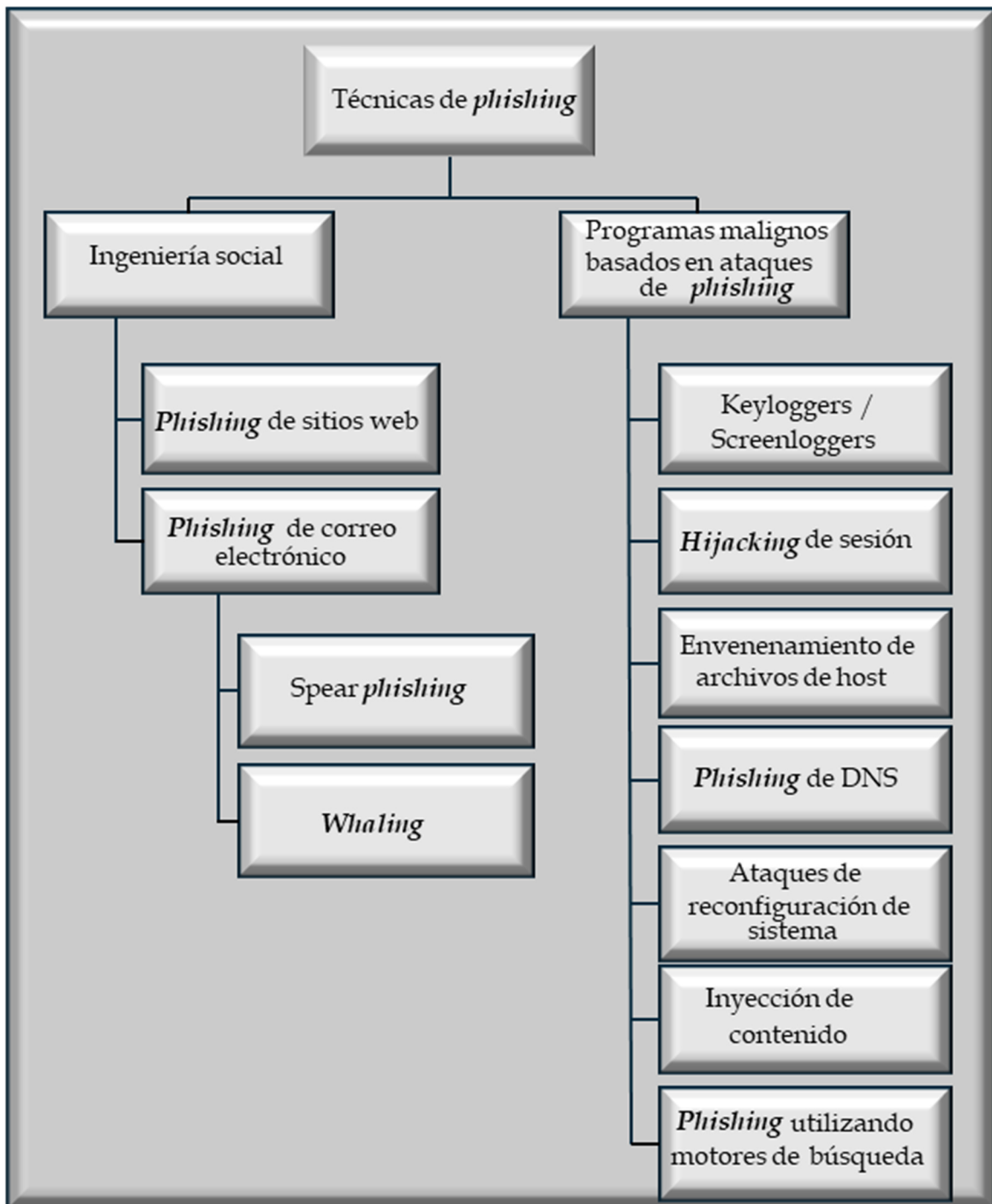
Existen publicaciones muy completas analizando todas esas técnicas, como la de Gupta, Tewari, Jain, & Agrawal (2017), y análisis profundos sobre el phishing en el contexto del correo electrónico, como el manuscrito de Chanti & Chithralekha (2020).

Ren y Susilo (2007) realizaron una propuesta *anti-phishing* por medio de un sistema denominado SEFAP (*Signed Email for Anti-Phishing*, correo electrónico firmado para *anti-phishing*). SEFAP es un sistema basado en varios esquemas de firma, que deben cumplir dos propiedades: basarse en identidad y tener capacidad de repudio. Que sea basado en identidad conlleva a eliminar la infraestructura de clave pública (PKI) y que posea la propiedad de repudio sirve para proteger la privacidad del emisor. SEFAP tiene la gran ventaja de poder ser integrada a cualquier servidor SMTP estándar. Su principal problema es que tanto emisor como receptor tienen que utilizar el servidor y cliente SEFAP, lo que añade una capa de complejidad y hace que no pueda utilizarlo cualquier usuario.

Con una orientación distinta y más enfocada al usuario, Kumaraguru et al. (2007) proponen formar al usuario mediante correos electrónicos de entrenamiento en los que se les instruye sobre los peligros potenciales de este tipo de ataque, enviándolos a las páginas falsas y enseñándoles a observar cómo la página que aparece en el enlace del correo no es la misma que han visitado. Aunque la propuesta podría ser eficaz, un descuido de un usuario disminuiría

la efectividad de todo el entrenamiento, así que lo mejor sería combinarla con alguna solución automatizada.

Figura 11. Técnicas utilizadas para llevar a cabo un ataque de phishing.



Fuente: Modificado de Gupta et al. (2017).

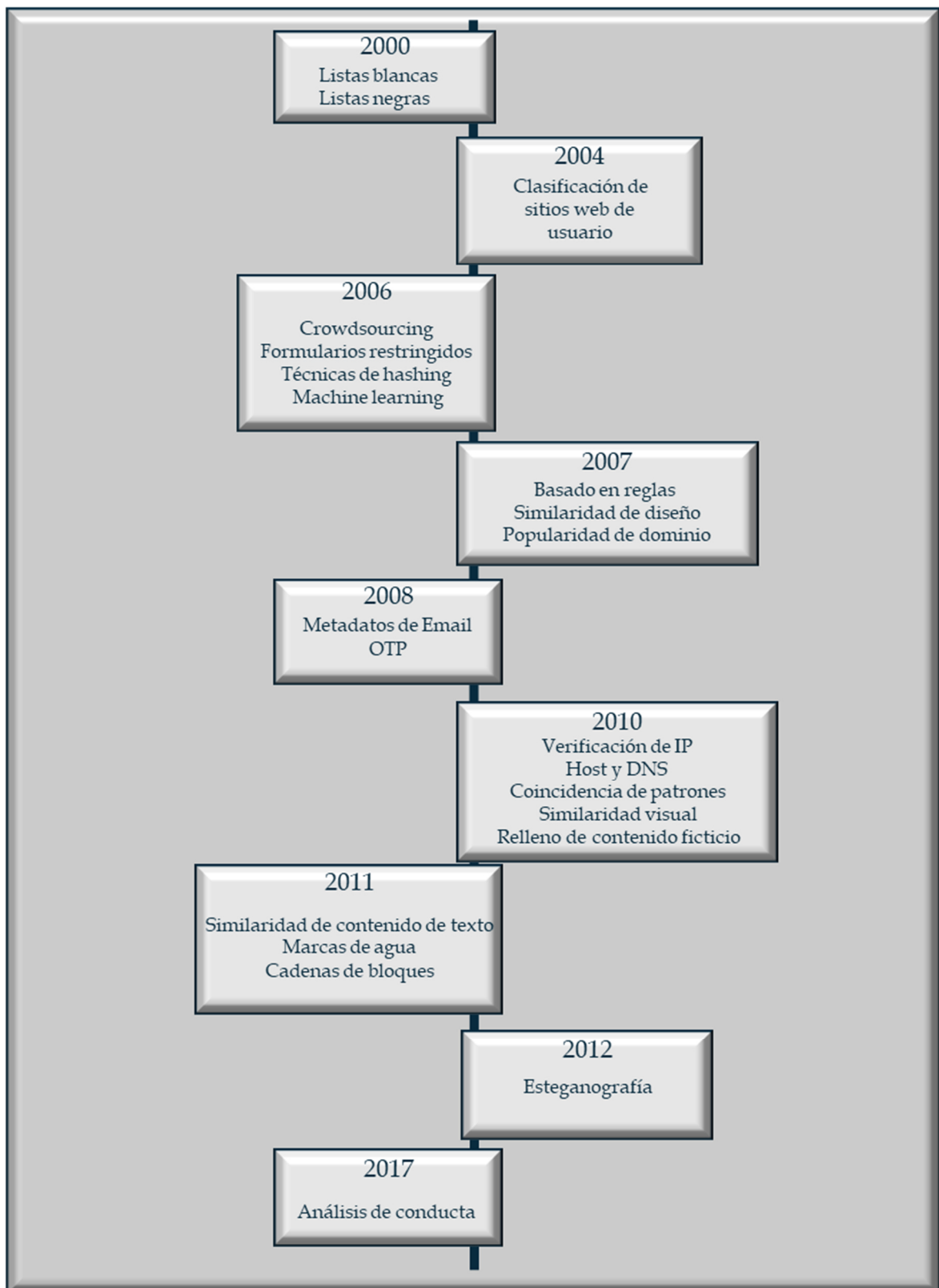
Con el fin de alejar al usuario de la complejidad de los certificados, algunos autores de la universidad de Miami idearon una solución denominada “Trusted Email” (Crain, Opyrchal, & Prakash, 2010), que utiliza un *proxy* instalado en la máquina del usuario y otro en el servidor *web* de la empresa, banco o institución con la que se desea comunicar el usuario. Los proxys son los encargados de la comunicación segura. Todo el proceso de verificación se realiza a través de un *plugin* que se instala en el navegador del usuario. Esta propuesta es una solución basada en la autenticación del usuario.

Los atacantes cambian la forma de realizar el ataque *phishing* constantemente. Esto hace que las técnicas de detección se vuelvan obsoletas en poco tiempo. Hay que tener en cuenta que el problema se debe a que SMTP es un protocolo sin autenticación y las mejores técnicas anti-*phishing* son las que dotan de autenticación al protocolo. Dentro de las técnicas de detección, en las que ya se ha producido el ataque, pero no ha llegado al usuario, existen dos enfoques: basadas en comportamiento y basadas en contenido. Hamid y Abawajy (2011) realizaron un estudio de diferentes técnicas que utilizaban alguno de esos enfoques y propusieron una solución híbrida.

Existen muchas soluciones y propuestas para atajar el problema del *phishing* en la actualidad. Chanti y Chithralekha (2020) presentan una clasificación de diferentes métodos anti-*phishing*, incluyendo las propuestas más novedosas que utilizan DNS basadas en *blockchain*, como Namecoin, Block- stack, Nebulis o Bitforest, entre otras. Estos autores desarrollaron una taxonomía que ayuda a comprender los diferentes enfoques planteados para la detección de *phishing*. Concluyen que ningún método es 100% confiable y seguro.

En la Figura 12 se puede ver cómo han evolucionado los métodos *anti-phishing* desde el año 2000. Los detalles de cada uno de esos métodos y sus aplicaciones están descritos en Gupta et al. (2017).

Figura 12. Evolución de los métodos anti-phishing desde el año 2000.



Fuente: Modificado de Gupta et al. (2017).

2.3. Correo no deseado

El correo no deseado (*spam*) es el principal problema hoy en día dentro de las amenazas al correo electrónico, no solo por el trastorno que genera, sino porque es la puerta que permite la entrada a mucho contenido potencialmente peligroso. El contenido no deseado es muy variado y los atacantes utilizan toda una serie de artimañas para hacer llegar al usuario publicidad, peligrosos virus informáticos, programas maliciosos y perpetuar ataques como el *phishing*, visto anteriormente.

En el año 2002, Amin reveló la existencia de los correos maliciosos dirigidos (TME), como una forma de ataque muy diferente al correo no deseado, el *phishing* y otros correos electrónicos no legítimos. Este autor hace un amplio estudio sobre las técnicas de detención de TME y los procedimientos de análisis y filtrado de emails.

El RFC5068 (Hutzler et al., 2007) realiza una serie de recomendaciones para el envío de un correo electrónico, incluyendo como mejores prácticas el uso de autenticación, autorización y trazabilidad temporal. Por otra parte, el RFC5039 (Rosenberg & Jennings, 2008) analiza el uso del protocolo de inicio de sesión (SIP), que también es susceptible a la amenaza de correo no deseado. El espacio de soluciones propuesto incluye: Filtrado de contenido, listas negras, listas blancas, comunicación basada en contenido, sistemas de reputación, ofuscación de la dirección, direcciones de uso limitado, pruebas de Turing, rompecabezas computacionales, acciones legales, entre otros. Todas estas soluciones se pueden aplicar al correo electrónico.

La mayoría de las técnicas utilizadas para defenderse del correo no deseado se basan en estudios de comportamiento o de contenido del correo. Se han realizado estudios empíricos para caracterizar cómo se comporta el correo no deseado, para obtener las cuentas de correo vulnerables y para conocer cuál es el contenido de estos. Para lograr ese objetivo, algunos autores han recogido y analizado millones de correos no deseados (Dhinakaran, Lee, Nagamalai, & Chae, 2007). En la misma línea de estudio, Jindal y Liu (2008) analizaron los correos no deseados de opinión y propusieron una técnica para detectar este tipo de ataque.

Lo más preocupante con el correo no deseado es que los atacantes hacen uso de él para perpetuar otros tipos de ataques como: *Ransomware*, *Spear phishing*, *Spoofing*, *Man-In-the-Middle Attacks*, *Whaling/Business Email Compromise*, *Key Loggers*, *Zero-Day Exploits*, *Social Engineering*. Karim et al. (2019) hicieron un estudio muy exhaustivo sobre las técnicas utilizadas para la detección del correo no deseado. Describieron tanto las técnicas basadas y como las no basadas en inteligencias artificial y aprendizaje de máquinas (*machine learning*) con el objetivo de contrarrestar todo este tipo de ataques. Dividieron el correo electrónico en

cuatro partes (Tabla 9) y analizaron los ataques frente las soluciones basándose en dicha partición.

Tabla 9. Explicación de las partes de un correo electrónico.

Parte	Identificado como	Datos	Descripción
A	Cabecera TCP/IP	IP origen IP destino TCP origen TCP destino	 El mensaje está contenido dentro de un paquete TCP/IP.  Las IP de origen y destino son lo más importante.
B	Sobre SMTP	HELO\EHLO server.ejemplo.com MAIL FROM: <ayuda@idk.com> RCPT TO: <usuario@gmail.com> DATA	 Identifica el remitente (MAIL FROM) y el destinatario (RCPT TO).  Son direcciones de correo electrónico reales, pero como son opcionales, son falsificables.  EHLO y DATA son comandos SMTP.
C	Cabecera SMTP	From: <ayuda@idk.com> To: <usuario@gmail.com> Subject: Su cuenta expirará pronto.	 From: y To: son solo visuales, pueden ser falsificados para ocultar el origen real.  Por defecto, toman los valores del sobre.
D	Cuerpo	Estimado usuario Puede dirigirse al sitio http://scammer para recibir la ayuda que necesita.	 Sección de la cabecera SMTP. Incluye el contenido del correo electrónico y archivos adjuntos.

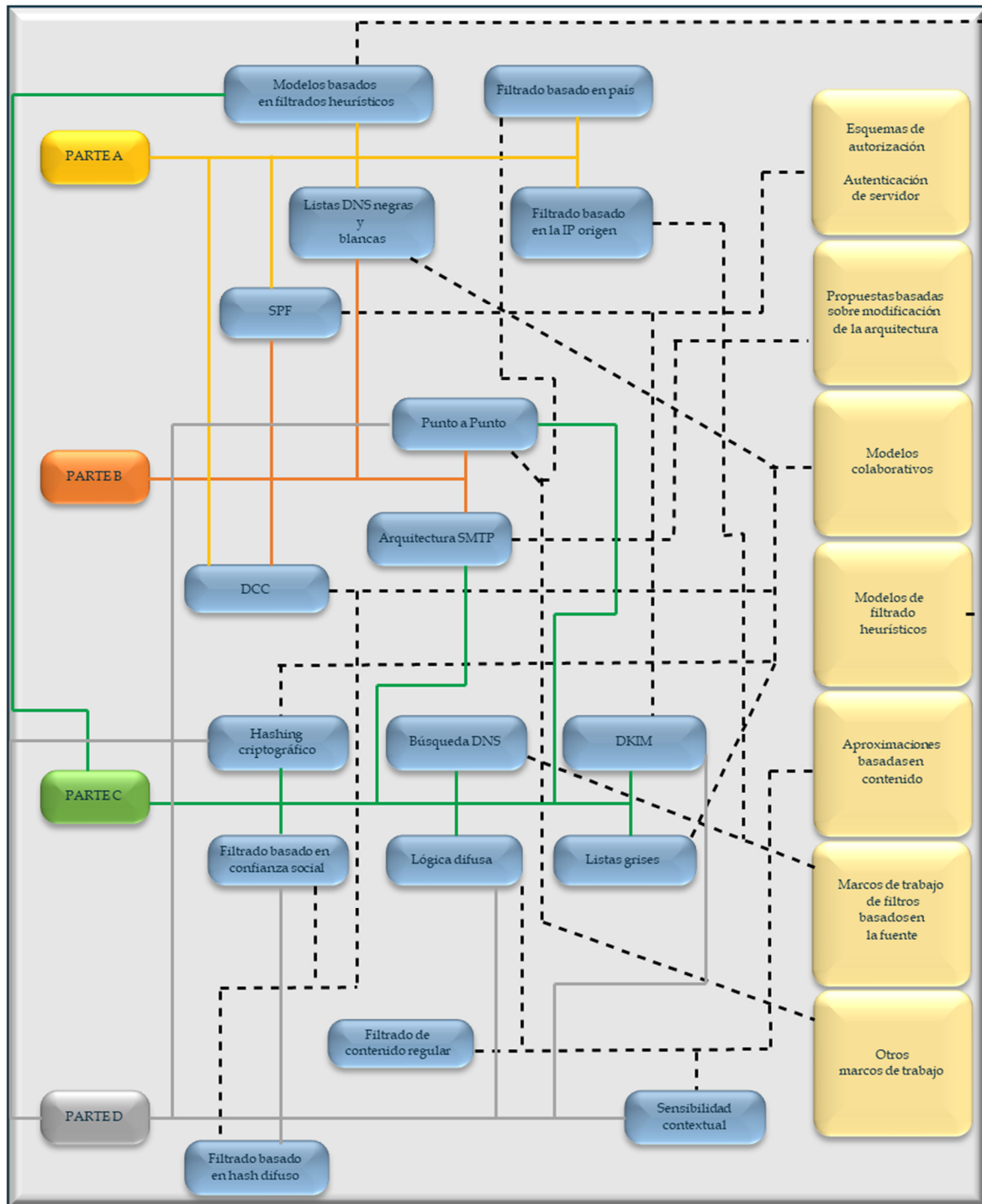
Fuente: Modificado de Karim et al. (2019).

Este enfoque permite clasificar muy bien las técnicas utilizadas frente a las vulnerabilidades y permite entender de una forma muy concreta lo que persigue cada técnica. Las técnicas utilizadas para contrarrestar el correo no deseado pueden tener como objetivo una o más partes, ya sea proporcionando filtros o realizando verificaciones muy específicas, donde ya han sido identificadas algunas vulnerabilidades o donde los atacantes hacen una serie de manipulaciones con el objetivo de no ser descubiertos.

En la Figura 13, se puede apreciar la relación entre las diferentes partes de un correo electrónico y el sistema de detección de correo no deseado utilizando técnicas no basadas en inteligencia artificial. Se observan en la parte central (cuadros azules) las diferentes técnicas de detección utilizadas; en la parte derecha, cómo estas técnicas pueden ser agrupadas (los

cuadros amarillos); y en la parte izquierda, la parte del correo donde la técnica es aplicada. Por ejemplo, sobre la parte C se puede aplicar la técnica de lógica difusa, que es una técnica basada en contenido. Algunas de estas técnicas serán brevemente descritas en el próximo apartado.

Figura 13. Interconexión entre las partes de un correo electrónico y las diferentes técnicas de detección de correo no deseado.



Fuente: Modificado de Karim et al. (2019).

El problema de estos algoritmos de detección convencionales es que con el tiempo los atacantes encuentran formas ingeniosas de evadirlos. Un campo emergente de estudio son las técnicas de detección y clasificación de correo no deseado utilizando inteligencia artificial. Aunque estas técnicas han sido utilizadas también de forma exitosa con el *phishing*, el objetivo final es detectar si el correo electrónico sigue o no unas reglas o políticas que puedan considerarse válidas para que el destinatario pueda acceder o no a su contenido. Entre las técnicas utilizadas están las inspiradas en organismos vivos (Eric Conrad, 2021; Ruano-Ordás, Fdez-Riverola, & R. Méndez, 2018), como los algoritmos de selección negativa (NSA) (Idris, Selamat, & Omatu, 2014; Idris et al., 2015), la optimización de enjambres de partículas (PSO) o la evolución diferencial (DE) (Brownlee, 2011). Todas estas técnicas intentan encontrar patrones, ya sea generando expresiones regulares con algoritmos genéticos para encontrar patrones de contenido o tratando de encontrar patrones de comportamiento de los atacantes.

Otras técnicas basadas en aprendizaje de máquinas han sido aplicadas persiguiendo el mismo objetivo de detectar correo no deseado. Las propuestas se dividen en dos grandes grupos: Algoritmos de aprendizaje automático supervisado y no supervisado. Los algoritmos de aprendizaje automático supervisado aprenden de un conjunto de datos ya etiquetados, donde la posible salida ya está definida. Aplicándolo al correo electrónico, el algoritmo repasa una gran cantidad de correo (aprendizaje) y construye un mapeo probabilístico entre la naturaleza de la entrada y la salida más probable (el resultado). El resultado podría ser categórico, en cuyo caso se denomina de clasificación, o numérico, en cuyo caso se denomina de regresión (Sathya & Abraham, 2013).

Por otra parte, en los algoritmos de aprendizaje automático no supervisado, no hay ningún dato etiquetado con el que trabajar, por lo tanto, no hay capacitación. El algoritmo intenta descubrir características comunes dentro de un grupo de correos proporcionados y reorganiza los datos en grupos encontrando los elementos comunes (Qian, Pathak, Hu, Mao, & Xie, 2010).

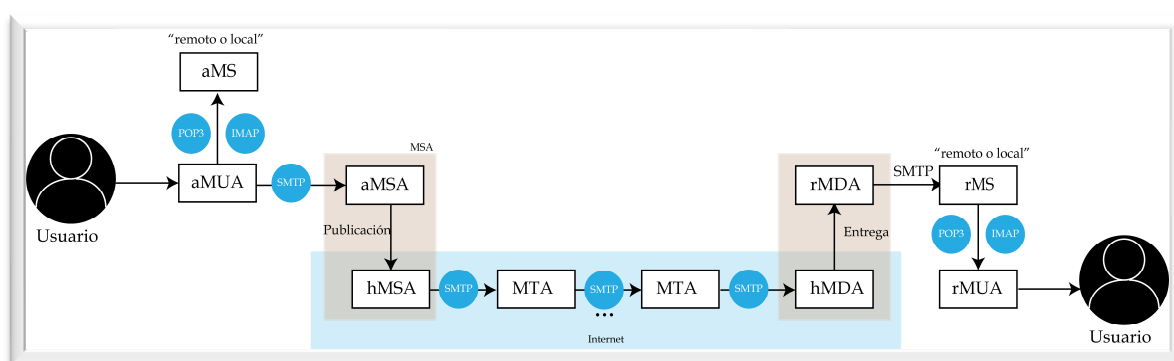
El campo de la aplicación de técnicas de inteligencia artificial a la detección de *spam* es muy amplio y no es el objetivo de este trabajo entrar en detalle en cada una de estas técnicas. Karim et al. (2019) exponen todas estas técnicas y realizan una comparativa muy detallada de la efectividad y aplicación de cada una de ellas.

2.4. Soluciones de seguridad

Cuando se estudia la seguridad en el correo electrónico se persiguen los mismos resultados que los deseados con la seguridad de la información en general: la confidencialidad, la disponibilidad e integridad de datos, en este caso del correo. La seguridad puede ser abordada con enfoques diferentes, como, por ejemplo, un enfoque desde el punto de vista de la seguridad ciudadana (Bellamy & McDonald, 2002).

Pero independientemente del enfoque, no hay que olvidar el objetivo cuando se aplica una solución técnica determinada: ¿el objetivo es mantener la confidencialidad, disponibilidad o la integridad? o ¿sigue un enfoque integral abordando todos o parte de esos objetivos? De igual manera, la solución debe ser usable, como se ha visto en apartados anteriores. La usabilidad es clave para la adopción de una solución de seguridad y para su difusión dentro de la comunidad de usuarios afectados. Con el objetivo de ejemplificar mejor las soluciones tecnológicas implementadas en torno a la arquitectura de correo electrónico, se utilizará como base la Figura 14, en la que se muestra los protocolos y servicios utilizados en la arquitectura de correo electrónico. Esta gráfica está basada en la Figura 10.

Figura 14. Modelo conceptual de la arquitectura de correo electrónico.



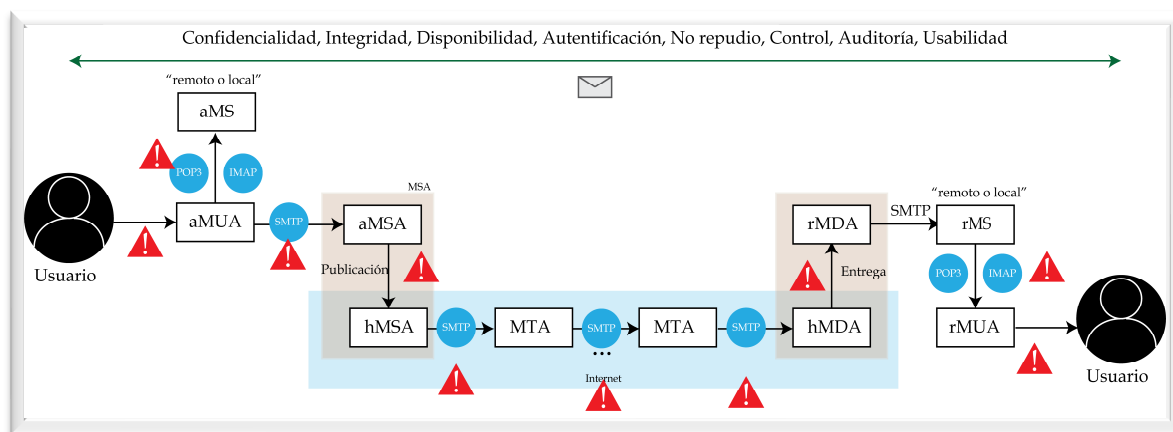
Fuente: Elaboración propia.

Dentro de la arquitectura ilustrada en la Figura 14, son muchos los “enlaces” que deben ser asegurados para lograr preservar con seguridad toda la comunicación desde el remitente hasta el destinatario cuando se envía un mensaje.

La Figura 15 muestra los puntos vulnerables (triángulos rojos) de la arquitectura de seguridad actual expuesta en este trabajo, donde los investigadores y empresas del sector de la seguridad de la información enfocan sus propuestas. La imagen también muestra los aspectos que se deben evaluar en las soluciones propuestas (confidencialidad, integridad, disponibilidad, autenticación, no repudio, control, auditoría, usabilidad), que serán descritos al final de este

apartado cuando se desarrolle la comparativa de las soluciones tecnológicas más extendidas. En este apartado se expondrán los aspectos que se tienen en cuenta en dichas propuestas, los aportes más notables de los grupos de investigación y las soluciones más relevantes o extendidas hoy en día.

Figura 15. Aspectos de seguridad de la arquitectura de correo electrónico y puntos clave de interés de las propuestas actuales de seguridad.



Fuente: Elaboración propia.

Los puntos vulnerables mostrados en esta figura son los que facilitan los ataques y aspectos discutidos en la sección “Estudio de la seguridad en el contexto del correo electrónico”. Cada uno de esos puntos vulnerables debe de ser reforzado para conseguir los objetivos de seguridad en el correo electrónico descritos en el mismo gráfico. Se puede ver que el usuario en sí es un punto vulnerable. Las técnicas de ingeniería social son las que directamente explotan esa vulnerabilidad. El uso de una técnica de defensa apropiada debería reforzar este punto y minimizar o eliminar por completo cualquier ataque.

a. Identificación y Autenticación

La identificación, cualidad que identifica de forma exclusiva un usuario dentro de un sistema, suele realizarse con la utilización de la dirección de correo electrónico para solicitar el acceso al servicio deseado. La identificación y la autenticación, cualidad que permite demostrar que el usuario es quien dice ser, son un tema de estudio dentro del campo de la seguridad informática y más aún, dentro del contexto de los servicios de correo electrónico existentes.

Un ejemplo de propuestas en este campo es la adopción de identificación y autenticación basada en correo electrónico (EBIA) en lugar de PKI (Garfinkle, 2003). Este autor argumenta que EBIA es una aproximación razonable para muchas aplicaciones comerciales y

gubernamentales debido a que proporciona una mejor usabilidad, privacidad, autonomía, resiliencia y requisitos de negocios del mundo real que la tecnología PKI.

Aunque **PKI** tiene diversas fortalezas en comparación con otras soluciones, tales como proporcionar no repudio, reemplazar los identificadores tradicionales como el SSN o el uso de dispositivos o tarjetas inteligentes para almacenar las claves, PKI presenta problemas que han evitado su expansión y adopción. Algunos de ellos son:

- 📁 **Usabilidad:** Los clientes PKI son muy difíciles de utilizar.
- 📁 **Coste:** Los usuarios tienen que comprar certificados en muchos casos.
- 📁 **Delegación:** Es muy difícil la delegación de autoridad para que, por ejemplo, un personal directivo delegue al personal de secretaría la gestión de su correo.

EBIA: Es más simple de utilizar. Usa una dirección de email como mecanismo de identificación y la recepción de un email como mecanismo de autenticación. Los autores abogan por el uso de mecanismos de seguridad adicionales para proteger las vulnerabilidades presentes al enviar un correo de confirmación sobre una red insegura. EBIA es de rápida difusión, ya que una dirección de email es más fácil de utilizar. La gran ventaja de EBIA sobre PKI se basa en el hecho que PKI requiere *software* especializado y una relación de confianza mutua con una CA, mientras que EBIA funciona con cualquier cliente de correo.

A nivel de requisitos de negocio, EBIA supera a PKI en:

- 🚫 PKI tiene que establecer un identificador personal. Las direcciones email son fáciles de crear y destruir. Los usuarios desean poder identificarse de forma diferente en muchos sitios *web*.
- 🚫 Los usuarios no desean ser identificados personalmente en algunos sitios *web*. EBIA lo permite, PKI no.
- 🚫 Es muy difícil para las organizaciones reaccionar ante pérdida de claves o errores de certificados con PKI, con EBIA no.
- 🚫 Si una clave en PKI es robada, el ladrón puede utilizarla una y otra vez sin que el dueño se entere.
- 🚫 Los usuarios suelen confiar más en las organizaciones donde obtienen sus direcciones email que en CA desconocidos o no visibles en PKI.
- 🚫 En las organizaciones, el uso de roles como “director financiero” o grupos como “técnicos de campo”, es sencillo con EBIA, ya que son listas de correo. Con PKI no es posible, ya que los certificados son unipersonales.

Con el mismo propósito que EBIA, proporcionar un método de identificación y autenticación, el cifrado basado en identidad (IBE) también es utilizado. Con IBE, se utiliza una dirección de

correo electrónico para crear un par de claves públicas y privadas para el cifrado de correo electrónico. IBE es diferente de EBIA, ya que IBE está basada en encriptación, pero se podría usar EBIA para distribuir IBE (Boneh & Franklin, 2003). Como se puede ver, estos mecanismos siguen utilizándose actualmente, y la seguridad de dichos métodos de identificación y autenticación dependen totalmente de la seguridad de los servidores de correo. Para proporcionar seguridad con estos mecanismos es necesario adoptar medidas de seguridad adicionales en los servidores de correo, debido a que **EBIA** e **IBE** son mecanismos de identificación y autenticación seguros basados sobre un sistema inherentemente inseguro.

Otro mecanismo similar para la identificación es el uso de una dirección de correo electrónico desechable (DEA), que ha sido también propuesto como un mecanismo de seguridad contra el *spam*. El uso de DEA junto con protocolos de direcciones de correo electrónico móviles (REAP) permite que un usuario que se quiera dar de alta con algún proveedor utilice un servidor REAP, que le suministra una cuenta de correo que puede ser eliminada si se recibe email o cambiada por el mismo servidor REAP si el usuario lo desea. El servidor se encargaría de realizar los cambios de cuenta de correo con los proveedores asociados (Seigneur & Jensen, 2003).

Existen propuestas basadas en PKI pero con cifrado ligero para autenticación de email (Adida, Hohenberger, & Rivest, 2005). La idea es preservar el uso natural del correo electrónico, sin entrar en conflictos con las expectativas de uso, como suele pasar con PKI. En este tipo de solución, cada dominio de email se convierte en una autoridad maestra para el esquema basado en identidad. El par de claves pública y privada es generado y la clave pública es distribuida vía DNS como un registro de texto asociado, y la clave privada es obtenida por IBE.

La proliferación de servicios de correo online ha originado la creación y utilización de herramientas que promulgan proteger la integridad y confidencialidad del usuario. En el año 2006, un artículo hizo referencia a **AQUINAS**, un sistema que permite cifrar el contenido utilizando un código de autenticación de mensajes (MAC) para el cifrado y estenografía para ocultar las claves y el texto cifrado (K. Butler, Enck, Plasterr, Traynor, & Mcdaniel, 2006). Las técnicas de autenticación biométrica han sido utilizadas también como esquema de autenticación (Rui & Yan, 2019). La huella digital fue planteada para ser utilizada dentro de sistemas de correo electrónico seguro, con una técnica que utilizaba IBE (Z. Wu et al., 2007).

Aunque es criticado por su complejidad, PKI sigue siendo una solución robusta y son muchos los autores que todavía apuestan en esa dirección. En el año 2012, Lina y Jiang propusieron el uso de tarjetas inteligentes dentro de un esquema de autenticación y cifrado para el correo electrónico. Los esquemas de autenticación han evolucionado mucho (Velásquez, Caro, & Rodríguez, 2018) y las técnicas de autenticación multifactor (MFA), donde el usuario tiene que

presentar más pruebas para autenticarse, tienen especial relevancia en nuestros días y pueden ser utilizadas para la autenticación de un servicio de correo electrónico (Jacomme & Kremer, 2021; Ometov et al., 2018; Velásquez et al., 2018).

La Identidad digital es la evolución de los mecanismos de identificación y autenticación para los servicios en línea a través de la utilización de tecnologías emergentes. Por medio de esta tecnología, un usuario puede identificarse universalmente, por lo que se considera un medio apropiado para la identificación y autenticación para los servicios de correo electrónico (Grassi, Garcia, & Fenton, 2020; National Institute of Standards and Technology, 2009).

b. Correo certificado

Los protocolos para correo electrónico certificado permiten que un emisor envíe un mensaje a un receptor, de tal forma que el receptor consigue el mensaje si y solamente si el emisor recibe una notificación de retorno correspondiente. El problema con este tipo de protocolos es que cualquiera puede simular enviar el mensaje.

Abadi y Blanchet (2005) proponen una manera de verificar este tipo de protocolos por medio de una herramienta. La idea general de este protocolo es que un tercero de confianza (TTP) intervenga en la comunicación, de esta forma, el emisor cifra una clave temporal con la clave pública del TTP. Esa clave temporal es utilizada para cifrar el mensaje. De esta forma, el receptor solamente puede descifrar el mensaje si tiene la clave. Para descifrarla, la envía al TTP para que la descifre, y es en ese momento, cuando el TTP le notifica al emisor que el receptor ha recibido el mensaje.

No solamente el problema de garantizar la recepción a través de email certificados compromete a un emisor y receptor, sino que también es necesario garantizar la recepción de muchos receptores de un email certificado. En esta línea dos protocolos de email certificados multi equipos fueron propuestos (J. Zhou, Onieva, & Lopez, 2005). Igualmente, se ha presentado una solución para correo electrónico certificado multi equipos sin la intervención de un TTP pero con soporte de *blockchain* (Hinarejos & Ferrer-Gomila, 2020). En dicha solución se preserva la confidencialidad del correo, incluso cuando está involucrada *blockchain*, y puede ser integrada en la infraestructura de correo electrónico existente si fuese necesario.

c. Soluciones integrales

Es muy importante definir los principios básicos de seguridad a la hora de diseñar una infraestructura de correo electrónico seguro. Uno de los estudios analizados propuso un diseño orientado a las necesidades de las organizaciones, acorde a los recursos y basado en los principios de la seguridad por diseño y la disponibilidad (Choukse, Singh, Laddhani, & Shahapurkar, 2012). La solución consiste en el desarrollo de una mezcla de las mejores soluciones propuestas hasta el momento, basándose en un análisis de amenazas y debilidades de las infraestructuras de correo electrónico.

Con respecto a las soluciones integrales, hay pocas propuestas en la literatura científica. El enfoque de esta línea suele sustentarse en recomendaciones basadas en modelos de seguridad (Al-Fedaghi & Alnasser, 2020), en diferentes soluciones existentes (Babraham, Alharbi, Alshiky, Alqurashi, & Kar, 2015) o en la utilización de marcos de trabajo con técnicas de cifrado (Bhatt & Sharma, 2019).

d. Soluciones en la nube

Por último, en el momento actual es frecuente que los servidores de correo se alojen en la nube. A la poca seguridad innata de este servicio, se le añade los problemas de seguridad propios de las soluciones en la nube (Alge, 2012). Una de las causas de pérdida de seguridad es que muchos proveedores omiten la verificación por razones de usabilidad, permitiendo ataques como la incriminación, la distribución de programas maliciosos y el espionaje (Hahn, Kunz, Schneider, & Vowé, 2012). La solución al problema de la nube requiere implementar servidores seguros de email en la nube utilizando tecnologías como la seguridad de la capa de transporte (TLS) (Rescorla, 2017), la autenticación SMTP y la capa de seguridad y autenticación simple (SASL) (Madi, Salehian, Masoumiyan, & Abdullah, 2012).

Hasta la fecha, diversos autores han varias propuestas para mejorar estos aspectos. Dos enfoques son los más utilizados. Por una parte, la utilización de MFA (Virgile & Yu, 2015). Por otra parte, el uso de técnicas de cifrado (Wei, Chen, Wang, Hu, & Ma, 2019, 2021; P. Xu, Jiao, Wu, Wang, & Jin, 2016).

e. Aspecto de seguridad considerados para evaluar y comparar las propuestas

Los aspectos de seguridad relacionados directamente con el correo electrónico son ocho:

- ❶ Confidencialidad.
- ❷ Integridad.
- ❸ No repudio.
- ❹ Autenticación.
- ❺ Disponibilidad.
- ❻ Control.
- ❼ Auditoria.
- ❽ Usabilidad.

Estos aspectos se evaluarán en cada una de las propuestas de seguridad estudiadas, se clasificarán para cada propuesta utilizando una nomenclatura de tres colores, como se describe en la Tabla 10, y se describirán en esta sección.

Tabla 10. Nomenclatura de colores utilizada para la evaluación de los aspectos de seguridad.

Color	Descripción
Verde	Aspecto considerado plenamente en la propuesta y asegurado desde que el remitente envía el mensaje hasta que llega a su receptor.
Naranja	Aspecto considerado en la propuesta y que tiene fallos, presenta vulnerabilidades o no asegura el mensaje desde que sale del remitente hasta que llega a su receptor.
Rojo	Aspecto no considerado ni tenido en cuenta o no aplica.

Fuente: Elaboración propia.

❶ Confidencialidad

La confidencialidad es una de las propiedades más deseadas en cualquier implementación o solución de correo electrónico. Se define como la propiedad que garantiza que solo las personas autorizadas pueden tener acceso a la información, en este caso al contenido del correo electrónico. En este aspecto es importante definir niveles de “confidencialidad” o niveles de acceso. Dentro del correo electrónico se puede tener confidencialidad a nivel del contenido del mensaje o a nivel de comunicación identificando los participantes.

❷ Integridad

La integridad es la propiedad que garantiza que solo las personas autorizadas puedan manipular o alterar la información, en este caso un correo electrónico. De la misma forma que en la confidencialidad, hay que garantizar que tanto la comunicación como el mensaje permanezcan inalterados por terceros no autorizados.

❸ No repudio

Garantizar que aquella persona, usuario o entidad que envió el mensaje o lo recibió no pueda negar o rechazar el hecho, es una propiedad deseable y necesaria para clasificar una solución de correo electrónico como segura.

❹ Autenticación

Esta propiedad garantiza que el autor y destinatario del mensaje es quien dice ser; identifica a la persona, usuario o entidad que genera y recibe el mensaje. Cuando se recibe un mensaje, garantiza que el remitente no ha sido suplantado por un tercero.

❺ Disponibilidad

Los mensajes de correo electrónico deberían estar disponibles a los usuarios autorizados cuando los necesiten. Esta propiedad garantiza la prevención de ataques DoS.

❻ Control

Si el usuario puede definir cuándo, cómo y quienes pueden acceder a la información de los correos, se puede decir que la propiedad de control está garantizada. El usuario debe poder definir el nivel de seguridad y los controles que desee a su información. Aunque no es un requerimiento de seguridad imprescindible, el control es necesario en esquemas de trabajo multiusuarios, donde muchos participantes pueden tener acceso parcial y otros total a los mensajes o información que se comparte, como es el caso de las listas de correo.

7 Auditoría

En un caso de uso organizacional, es necesario poder determinar qué, cuándo, cómo y quién realiza acciones sobre el sistema. Esta propiedad permite poder controlar o no, a elección de los usuarios o los administradores, todos los aspectos de la comunicación, sin perder en ningún momento las propiedades de seguridad anteriormente definidas. Debería poderse definir el esquema de auditoría a utilizar.

8 Usabilidad

La usabilidad como propiedad de seguridad facilita la adopción de la solución o propuesta. Es una propiedad deseable, ya que una buena propuesta de seguridad puede ser descartada por la complejidad de uso o implantación.

f. Propuestas de seguridad más aceptadas por las organizaciones

STARTTLS es una extensión de SMTP que permite el uso de seguridad en la capa de transporte TLS para proporcionar una comunicación sobre Internet privada y autenticada. TLS es ampliamente utilizado por el protocolo de transferencia de hipertexto (HTTP), pero ha sido utilizado para añadir seguridad a otros protocolos como SMTP (Hoffman, 2002).

La comunicación entre cliente y servidor se realiza como se muestra en la Figura 16. Después del envío del comando STARTTLS, se inicia la negociación típica TLS. Una vez acordados los niveles de seguridad, los algoritmos utilizados y el intercambio de claves, se procede al envío del correo de forma cifrada (color verde). El problema con este esquema es que hay comunicación en texto plano (color rojo) que puede alterar el proceso de negociación. Un atacante podría modificarlo y la comunicación quedaría comprometida, se daría un ataque de hombre en medio (MITM) que ocasionaría una degradación del cifrado.

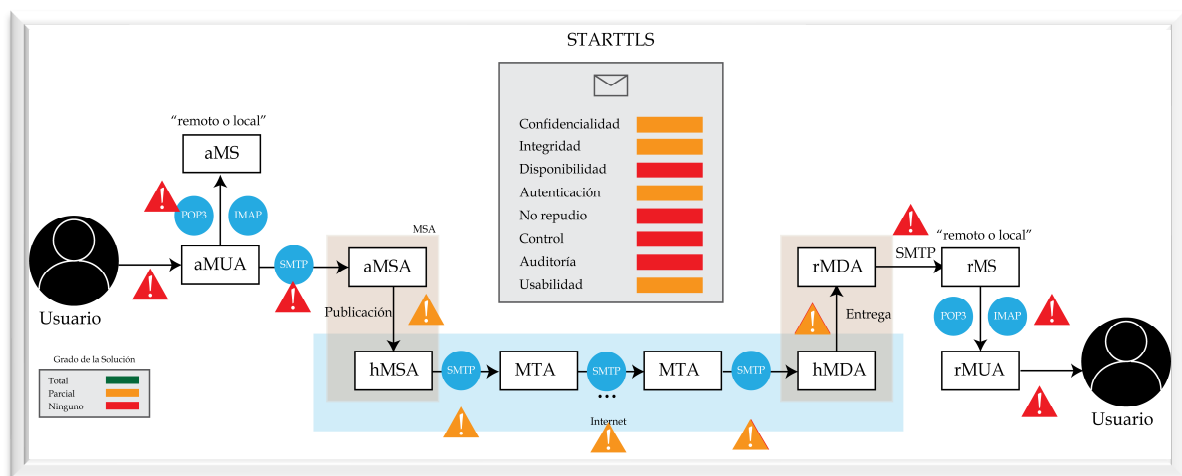
La evaluación de seguridad de esta solución se puede ver en la Figura 17. La evaluación es representada con 3 colores. Si el aspecto a evaluar está en rojo, significa que la propuesta no ofrece ninguna solución en ese aspecto, el grado de la solución es “ninguno”. Si es verde, significa que es “total”, la propuesta está focalizada a dar una solución completa en ese aspecto. Si es naranja, significa que la solución es “parcial”, lo que indica que, aunque la propuesta da solución a ese aspecto, sigue habiendo alguna vulnerabilidad que puede ser explotada o no aplica la solución a ese aspecto de seguridad en todo el canal de comunicación.

Figura 16. Sesión STARTTLS de SMTP entre cliente y servidor.



Fuente: Modificado de Hoffman (2002).

Figura 17. Evaluación de seguridad de STARTTLS.



Fuente: Elaboración propia.

STARTTLS solo asegura parcialmente la comunicación, ya que de ninguna manera verifica si el servidor acepta o no TLS, ni tampoco la validez del certificado. Si se da un ataque MITM, la comunicación cifrada puede que nunca se dé y queda comprometida la confidencialidad, integridad y la autenticación (dada por el protocolo TLS). De igual manera, estos tres aspectos no son tenidos en cuenta por la solución en otros puntos de la arquitectura, como, por ejemplo, la comunicación entre el MUA y el MSA.

Por otra parte, la usabilidad de esta solución depende la facilidad de uso de los protocolos TLS, que ya de por si es compleja por el uso de los certificados y la complejidad asociada a la gestión de estos. STARTTLS fue concebido para poner una capa de seguridad a nivel de SMTP entre los diferentes MTA o actores que tienen comportamiento MTA, como el caso de la comunicación hMDA a rMDA o aMSA a hMSA, dependiendo de la implementación. La última versión de TLS es la versión 1.3, que se creía segura, pero un reciente trabajo ha encontrado que un atacante puede perpetuar una variación del ataque Bleichenbacher (Bleichenbacher, 1998) con una degradación a TLS 1.2 (Ronen et al., 2019).

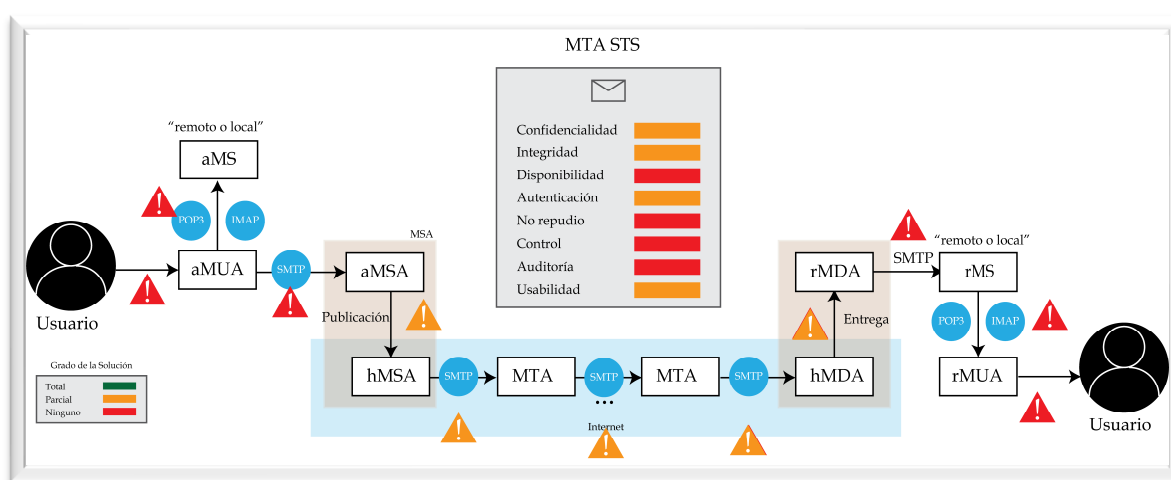
Seguridad de transporte estricta del agente de transferencia de mensajes (MTA STS)

La seguridad de transporte estricta del agente de transferencia de mensajes (MTA STS) es el resultado del esfuerzo de muchas compañías del sector involucradas (Google, Microsoft, Yahoo, Comcast, LinkedIn, entre otras) en hacer el correo electrónico más seguro (Margolis, Risher, et al., 2018). STARTTLS fue creado para mejorar la seguridad del correo electrónico, pero debido a su vulnerabilidad por un ataque MITM, MTA STS se creó para mejorar los aspectos donde el uso de TLS falla, como, por ejemplo, utilizar certificados digitales validados cuando se haga una conexión SMTP, de tal forma que la conexión entre los servidores va cifrada y se evita el MITM y, por ende, la degradación del cifrado.

Este mecanismo permite que los MTA que utilizan el protocolo SMTP cumplan con la seguridad utilizando TLS para la protección de todas las conexiones SMTP entre MTA. Si un MTA no cumple con la especificación TLS, se le debe denegar la entrega del correo. Con esto se evita la vulnerabilidad presente al utilizar STARTTLS. A su vez, con el objetivo de mejorar la seguridad, se propone utilizar el estándar para generar informes de problemas de conectividad con TLS (TLS-RPT). De esta forma, se reciben avisos desde los MTA en casos de problemas de entrega de correo y se puede actuar en consecuencia (Margolis, Brotman, Ramakrishnan, Jones, & Risher, 2018).

En la Figura 18, se puede ver la evaluación de MTA STS basada en los aspectos de seguridad considerados en este trabajo como claves a la hora de adoptar una solución. Hay que anotar que MTA STS es una mejora a STARTTLS y, por lo tanto, no añade más aspectos de seguridad y adolece de los mismos problemas que STARTTLS con respecto a las vulnerabilidades de TLS. Sin embargo, el ataque MITM no puede ser perpetuado, lo cual supone una mejora importante. Al añadir otros actores a la arquitectura, como terceros de confianza (por ejemplo, una CA para validar certificados), servidores de políticas para verificar si el MTA acepta TLS y los servicios de resolución de nombres DNS, se añaden las vulnerabilidades que puedan tener esos servicios.

Figura 18. Evaluación de seguridad de MTA STS.



Fuente: Elaboración propia.

Aunque este mecanismo frustra los ataques mencionados que pueden afectar a STARTTLS, otros problemas pueden presentarse y que están documentados en la sección 10 de la especificación (Margolis, Risher, et al., 2018), entre ellos los ataques DoS, la divulgación de las políticas de seguridad, el uso de certificados no validos propiedad de los atacantes, la creación de políticas débiles y los problemas que comprometen PKI.

Protocolo Dane

El Sistema de Autenticación basada en DNS de Entidades Nombradas, llamado DANE (Dukhovni & Hardaker, 2015), se fundamenta en las extensiones de seguridad del sistema de nombre de dominios (DNSSEC). Tiene como objetivo mejorar el escenario de TLS y las CA (Arends, Austein, Larson, Massey, & Rose, 2005).

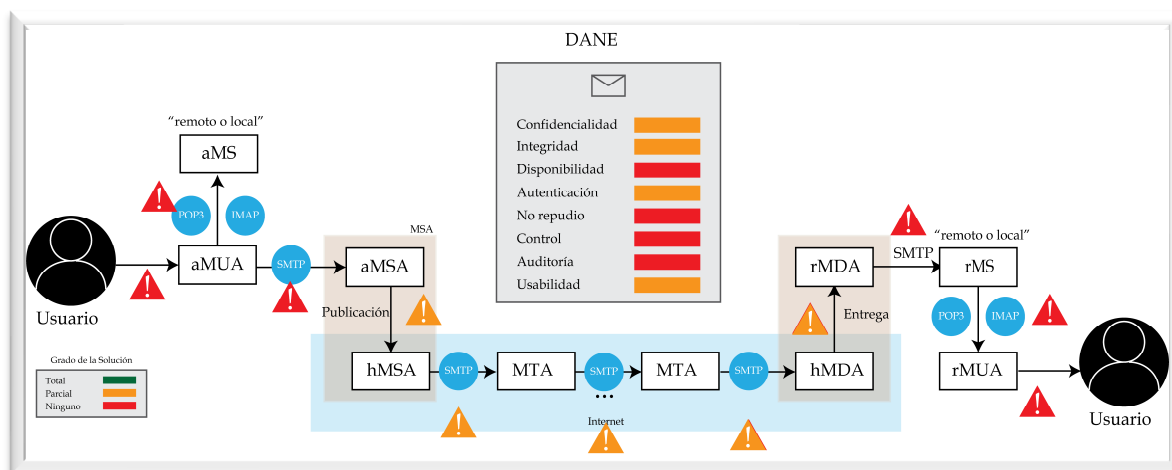
El uso de los certificados tiene el problema de que la firma del certificado solo tiene sentido si se confía plenamente en la clave que lo ha firmado, lo que lleva a que se debe tener una lista de CA disponible a mano.

DNSSEC funciona como las CA, siguiendo una cadena de confianza, pero se diferencia de las CA en que un dominio concreto solamente será firmado por su dominio padre. Así que DANE utiliza la infraestructura DNSEC como almacenamiento de las firmas de las claves y de los certificados de cualquier servicio TLS. DANE funciona como un servicio TLS normal, pero cuando el cliente obtiene el certificado para validar el servicio, hace una petición DNS para ver si concuerda con el certificado almacenado. Si coinciden, todo está correcto.

DNSSEC no está muy extendido, los navegadores no lo soportan y son pocos los sitios que lo tienen implementado. Recientes estudios adaptación (H. Lee, Girish, van Rijswijk-Deij, Kwon, & Chung, 2020) informaron sobre la poca adopción de DANE y su lento crecimiento, los problemas con la configuración y validación de los registros utilizando DNSSEC. Por otra parte, indican que a nivel operacional para los administradores es fácil su adaptación.

La Figura 19 muestra la evaluación del DANE según los aspectos de seguridad considerados en este trabajo. Es necesario tener en cuenta que DANE no tiene como objetivo proteger todo el tráfico SMTP, y hay que esperar a que la adopción de DNSSEC y DANE sea universal. Si DNSSEC se ve comprometida, no es posible recurrir a la CA pública para evitar ataques MITM y el atacante puede publicar asociaciones de certificados.

En este mismo gráfico se puede ver cómo DANE asegura una parte de todo el canal de comunicación que recorre un mensaje. Por lo tanto, según la evaluación, solo la confidencialidad, integridad, autenticación, en esa sección del canal puede ser analizada. Por ello es calificada como “parcial”, ya que no protege todo el tráfico, solo entre los MTA. Debido a su complejidad, su adopción es lenta y necesita mucho mantenimiento. Por lo tanto, su evaluación de “usabilidad” es parcial, indicador de que es compleja para la adopción y uso.

Figura 19. Evaluación de seguridad de DANE.

Fuente: Elaboración propia.

DarkMail

Es una nueva aproximación para el correo electrónico que tiene como objetivo sustituir totalmente a SMTP y a los otros protocolos de acceso como POP e IMAP.

El proyecto (Phil Zimmermann, Jon Callas, Mike Janke, 2018) se puede leer bajo el nombre de “entorno de correo de Internet oscuro” (DIME). Propone el protocolo de transferencia de correo oscuro (DMTP) y protocolo de acceso a correo oscuro (DMAP).

Este proyecto nació de otro denominado Lavabit, que según su autor (Levison, 2014), fue dado de baja ante las presiones del gobierno americano. Según Felician (2013), la razón principal es que los gobiernos quieren tener el control del contenido de los mensajes por razones de seguridad nacional y este tipo de servicio no se los permite, ya que no se deja rastro o es muy difícil acceder a ellos.

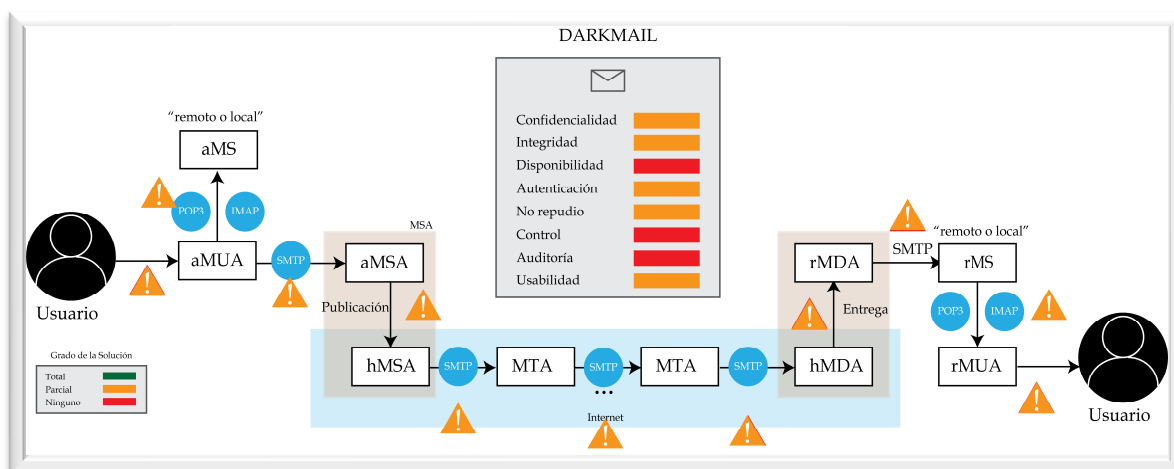
La idea de DarkMail (Phil Zimmermann, Jon Callas, Mike Janke, 2018) es eliminar la cabecera del mensaje reemplazándola con información de enrutamiento. Solamente expone las direcciones IP, ya que los creadores de DarkMail argumentan que Tor y VPNs las necesitan también y es imposible eliminarlas. Proporciona seguridad punto a punto sobre el enrutamiento y el mensaje. Todas las claves son generadas en el punto final y los servidores no las conocen. Utiliza el Estándar de cifrado avanzado (AES) o equivalente para cifrar el contenido y, a pesar de que continúa utilizando DNS, la intención es utilizar otro sistema.

Hay una implementación de DarkMail llamada Lavabit Flow (Levison & Delgado, 2021) y se espera que crezca en los próximos años. Es una plataforma de comunicaciones de código abierto y segura de extremo a extremo para mensajería asincrónica a través de Internet. Proporcionan tres niveles de cifrado (de confianza, cauteloso y paranoico), según en el lugar donde se almacena la clave. Actualmente, solo tienen implementado el modo de “confianza”, donde el servidor gestiona todos los problemas de privacidad, de forma que los usuarios deben confiar en el servidor. Todo el envío se realiza mediante SMTP y los mensajes son recibidos mediante POP o IMAP.

La evaluación de la seguridad según los aspectos considerados en este trabajo se puede ver en la Figura 20. Aunque DarkMail ofrece cifrado en todo el tráfico, es un sistema federado, lo que lleva a tener que confiar en terceros, ya sea por validación, gestión de certificados o la misma gestión de las claves. Como todos los servicios de Internet, se cuenta con el servicio DNS y, por lo tanto, parte de su seguridad dependerá de si se adopta o no DNSSEC, el cual proporciona protección contra la manipulación, pero no previene de bloqueos del servicio, ni tampoco la confidencialidad de las consultas está asegurada.

Se observa que DMTP utiliza el mismo esquema de seguridad que MTA STS. En consecuencia, su seguridad será tan fuerte como la del protocolo TLS que se utilice o lo bien configurado que esté el MTA. Con esta solución existe un alto grado de confidencialidad e integridad en relación con el contenido del mensaje, pero algunos aspectos de confidencialidad, como los orígenes y destino de la comunicación, pueden ser expuestos a los atacantes. Tampoco hay mecanismos que faciliten el control, ni es posible realizar actividades de auditoría. La solución está en desarrollo y, aunque hay una implementación, todavía tiene muchos elementos de base que resolver.

Figura 20. Evaluación de seguridad de DARKMAIL.



Fuente: Elaboración propia.

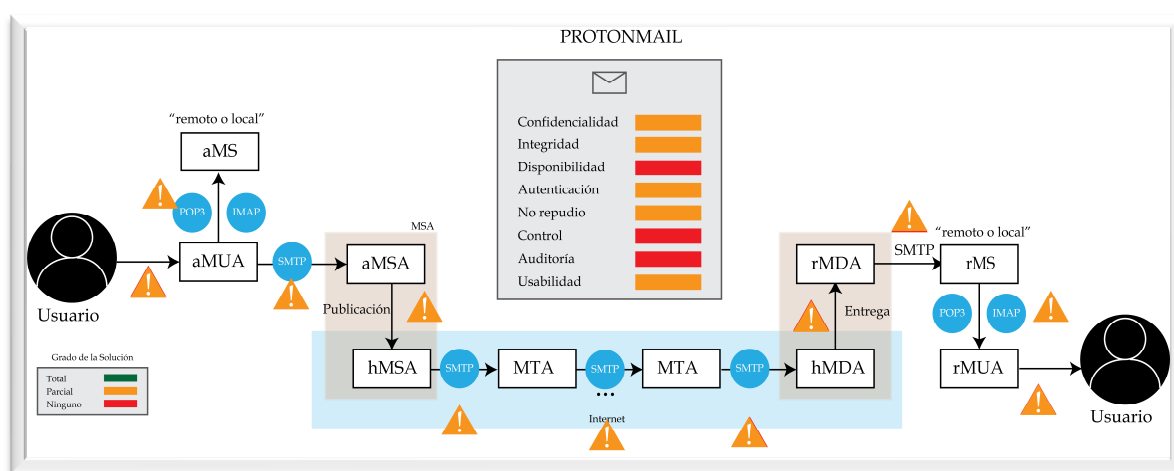
Protonmail

Es una solución de correo online que argumenta ofrecer cifrado extremo a extremo, de forma que los servidores solamente almacenan los datos de forma cifrada y los usuarios son los únicos que conocen la contraseña para descifrar el buzón (Andy Yen & Butler, 2021). Utiliza criptosistemas de clave pública, de manera que cuando un usuario crea una cuenta, el navegador genera un par de claves RSA, una pública y otra privada. La clave pública es utilizada para cifrar los mensajes del buzón utilizando AES-256. Las claves pública y privada son almacenadas en los servidores de Protonmail de forma cifrada. Para proteger todo el tráfico entre los clientes y los servidores se utiliza TLS.

La arquitectura de esta solución ya ha sido evaluada y analizada por Kobeissi (2018). El estudio expone serios defectos de la arquitectura de seguridad de esta solución. Comenta que no cumple los objetivos de seguridad autoproclamados cuando estos se someten a análisis exhaustivo. Critica su aplicación *web*, argumentando que las características que proporciona son insuficientes para garantizar la seguridad, y propone utilizar una aplicación que se ejecute localmente y eliminar completamente la solución *web*. Debido a las vulnerabilidades de ataques de diccionario de las claves PGP que se almacenan en los servidores, queda comprometida la confidencialidad e integridad.

La evaluación de la seguridad de Protonmail está ilustrada en la Figura 21. Dado que es una solución que abarca todo el canal de comunicación de una arquitectura de correo, está al mismo nivel que DarkMail, con un grado de seguridad parcial en el que muchos actores participan, y está expuesta a las consideraciones de seguridad de cada uno de ellos.

Figura 21. Evaluación de seguridad de Protonmail.



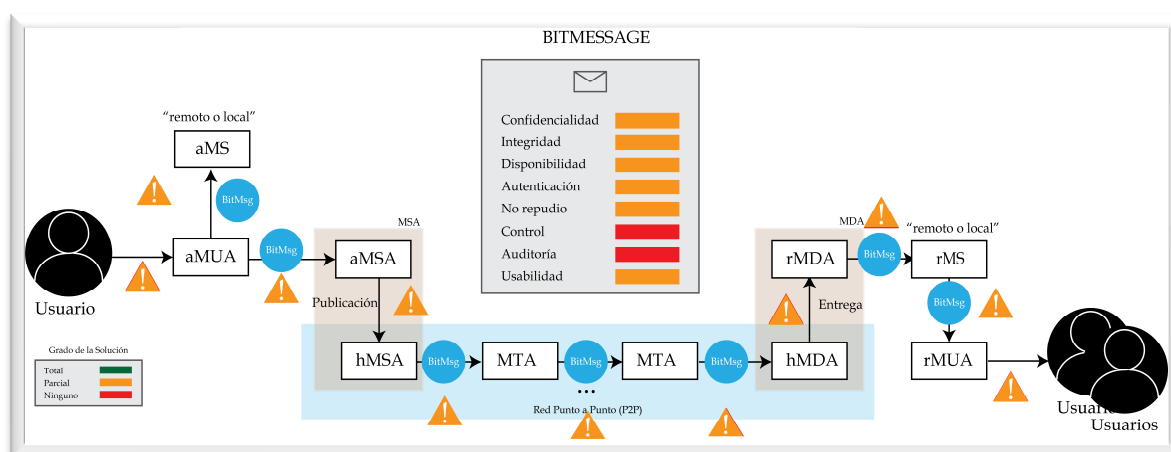
Fuente: Elaboración propia.

Bitmessage

Utiliza un sistema totalmente descentralizado bajo un protocolo de comunicación punto a punto (P2P) e inspirado en Bitcoin y se proclama como una alternativa al correo electrónico tradicional (Warren, 2021). El mensaje es enviado a todos los usuarios en la red, pero solamente el receptor puede decodificarlo con su clave privada. Los usuarios se pueden suscribir a mensajes de una dirección particular (Warren, 2012).

Algunos aspectos de su seguridad han sido estudiados y discutidos por la comunidad científica (Kovacs, Karakatsanis, & Svetinovic, 2014). Bitmessage permite a los usuarios generar direcciones basadas en una frase de contraseña. Si la misma frase es utilizada por dos usuarios distintos, la aplicación generará la misma dirección y se usarán las mismas claves privadas para el cifrado, lo que genera un gran problema de seguridad y la confidencialidad, autenticación e integridad quedan comprometidas. Otro problema es el almacenamiento de las claves privadas que se guardan en la máquina del usuario sin ningún tipo de protección.

En la Figura 22, se representa el resultado de la evaluación de Bitmessage realizada en la presente investigación, donde resaltan dos elementos. En Bitmessage, como en toda red P2P, los MTA aparecen embebidos en la red y toda la carga de envío, recepción y lectura queda a cargo de los MUA, MS, MDA y MSA. Se observa que la auditoría y el control no aplican en este tipo de solución. El único protocolo utilizado en todo el recorrido del mensaje es el Bitmessage por lo que se concluye que el resto de los aspectos de seguridad estudiados solo son cubiertos parcialmente. Siendo una plataforma de P2P, Bitmessage aporta una solución al tema de disponibilidad, característica propia de las redes P2P (Bhagwan, Savage, & Voelker, 2003; Yu Wu, Yang, Qu, Xiao, & Dai, 2011), pero es una solución parcial debido a que la disponibilidad del mensaje queda a cargo de la confianza que se deposite en los nodos o a la disponibilidad basada en los incentivos de los mismos.

Figura 22. Evaluación de seguridad de Bitmessage.

Fuente: Elaboración propia.

El método de Bitmessage para prevenir el correo no deseado es la utilización de una prueba de trabajo, lo cual obliga al usuario que envía el mensaje a usar poder computacional para disuadir el uso de envío masivo de mensajes. Esta técnica desperdicia poder computacional y hace que su uso en la práctica sea ineficiente. Para abordar este problema, Shi et al. (2021) proponen un mecanismo basado en prueba de espacio, que requiere que el usuario dedique una cierta cantidad de espacio en disco para enviar un mensaje. Otra propuesta (Schaub & Rossi, 2015) se focaliza en realizar una mejora al algoritmo de prueba de trabajo mencionado, de tal forma que los usuarios maliciosos se vean más afectados que los usuarios legítimos.

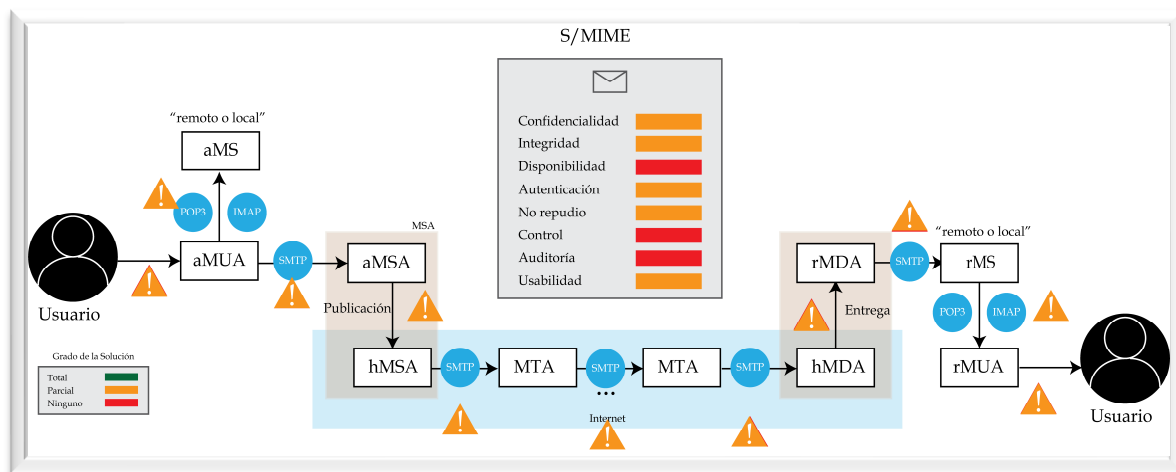
Extensiones de Correo de Internet Multipropósito Seguro (S/MIME)

Algunos mecanismos de seguridad pueden ser aplicados al contenido del mensaje, en especial al cuerpo del mensaje, y es allí donde esta solución hace su cometido. Este estándar se basa en los criptosistemas de clave pública. Con S/MIME lo que se cifra es el mensaje y no la comunicación (J. Schaad, B. Ramsdell, & Turner, 2019).

Cada una de las secciones del mensaje se cifra utilizando una clave de sesión. La clave de sesión se inserta en cada uno de los encabezados de las secciones y se cifra con la clave pública del destinatario, de modo que únicamente el destinatario puede leerlo descifrando con su clave privada. De igual modo, la firma del mensaje se cifra con la clave privada del remitente, pudiendo ser verificada la integridad del mensaje.

La Figura 23 muestra la evaluación de S/MIME basado en los mismos aspectos de seguridad que se han analizado en las otras soluciones. La utilización de la firma y el cifrado desde el origen (MUA) asegura la confidencialidad, el no repudio y la integridad del mensaje. No obstante, recientes estudios han demostrado que esta solución presenta vulnerabilidades que pueden ser explotadas por los atacantes (Khacef & Pujolle, 2019; Müller, Brinkmann, Poddebniak, Böck, et al., 2019; Müller, Brinkmann, Poddebniak, Schinzel, & Schwenk, 2020; Poddebniak et al., 2018; Schwenk et al., 2020). S/MIME es una solución aceptable, pero necesita ser ajustado en muchos aspectos como se comenta en los trabajos mencionados para evitar MITM, exfiltración de correo (EFAIL) y ataques de contenido, entre otros. Considerando esas vulnerabilidades, se han evaluado y calificado los aspectos de seguridad de S/MIME como asegurados parcialmente, siendo la disponibilidad, el control y la auditoría no aplicables para esta solución.

Figura 23. Evaluación de seguridad de S/MIME.



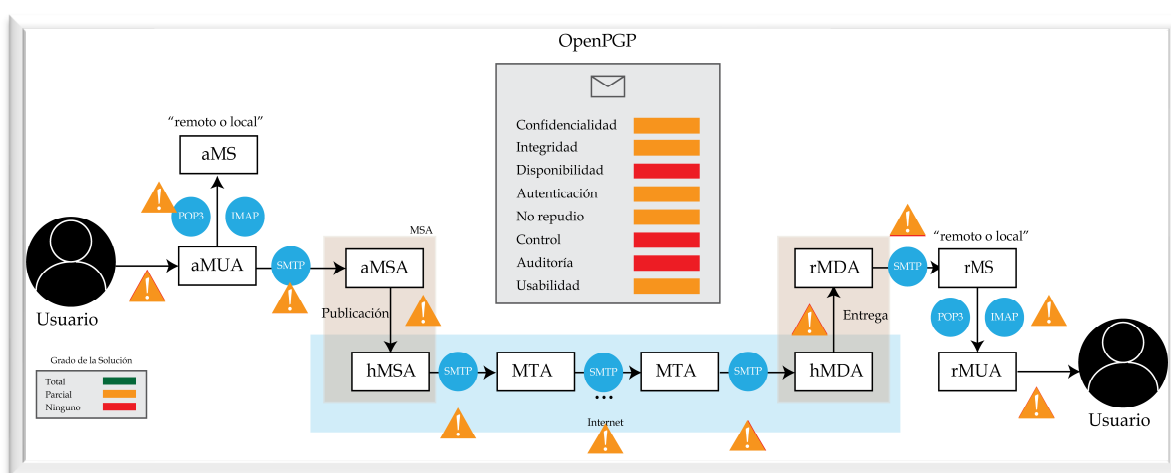
Fuente: Elaboración propia.

Privacidad Bastante Buena Abierta (OpenPGP)

La Privacidad Bastante Buena Abierta (OpenPGP) es una propuesta del año 1998 y actualizada en el año 2007, que tiene como objetivo proporcionar funciones de cifrado, descifrado, firma y gestión de claves (Callas et al., 2007). No es una solución orientada al correo electrónico, pero ha sido utilizada para el cifrado y la firma en otras soluciones de seguridad de correo presentadas previamente (Andy Yen & Butler, 2021; Elkins, Torto, Levien, & Roessler, 2001; Warren, 2021). Es ampliamente utilizada por muchas comunidades de usuarios en Internet para cifrar y muchos clientes de correo electrónico ofrecen seguridad con OpenPGP a través de firma y cifrado de los mensajes. Es una solución que no es muy escalable y presenta muchos problemas de usabilidad, como ocurre también con la propuesta S/MIME (Reuter, Boudaoud, Winckler, Abdelmaksoud, & Lemrazzeq, 2020).

La evaluación de OpenPGP que se ha realizado en la presente tesis (Figura 24) es muy parecida a S/MIME, ya que tienen los mismos objetivos. La utilización de la firma y el cifrado desde origen (MUA) asegura la confidencialidad, el no repudio y la integridad del mensaje, como en S/MIME. Pero de igual manera, recientes estudios han demostrado que OpenPGP presenta vulnerabilidades que pueden ser explotadas por los atacantes (Müller, Brinkmann, Poddebniak, Böck, et al., 2019; Müller et al., 2020; Müller, Brinkmann, Poddebniak, Schinzel, & Schwenk, 2019; Poddebniak et al., 2018; Schwenk et al., 2020).

Figura 24. Evaluación de seguridad de OpenPGP.



Fuente: Elaboración propia.

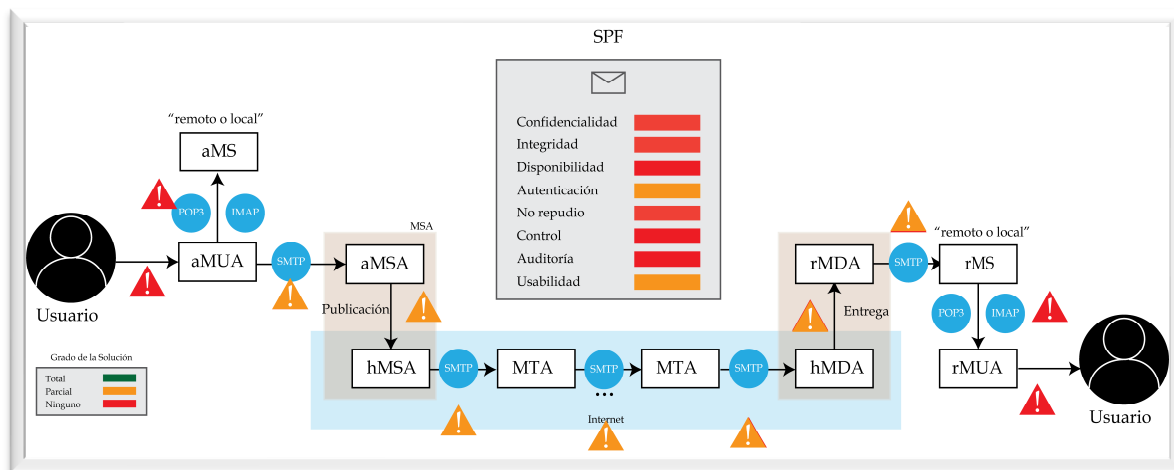
Marco de Políticas del Remitente (SPF)

El marco de políticas del remitente (SPF) es un sistema cuyo objetivo principal es proteger los correos electrónicos contra la falsificación de direcciones y, por consiguiente, minimizar los ataques de *phishing*. Utiliza DNS para identificar los nombres de dominio de los servidores SMTP que han sido autorizados para el transporte de correo (Wong & Schlitt, 2006). SPF verifica el remitente que aparece en el sobre del mensaje, es decir el campo MailFrom, y consulta el registro SPF para comprobar si el dominio del remitente está autorizado, antes de publicar el mensaje. SPF está bastante extendido debido a que facilita la identificación y autenticación de origen por parte de los servidores SMTP destino.

SPF da solución parcial a los aspectos de seguridad de autenticación y usabilidad, ya que se focaliza en la autenticación de los servidores y es de fácil manejo dada su facilidad de configuración (Figura 25). Los registros SPF se pueden tornar obsoletos. Por esta razón muchos proveedores utilizan mecanismos adicionales a SPF. En este contexto, no pasar la autenticación SPF no es garantía de bloqueo.

En SPF, si el mensaje es reenviado, el registro deja de tener sentido. SPF es muy complicado de mantener puesto que no tiene forma de generar informes para hacerle seguimiento. SPF no verifica la dirección que aparece en el campo From: de la cabecera del mensaje, que es el componente que ven los usuarios y puede ser suplantado. Además de las vulnerabilidades comentadas, existe evidencia de algunos ataques que pueden sobrepasar la seguridad prometida por SPF (Ajina & Kumar, 2019; K. Shen et al., 2020).

Figura 25. Evaluación de seguridad de SPF.



Fuente: Elaboración propia.

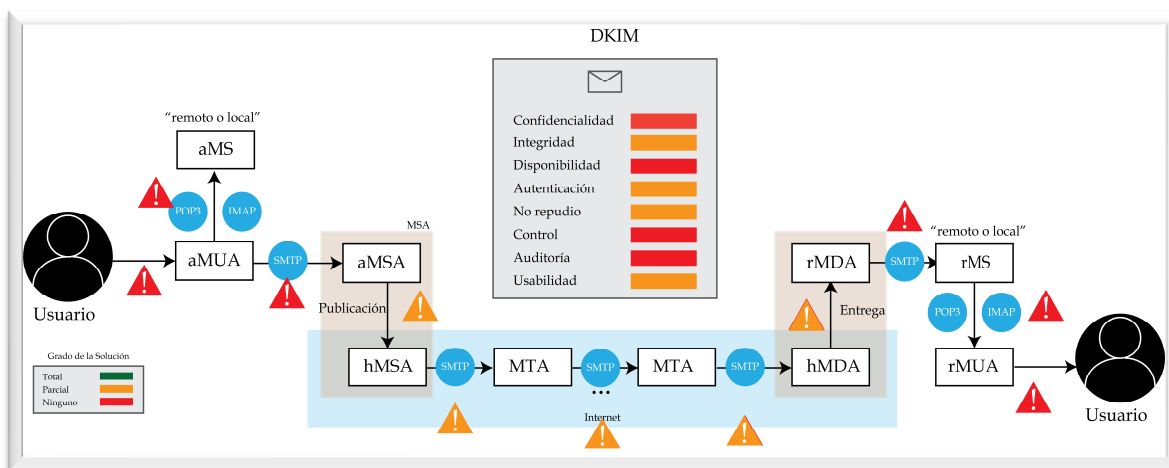
Correo electrónico identificado con clave de dominio (DKIM)

El correo electrónico identificado con clave de dominio (DKIM) crea un marco de autenticación a nivel de dominio para el correo electrónico utilizando la tecnología de clave pública y el DNS para probar la procedencia y contenido del correo electrónico (Crocker, 2011). El objetivo de DKIM es proteger la identidad del firmante y la integridad de los mensajes mientras que conserva la funcionalidad del correo como es conocida hoy en día.

La idea básica es que el servidor del remitente añada una firma al correo electrónico para que el servidor del destinatario pueda verificarla accediendo a un registro DNS anteriormente publicado por el remitente. Es un mecanismo de autenticación basado en DNS y, al igual que SPF, requiere de la creación de un registro DNS, pero añade más complejidad al destinatario, que tiene que gestionar la validación de las firmas. DKIM añade una firma a la cabecera del correo electrónico saliente y expone la clave pública de su dominio para que los servidores puedan verificar la firma. Como SPF, DKIM ayuda a identificar el correo no deseado y la manipulación de este desde el MTA remitente hasta el MTA del destinatario.

En la evaluación de seguridad de DKIM realizada en la Figura 26 la autenticación, la integridad y el no repudio se clasifican como resueltos parcialmente ya que, aunque su solución es el objetivo principal de DKIM, esta no se produce en todo el circuito, por lo que DKIM no protege la ruta del mensaje desde el MUA hasta el MTA. Por otra parte, estudios de seguridad realizados con DKIM han desvelado problemas de seguridad al implementar una solución con DKIM (Herzberg, 2009; Konno, Kitagawa, & Yamai, 2020; K. Shen et al., 2020).

Figura 26. Evaluación de seguridad de DKIM.



Fuente: Elaboración propia.

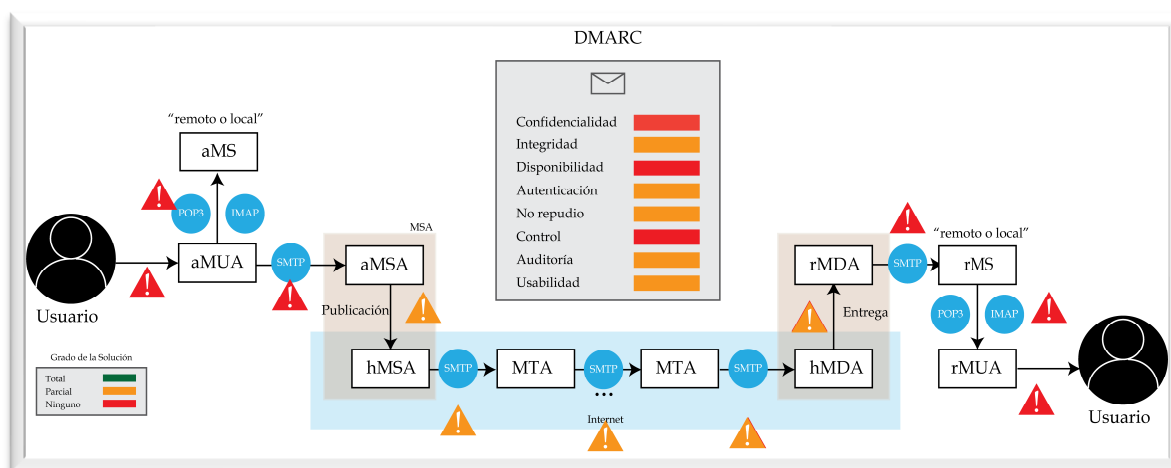
Autenticación, Informes y Conformidad de Mensajes basada en Dominios (DMARC)

La Autenticación, Informes y Conformidad de Mensajes basada en Dominios (DMARC) hace uso de los métodos de autenticación existentes para la autenticación de los mensajes de correo entrantes, con el objetivo de que el destinatario se asegure que el mensaje está alineado con la información que tiene sobre el remitente. Si el mensaje se considera “no alienado” con las políticas definidas, DMARC incluye una guía paso a paso de cómo se deben gestionar esos mensajes (Kucherawy & Zwicky, 2015).

DMARC se basa en las especificaciones DKIM y SPF. Permite que una organización publique una política indicando qué mecanismos utilizará (DKIM, SPF o ambos) para enviar correo electrónico desde ese dominio, qué hacer con el campo FROM: qué hacer cuando ocurran fallos y cómo generar los informes de las acciones realizadas bajo esas políticas.

Dado que DMARC se basa en SPF y DKIM, mejora los aspectos de seguridad que estos proporcionan, pero no está exento de los problemas o vulnerabilidades comentadas en su respectivas secciones, así como de problemas propios como la no detección de correos no deseados que no vienen de suplantación y la generación de falsos positivos (Dan, Kitagawa, Sakuraba, & Yamai, 2019; Kambourakis, Gil, & Sanchez, 2020; Shang, Zhu, & Liu, 2020; K. Shen et al., 2020).

La evaluación realizada a DMARC se puede ver en la Figura 27. DMARC no ofrece solución a la confidencialidad, disponibilidad ni a aspectos de control. Los aspectos de seguridad se centran en la autenticación e integridad a nivel de dominio, no a nivel del usuario, y su seguridad se aplica a nivel de MTA. El no repudio se garantiza con el uso de la firma DKIM y de igual manera a nivel de dominio. En cuanto a la usabilidad, tiene la misma complejidad que SPF y DKIM.

Figura 27. Evaluación de seguridad de DMARC.

Fuente: Elaboración propia.

g. Tabla comparativa de las soluciones y propuestas de seguridad descritas

En la Tabla 11, se puede observar si las soluciones de seguridad discutidas anteriormente garantizan o no cada una de las propiedades de seguridad deseadas en una implementación de una arquitectura de correo electrónico. Confidencialidad, integridad y autenticación son las propiedades más desarrolladas, pero se puede concluir con la tabla que la mayoría de las soluciones solo tratan algunos aspectos muy específicos de las propiedades y dejan de lado otros aspectos que son igual de importantes o deseables en muchos casos.

La columna “Fallos”: indica si la solución presenta algún fallo de seguridad de base o se han identificado algunos fallos por medio de estudios realizados por algunos autores. Se puede ver que todas las soluciones estudiadas presentan algún fallo, detalle para tener en cuenta previamente a su adopción.

La columna “Circuito” indica si la solución ofrece seguridad en todo el recorrido del mensaje, desde que sale del remitente hasta que llega a su destinatario. Se puede ver que las soluciones suelen ser arquitecturas desarrolladas desde cero y no son servicios añadidos a la arquitectura actual.

Tabla 11. Aspectos de seguridad de las soluciones de seguridad más extendidas.

Aspectos de seguridad										
Soluciones	Confidencialidad	Integridad	No repudio	Autenticación	Disponibilidad	Control	Auditoría	Usabilidad	Fallos	Circuito
STARTTLS	x	x	x	x	x	x	x	x	Sí	Parcial
MTA STS	x	x	x	x	x	x	x	x	Sí	Parcial
DANE	x	x	x	x	x	x	x	x	Sí	Parcial
DARK MAIL	x	x	x	x	x	x	x	x	Sí	Completo
PROTONMAIL	x	x	x	x	x	x	x	x	Sí	Completo
BITMESSAGE	x	x	x	x	x	x	x	x	Sí	Completo
S/MIME	x	x	x	x	x	x	x	x	Sí	Completo
OpenPGP	x	x	x	x	x	x	x	x	Sí	Completo
DKIM	x	x	x	x	x	x	x	x	Sí	Parcial
SPF	x	x	x	x	x	x	x	x	Sí	Parcial
DMARC	x	x	x	x	x	x	x	x	Sí	Parcial

Nota: Interpretación de colores según la Tabla 10 .

Fuente: Elaboración propia.

Otra conclusión de esta evaluación es que ninguna de las soluciones analizadas tiene en cuenta aspectos de disponibilidad, control o auditoría. Esos aspectos suelen añadirse como una capa más, como los registros de auditoría de algunos servidores de correo. Los mecanismos de control de acceso y la auditoría son características que no se tienen en cuenta en la arquitectura actual de correo electrónico y tampoco hay extensiones que la consideren. La tendencia actual es proporcionar una arquitectura que ofrezca total control para el usuario, pero no ofrecen mecanismos para gestionar ese control, por ejemplo, a través de permisos o control de accesos gestionados por el mismo usuario.

IV. TECNOLOGÍA BLOCKCHAIN

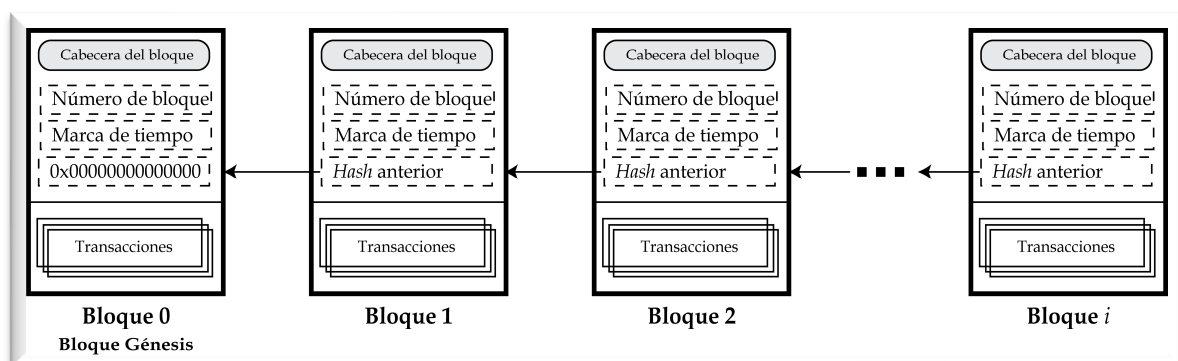
1. Introducción

En el año 2008 apareció el primer documento que definía el concepto de *blockchain*. Acto seguido, *blockchain* fue utilizada como estructura base sobre la que se implementó Bitcoin, un sistema descentralizado de una moneda digital que permite la distribución de *bitcoins* utilizando un programa P2P (Nakamoto, 2008). La identidad de su autor, identificado bajo el seudónimo Satoshi Nakamoto, continúa siendo desconocida hoy en día.

La tecnología de Bitcoin utiliza un libro mayor de cuentas como una cadena de bloques distribuida. Todos los participantes tienen una copia, en la que se registran y se verifican todas las transacciones de *bitcoins* solicitadas por los participantes, proporcionando transparencia a todos ellos. Lo más innovador de Bitcoin es que da una solución a un problema denominado “el doble gasto”, lo que evita que un participante pueda utilizar el mismo dinero virtual en más de una transacción sin depender de ninguna entidad confiable y centralizada (Chohan, 2018).

Las transacciones son organizadas como una cadena de bloques, donde los nuevos bloques se van expandiendo de forma jerárquica, utilizando técnicas criptográficas para asegurar la integridad. Para añadir un nuevo bloque a la cadena, es necesario un consenso entre los nodos por medio de una competencia que es realizada por los nodos en la red (Crosby, Nachiappan, Pattanayak, Verma, & Kalyanaraman, 2016). Además de las transacciones, el resumen del bloque (*hash*) y el del bloque anterior también es almacenado, creando un enlace criptográfico de la cadena. La Figura 28 muestra una cadena simple donde se pueden ver gráficamente estos conceptos.

Figura 28. Esquema simple de una cadena de bloques.



Fuente: Modificado de Kolb et al. (2020).

Además de la admisión de los nuevos nodos, el procedimiento de consenso gestiona el protocolo de lectura para la verificación de la cadena de bloque y la coherencia de toda la información con la gestión de cada una de las copias en cada uno de los nodos participantes (Shijie Zhang & Lee, 2020). Una vez se ha añadido un bloque y todas las transacciones que contiene, no es posible alterarlo, asegurando así, que la información contenida en el bloque no pueda ser comprometida ni alterada. El libro mayor se comporta como una base de datos de transacciones entre pares, distribuida, persistente, inalterable y verificable.

Blockchain garantiza a un sistema como Bitcoin, que las transacciones están aseguradas de problemas conocidos de seguridad, privacidad y confianza, como el problema del doble gasto, la divulgación no autorizada de transacciones privadas, la dependencia de una autoridad central confiable y la falta de confianza de los sistemas descentralizados. La seguridad en *blockchain* es un tema de especial interés entre los investigadores (Leng, Zhou, Zhao, Huang, & Bian, 2020; R. Zhang, Xue, & Liu, 2019).

Blockchain no solamente es útil en el campo de las monedas digitales. Bitcoin es una de las variadas implementaciones que *blockchain* puede tener en el mundo financiero (Ali, Ally, Clutterbuck, & Dwivedi, 2020) y su uso se ha extendido a otras áreas de estudio, con diversas aplicaciones en:

- ▣ Agricultura (Bermeo-Almeida et al., 2018).
- ▣ Privacidad (Calvaresi, Dubovitskaya, Calbimonte, Taveter, & Schumacher, 2018; de Haro-Olmo, Varela-Vaca, & Álvarez-Bermejo, 2020; S. Shi et al., 2020).
- ▣ Transporte (Batta, Gandhi, Kar, Loganayagam, & Ilavarasan, 2020; Pournader, Shi, Seuring, & Koh, 2020).
- ▣ Subcontratación masiva (*crowdsourcing*) (Gong, van Engelenburg, & Janssen, 2021; Kogias et al., 2019; Ma, Sun, Lei, Qin, & Lu, 2020; More & Motwani, 2021; J. Zhang, Cui, Ma, & Yang, 2019).
- ▣ Internet de las cosas (Ekramifard, Amintoosi, & Seno, 2020; Lo et al., 2019; Lone & Naaz, 2021; Masood & Faridi, 2021; Tran, Ali Babar, & Boan, 2021).
- ▣ Negocios (Alkhudary, Brusset, & Fenies, 2020; Garcia-Garcia, Sanchez-Gomez, Lizcano, Escalona, & Wojdyski, 2020; Grover, Kar, & Vigneswara Ilavarasan, 2018).
- ▣ Almacenamiento seguro (Aslam, Manzoor, & Babar, 2019; Liang, Fan, Li, Zhang, & Gaudiot, 2020; Zahed Benisi, Aminian, & Javadi, 2020).
- ▣ Hay una industria en crecimiento alrededor de *blockchain* y muchos intereses por parte de los gobiernos y el mundo académico (Bodkhe et al., 2020).

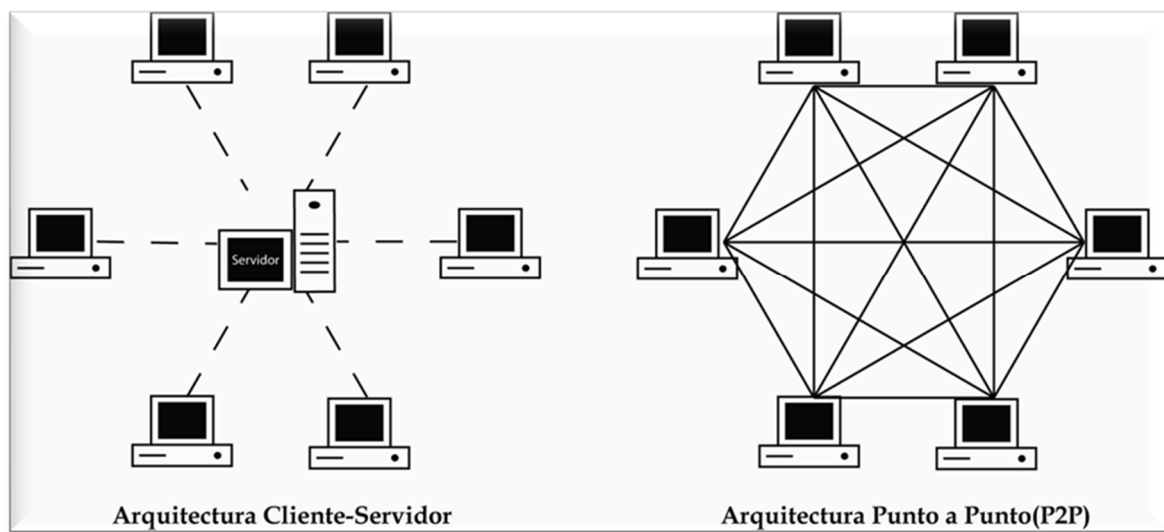
2. Estructura y Arquitectura de *Blockchain*

2.1. Tipología

Blockchain en su definición más simplista es una red P2P que sirve como un libro mayor público. La arquitectura de una red P2P permite a las redes basadas en esa arquitectura autoorganizarse, escalar y seguir funcionando aún en presencia de fallos o problemas en los nodos participantes, incluso con una población de nodos variable y sin un servidor y gestor central que la administre (Androutsellis-Theotokis & Spinellis, 2004).

La arquitectura de una red P2P se representa gráficamente como un conjunto de nodos interconectados que tienen el mismo rol. En esto se diferencia del modelo tradicional, donde existe un nodo con el rol de servidor, que es la autoridad de la red, y un conjunto de nodos con el rol de cliente, que se comportan como subordinados. En el esquema de la Figura 29 se aprecia la diferencia de la arquitectura P2P y la arquitectura tradicional de cliente y servidor.

Figura 29. *Arquitectura cliente-servidor y P2P.*



Fuente: Elaboración propia.

Blockchain, por otra parte, haciendo uso de la arquitectura de P2P, tiene distribuido el libro mayor entre todos los nodos participantes, lo que la convierte en una arquitectura distribuida. Con *blockchain* no hay un punto central de decisión o control, las decisiones se toman por consenso entre los participantes, de tal forma que el control no está centralizado. Se puede definir de forma general como una arquitectura descentralizada (Chatterjee & Chatterjee, 2018).

Esta descentralización se logra con el control a través del consenso de los participantes, pero si unos pocos participantes tomaran el control, esa característica desaparecería. Aunque puede ser problema en el caso de aplicaciones financieras como Bitcoin (Aggarwal & Kumar, 2021a; Praveen, Anand, Singh, & Ranjan, 2021; Ye, Li, Cai, Gu, & Fukuda, 2018), esta configuración puede ser deseable en otras topologías utilizadas por otras aplicaciones más específicas de *blockchain* (Pahlajani, Kshirsagar, & Pachghare, 2019).

No existe una única configuración o topología a la hora de crear una solución basada en una arquitectura de *blockchain*. Para *blockchain* se distinguen tres topologías diferentes que condicionan cómo los participantes interactúan en la red (ya sea ingresando al sistema, consultando o modificando el libro mayor) y quiénes tienen el control a la hora de tomar decisiones (Khettry, Patil, & Basavaraju, 2021). Estas tres tipologías son:

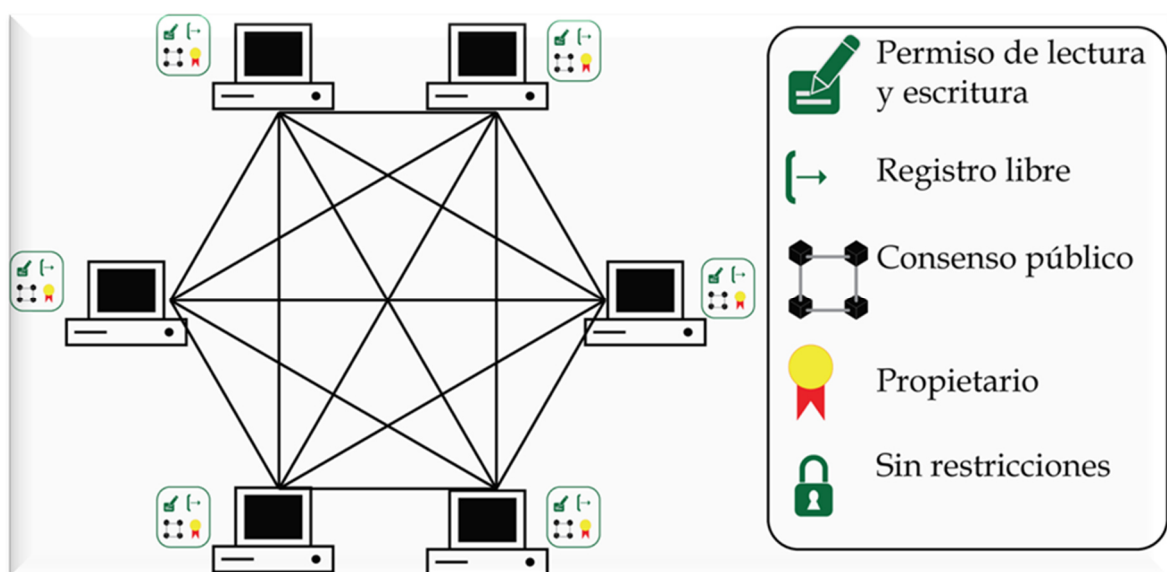
- ① Pública (Irresberger, John, & Saleh, 2020).
- ② Privada (Pahlajani et al., 2019).
- ③ De consorcio o federada (Zavolokina, Ziolkowski, Bauer, & Schwabe, 2020).

a. *Blockchain* pública (*Public Blockchain*)

La *blockchain* pública es la topología con menos restricciones de las tres tipologías existentes. Por ello se puede decir que está abierta al público y no es necesario solicitar permiso para participar (Aggarwal & Kumar, 2021c; Khettry et al., 2021). Está totalmente descentralizada, donde cada nodo participante y de forma ilimitada, tiene una implementación de la plataforma, puede unirse a la red, leer el libro mayor con todas sus transacciones, realizar transacciones y participar en el proceso de consenso para determinar qué bloques deben ser agregados a la cadena.

Para participar en el proceso de consenso, se requiere que los nodos resuelvan acertijos criptográficos muy complejos, lo que pueden consumir una gran cantidad de energía. Ejemplos de implementaciones de *blockchain* pública son las criptomonedas, como, por ejemplo Bitcoin y Ethereum (Irresberger et al., 2020). La Figura 30 muestra de forma gráfica esta tipología.

Figura 30. *Blockchain* pública.



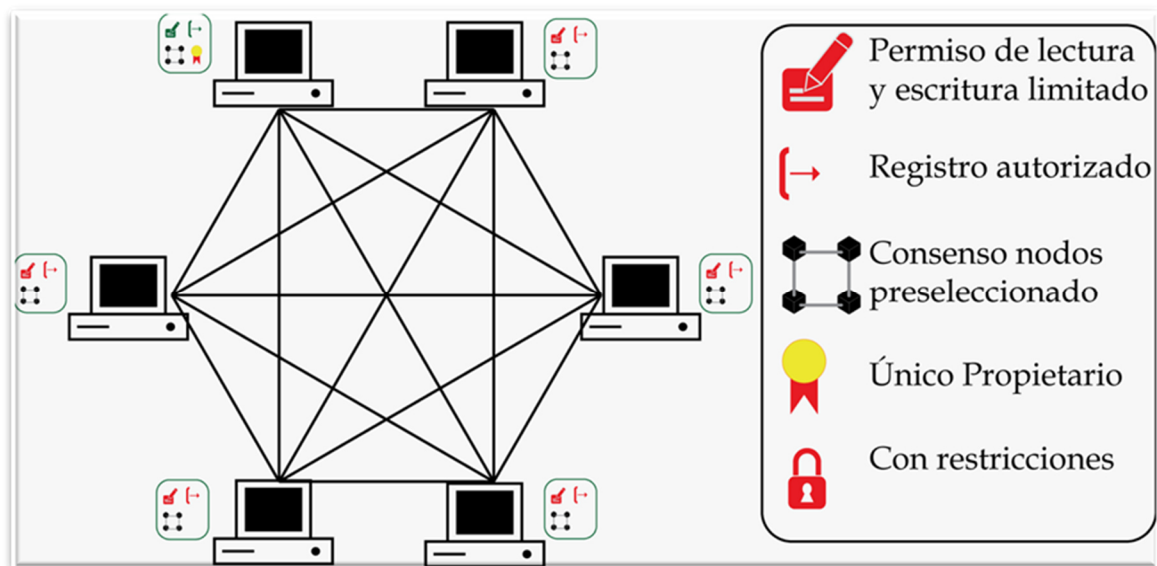
Fuente: Elaboración propia.

b. Blockchain privada (*Private Blockchain*)

Cuando en una *blockchain* existe un propietario que se reserva el derecho a proporcionar el acceso, la lectura y la escritura del libro mayor a los participantes, se denomina a esta topología *blockchain* privada (Kwak, Kong, Cho, Phuong, & Gim, 2019). Este propietario se suele denominar autoridad central y es el responsable de realizar las modificaciones en la red. Por este motivo, una implementación basada en esta topología se considera centralizada.

En la Figura 31 se muestra un esquema de *blockchain* privada. La cantidad de nodos participantes es limitada, ya que se requiere permiso de acceso. Aunque todos los nodos participantes podrían participar en el consenso, la autoridad central selecciona los nodos habilitados para realizar esa actividad.

Figura 31. Blockchain privada.



Fuente: Elaboración propia.

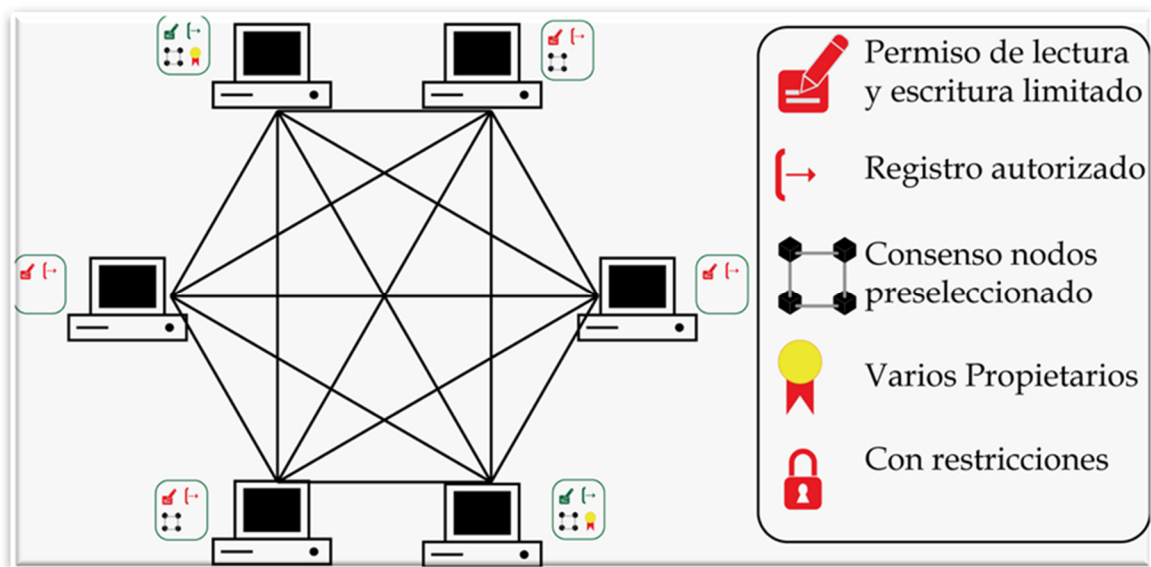
Esta topología es adecuada para las organizaciones privadas que implementan alguna solución propietaria, como, por ejemplo, organizaciones financieras. Existen plataformas que permiten la creación de este tipo de aplicaciones, como Hyperledger Fabric, Ethereum Geth, Libra, Quorum, MultiChain y R3 Corda (Dabbagh, Choo, Beheshti, Tahir, & Safa, 2021; Polge, Robert, & Le Traon, 2020).

c. **Blockchain de consorcio o federada (*Consortium Blockchain*)**

Esta topología permite a varias empresas por común acuerdo llegar a un consenso para interactuar entre sí utilizando una *blockchain*. Con este modelo, los nodos con autoridad son preseleccionados, a diferencia de una *blockchain* privada, donde un nodo posee toda la autoridad. Es un esquema parcialmente descentralizado porque un grupo de nodos se selecciona para realizar las tareas de consenso e interactuar entre sí y el acceso de lectura y escritura está restringido a cierto número de nodos. Dada la participación de varias empresas, en esta topología se proporcionan datos privados y públicos y para participar en la red es necesaria una invitación (Y. Chen et al., 2020; Dib, Brousmiche, Durand, Thea, & Hamida, 2018; Zavolokina et al., 2020).

HyperLedger Fabric y Ethereum son dos ejemplos de marcos de trabajo para desarrollar este tipo de *blockchain* (Sajana M; Sethumadhavan, M, 2018) y son muchas las aplicaciones que las empresas pueden poner en funcionamiento con *blockchain privadas* para dar servicio a sus clientes o para negocios entre las mismas empresas. En especial, se pueden ver aplicaciones a nivel financiero (Nathan & Jacobs, 2020), almacenamiento masivo de datos (X. Chen et al., 2020), energía (Fu, Dong, & Ju, 2020; Shaomin Zhang, Pu, Wang, & Dong, 2019), transporte (Di Wang & Zhang, 2020), Internet de las cosas (Huang et al., 2019), derechos digitales (Yue Wu, Lu, Yu, & Luo, 2020), votación (K. Li, Li, Hou, Li, & Chen, 2018), salud (Zhao, Wen, Huang, Zhou, & Yang, 2020), bolsa (Al-Shaibani, Lasla, & Abdallah, 2020), entre otras. La Figura 32 muestra gráficamente los conceptos asociados a esta topología.

Figura 32. *Blockchain de consorcio.*

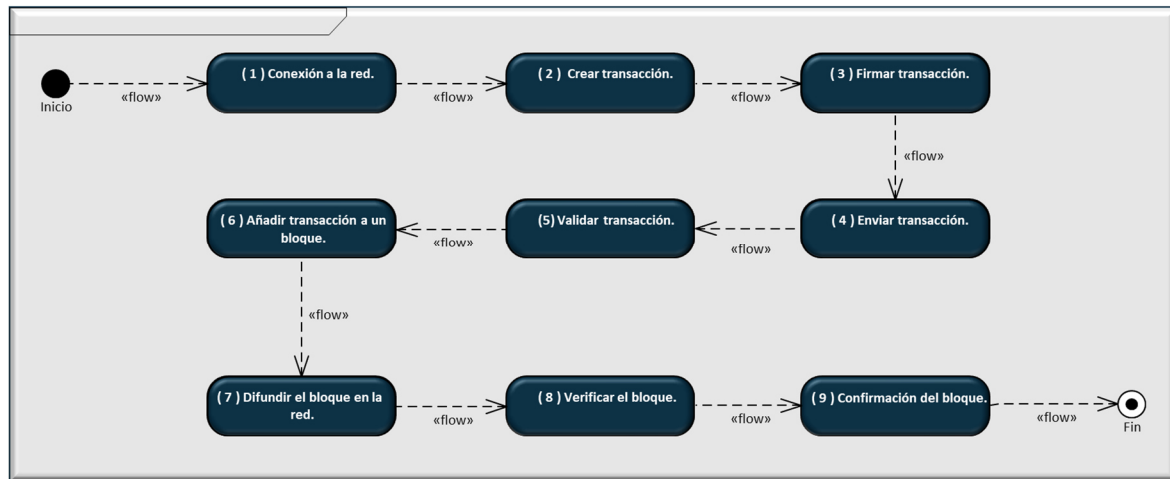


Fuente: Elaboración propia.

2.2. Ciclo de vida de una transacción en *blockchain*

Blockchain se basa en una estructura que se comporta como una base de datos distribuida y segura de registro de transacciones. La vida de una transacción se inicia con su creación hasta que aparece inmutable y visible para los demás participantes de la red. El ciclo de vida de una transacción se puede esquematizar como lo muestra la Figura 33.

Figura 33. Ciclo de vida de una transacción en *blockchain*.



Fuente: Elaboración propia.

- ❶ **Conexión a la red:** Para explicar contextualmente cómo funciona *blockchain*, se parte de la premisa de que un usuario de la red está en posesión de un identificador criptográfico único que será utilizado para realizar sus transacciones y un par de claves privadas y públicas que debe asegurar (Pal, Alam, Thakur, & Singh, 2021). En el caso de Bitcoin, el usuario está en posesión de una billetera electrónica (*wallet*) y el identificador es una dirección, denominada comúnmente dirección Bitcoin.
- ❷ **Crear transacción:** Los usuarios identificados inician una transacción en la red. La transacción podría estar relacionada con un contrato, una criptomoneda o cualquier otro activo, dependiendo de la aplicación de *blockchain* utilizada. En el caso de Bitcoin, el usuario selecciona la dirección destino donde desea enviar una cantidad determinada de *bitcoins*, ingresa dicha cantidad y finaliza la transacción (Atzei, Bartoletti, Lande, & Zunino, 2018).
- ❸ **Firmar transacción:** La transacción es firmada con la clave privada del remitente, para demostrar que el usuario que crea la transacción no puede negar que hay participado en la transacción (no repudio) (Fang et al., 2020). En el caso de Bitcoin, la billetera es la

encargada firmar la transacción con la clave privada del remitente, garantizando así que es el dueño de los *bitcoins* que va a transferir.

- ④ **Enviar transacción:** La transacción es enviada a todos los nodos de la red (El-Ansary, Alima, Brand, & Haridi, 2003; Hao et al., 2020).
- ⑤ **Validar transacción:** Los nodos de la red validan la transacción. En el caso de Bitcoin, se debe verificar si el remitente tiene suficiente cantidad de dinero para realizar transacciones legítimas (Ferdous, Chowdhury, & Hoque, 2021).
- ⑥ **Añadir transacción a un bloque:** Unos nodos con capacidad de tomar decisiones incluyen la transacción validada en un bloque, ya que la transacción debe ser representada en un bloque en toda la red. En el caso de Bitcoin, los nodos que añaden la transacción a un bloque se denominan mineros. Aunque cualquier nodo puede ser minero, solo los nodos que computacionalmente tienen capacidad para realizar esta tarea son los que realmente participan en esa decisión. La tarea que deben realizar los nodos mineros en Bitcoin para la generación de un bloque es resolver un problema criptográfico. Esto lo realizan con un algoritmo de consenso denominado prueba de trabajo (Shijie Zhang & Lee, 2020).
- ⑦ **Difundir el bloque en la red:** Los nodos que han añadido el bloque lo difunden a todos los participantes de la red. En el caso de Bitcoin, los nodos mineros que han resuelto el problema son los que difunden dichos bloques (Shahsavari, Zhang, & Talhi, 2020).
- ⑧ **Verificar el bloque:** Los miembros de la red analizan y verifican el bloque recibido. La verificación del bloque se realiza con el apoyo del algoritmo de consenso. En el caso de Bitcoin, los nodos que verifican el bloque aprueban que se incluya en *blockchain* solamente si el minero que lo ha enviado tiene el resultado correcto del problema planteado. Es en ese instante es cuando el destinatario puede ver la primera confirmación (Gol, 2019).
- ⑨ **Confirmación del bloque:** El nuevo bloque aprobado se adjunta a la red, se vuelve permanente e inmutable y la transacción queda confirmada. En el caso de Bitcoin, cada que se van añadiendo más bloques, se realizan más confirmaciones, asegurando la transacción (Kovalchuk et al., 2018).

2.3. Conceptos criptográficos utilizados en *blockchain*

Hay cinco conceptos criptográficos que deben ser considerados para entender la arquitectura de *blockchain*:

- ✎ Funciones de resumen (*hash*).
- ✎ Punteros *hash*.
- ✎ Árboles de Merkle.
- ✎ La firma digital.
- ✎ Las direcciones o seudónimos.

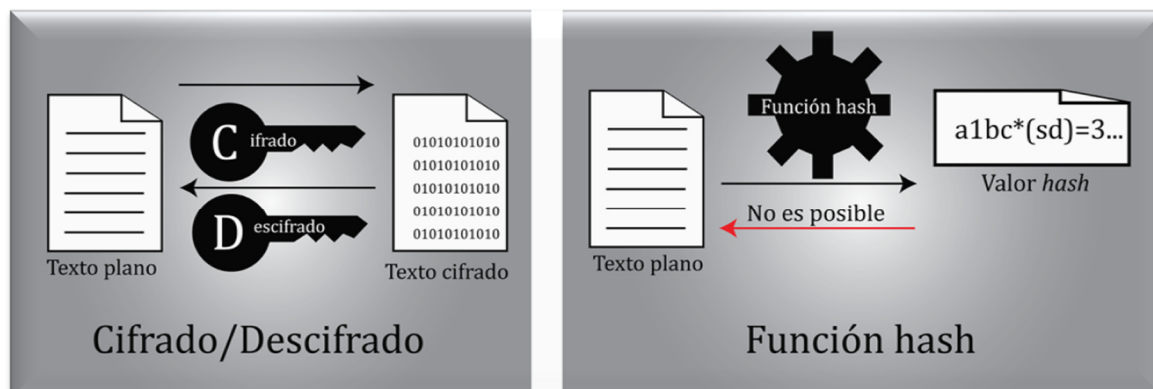
Todos estos aspectos combinados ayudan a prevenir el problema del doble gasto de los sistemas distribuidos de monedas digitales y asegurar la no modificación retrospectiva de cualquier dato de una transacción añadida con éxito en *blockchain*.

a. Funciones de resumen o funciones *hash*

Las funciones *hash* gozan de popularidad tanto en el mundo académico como fuera de él. Se pueden definir como funciones capaces de transformar un bloque binario de tamaño variable en un bloque binario de tamaño fijo. Estas funciones no se utilizan para el cifrado o descifrado, pero son indispensables para verificar la integridad de los datos que están procesando (Menezes, Van Oorschot, & Vanstone, 1996).

Al resultado de la transformación se le suele denominar *hash*, *digest* o resumen, y no es posible con ese resumen obtener los datos originales que fueron procesados. En la Figura 34 se puede ver un esquema contextual de las funciones de cifrado/descifrado y las funciones de resumen para entender su diferencia y objetivo.

Figura 34. Funciones de cifrado/descifrado y funciones *hash*.



Fuente: Elaboración propia.

La firma digital fue uno de sus primeros usos, en cuanto a que el resultado de una firma suele ser un resumen de todos los elementos de los datos en lugar de todos los elementos. El valor *hash*, siendo más corto, permite que se realicen cálculos más rápidos, lo que origina protocolos de firma más eficientes. Además de este uso inicial, las funciones *hash* se han aplicado a otras áreas relacionadas como la protección de contraseñas, árboles de Merkle, derivación de claves, integridad de datos, detección de datos corruptos, presencia de virus, protocolos de identificación, eliminación de datos redundantes, entre otros usos (Katz & Lindell, 2014).

Dentro de *blockchain*, el uso de las funciones *hash* se puede dividir en seis categorías (L. Wang, Shen, Li, Shao, & Yang, 2019):

- 1 Prueba de trabajo o minería de monedas (PoW) (Gol, 2019).
- 2 Generación de direcciones (Chan, Chin, & Goh, 2020).
- 3 Generación de bloques (con el uso de árboles de Merkle) (Merkle, 1988).
- 4 Firma digital (Fang et al., 2020).
- 5 Generación de números pseudoaleatorios (PNG) (Bhattacharjee, Maity, & Das, 2018).
- 6 Utilización de pruebas de conocimiento cero (ZKP) como el mecanismo Fiat-Shamir (Hasan, 2019).

La importancia de las funciones *hash* dentro de *blockchain* se debe principalmente a las siguientes propiedades (Rogaway & Shrimpton, 2004):

-  **Dependencia a nivel de bits:** El resumen de un mensaje depende completamente de todos los bits del mensaje. Por ende, si cambia un bit del mensaje, el valor del resumen o *hash* debe cambiar, aproximadamente, la mitad de los bits.
-  **Resistencia a la preimagen:** Esta propiedad indica que la función resumen computacionalmente debe ser difícil de revertir. Esto garantiza la integridad.
-  **Resistencia a la segunda preimagen:** No debe ser posible encontrar el mismo resumen en dos mensajes distintos, no deben existir dos mensajes con el mismo valor *hash*. Esta propiedad considera que uno de los mensajes es conocido e intenta localizar un mensaje diferente con el mismo valor *hash*.
-  **Resistencia a colisiones:** Debe ser computacionalmente difícil encontrar dos mensajes que tengan el mismo valor *hash*. En este caso, no se imponen restricciones sobre los mensajes. No es recomendable utilizar una función resumen si ésta es vulnerable a la resistencia a colisiones.

El cumplimiento de las propiedades anteriores garantiza que las funciones que cumplen todas las propiedades no son vulnerables y se evita que a partir de un *hash* se pueda recuperar el contenido del mensaje y hace que no sea posible encontrar otro mensaje con el mismo *hash*.

Las principales funciones utilizadas en *blockchain* suelen ser RIPEMD-160 (Bosselaers, Dobberty, & Preneel, 1997) y SHA-256 (Gallagher, 2015). Hay varios trabajos académicos que describen profundamente estas funciones y realizan análisis comparativos del desempeño y sus propiedades de seguridad que pueden ser consultados (Long, 2019; Parmar & Jit, 2021; Pittalia, 2019; Raghuvanshi, Khurana, & Bindal, 2014; Thirupalu, Scholar, & Reddy, 2020).

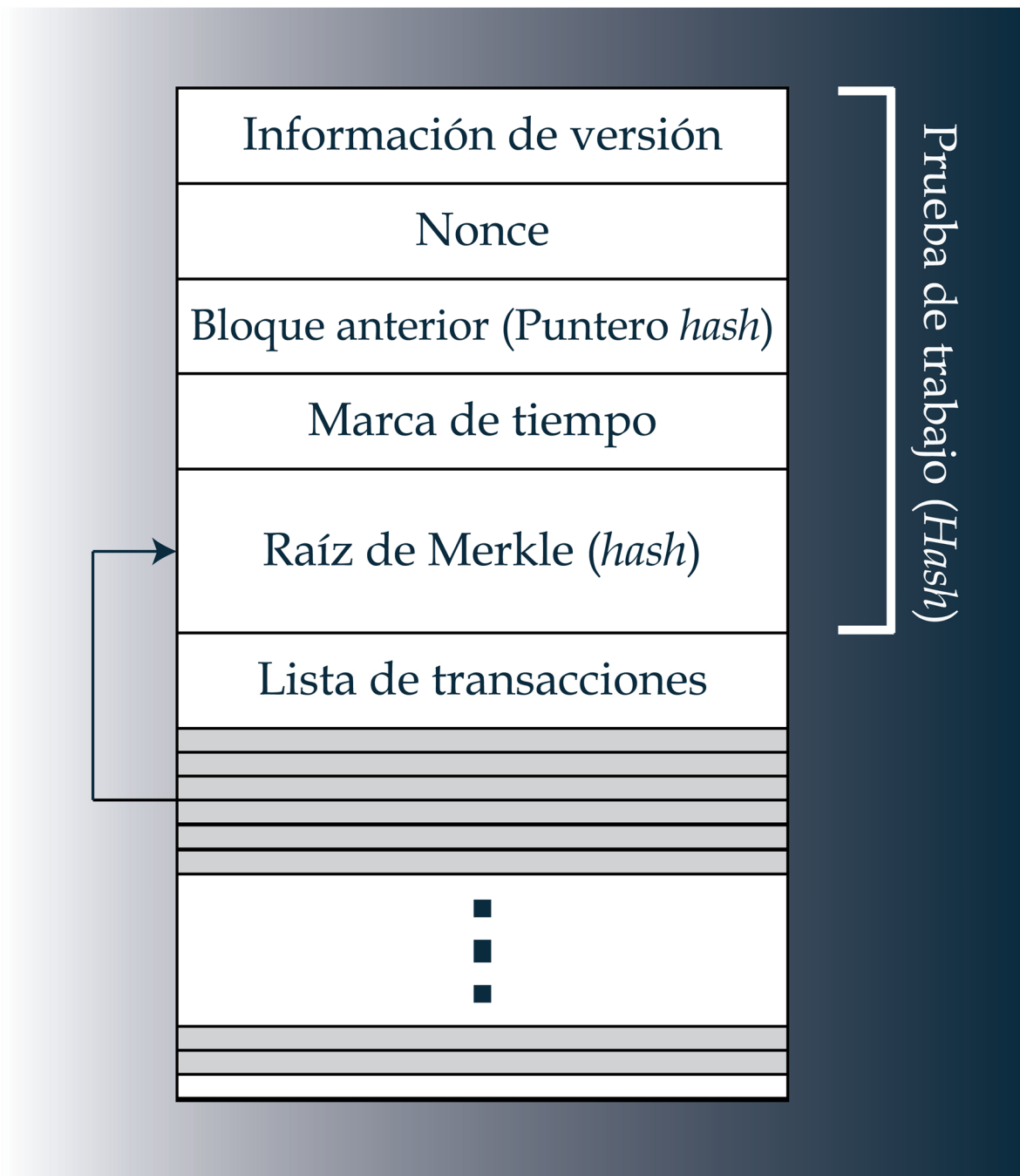
b. Punteros *hash*

La Figura 35 muestra una estructura de bloque simplificada de *blockchain* utilizada con el algoritmo de consenso de PoW, pero sirve para explicar el concepto de bloque detalladamente. Un bloque está compuesto por un encabezado y un cuerpo, que contiene una lista de transacciones. El encabezado del bloque contiene varios campos, entre los cuales está el número de versión para rastrear el *software* de las actualizaciones de protocolo, una marca de tiempo, el tamaño del bloque y el número de transacciones contenidas en el bloque. El campo raíz de Merkle representa el valor *hash* de las transacciones contenidas en el bloque, como se ha explicado en el apartado anterior. Algunos campos son utilizados específicamente por el algoritmo de consenso de PoW, como el campo “*nonce*” y el campo “*dificultad*”, que serán explicados en la sección de algoritmos de consenso.

Los bloques se enlazan por medio de punteros formando una cadena de bloques. Los punteros son punteros *hash* que almacenan la posición del bloque anterior más el *hash* del bloque anterior (Kolb et al., 2020). Dado que esta información es pública, cualquier persona puede verificarla y comprobar si ha sido manipulada.

La modificación del contenido de los bloques es un proceso complicado porque requiere cambiar los punteros *hash* de todos los bloques anteriores, una actividad computacionalmente pesada (Saad et al., 2019). Aunque el atacante pudiera realizar dicha manipulación, no podría modificar la cabecera del primer bloque y ésta permitiría descubrir el ataque. Este primer bloque se conoce como el bloque Génesis y es generado primeramente cuando se construye el sistema. Todo lo anteriormente expuesto demuestra que la cadena tiene la propiedad de ser resistente a la manipulación.

Figura 35. Estructura de bloque simplificada.

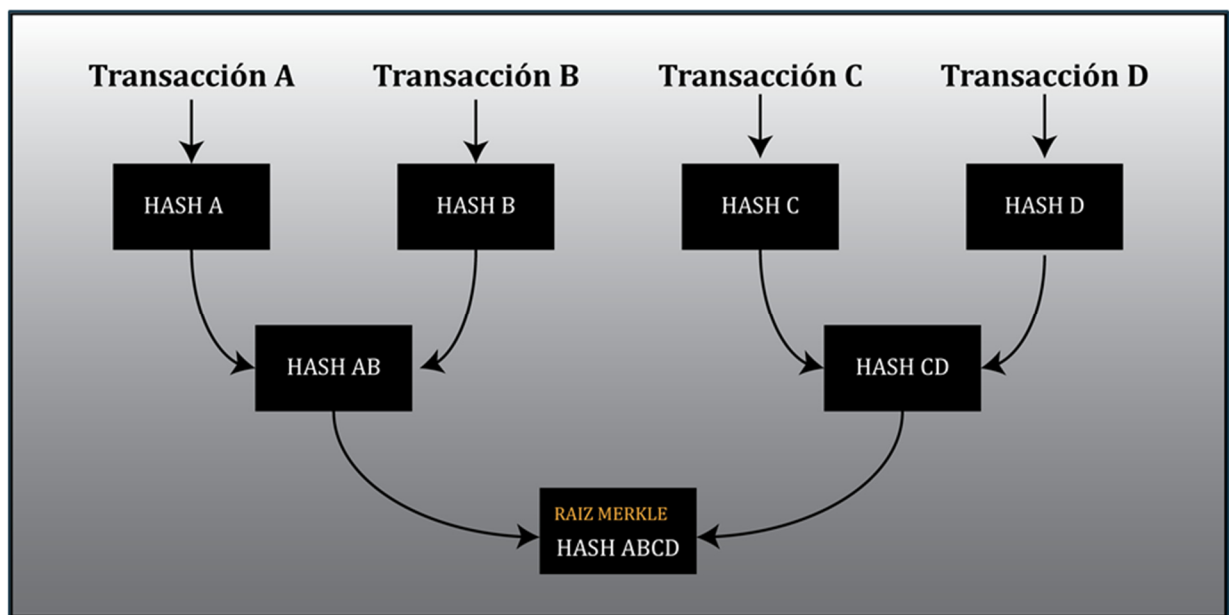


Fuente: Modificado de Labbi, Kannouf, Chahid, Benabdellah, & Azizi (2019).

c. Árboles de Merkle

Los árboles de Merkle son una estructura de organización y búsqueda muy utilizada que fue creado por Ralph Merkle en el año 1982 (Merkle, 1982). Se define como un árbol de búsqueda binario con sus nodos vinculados uno a otro mediante la utilización de punteros *hash*. La estructura va agrupando estos nodos en grupos disjuntos, de forma que dos nodos en el nivel inferior se agrupan en uno en el nivel principal, y para cada par de nodos de nivel inferior, el algoritmo crea un nuevo nodo de datos, que contiene el valor *hash* de cada uno. Esto se va repitiendo hasta llegar a la raíz del árbol como se muestra en la Figura 36.

Figura 36. Árbol de Merkle.



Fuente: Elaboración propia.

La importancia de esta estructura radica en que puede evitar la modificación de los datos. La modificación es evitada debido a que, si un atacante intenta realizar una alteración, el valor *hash* del nodo padre se alterará, lo que le obliga a tener que cambiar todos los nodos de la ruta desde el nodo inferior hasta el superior. Se puede detectar fácilmente si ha habido manipulación verificando si el puntero *hash* del nodo raíz coincide o no con el *hash* del puntero utilizado. La pertenencia de un nodo al árbol de Merkle se puede verificar en un tiempo logarítmico calculando *hashes* en la ruta y comprobando el valor *hash* con la raíz (Nakamoto, 2008).

d. La firma digital

Cuando se firma digitalmente un documento se está asegurando su autoría de forma similar a como lo puede garantizar la firma manuscrita, pero en formato electrónico. El uso de la firma digital garantiza que los mensajes o documentos recibidos por medios electrónicos procedieran efectivamente de sus legítimos autores y evita posibles suplantaciones. La mayoría de los países consideran la firma digital con la misma validez legal que la firma manuscrita (Martínez, Hernández-Álvarez, & Encinas, 2020).

Una firma digital además de ser fácil de calcular y difícil de falsificar, debe proporcionar suficientes garantías de que depende totalmente de la firma, el documento que ha firmado y el momento que realizó dicha firma. Para poder crear una firma y su posterior verificación, se utilizan protocolos criptográficos que permiten acreditar la autoría de un documento, garantizar su integridad (el documento no se ha modificado durante el proceso de entrega) y el no repudio por parte de los participantes.

El no repudio de la información es muy importante para la seguridad en *blockchain*. Para garantizar el no repudio de la información y en especial en *blockchain*, se aplican algunas tecnologías de seguridad, como la firma digital (L. Wang et al., 2019), la autenticación de identidad y el sellado de tiempo (Haber & Stornetta, 1991). No repudio significa que los participantes no pueden negar la transacción y el comportamiento en la transacción en *blockchain*. El propósito del servicio de no repudio es recopilar, mantener, proporcionar y verificar la evidencia innegable sobre los mensajes del transmisor al receptor. El servicio de no repudio puede involucrar los servicios de un TTP, pero dada la naturaleza descentralizada de *blockchain*, ese aspecto se debe lograr por consenso (Fang et al., 2020).

En *blockchain*, el no repudio involucra dos aspectos. Uno de ellos es que la información enviada no se puede negar, es decir si A envió un mensaje a B, A no puede negar que lo hizo. El otro aspecto es que B no puede afirmar que no lo recibió. Las firmas digitales en los sistemas que usan *blockchain* utilizan técnicas de cifrado asimétrico que son típicas de las ecuaciones de curva elíptica (Monika, Tomar, Kumar, & Kumar, 2020; Oswald, 2005) para garantizar el rechazo de la información.

El algoritmo de firma digital de curva elíptica (ECDSA) (Johnson, Menezes, & Vanstone, 2001) y el algoritmo de firma digital de la curva de Edward (EdDSA) (S. Josefsson & I. Liusvaara, 2017) son los dos esquemas de firma digital que se utilizan con frecuencia en *blockchain*. Ambos se basan en la dureza de la versión de curva elíptica del problema del logaritmo discreto. ECDSA funciona sobre una curva elíptica general y ahora se usa en Bitcoin y Ethereum, mientras que

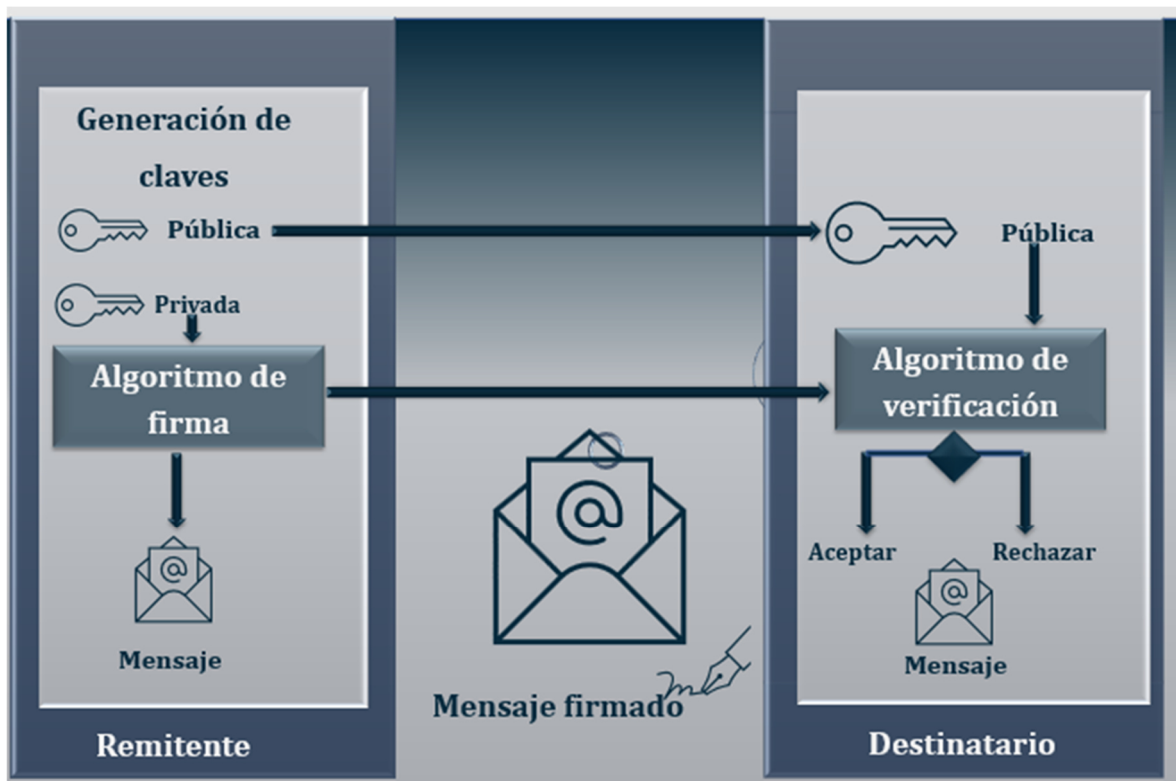
EddSA funciona sobre una curva de Edward y ahora se usa en otras implementaciones de *blockchain* como Naivecoin y Monero.

ECDSA garantiza la generación de firmas únicas e irrepetibles para cada conjunto de claves (privadas y públicas) generadas y la imposibilidad práctica de falsificar las firmas digitales. Esto realmente lo logra debido a que la potencia computacional necesaria para poder romperla está fuera de los límites actuales. ECDSA demuestra ser resistente a la falsificación en presencia de un ataque de mensaje elegido basado en un grupo genérico y la función *hash* resistente a colisiones (H. Mayer, 2016).

Todo esquema de firma digital tiene como mínimo los siguientes componentes:

- ▣ El algoritmo de generación de claves: Crea una clave denominada clave privada que se utiliza para firmar mensajes y otra llamada clave pública que es utilizada para validar si el mensaje ha sido firmado con la correspondiente clave privada.
- ▣ El algoritmo de firma: Es el encargado de producir una firma mediante el uso de la clave privada.
- ▣ El algoritmo de verificación: Toma una firma, un mensaje y una clave pública como entradas. Valida la firma del mensaje con la clave pública y devuelve un valor booleano indicando si es válida o no.

La Figura 37 muestra un esquema simplificado del proceso de firma electrónica en que se puede apreciar la generación de las claves por parte del remitente, el algoritmo de firma que toma la clave privada y el mensaje para generar un mensaje firmado (que está compuesto por la firma y el mensaje propiamente dicho), y por parte del destinatario se puede ver que tomando la clave pública del remitente, ejecuta el algoritmo de verificación para determinar si se acepta el mensaje al estar firmado correctamente o no se acepta el mensaje si hay un error de verificación de la firma.

Figura 37. Esquema simplificado de firma electrónica.

Fuente: Elaboración propia.

e. Las direcciones o seudónimos

Como se pudo ver en la sección anterior, la autenticidad de un mensaje puede ser validada mediante la utilización de la firma digital. Fuera de *blockchain*, el esquema de administración de la clave privada y pública se realiza utilizando un TTP por medio de PKI, en el que se realiza un acuerdo vinculante entre las respectivas identidades de las entidades interesadas, mediante una identificación (un documento de identidad, por ejemplo) y sus claves públicas. Para realizar la vinculación se registran y emiten certificados con una CA. El proceso de verificación de firma se traduce automáticamente en la verificación de la identidad del firmante en función del nivel de garantía de la vinculación. Entre más confiable sea la CA, mejor será esa garantía. Dado este esquema, la clave pública se puede ver como una identidad en estos escenarios (Lozupone, 2018).

Con *blockchain*, se adopta la idea de identidad descentralizada, con lo cual no es necesaria la utilización de una autoridad certificadora central. En este caso son los usuarios quienes generan los pares de claves y pueden generar tantas como quieran. Los *hashes* de las claves públicas que generan son denominados “direcciones” en el caso de Bitcoin. Estas identidades son seudónimos que han creado los propios usuarios (Chan et al., 2020).

2.4. Algoritmos de consenso

Cuando se trabaja con una *blockchain* descentralizada, surge el problema de como los nodos deben ignorar o agregar un bloque a la copia que tienen del libro mayor. Para este proceso se utiliza un algoritmo de consenso que tiene como objetivo buscar que la mayoría de la red acuerde la actualización del libro mayor global y prevenir ataques maliciosos (Gol, 2019).

El consenso se define aproximadamente como un acuerdo entre varias partes, pero en el caso de una red distribuida como *blockchain*, esto es un gran desafío, puesto que no hay un nodo central que garantice que haya libros mayores idénticos en cada uno de los nodos distribuidos. Por lo ello, es necesario la existencia de algún protocolo para garantizar la consistencia entre los nodos (Khettry et al., 2021).

Los mecanismos de consenso en *blockchain* ayudan a eliminar el problema del doble gasto (Chohan, 2018) y el problema de los generales bizantinos (Lamport, Shostak, & Pease, 1982). En el problema de los generales bizantinos, un grupo de generales que comanda una parte de un ejército bizantino rodea una ciudad. El ataque fracasaría si solo una parte de los generales atacara la ciudad. Los generales necesitan comunicarse para llegar a un acuerdo sobre si atacar o no. Sin embargo, puede haber traidores dentro de los generales. El traidor podría enviar diferentes decisiones a diferentes generales. Este es un entorno sin confianza. Lograr el consenso en este tipo de problemas es un desafío (Monrat, Schelén, & Andersson, 2019).

Los algoritmos de consenso tienen como objetivos (Gol, 2019):

- 👉 **Llegar a un acuerdo:** Se deben reunir todos los acuerdos del grupo de la mejor manera posible.
- 👉 **Colaboración:** Cada uno de los miembros del grupo trabaja para lograr un mejor acuerdo dirigido a los intereses de todo el grupo.
- 👉 **Cooperación:** Cada miembro del equipo dejará de lado sus propios intereses y trabajará en equipo.
- 👉 **Igualdad de derechos:** Igualdad para todos los miembros en la votación.
- 👉 **Participación:** Todos los miembros de la red necesitan participar en la votación. Nadie se quedará fuera o puede quedarse fuera sin un voto.
- 👉 **Actividad:** Todos los miembros tienen la misma responsabilidad y son igualmente activos.

Además de cumplir con los objetivos mencionados anteriormente, el libro mayor necesita un algoritmo de consenso eficiente y seguro, que debe ser tolerante a fallos y garantizar que:

- ✎ Todos los nodos almacenen simultáneamente una *blockchain* idéntica.
- ✎ No depender de una autoridad central para evitar que adversarios malintencionados interrumpan la coordinación del algoritmo de consenso.

Existen muchos algoritmos de consenso disponibles. Es importante comprender sus mecanismos, propiedades y propósitos a la hora de elegir una plataforma de *blockchain* adecuada que se adapte a la solución del problema que se plantea. Por esa razón, para determinadas aplicaciones de *blockchain*, una plataforma se adapta mejor que otra y un algoritmo de consenso es más apropiado que otro (Bano et al., 2019). Con el objetivo de lograr más velocidad en el consenso, se han adoptado grafos acíclicos dirigidos en lugar de cadenas puras de bloques, pero aún se debate su seguridad relativa (Benčić & Žarko, 2018).

a. Propiedades de los algoritmos de consenso

Se hay identificado diferentes propiedades presentes en los diferentes algoritmos de consenso que facilitan la clasificación de los mismos (Bano et al., 2019; Cachin & Vukolić, 2017; Nijssse & Litchfield, 2020). Estas propiedades permiten también la evaluación y valoración de la utilización de un algoritmo de consenso en una determinada aplicación basado en sus propiedades (Ferdous et al., 2021). Se pueden catalogar de forma general cuatro grupos de propiedades muy diferenciadas:

- ✓ Propiedades estructurales.
- ✓ Propiedades de recompensa y bloque.
- ✓ Propiedades de seguridad.
- ✓ Propiedades de desempeño.

Propiedades estructurales

Estas propiedades definen como los diferentes nodos en la red de *blockchain* están estructurados para participar dentro del algoritmo de consenso. Determinar la naturaleza de esa participación está en el ámbito de estas propiedades, ya que hay que definir, entre otras cosas, si todos los nodos participan, cómo se organizan los participantes, qué mecanismos de participación existen, si se eligen al azar o ya están fijos, si todos los nodos son igual de importantes o no, cómo puede un nodo unirse a la participación y cómo se realiza dicha elección (Ferdous et al., 2021).

Las propiedades estructurales (Figura 38) se pueden dividir en tres aspectos:

- ☞ **Tipo de nodo:** En este aspecto, el algoritmo debe definir si hay alguna clasificación especial con los nodos que participan o si cumplen algún rol en especial.
- ☞ **Tipo de estructura:** Esta propiedad se refiere a la manera como se organizan los nodos que participan. La organización puede ser simple si los nodos se organizan en un solo grupo, denominado comité, o múltiple si hay varios comités, en los cuales se definen o no reglas especiales para la incorporación de participantes.
- ☞ **Mecanismo subyacente:** Esta propiedad indica como se realiza la selección de los nodos participantes. Puede ser por votación, en la que se requiere intercambiar los resultados en la red antes de agregar el bloque a la cadena de bloques. Si un nodo desea agregar un bloque a su cadena, se debe verificar que al menos x (x como un umbral establecido) nodos estén de acuerdo con él (Pahlajani et al., 2019). En los mecanismos que están basados en una demostración, los nodos que se unen a la red necesitan resolver un problema criptográfico para obtener el derecho de agregar el bloque.

Figura 38. Propiedades estructurales de los algoritmos de consenso.

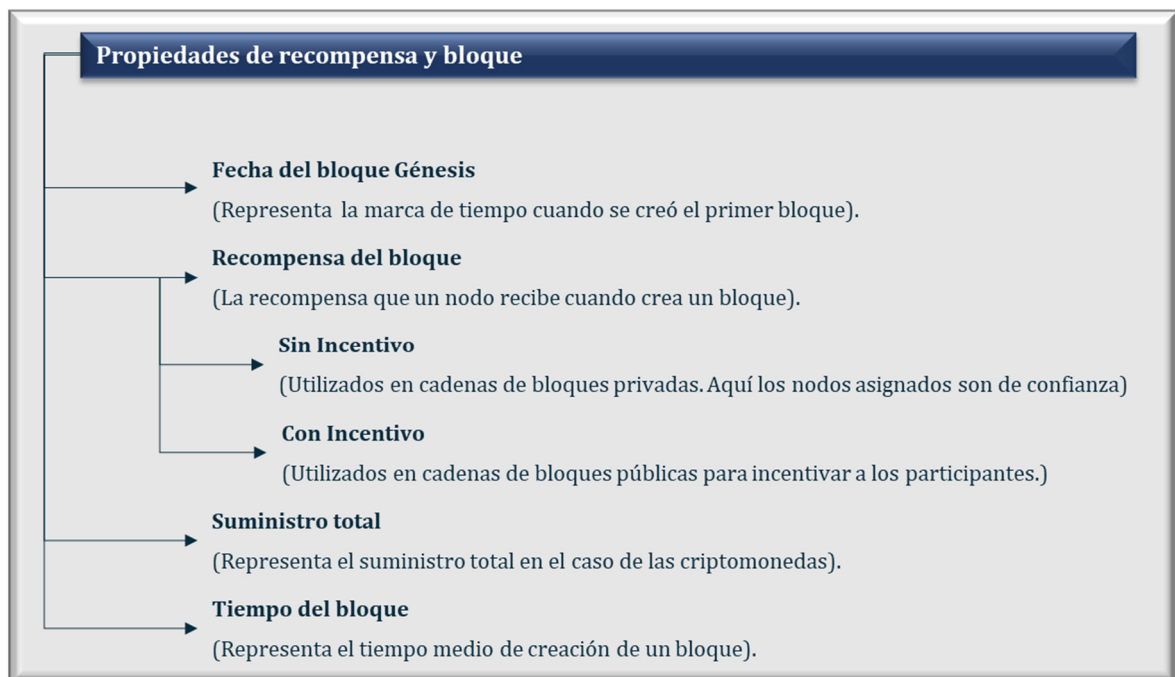


Fuente: Elaboración propia.

Propiedades de recompensa y bloque

Estas propiedades son métricas cuantitativas que ayudan a diferenciar diferentes sistemas o implementaciones de soluciones con *blockchain*. En especial las criptomonedas son las más interesadas en este tipo de propiedades. Aunque no caracterizan un algoritmo de consenso como tal, generan un impacto en cómo se desarrolla el consenso. Por ejemplo, en el caso de Bitcoin, el interés que genera la recompensa entre los mineros permite que se pueda alcanzar el consenso (Cachin & Vukolić, 2017). Estas propiedades están organizadas en la Figura 39 con sus respectivos comentarios.

Figura 39. *Propiedades de recompensa y bloque de los algoritmos de consenso.*



Fuente: Elaboración propia.

Propiedades de seguridad

Entre las propiedades de los algoritmos de consenso que más interés generan se encuentran las propiedades de seguridad a causa de que son los algoritmos que modifican los bloques de la cadena. Los estudios sobre la seguridad de los algoritmos de consenso se suelen centrar en cuatro aspectos:

- ⊕ La **autenticación de los nodos**: En este caso surge el interrogante de si los nodos que participan están realmente autenticados, dando respuesta a la preocupación de si son conocidos y que tan confiables son.
- ⊕ El **no repudio**.
- ⊕ La **resistencia a la censura**.
- ⊕ Los **vectores de ataque** que el algoritmo puede soportar.

De este último vale la pena destacar tres vectores de ataque que un algoritmo de consenso debe soportar obligatoriamente:

- 1 Tolerancia al adversario**: Cuántos nodos bizantinos máximos pueden ser soportados (Praveen et al., 2021).
- 2 Protección Sybil**: En este tipo de ataque, un atacante puede duplicar su identidad para lograr tener ventajas ilícitas. En el caso de *blockchain*, el atacante podría crear o controlar tantos nodos como necesite para tener influencia sobre el consenso (Aggarwal & Kumar, 2021a; Douceur, 2002).
- 3 Resistencia a la DoS**: Implica que el algoritmo está diseñado para contrarrestar este tipo de ataque (Mirkin et al., 2020).

Propiedades de desempeño

Se utilizan cinco propiedades de desempeño para medir el rendimiento del algoritmo de consenso en la red distribuida de *blockchain* (Fan, Ghaemi, Khazaei, & Musilek, 2020):

- 1 Tolerancia a fallos:** Indica el máximo número de nodos fallidos que pueden ser tolerados. En caso de fallar algunos nodos, debe ser transparente la ejecución del consenso.
- 2 Rendimiento:** Se refiere al número de transacciones que el algoritmo puede procesar por segundo. Es esencial en el caso de las criptomonedas y es un tema recurrente en el mundo académico.
- 3 Escalabilidad:** Relacionada con el crecimiento de *blockchain*. Considera si se puede asegurar que *blockchain* puede crecer en tamaño y funcionalidad sin perder el desempeño el consenso.
- 4 Latencia:** Cómo se comporta el tiempo de respuesta desde que se envía una transacción hasta que se llega a un consenso. Si el sistema de cadena de bloque utilizado es crítico, esta propiedad es clave para el desempeño de todo el sistema.
- 5 Consumo de Energía:** Se refiere a la cantidad de energía necesaria para el funcionamiento óptimo del consenso. Esta propiedad ha generado mucha controversia con los primeros algoritmos de consenso desarrollados, como el algoritmo de PoW utilizado en Bitcoin.

b. Algoritmos de consenso más utilizados

En los sistemas de *blockchain* existentes, hay cuatro mecanismos principales de consenso que suelen ser utilizados (Xiaoqi Li, Jiang, Chen, Luo, & Wen, 2020):

- 1 Prueba de trabajo (PoW) (Meneghetti, Sala, & Taufer, 2020).
- 2 Prueba de participación (PoS) (Saleh, 2021).
- 3 Tolerancia a fallas bizantinas práctica (PBFT) (M. Castro & Liskov, 1999).
- 4 Prueba de participación delegada (DPoS) (Skh Saad & Raja Mohd Radzi, 2020).

A continuación, se describen estos cuatro algoritmos dada su relevancia y uso en *blockchain*. Posteriormente, se explicarán otros mecanismos de consenso propuestos para algunas aplicaciones de *blockchain* específicas o que están en estudio.

Prueba de trabajo (PoW)

Un mecanismo de PoW implica dos tipos de nodos: probador y verificador. El probador realiza una tarea computacional de uso intensivo de recursos con la intención de lograr un objetivo y lo presenta a un verificador o un conjunto de verificadores para su validación. Un verificador requiere menos recursos que un probador (Ferdous et al., 2021). Con este objetivo, en 1993, Dwork y Naor propusieron utilizar PoW para combatir el correo no deseado. La idea central era obligar al remitente de un correo electrónico a resolver un rompecabezas matemático que requería muchos recursos y la solución tenía que adjuntarse al correo como prueba de haberla realizado. A la recepción del correo, el destinatario lo aceptaba solo si la solución se podía verificar con éxito.

PoW aplicado a *blockchain* fue propuesto por Nakamoto (2008) para resolver el problema de los generales bizantinos en *blockchain* de Bitcoin. PoW es uno de los algoritmos más utilizados en *blockchain* relacionadas con criptomonedas, representando más del 90% de la capitalización total de las criptomonedas existentes en 2016 (Gervais et al., 2016). En el momento actual, PoW domina la descentralización (Irresberger et al., 2020).

PoW requiere que los nodos de la red resuelvan funciones matemáticas complejas unidireccionales antes de que puedan agregar bloques a la cadena de bloques. El proceso de encontrar pruebas correctas que resuelven funciones criptográficas se llama minería, y los nodos o individuos que participan en este proceso se denominan mineros (Gamage, Weerasinghe, & Dias, 2020).

Este algoritmo de consenso para los participantes de *blockchain* implica (Meneghetti et al., 2020):

- ✎ Un desafío C que los mineros deben resolver para crear nuevos bloques.
- ✎ Un criterio para distinguir los bloques adecuados de los obsoletos.

Con el objetivo de resolver el desafío C en PoW, cada nodo calcula el valor *hash* del encabezado del bloque. El encabezado contiene valores “nonce”, que al ser modificado condiciona el *hash* del bloque. Es por lo que el minero a menudo cambia ese valor “nonce” para obtener diferentes valores *hash*. El valor calculado tiene que cumplir el objetivo propuesto. En Bitcoin, el valor *hash* debe empezar por una cantidad de ceros ya predefinidos. El nodo (probador) al resolver el desafío lo transmite inmediatamente a otros nodos (verificadores) y esos nodos verificarían la exactitud del valor *hash*. Tras la validación, otros mineros agregan este nuevo bloque a la cadena de bloques (Khettry et al., 2021).

Para resolver el desafío, los probadores deben utilizar bastantes recursos para llegar a una solución, de tal modo que probabilísticamente, los probadores que tienen la mayor parte de la potencia de la CPU tienen más oportunidades de encontrar la respuesta en el algoritmo de PoW. Como recompensa, al poner a disposición ese poder computacional y la electricidad para resolver el desafío, reciben monedas recién acuñadas, que también se denominan recompensa en bloque.

Los mineros (probadores) compiten entre sí para resolver ese desafío, que dinámicamente se va volviendo cada vez más complejo, lo que asegura que el intervalo de bloque sea consistente a medida que mejora el poder computaciones de los probadores. La dificultad de PoW es una característica intrínseca de la seguridad, ya que evita inundar la red con mensajes y da la oportunidad a los nodos honestos de estar unificados (Kiayias & Panagiotakos, 2015).

Debido al gran esfuerzo requerido para calcular el valor *hash* del bloque, y sabiendo la cantidad de recursos que consume, las *blockchain* basadas en PoW tienen un posible riesgo de centralización para los probadores (Gamage et al., 2020). En las criptomonedas de PoW pura, la seguridad depende totalmente de los mineros y los ingresos de la minería de toda la red. La suma de todos los ingresos de los mineros es una medida directa de la competencia entre las diferentes criptomonedas basadas en PoW (King, 2013).

Puede darse el caso de que más mineros resuelvan el desafío antes de que se notifiquen las soluciones a los demás nodos o que mineros deshonestos intentar generar ataques maliciosos (como un doble gasto). En esas situaciones, se generaría una bifurcación (*fork*) (Meneghetti et al., 2020). Con el objetivo de resolver ese tipo de incidencias, se utilizan 3 criterios para seleccionar de forma unificada la mejor ramificación:

- ✱ La bifurcación más antigua gana: Esta regla indica que, en presencia de bifurcaciones, un minero honesto agregará bloques solo en la ramificación que se creó primero (la marca de tiempo más antigua).
- ✱ La bifurcación más larga gana: Esta regla se basa solo en la geometría de la cadena. La cadena principal es simplemente la más larga, así todos los bloques no incluidos en esta cadena deben considerarse obsoletos y descartarlos.
- ✱ GHOST: Es actualmente la estrategia que se considera más segura (Sompolinsky & Zohar, 2013), y considera como cadena principal aquella donde más potencia computacional ha gastado.

En particular, un algoritmo de PoW en una *blockchain* tiene dos propósitos críticos (Ferdous et al., 2021):

- ⚙ Un mecanismo de disuasión contra el ataque de Sybil. La inversión monetaria de un nodo probador en PoW es muy alta, de manera que para que un ataque Sybil sea exitoso, la inversión del atacante será proporcional al número de identidades Sybil, superando a nivel económico la ventaja obtenida con ese tipo de ataque (Baza et al., 2020).
- ⚙ El mecanismo PoW se usa para lograr el consenso distribuido requerido cuando ocurre una bifurcación en una *blockchain* (Finlow-Bates, 2018).

Prueba de participación (PoS)

En PoS, la creación de un bloque depende de la participación retenida en lugar de la potencia computacional (Shijie Zhang & Lee, 2020). PoS no necesita ajustar un valor “nonce” muchas veces, debido a que la clave está en que los probadores resuelvan el problema con la cantidad de monedas que tienen en posesión, según la siguiente formula:

$$SHA256(\text{marca de tiempo, hash anterior...}) < \text{Objetivo} \times \text{moneda}.$$

PoS es un protocolo de consenso de ahorro de energía que aprovecha una forma del incentivo de la moneda interna en lugar de consumir mucha potencia computacional para llegar a un consenso. Ante esta situación, cuantas más monedas tenga el probador, mayores serán las probabilidades para crear un bloque. En PoS se parte del hecho que las personas que poseen más dinero tienen menos probabilidades de atacar la red (Khettry et al., 2021). Los nodos del sistema se eligen arbitrariamente y, en el caso de que el nodo tenga una cantidad particular de monedas almacenada en su billetera, estará en condiciones de ser seleccionado (Gol, 2019).

La idea central de PoS se desarrolla en torno al concepto de demostración de riqueza por parte del nodo. Para este fin deben bloquear una cierta cantidad de sus monedas, llamada participación, en una cuenta de depósito como garantía para poder participar en el proceso de creación del bloque. El nodo que custodia su participación se conoce como parte interesada, líder, falsificador o acuñador (*minter*) en la terminología de PoS (Ferdous et al., 2021). Si el acuñador tiene un comportamiento inadecuado, puede perder el dinero.

En el momento en el que un nodo interesado deposita su participación, se convierte en miembro de un grupo exclusivo, cuyos miembros pueden participar en el proceso de creación del bloque. Si un nodo tiene la oportunidad de crear un nuevo bloque, recibirá una recompensada de dos formas diferentes: puede cobrar las tarifas de las transacciones del

bloque o recibir una cierta cantidad de monedas como un tipo de interés en base a su participación.

Se ha argumentado que incentivar junto con penalizar, puede proporcionar un nivel de seguridad similar al que proporciona PoW, además de ofrecer otros beneficios para *blockchain*:

- ⌚ La eficiencia energética que la hace sea más sostenible a largo plazo.
- ⌚ La mitigación de la centralización que está menos afectada por el fenómeno de las economías de escala.
- ⌚ La seguridad económica explícita que disuade cualquier ataque de mala conducta.
- ⌚ La generación de un suministro inicial a los nodos para su participación haciendo uso de un proceso de pre minería o un proceso de transición PoW a PoS, un proceso que recompensa la creación de un bloque ofertando una tarifa de transacción dentro del bloque acuñado o por medio de una tasa de interés, y un proceso de selección del siguiente nodo que creará el bloque por medio de un enfoque encadenado, BFT o delegado (Ferdous et al., 2021).

PPcoin fue la primera criptomoneda en utilizar PoS haciendo uso del concepto “edad de la moneda” (*coin-age*) al introducirlo en la fórmula para resolver el desafío propuesto en PoS (King & Nadal, 2012). Además de PPcoin, muchas otras criptomonedas han adoptado PoS, tales como BlackCooIn, Nxt (Gamage et al., 2020) y Casper de Ethereum entre otras (Buterin, Reijersbergen, Leonardos, & Piliouras, 2020).

Tolerancia a fallas bizantinas práctica (PBFT)

La Tolerancia práctica a falla bizantina (PBFT) es un protocolo de tolerancia a fallos bizantino con baja complejidad de algoritmo y alta practicidad en sistemas distribuidos (M. Castro & Liskov, 1999). Este protocolo fue desarrollado con el objetivo de llegar a un consenso en presencia de nodos bizantinos, donde los nodos pueden estar bajo el control de un adversario y actuar maliciosamente contra el objetivo común de llegar al consenso. En un sistema con n nodos, si PBFT puede tolerar $f < n/3$ nodos bizantinos, se considera óptimo (Cachin & Vukolić, 2017).

PBFT y otros protocolos de consenso emplean el concepto de replicación para lograr resistencia contra fallas, más no escalabilidad (Bano et al., 2019). Similar a cualquier protocolo de consenso tolerante a choques, estos protocolos también utilizan el concepto de puntos de vista / épocas, donde se elige un líder en cada punto de vista para ordenar los mensajes a transmitir, y se supone que otros nodos honestos siguen las instrucciones del líder (Ferdous et al., 2021).

PBFT contiene cinco fases para lograr ese consenso: solicitud, preparación previa, preparación, compromiso y respuesta. PBFT garantiza que los nodos mantengan un estado común y toman una acción consistente en cada ronda de consenso (Shijie Zhang & Lee, 2020). Los participantes de la red se conocen y pueden distinguir cuál es importante y cuál no. De esta forma, un nodo se postula como el siguiente nodo solo si dos tercios del total de nodos votan a favor de él, y es entonces cuando el nodo envía su solicitud a todos los demás nodos de la red (Khettry et al., 2021).

Algunas criptomonedas como EOS utilizan un protocolo de consenso combinado con PBFT para la validación y creación de los bloques con el objetivo de reducir el tiempo necesario para cada ronda del consenso (Larimer, 2017). Otras soluciones como Hyperledger Fabric, una plataforma para soluciones de libro mayor distribuido, utilizan PBFT como su algoritmo de consenso (Sajana M; Sethumadhavan, M, 2018). PBFT es utilizado también para satisfacer las necesidades de organizaciones como los bancos, los sistemas de nómina o las cadenas de suministro (Khettry et al., 2021).

Prueba de participación delegada (DPoS)

PoS resuelve los dos grandes problemas de PoW: el alto consumo de energía y la baja capacidad de transacciones. Pero la descentralización sigue siendo un problema para los dos algoritmos.

DPoS permite que los nodos que tienen participación voten utilizando puntuaciones de reputación u otros mecanismos para elegir los nodos que se encargarán de crear los bloques (He, Tang, & Wang, 2020). Con este sistema de votación, los interesados otorgan el derecho de crear bloques a los delegados que apoyan, en lugar de crearlos ellos mismos, reduciendo así su consumo de energía computacional a 0. Si los delegados no pueden generar bloques en su turno, serán despedidos y los interesados seleccionarán nuevos nodos para reemplazarlos (Shijie Zhang & Lee, 2020).

Los líderes o testigos, que son los que proponen un bloque, se seleccionan en cada ronda. La forma en que se elige a ese líder depende del sistema respectivo. El líder recibe recompensas por crear cada bloque y, por el contrario, recibe una penalización y es eliminado del grupo de validadores si realiza alguna actividad maliciosa. Todos los delegados que poseen alguna participación (como en DoS) compiten entre sí para conseguir ser el próximo líder. Cada delegado puede ofrecer diferentes niveles de incentivos a sus votantes, como, por ejemplo, distribuir una cierta fracción de su recompensa entre ellos (Ferdous et al., 2021).

Los validadores de bloques ejecutan nodos completos, aunque no todos los validadores son productores de bloques. Cualquiera puede convertirse en validador ejecutando un nodo completo. Debido a la lista fija de productores de bloques, la DPoS permite generar un nuevo bloque a una tasa fija con requisitos computacionales mínimos, lo que la hace más escalable que PoW y PoS (Shijie Zhang & Lee, 2020), en virtud de que puede procesar más transacciones en mucho menos tiempo y casi sin costo en comparación con los otros algoritmos.

Si la mayoría de los productores fallan, es decir no pueden crear bloques, DPoS puede seguir funcionando, ya que la comunidad puede votar para reemplazar a los productores fallidos hasta que se pueda reanudar la participación plena. DPoS sacrifica el concepto de descentralización en *blockchain* para lograr un alto rendimiento de transacciones, lo que acarrea muchas críticas por parte de algunos autores, que la consideran como un mecanismo de consenso inadecuado para *blockchain* que gestionan transacciones (Gamage et al., 2020).

Además de una menor descentralización y, dada la velocidad de procesamiento de las transacciones, puede generar muchas más fallas y algunos desafíos de seguridad, siendo más propenso a ataques (Skh Saad & Raja Mohd Radzi, 2020). Algunos sistemas de *blockchain*, como EOS, BitShares, Steemit y Lisk, usan este algoritmo de consenso (Gamage et al., 2020).

Prueba de investigación (PoR)

La Prueba de investigación (PoR) es un enfoque que combina dos pruebas: la prueba PoS con la prueba de Infraestructura Abierta de Berkeley para la Computación en Red (BOINC) (Anderson, 2020). BOINC es una plataforma de computación en rejilla (*grid*) utilizada por investigadores de diferentes dominios, que permite explotar los recursos informáticos inactivos de las computadoras personales en todo el mundo. Con la prueba de BOINC, un investigador tiene que demostrar su contribución al trabajo de investigación de BOINC.

El mecanismo PoR es aprovechado por Gridcoin (2018), una criptomoneda que cualquiera puede ganar compartiendo sus recursos informáticos con el proyecto BOINC. El mecanismo de PoS utilizado aquí es similar al algoritmo tradicional de PoS. Cualquiera puede convertirse en un acuñador, conocido como inversor (*investor*), al poseer una cierta cantidad de Gridcoin, y participar en el proceso de acuñación. Además, otros usuarios, conocidos como investigadores (*Researchers*), también pueden participar en el proceso. Asimismo, un inversor puede ser un investigador y, por lo tanto, puede aumentar la cantidad de monedas ganadas en la cuadrícula (Kondru, Saranya, & Chacko, 2021).

Prueba de quemado (PoB)

La Prueba de quemado o grabación (PoB) es un algoritmo de consenso propuesto por Ian Stewart en el que los mineros necesitan quemar sus monedas para participar en el proceso de consenso (minería) (Karantias, Kiayias, & Zindros, 2020), como alternativa a PoW en la que los mineros deben invertir en la construcción de una plataforma minera para poder participar. Quemar monedas significa enviar monedas a una dirección de la que no se tiene la clave privada y, por lo tanto, nunca se puede usar. La cantidad que se quema tiene una correlación positiva con la posibilidad de ser seleccionado para participar en el consenso. Slimcoin es una criptomoneda que utiliza la idea de PoB en combinación con PoW y PoS (P4Titan, 2014), creando así un mecanismo de consenso híbrido.

Prueba de velocidad de participación (PoSV)

La Prueba de velocidad de participación (PoSV) es una alternativa a PoW y PoS para asegurar una red P2P y confirmar transacciones de Reddcoin, una criptomoneda creada específicamente para facilitar las interacciones sociales en la era digital (L. Ren, 2014). Utiliza un mecanismo llamado velocidad de la participación, junto con un algoritmo de PoS tradicional. Conceptualmente, la velocidad de la participación refleja la noción de la velocidad del dinero, un término económico que mide la frecuencia del flujo de dinero dentro de la sociedad. La velocidad de participación pretende aumentar el flujo de participación durante el mecanismo de consenso de PoS.

Esto se puede lograr si se anima a los interesados a participar activamente en el mecanismo de consenso utilizando su criptomoneda, en lugar de mantener sus monedas fuera de línea. La idea es contrarrestar la falta de participantes en PoS, de forma que estos puedan contribuir en la verificación de las transacciones en la red que use PoSV. El mecanismo utilizado por PoSV para animar la participación es la utilización de una función de envejecimiento de monedas no lineal, de modo que los que participan en el consenso más activamente ganarán más que los que no lo hacen. En particular, la ganancia es más rápida en los primeros días y semanas que en las semanas posteriores (Ferdous et al., 2021).

Prueba de importancia (PoI)

En PoS, cuantas más monedas tiene el probador en la cuenta, más gana, lo que significa que los más ricos se vuelven más ricos y se obtiene más incentivo por acumular que por gastar. Con el objetivo de solucionar este problema se creó el consenso de Prueba de importancia (PoI),

introducido por la una plataforma de *blockchain* New Economy Movement (NEM) con una criptomoneda asociada llamada XEM (Kraus, Kraus, & Manzhura, 2019).

De la misma manera que en PoS, los nodos necesitan “ceder” una cantidad de moneda, lo que les da una calificación o puntaje para participar en el consenso. Pero en la PoI, este puntaje incluye más variables. Por ejemplo, a los nodos que acumulen más de 10.000 XEM se les asigna la misma importancia, y con monedas de 9B XEM, la cantidad de nodos no puede superar los 900K. A medida que pasa el tiempo y varían la cantidad de transacciones de XEM, la importancia de los nodos cambia (Ferdous et al., 2021).

Principalmente se tiene en cuenta:

\$ **Transferencias netas:** Cuánto se ha gastado en los últimos 30 días. Las transacciones más recientes tienen más ponderación.

\$ **Cantidad de dinero adquirido:** Con el propósito de crear bloques.

\$ **Nodos de clúster:** Las cuentas que forman parte de clústeres de actividad interconectados tienen un peso ligeramente mayor que los centros (que vinculan los clústeres, pero no forman parte de ellos).

En NEM, la importancia de una cuenta depende solo de las transferencias netas de XEM desde esa cuenta. La puntuación de importancia premia la difusión de XEM, mientras que la acumulación será penalizada (Y. Lai, 2018).

Prueba de cooperación (PoC)

La Prueba de cooperación (PoC) es un algoritmo de consenso introducido por la criptomoneda FairCoin (König, Duran, Fessler, & Alton, 2018). Este algoritmo utiliza unos tipos especiales de nodos denominados nodos validados cooperativamente (CVN). Estos CVN funcionan como los validadores en DPoS, ya que pueden crear bloques en FairCoin por medio de PoC, pero con la diferencia que cada nodo CVN está autenticado por su correspondiente identificador de FairCoin.

Es de confianza porque cumple con una serie de reglas y requisitos técnicos impuestos por la comunidad. Los nodos candidatos deben participar en actividades de la comunidad, como ejecutar un nodo local o contribuir a problemas técnicos o de gestión con Faircoin. Estas actividades deben de ser confirmadas por mínimo dos miembros activos de la comunidad.

El nodo candidato también debe cumplir una serie de requisitos técnicos de disponibilidad y de hardware específico para la generación de las firmas.

En cuanto al algoritmo, los bloques en Faircoin se crean de forma rotatoria cada tres minutos por uno de los CVN. Para crear un nuevo bloque, es necesario seleccionar un CVN utilizando un mecanismo de votación determinista llevado a cabo individualmente por cada CVN de la red (Ferdous et al., 2021; König et al., 2018).

Tolerancia a fallas bizantinas simplificada (SBFT)

La Tolerancia a fallas bizantinas simplificada (SBFT) es un sistema de *blockchain* con autorización tolerante a fallas bizantinas de última generación que aborda los desafíos de escalabilidad, descentralización y replicación geográfica a escala mundial. SBFT está optimizado para la descentralización y puede manejar fácilmente más de 200 réplicas activas en una implementación a escala mundial real (Golan Gueta et al., 2019).

En SBFT, un bloque puede reunir todas las transacciones, agruparlas consecuentemente en otro bloque y, finalmente, validarlas todas. El generador del bloque aplica las mismas reglas que todos los nodos siguen para validar todas las transacciones. Una vez creado el nuevo bloque, un firmante de bloque puede validarlos y agregar su propia firma. Por eso, si a alguno de los bloques le falta alguna firma será rechazado (Gol, 2019). SBFT se basa en PBFT, pero mejora la cantidad de mensajes que tienen que ser enviados entre los nodos delegados en PBT para lograr el consenso, ya que los nodos son agrupados en delegaciones y los nuevos bloques son mantenidos por una delegación específica (Golan Gueta et al., 2019).

Prueba de actividad (PoA)

La Prueba de autoridad (PoA) utiliza lo mejor de PoW y PoS (Gol, 2019). El consenso se inicia como en PoW, con varios nodos compitiendo entre sí para encontrar un nuevo bloque. Una vez se encuentra un nuevo bloque, se empieza a utilizar PoS, y el bloque contiene solo un encabezado y la identificación de recompensa del nodo que generó el bloque. Inmediatamente después de la generación del bloque, se selecciona un nuevo grupo aleatorio de validadores de la red según los detalles del encabezado, los cuales están obligados a validar o firmar el nuevo bloque.

Como en PoS, entre mayor sea la aportación del validador, más posibilidades tendrá de tener el rol de firmante. Los validadores seleccionados deben firmar el bloque, cambiando el estado del bloque a “completo”, se añade a *blockchain* y las transacciones empiezan a registrarse en dicho bloque. Si por alguna razón, algunos de los firmantes seleccionados no están disponibles para firmar el bloque, el proceso pasa al siguiente bloque ganador con un nuevo conjunto de validadores, hasta que el bloque está completo (Bentov, Lee, Mizrahi, & Rosenfeld, 2014).

En este algoritmo, las recompensas son divididas entre el generador del bloque (minero) y los diferentes validadores que contribuyeron para firmar el bloque. Dado que el sistema PoA se basa en PoW y PoS, es criticado debido a que es necesaria la utilización de muchos recursos de energía para extraer bloques en la fase de PoW. Por otra parte, acaparadores de monedas pueden generar centralización de poder a la hora de firmar y acumular recompensas (Supreet, Vasudev, Pavitra, Naravani, & Narayan, 2020).

Prueba de publicación (PoP)

La Prueba de publicación (PoP) se usa en Bitcoin para verificar si en una determinada fecha y hora se ha realizado una publicación (Aggarwal & Kumar, 2021d). Al realizar este proceso de consenso, es necesario codificar un *hash* de un determinado texto sin ningún formato dentro de una *blockchain* de Bitcoin.

El objetivo es autenticar que cierta información fue publicada en una fecha determinada, un concepto denominado sellado de tiempo (Tschorsch & Scheuermann, 2016). Este mismo concepto más específico es aplicado con CommitCoin, donde se genera una dirección Bitcoin que codifica la información de un documento respectivo (J. Clark & Essex, 2012).

Tolerancia a fallas bizantinas delegada (dBFT)

La Tolerancia a fallas bizantinas delegada (dBFT) fue desarrollado por el equipo de NEO (2017). NEO es una plataforma de *blockchain* impulsada por la comunidad que utiliza la tecnología *blockchain* y la identidad digital con el fin de digitalizar activos y gestionarlos utilizando contratos inteligentes (Elrom, 2019). NEO es similar en muchos aspectos a Ethereum. dBFT es una variante de PBFT, pero se han modificado el modelo de red (de Cliente/Servidor a P2P), la regla de permiso (de fijo a dinámico) y procedimiento de aplicación (de 3 fases a 2 fases) (Q. Wang et al., 2020).

Con dBFT se utilizan tres tipos de nodos: **delegados, orador y los nodos ordinarios**. Si se representara la red como un país, los ciudadanos serían los nodos ordinarios, los representantes, los delegados; y los políticos, los oradores que deben garantizar que todo vaya bien. Los nodos ordinarios son esencialmente poseedores de tokens NEO y mantienen un interés en *blockchain* mediante la votación. Los delegados son los nodos elegidos por los nodos ordinarios para presentar solicitudes a los nodos oradores. Los oradores son elegidos al azar para seguir las solicitudes de los nodos ordinarios.

Es un algoritmo de finalidad perfecta, donde todas las transacciones están 100 por ciento confirmadas después de la primera confirmación. La *blockchain* no puede bifurcarse con dBFT y está totalmente centrado en la empresa, ya que tiene en cuenta los casos de uso regulatorios y comerciales (Holbrook, 2020).

Como desventajas, tiene una alta centralización debido al bajo número de nodos de la red NEO y los delegados están controlados por el consejo de NEO. En caso de desacuerdo entre los miembros, la red NEO no se puede bifurcar, lo que puede ser una preocupación si ocurre un desacuerdo entre los miembros.

El dBFT utiliza tecnología de identidad digital, lo que significa que los delegados pueden ser una persona física o institución, por lo tanto, pueden estar sujetos a regulación judicial y la propiedad de los activos se puede congelar, revocar, heredar, recuperar y transferir. Esta regulación facilita el registro de activos financieros compatibles en la red NEO (Holbrook, 2020).

Prueba de capacidad

La Prueba de capacidad es un algoritmo de consenso utilizado para trazar. Mientras que en PoW los mineros utilizan la potencia computacional para resolver un desafío, en la Prueba de capacidad, las soluciones se almacenan previamente en memoria, entonces los mineros utilizan estos datos para dibujar un gráfico. A este proceso se le denomina trazado. Los mineros que tienen más capacidad de almacenamiento tienen más probabilidad de participar en el proceso de consenso (Aggarwal & Kumar, 2021d).

Prueba de peso

Tanto la Prueba de peso como PoS consideran la participación, pero la Prueba de peso además añade otros factores, que se consideran los factores ponderados (Gol, 2019). Algunas criptomonedas como Algorand (J. Chen & Micali, 2019), Filecoin (Benet & Greco, 2017) y Chia (Cohen & Pietrzak, 2019) implementan la prueba de peso. Filecoin considera la cantidad de información como un sistema de archivos interplanetario (IPFS) (Baumgart & Mies, 2007) que un usuario puede almacenar y pondera ese problema. Otras ponderaciones incluyen pruebas de reputación o pruebas de espacio-tiempo.

Prueba de tiempo transcurrido (PoET)

La Prueba de tiempo transcurrido (PoET) es un algoritmo de consenso donde la elección del nodo minero es aleatoria, basándose en el tiempo que los mineros han esperado para la creación del bloque. El tiempo de espera aleatorio es justo para todos los nodos de la red,

asegurando que cada participante tenga una oportunidad justa de proponer un nuevo bloque y participar en el proceso de votación. El nodo cuya espera termina primero produce un nuevo bloque (Aggarwal & Kumar, 2021d).

Basado sobre la plataforma de computación *Software Guard Extensions* (SGX), Intel propuso PoET para *blockchain*. PoET utiliza el conjunto de instrucciones de SGX, que se ejecutan en un entorno privado del resto de las aplicaciones para dar mayor seguridad u confianza. Existen preocupaciones debido a que PoET depende de SGX y este último podría ser secuestrado (Van Bulck et al., 2019). Hyperledger Sawtooth, una solución empresarial para la construcción y ejecución de *blockchains*, utiliza el algoritmo de consenso PoET, que aprovecha el SGX de Intel para implementar su sistema de consenso.

PoET es un consenso *plug-and-play*, de bajo consumo de energía. que resuelve el problema de selección aleatoria de líderes sin requerir muchos recursos ni incentivos. Entre sus desventajas, destaca la falta de portabilidad que ofrece debido a la dependencia de los requisitos de *software* y *hardware* patentados de Intel y que tiene unos costes más altos que otras plataformas de *blockchain* (Holbrook, 2020).

Prueba de autoridad (PoAT)

La Prueba de autoridad (PoAT) ha sido propuesta como un mecanismo de consenso tolerante a fallas bizantinas que está integrado con los protocolos de Ethereum. En PoAT, un subconjunto de nodos se designan como autoridades de validación y cooperan para mantener un funcionamiento estable de *blockchain* (Kolb et al., 2020). El nodo validador actúa como líder temporalmente, agrega las transacciones que van apareciendo en la red y propone el siguiente bloque a añadir en la cadena. El bloque será aceptado solo si la mayoría de validadores lo aprueba.

El consenso de PoAT es simple, rápido y evita el esfuerzo computacional de PoW. Ya que toda la confianza recae sobre los validadores y no existen ni incentivos o penalizaciones, no es adecuado en *blockchain* pública, pero es muy atractiva en un modelo autorizado donde los validadores son fijos y tienen un interés personal en el éxito de *blockchain*. PoAT ha sido implementado en dos protocolos diferentes de Ethereum del cliente Geth: el protocolo Clique y Aura (Ekparinya, Gramoli, & Jourjon, 2020). Otra implementación de PoAT se puede encontrar en el marco de trabajo CCF conocido como Coco (Microsoft, 2017).

Prueba de participación arrendada (LPoS)

La Prueba de participación arrendada (LPoS) fue introducido en Estados Unidos por la plataforma Waves y es un algoritmo ahorrador de energía, como PoS (Dominguez Perez, Ibarra, Alejandro, Rumayor, & Lara-Alvarez, 2020). Basada en PoS, la LPoS cambia el sistema de participación proporcionando a los usuarios con menos recursos la opción de participar activamente en el proceso de consenso. A cambio de esa participación, la recompensa se distribuye porcentualmente entre los nodos participantes.

Los participantes se pueden unir al depositar solidariamente sus monedas en un nodo. Esto permite mantener la red con éxito, ya que posibilita que el sistema forme una comunidad centralizada en una plataforma descentralizada. Con este sistema, los pequeños propietarios pueden tener una probabilidad de participación, arrendarán sus monedas a la red y aprovecharán el beneficio obtenido del consenso. El propósito principal de la plataforma Waves era ayudar a los pequeños inversores y esto se ha logrado mediante el algoritmo de consenso proporcionando transparencia (Gol, 2019).

2.5. Evolución de *blockchain*

Blockchain ha ido evolucionando conforme han ido surgiendo nuevas plataformas que satisfacen los requerimientos de las aplicaciones de educación, logística, cuidado de la salud, robótica, entre otras. Dicha evolución ha sido clasificada en cuatro capas o niveles, que se describirán en esta sección (Bhutta et al., 2021).

a. Divisas basadas en *blockchain*, denominado Blockchain 1.0

La primera aplicación de la tecnología *blockchain* fueron las criptomonedas, siendo éstas un medio de intercambio digital que hace uso de *blockchain* para su funcionamiento. Este medio digital de intercambio financiero se ha extendido porque permite la utilización de dinero “efectivo” en Internet, y su uso como sistema de pago digital facilita las transacciones financieras en línea (Xiaofan Li & Whinston, 2020).

Blockchain 1.0 es un libro mayor distribuido utilizado para almacenar las transacciones de “efectivo digital” entre dos partes. Todos los usuarios conectados por una red P2P gestionan ese libro mayor. Cualquier cambio en el libro mayor requiere un consenso entre más de la mitad de los usuarios de la red (Bhutta et al., 2021). El algoritmo utilizado mayoritariamente durante este periodo fue PoW, sobre un libro mayor distribuido en *blockchains* públicas (M. S. Lee & Kim, 2021).

Bitcoin es una criptomoneda que no está respaldada por el Banco Central Europeo ni la mayoría de los gobiernos y fue la primera aplicación de *blockchain*. Es utilizada actualmente para muchas actividades como, por ejemplo, reservar hoteles, compras, transacciones financieras, comprar videojuegos, etc.

Entre los beneficios de las criptomonedas basadas en la tecnología *blockchain* se pueden listar:

- ⊗ Descentralización de la criptomoneda y de las transacciones financieras.
- ⊗ Descentralización del almacenamiento de datos debido al uso de una base de datos distribuida.
- ⊗ Eliminación de una autoridad central.
- ⊗ Soporte a las transacciones a prueba de manipulaciones.
- ⊗ Transparencia en una red P2P al exponer al público el libro mayor.
- ⊗ Utilización de un algoritmo de consenso para asegurar las transacciones. En Bitcoin, por ejemplo, se utiliza PoW para asegurar todas las transacciones (Aggarwal & Kumar, 2021e).

Otra criptomonedas consideradas de esta generación son Litecoin (Simonite, 2013), Monero (Saberhagen, 2013) y Ripple (Armknecht, Karame, Mandal, Youssef, & Zenner, 2015).

b. Utilización de contratos inteligentes (*Smart Contracts*), Blockchain 2.0

La segunda generación de *blockchain* se inicia con los contratos inteligentes y la aparición de servicios financieros en torno a *blockchain* que fueron introducidos en el 2010. En esta generación aparecieron Ethereum e Hyperledger como marcos de trabajo relevantes que permitían generar aplicaciones con *blockchain* (Bodkhe et al., 2020).

Un contrato inteligente es un acuerdo entre los nodos de una red. Dicho contrato se ejecuta en una *blockchain* y se almacena en un libro mayor de contabilidad público. Se utiliza para facilitar, verificar o hacer cumplir digitalmente la negociación de las partes interesadas. Las transacciones de la red, que hacen uso de un contrato inteligente, se procesan y ejecutan automáticamente. Cada vez que ocurre una transacción entre varios nodos, se invoca una función que llama al contrato inteligente y lo procesa. La transacción se puede enviar a cualquier nodo de *blockchain* y se transmite a toda la red para que todos los nodos puedan leer la transacción (Aggarwal & Kumar, 2021b).

A diferencia del método tradicional utilizado por las personas u organizaciones para llegar a acuerdos legales, donde se recurre a un tercero como un notario o abogado, no es necesario ese tercero al utilizar un contrato inteligente (Kölvart, Poola, & Rull, 2016). En este caso el acuerdo es administrado y controlado por la red que da soporte a *blockchain*. Esta autonomía

es una de las propiedades destacables de los contratos inteligentes, además del ahorro económico por la ausencia de un tercero.

Otra de sus propiedades es la precisión, puesto que con ellos se evitan errores que surgen cuando se cumplimentan manualmente. Los contratos inteligentes también generan confianza ya que los documentos o ficheros que se comparten están cifrados y distribuidos públicamente en el libro mayor. Por último, el hecho de tenerlos distribuidos en los nodos de la red P2P, asegura la copia de seguridad, de tal forma que el contrato permanece de forma inmutable en la red como respaldo (Aggarwal & Kumar, 2021b).

En las plataformas diseñadas para el desarrollo de contratos inteligentes, como Ethereum e Hyperledger, los participantes se unen a la red y pueden solicitar la ejecución de un contrato particular para una transacción. El historial de estas transacciones se almacena en *blockchain*, que es gestionada por la red de manera parecida a las monedas digitales, de modo que el estado del contrato y los activos de los participantes son determinados por la secuencia de la transacción en *blockchain* (Bhutta et al., 2021). Los protocolos de consenso utilizados por *blockchain* tienen que resolver cualquier conflicto entre las partes contractuales (Nijse & Litchfield, 2020).

Además de Ethereum e Hyperledger, hoy en día existen muchas otras opciones disponibles para desarrollar aplicaciones de *blockchain* que permiten la creación de contratos inteligentes, entre ellas, Corda, Stellar, Rootstack, EOSIO y NEO (Aggarwal & Kumar, 2021c; Hewa, Ylianttila, & Liyanage, 2021; Mandloi & Bansal, 2020).

En cuanto al lenguaje de programación utilizado a la hora de desarrollar contratos inteligentes, algunas plataformas utilizan lenguajes propietarios como Solidity (Ethereum y Rootstack), otros lenguajes más estándar como Java (Hyperledger y Corda), Python (Stellar), C++ (EOSIO), y C# (NEEO), entre otros.

Los contratos inteligentes no están exentos de ataques (Sayeed, Marco-Gisbert, & Caira, 2020) y algunas plataformas han sufrido numerosos ataques que han costado millones de dólares, como el caso de la Organización Autónoma Descentralizada (DAO). Las DAO son organizaciones autónomas muy innovadoras, que no tienen oficinas, ni empleados y se basan en contratos inteligentes y *blockchain* (Falkon, 2017).

c. Aplicaciones descentralizadas (dApps), el Blockchain 3.0

Con la utilización de contratos inteligentes aparecen otro tipo de aplicaciones de *blockchain* diferentes a las criptomonedas. Estas aplicaciones son solo una de los muchos usos de las tecnologías de libro mayor distribuido (DLT) (Chowdhury et al., 2019). Estos libros distribuidos pueden contener cualquier tipo de información, no necesariamente relacionada con temas financieros. Todas las aplicaciones de la tecnología *blockchain* que utilizan contratos inteligentes con usos diferentes a las criptomonedas son referidas comúnmente como Blockchain 3.0. Aunque son aplicaciones conceptualmente independientes de las criptomonedas, suelen beneficiarse de estar integradas con ellas, como es el caso de Bitcoin o Ethereum (Di Francesco Maesa & Mori, 2020).

Las dApps pueden ser utilizadas en numerosas áreas como la salud, gobernanza, Internet de las cosas, cadena de suministro, negocios, ciudades inteligentes, juegos, entre otras aplicaciones (Cai et al., 2018).

El objetivo de la generación de la Blockchain 3.0 fue actualizar la generación anterior focalizándose en resolver los problemas de escalabilidad, interoperabilidad, privacidad, desempeño, coste y eficiencia presentes en la generación anterior. No obstante, dada la gran dominación de Bitcoin en el mundo de las criptomonedas, los problemas de escalabilidad han ido surgiendo. Para poder medir la escalabilidad de Bitcoin, varias medidas han sido utilizadas, entre ellas están el rendimiento máximo, la latencia, el tiempo de arranque y el coste por transacción confirmada (Croman et al., 2016).

Para resolver los problemas de escalabilidad, sin perder la seguridad y descentralización, varias aproximaciones han sido propuestas (Q. Zhou, Huang, Zheng, & Bian, 2020). Las soluciones centradas en los algoritmos de consenso, la red y la estructura de datos son ejecutadas en *blockchain* (*on-chain*), a diferencia de otras propuestas que escalan *blockchain* por medio de métodos fuera de la cadena (*off-chain*), como los canales y cadenas laterales y los protocolos de cadenas cruzadas (Q. Zhou et al., 2020).

Blockchain por defecto está diseñada para ser descentralizada, autosuficiente e independiente, de manera que un nodo verificador solo tiene que descargar la cadena y empezar a funcionar. Con la utilización de *blockchain* en otras aplicaciones diferentes a las criptomonedas, surge la idea de poder comunicar dos cadenas diferentes. Pero con el modelo inicial de *blockchain* esto no está considerado y, al parecer, es imposible de realizar sin modificar la definición inicial de *blockchain* (Lafourcade & Lombard-Platet, 2020).

Las *blockchains* normalmente son sistemas que operan en entornos independientes, debido a que no hay mecanismos ni confianza que permita que varias *blockchains* interactúen entre sí. Diversos investigadores están trabajando en cómo superar esas barreras, con el objetivo de resolver los problemas de escalabilidad y conectividad entre las diferentes plataformas de *blockchain* (Qasse, Talib, & Nasir, 2019).

Otro reto para resolver en beneficio de las dApps es la privacidad. *Blockchain* como base de datos distribuida tiene ciertas ventajas con respecto a la protección de la privacidad (a prueba de manipulaciones, anonimato y estabilidad de la red), ya que ayuda en la resolución de los problemas de privacidad a los que están expuestos algunos servicios centralizados basados en *blockchain* (sistemas de estacionamiento inteligente, votación, etcétera) (Dan Wang, Zhao, & Wang, 2020).

No obstante, la arquitectura de *blockchain* conlleva algunos efectos adversos con respecto a la protección de la privacidad, entre los cuales están la privacidad de la identidad del usuario y de sus transacciones, la administración de sus claves y aspectos regulatorios, como la ley de protección de datos (Bernal Bernabe, Canovas, Hernandez-Ramos, Torres Moreno, & Skarmeta, 2019). Para resolver estos problemas, otros autores han propuesto soluciones como la mezcla de monedas, pruebas de conocimiento cero, firmas en anillo, cifrado homomórfico, direcciones ocultas, computación segura de múltiples partes y ambientes de ejecución segura (Dan Wang et al., 2020).

Otros retos relacionados con las dApps tienen que ver con su desempeño, coste y eficiencia. En este aspecto se prevé que Blockchain 3.0 sea una tecnología disruptiva con potencial para ser utilizada en aplicaciones en varias industrias. Debido al surgimiento de muchas más plataformas de *blockchain* diferentes, es esencial conocer y poder evaluar su rendimiento para comprobar la viabilidad de ser utilizadas en diferentes casos de uso y escenarios. En este aspecto, algunos grupos de investigación (Dabbagh et al., 2021; Fan et al., 2020; Park & Li, 2021; Yadav & Shevkar, 2021) han realizado varias propuestas para formalizar y modelar cómo puede ser llevada a cabo esa tarea de valoración de desempeño.

d. La *blockchain* en la industria 4.0, denominada Blockchain 4.0

La Industria 4.0 (la cuarta revolución industrial) es conocida por un cambio de paradigma en la producción: una transición de la producción estática y controlada de forma central a la fabricación dinámica y descentralizada, que tiene como objetivo aumentar la calidad, la personalización y la eficiencia de los productos (Neugebauer, Hippmann, Leis, & Landherr, 2016).

En este aspecto existen muchos desafíos para que las empresas adopten tecnologías que ayuden a cumplir con los objetivos propuesto en la Industria 4.0, como los sistemas de producción autónomos y la cooperación interorganizacional flexible. Por su naturaleza, *blockchain* encaja de forma natural para ayudar a resolver esos desafíos (Laabs & Dukanović, 2021).

La utilización de *blockchain* desde la perspectiva empresarial puede ofrecer a la industria los siguientes beneficios relevantes de seguridad y de negocio (Laabs & Dukanović, 2021):

- ☑ **Inmutabilidad:** Este beneficio es muy importante si lo que se busca es hacer seguimiento de activos digitales y generar pistas de auditoría.
- ☑ **Firmas digitales:** Además de aportar autoridad a la red, asegura que cualquier interacción de activos digitales sea verificable y no repudiable.
- ☑ **Redes sin confianza:** Para los casos en los que exista desconfianza entre las partes, la tecnología *blockchain* facilita las interacciones entre sí sin la necesidad de un intermediario de confianza. Con *blockchain*, todas las transacciones son válidas y, si es necesario, se pueden utilizar mecanismos de consenso que proporcionen un depósito en garantía y custodia de los activos.
- ☑ **Transparencia:** Todos los participantes en la red pueden ver todo en *blockchain*, incluso pueden verificar el contenido exacto de un contrato inteligente antes de firmarlo. La inmutabilidad garantiza que el contrato no cambiará.
- ☑ **Conciliación rápida y menores costos de contratación:** Al no necesitar intermediarios de confianza, la liquidación entre las partes puede ser más rápida, sobre todo en los casos donde los intermediarios causan retrasos normalmente durante el proceso. Muchos costes de contratación, facturación y liquidación desaparecen.
- ☑ **Procesos automatizados de varios pasos:** Los contratos inteligentes pueden facilitar los procesos y flujos de trabajo, ya que están completamente automatizados y distribuidos. De esta forma, las partes conocen de antemano los posibles resultados de antemano, incluso traspasando las fronteras de la empresa.
- ☑ **Digitalización de activos:** *Blockchain* puede almacenar información de activos físicos y no físicos del mundo real, pudiéndose rastrear, comerciar o dividir, algo muy difícil en el mundo real. *Blockchain* facilita la representación digital de muchos activos, ofreciendo beneficios significativos en comparación con la forma habitual de seguimiento de activos físicos, como las transferencias, el seguimiento con papel y los acuerdos legales, así como la necesidad de contar con terceros de confianza.

- ☑ **Redundancia:** Debido a su naturaleza distribuida, las *blockchain* son menos propensas a puntos únicos de falla. La tolerancia contra fallas en los nodos, los problemas de conectividad y la pérdida de datos son un beneficio importante en comparación con los sistemas centralizados.

Todos estos beneficios por la adopción de *blockchain* ayudan en gran medida a los desafíos propuestos por la Industria 4.0, los cuales son (Laabs & Dukanović, 2021):

- 📁 Adoptar líneas de producción descentralizadas y autoorganizadas con todos los datos relevantes disponibles.
- 📁 Tener a disposición plataformas que faciliten el almacenamiento, intercambio y comercialización de datos.
- 📁 Gestionar grandes cantidades de tráfico de red.
- 📁 Proporcionar toda la información de producción juntos con los productos para facilitar la comunicación con su entorno.
- 📁 Contratación dinámica y gestión de empresas virtuales.
- 📁 Proporcionar una sólida confianza mutua entre las empresas.
- 📁 Estar en posesión de infraestructuras muchos más fiables.
- 📁 Proporcionar seguridad a las plataformas de servicio y las máquinas.
- 📁 Autenticación e integridad de las máquinas, equipos y servicios.
- 📁 Facilitar la protección del conocimiento o protección de la propiedad intelectual.
- 📁 Capacidad segura de poder enchufar y comenzar a trabajar.

Para lograr una integración perfecta con las necesidades y demandas comerciales de la Industria 4.0, estas especificaciones y la gran cantidad de dApps que van apareciendo requieren de una plataforma y una arquitectura de recursos empresariales que pueda integrar varios servicios y permita comunicaciones entre cadenas. Estas plataformas deben proporcionar automatización e integración con otras diferentes plataformas de forma transparente y coherente, con un grado mayor de confianza y privacidad.

Es en esos aspectos donde encaja Blockchain 4.0, permitiendo que los equipos técnicos de las organizaciones realicen la integración de todos o la mayoría de los procesos de negocio, garantizando la seguridad de todos los actores (Srivastava, Bhattacharya, Singh, & Mathur, 2018). Estas plataformas deben apoyar a la gestión de la cadena de suministro, los sistemas de gestión financiera, flujos de trabajo y gestión de activos. Entre las plataformas disponibles en el mercado que proporcionan o están trabajando en desarrollar soluciones para la Industria 4.0, están Hyperledger, Corda, EOS, IOTA, Ethereum, Quorum, OpenChain, Stellar, entre otras (Aggarwal & Kumar, 2021c).

2.6. Arquitectura de *blockchain*

Una vez descritos en los apartados anteriores de esta sección todos los componentes estructurales de *blockchain*, sus actores principales y cómo han ido evolucionado dichos componentes para ir dando solución a las necesidades que van demandando los individuos y la industria. Se finalizará este apartado describiendo la arquitectura de *blockchain*, que es la base sobre la que se construyen gran parte de sus aplicaciones.

a. Arquitectura basada en un libro mayor único

Este tipo de arquitectura fue utilizada para las aplicaciones de Blockchain 1.0, aunque ha permanecido hasta *Blockchain* 3.0. La arquitectura difiere un poco si la *blockchain* es pública, privada o híbrida, refiriéndose esta última a una arquitectura que utiliza transacciones privadas en una cadena pública.

En cuanto a las arquitecturas que utilizan una *blockchain* pública, estas presentan los participantes como pares y nodos. Un nodo puede ser simple, completo o de minado. Por medio de llamadas de procedimiento remoto (RPC), los clientes (usuarios) que envían o emiten una transacción, se conectan a *blockchain* y a un servicio de integración para conectarse a servicios externos. Los servicios externos son utilizados si se requieren datos externos para la validación, como, por ejemplo, datos climáticos, valores del mercado de acciones o datos de divisas. Si el servicio externo es malicioso, puede generar dudas de la validez de la transacción (R. Lai & Lee Kuo Chuen, 2018).

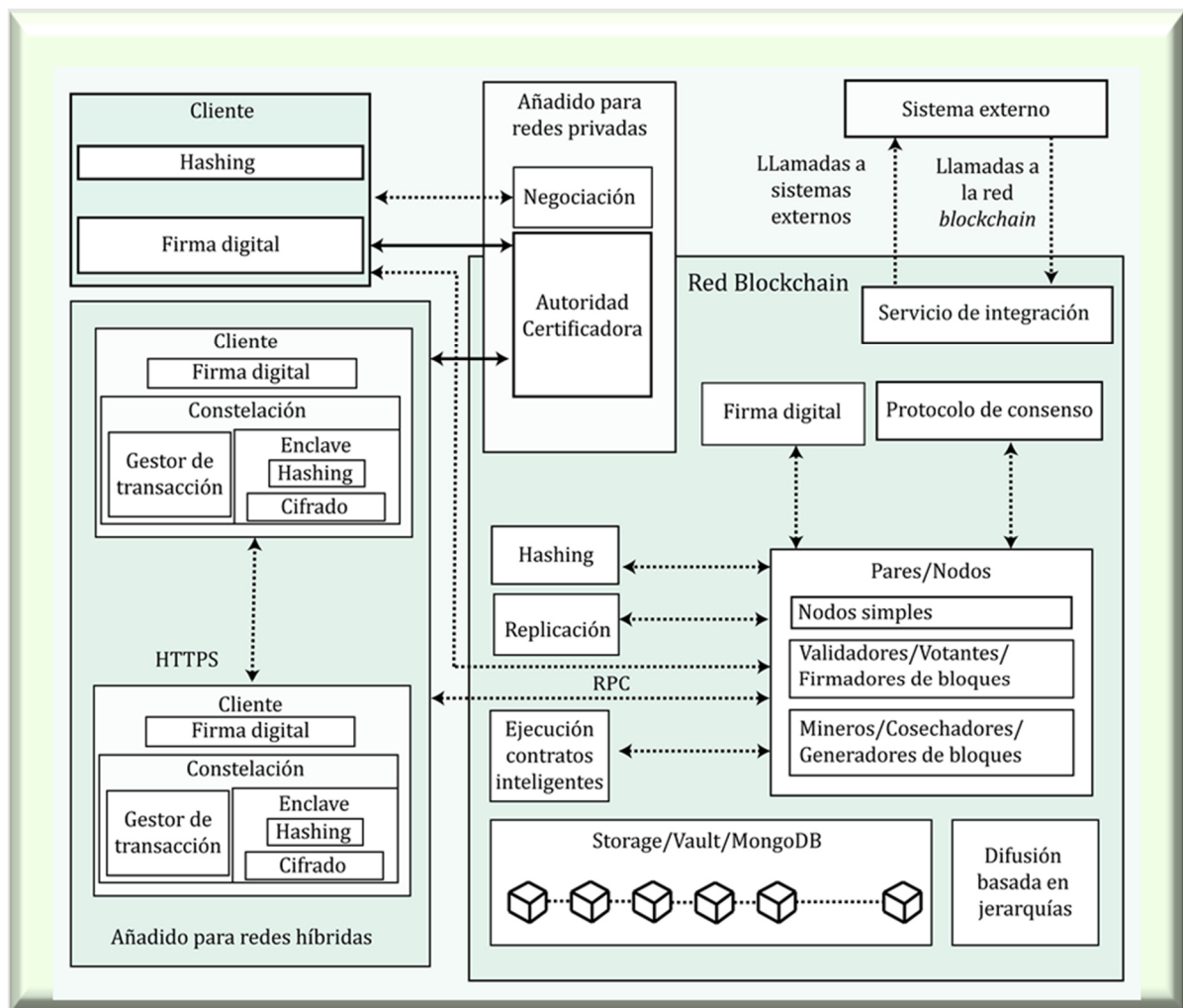
Ethereum es una plataforma que puede utilizarse para crear soluciones basadas en este tipo de arquitecturas (Ferretti & D'Angelo, 2020). Los dominios de aplicación apropiados para este tipo de arquitectura son el transporte, las cadenas de suministro, la gestión de proyectos, la propiedad intelectual, las finanzas y la energía. Pero no es aconsejable para aplicaciones privadas (como la atención médica, la educación, la gobernanza y la política nacional) debido a que cualquiera puede unirse a la red, los datos son públicos y no se emplean mecanismos de control de acceso.

Las arquitecturas que utilizan una *blockchain* privada presentan algunos componentes adicionales que permiten resolver los problemas de privacidad y control de acceso que presentan las *blockchains* públicas. Estos componentes adicionales son una CA y un proceso de negociación (*handshaking*). En este tipo de arquitectura, todos los nodos deben autenticarse y autorizarse por medio de roles para unirse a la red. El proceso de negociación se encarga de asegurar que los nodos participantes son auténticos y se establecen conexiones con ellos (Ismail & Materwala, 2019).

Diferentes plataformas puede ser utilizadas para este tipo de arquitectura, como es el caso de Hyperledger Burrow (Kuhlman, Middleton, Bollen, & Davis, 2017), Chain Core (Samman, 2016), Hyperledger Sawtooth (Ampel, Patton, & Chen, 2019), Hydrachain (Chakravarty et al., 2020), Hyperledger Iroha (Aggarwal & Kumar, 2021f), Burst (Gauld, von Ancoina, & Stadler, 2017), NEM (Y. Lai, 2018) y BigchainDB (BigchainDB GmbH, 2018).

En cuanto a las aplicaciones que siguen una arquitectura híbrida, la plataforma Quorum utiliza un bloque de construcción denominado Tessera para la arquitectura pública (S. Li, Wang, & Du, 2020). Tessera permite la utilización de cifrado, descifrado y distribución de transacciones privadas sobre una *blockchain* pública. Tessera permite el envío de transacciones de forma privada mediante cifrado. Incluye un gestor de transacciones y un enclave. El administrador de transacciones mantiene los datos de la transacción privados y seguros, ya que transmite los datos cifrados con *hash* a la red. Las operaciones de *hash* y cifrado/descifrado las realiza Enclave. Tessera se puede utilizar en el dominio de los bienes raíces, redes sociales, industria minorista, salud e investigación (Ismail & Materwala, 2019).

Todos los datos de las transacciones son almacenados en el libro mayor cuando se utilizan arquitecturas públicas y privadas. Sin embargo, la arquitectura híbrida realiza menos operaciones porque los nodos que no están involucrados en una transacción privada no almacenan los datos de la transacción, solo el *hash*. La Figura 40 muestra un esquema de las arquitecturas basadas en un libro mayor único que se comentan en esta sección.

Figura 40. Arquitectura de cadena de bloques pública, privada e híbrida basada en un libro mayor único.

Fuente: Modificado de Ismail & Materwala (2019).

b. Arquitectura basada en múltiples libros mayores para una red privada

Esta arquitectura se basa en la utilización de varios libros mayores distribuidos en grupos de nodos que comparten *blockchains* de forma privada. La idea central es agrupar nodos que realicen transacciones confidenciales y privadas entre ellas. Estos subgrupos se encuentran dentro de organizaciones o grupos de organizaciones formando una *blockchain* de topología federada (Ismail & Materwala, 2019).

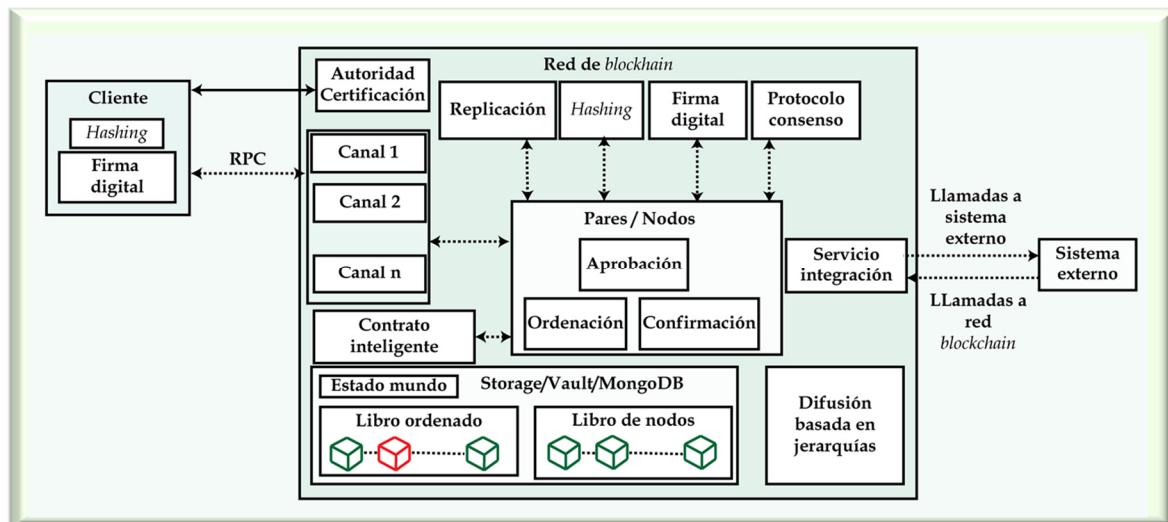
La idea fue concebida en una plataforma denominada Hyperledger Fabric (Aggarwal & Kumar, 2021f). Esta plataforma divide la red en canales donde los miembros realizan transacciones privadas entre ellos, las cuales no pueden ser vistas por otros miembros de otros canales dentro de la misma red federada. Para realizar transacciones privadas entre miembros de un

canal, Hyperledger utiliza el concepto de colección. Las transacciones dentro de una colección no pueden ser vistas por los otros miembros del mismo canal.

Dentro de esta arquitectura, el libro mayor está dividido en dos componentes: un componente que registra el estado del libro mayor, denominado el estado mundial, y el libro mayor en sí, que contiene los bloques validados y confirmados. Hay tres tipos de nodos o pares, los nodos de aceptación o aprobación (*Endorsing Peers*), los nodos de ordenación (*Orderes Peers*) y los nodos de confirmación (*Committing Peers*).

Los nodos de aprobación son los que ejecutan los contratos inteligentes (llamados código encadenado o *Chain Code*) de las transacciones propuestas por los clientes y los validan, pero no realizan ninguna modificación a la *blockchain*, solo firman la transacción y envían la respuesta al cliente. Cuando el cliente recibe las respuestas de los nodos de aprobación, estas respuestas son enviadas a los nodos de ordenación, que generan un bloque recibido de diferentes clientes para luego enviarlo a todos los nodos de confirmación. Cada nodo de confirmación verifica si el bloque es válido de acuerdo con las políticas de aprobación, y aplica las transacciones validadas en el estado mundial y las registra en el libro mayor. Por último, las aplicaciones son notificadas del éxito o fallo de las transacciones (X. Xu, Weber, & Staples, 2019). En la Figura 41 se puede ver un esquema de esta arquitectura.

Figura 41. Arquitectura de blockchain basada en múltiples libros mayores.



Fuente: Modificado de Ismail & Materwala (2019).

Esta arquitectura es apropiada cuando varias organizaciones requieran confidencialidad en las transacciones entre diferentes subgrupos de los participantes, como es el caso de Universidades colaboradoras, hospitales y grupos empresariales.

Hyperledger Fabric no es la única opción en el mercado para implementar soluciones de este tipo, sino que hay otras plataformas disponibles para ello, como Ethereum, Quorum, Multichain y Corda, permiten el desarrollo de aplicaciones de *blockchain* con permisos y con topología federada (Polge et al., 2020).

c. Arquitectura basada en interoperabilidad

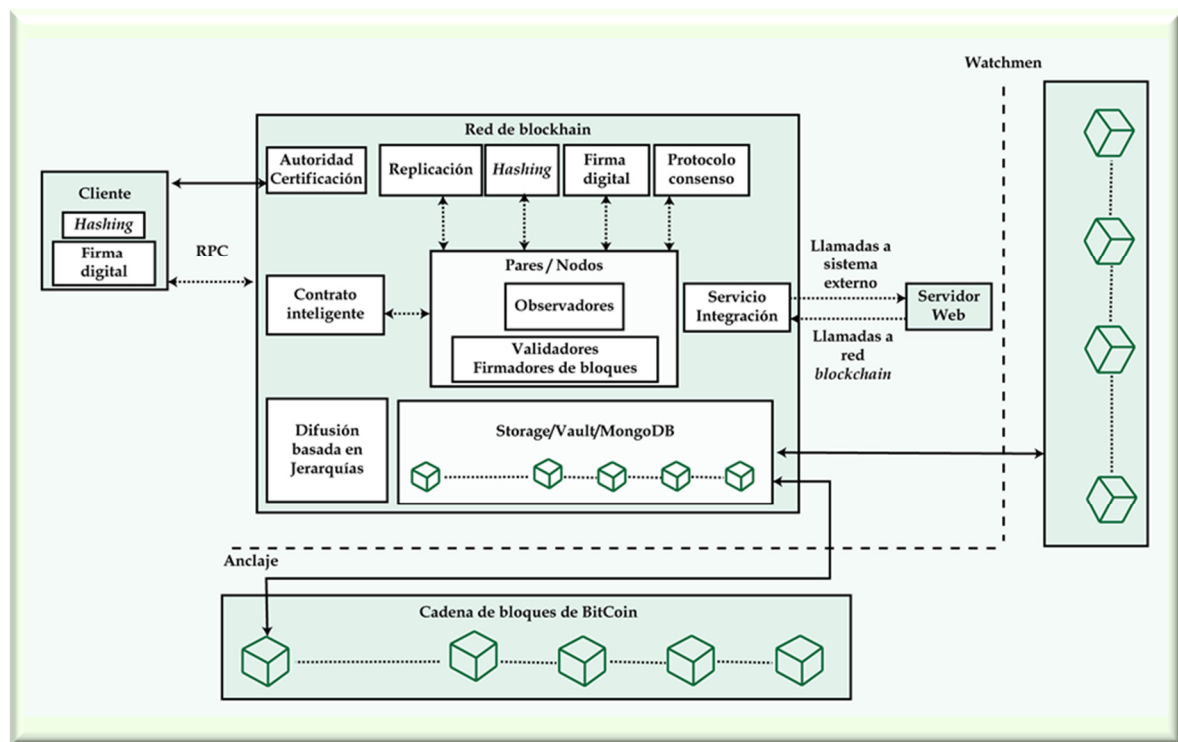
Hoy en día, en muchos dominios de aplicación se está estudiando la viabilidad de poner adoptar *blockchain* como una solución para sus aplicaciones. Con el objetivo de mejorar aspectos a los que *blockchain* puede dar solución, muchas plataformas que se han desarrollado admiten diferentes lenguajes de programación, formas de contratos inteligentes y muchos protocolos de comunicación distintos, lo que hace muy difícil la interacción entre diferentes *blockchains* (Lafourcade & Lombard-Platet, 2020).

Este problema se ha abordado a través del uso de lo que denominan “*sidechains*”, que es una *blockchain* enlazada a otra cadena, que posibilita la interacción entre ellas (Singh et al., 2020). Este concepto se le conoce como arquitectura de interoperabilidad y su finalidad es permitir la interacción de *blockchains* entre redes públicas y privadas y mejorar la seguridad de una *blockchain* al vincularla con otra *blockchain* (Siris et al., 2019). Algunas plataformas como Elements (<https://elementsproject.org>), Openchain (<https://www.openchainproject.org>) y Lisk (<https://lisk.com>) permiten el desarrollo de este tipo de *blockchain* con específicas implementaciones (Samanta et al., 2021).

La Figura 42 representa un esquema general de este tipo de arquitectura. En ella se puede apreciar el uso de dos cadenas, una principal, que como en el caso de Openchain es una *blockchain* Bitcoin, y una cadena lateral, que es la propia de OpenChain. También se observan los dos elementos diferenciadores de esta arquitectura:

- ✎ El proceso denominado **anclaje** (*anchoring*), que añade un *hash* acumulativo de la cadena lateral en la principal.
- ✎ El servicio **Watchmen**, que transfiere de forma verificada y segura diferentes activos entre cadenas. Este servicio por ejemplo es implementado en Lisk.

Figura 42. *Arquitectura de blockchain basada en interoperabilidad.*



Fuente: Modificado de Ismail & Materwala (2019).

3. Propiedades de seguridad y privacidad de *blockchain*

Blockchain por defecto garantiza una serie de características de seguridad inherentes: consistencia, resistencia a la manipulación, resistencia a ataques DoS, anonimato y resistencia al ataque de doble gasto.

Consistencia

Consistencia en el contexto de un libro mayor distribuido en una *blockchain* se refiere a la propiedad de que todos los nodos tienen el mismo libro mayor al mismo tiempo. La consistencia que proporcionan algunas aplicaciones de *blockchain*, como Bitcoin, es definida por algunos autores como eventual o débil (Tseng, 2017), mientras que otros la consideran fuerte (Decker, Seidel, & Wattenhofer, 2016).

La consistencia eventual se logra cuando se intenta un equilibrio entre disponibilidad y consistencia, asegurando que todas las actualizaciones finalmente sean propagadas a todos los nodos (Vogels, 2009). Con esto se logra que haya una alta disponibilidad de la información y una baja latencia, aunque no está claro cuando son los datos consistentes, por lo que una consulta puede generar datos obsoletos.

Por otra parte, la consistencia fuerte significa que todos los nodos tienen el mismo libro mayor al mismo tiempo, por lo tanto, no se puede consultar los datos actualizados hasta que no se actualicen los nuevos datos en todos los nodos.

El teorema de Consistencia, Disponibilidad y Tolerancia a la Partición (CAP), se utiliza para definir las propiedades transaccionales de los sistemas distribuidos. Este teorema indica que cualquier sistema distribuido puede tener solamente dos de esas tres propiedades. La consistencia indica que cada nodo recibe la actualización más reciente, la disponibilidad indica que los datos siempre están disponibles y la tolerancia a partición indica que el sistema siempre estará realizando operaciones, aunque algún subconjunto de nodos falle (Gilbert & Lynch, 2002). Parece ser que este teorema ha sido violado por *blockchain* de Bitcoin, ya que sí que logra simultáneamente las tres propiedades, pero no lo hace hasta después de un periodo de tiempo (R. Zhang et al., 2019).

Resistencia a la manipulación

En el contexto de *blockchain*, la resistencia a la manipulación se refiere a que cualquier información de transacción almacenada en *blockchain* no puede manipularse durante y después del proceso de generación de los bloques. Cualquier intento de manipulación en *blockchain* no puede tener éxito, ya que cada transacción es comprimida por una función *hash*, como SHA-256, y luego firmada usando un algoritmo de firma segura, como ECDSA.

Un nodo no puede alterar la información contenida en los bloques debido a que sería detectado por otros nodos debido a los mecanismos de consenso y esto está garantizado por la imposibilidad de falsificar el algoritmo de firma segura. Además de este sistema de verificación, un adversario fallará en sus intentos de modificar cualquier dato contenido en los bloques históricos gracias al encadenamiento *hash*.

Resistencia a los ataques de denegación de servicio distribuido (DDoS)

El ataque de denegación de servicio distribuido (DDoS) es perpetuado sobre un host logrando que se interrumpan sus servicios para los usuarios previstos. Los atacantes pretenden sobrecargar el sistema inundándolo con solicitudes sin sentido, y de esta forma, paralizan el sistema para que no pueda proporcionar sus servicios (McDowell, 2009). Un ataque DDoS es un ataque distribuido, es decir, el ataque de inundación de tráfico entrante se origina desde muchas fuentes distribuidas a través de Internet (Mahjabin, Xiao, Sun, & Jiang, 2017).

La gran preocupación de este tipo de ataque en *blockchain* está relacionada con la preocupación de no tener una *blockchain* disponible al eliminar nodos de forma parcial o completa. Este ataque no se puede perpetuar debido a la naturaleza totalmente

descentralizada del sistema y al protocolo de consenso, que asegura que el procesamiento de las transacciones en *blockchain* pueda continuar incluso si son varios los nodos que fallan.

Cuanto más grande se vuelve la red, más difícil es tener éxito en un ataque DDoS, ya que el atacante tendría que poseer mucha capacidad computacional para superar la capacidad computacional agregada de todos los nodos de la red (Saad et al., 2020).

Resistencia a los ataques de doble gasto

Este ataque se refiere a las implementaciones de moneda digital utilizando *blockchain*, entre las que se puede citar a Bitcoin (Chohan, 2018). Se considera un problema de seguridad general debido a la facilidad de reproducir la información digital, como, por ejemplo, *tokens* digitales o moneda electrónica, y enviarla a muchos destinatarios.

Para evitar el doble gasto, el protocolo de consenso utilizado (PoW en Bitcoin) verifica todas las transacciones utilizando los registros de transacciones en su *blockchain*, permitiendo que todos los nodos verifiquen públicamente las transacciones antes de realizar cualquier modificación a nivel global en *blockchain*. De esta forma, se asegura de que el remitente de cada transacción solo gaste la cantidad de moneda digital que posee legítimamente.

La combinación de transacciones firmadas con firmas digitales y la verificación pública de transacciones con un consenso mayoritario garantizan que la *blockchain* utilizada en las implementaciones de moneda digital pueda ser resistente al ataque de doble gasto.

Resistencia al ataque de consenso a la mayoría (51%)

El protocolo de consenso tiene el riesgo de poder ser engañado. Uno de esos engaños es el ataque del 51% en el contexto del doble gasto. Este tipo de ataque puede producirse si un nodo minero o grupo de nodos mineros controlan más del 51% de la potencia informática para mantener la *blockchain*, lo que les da la posibilidad de poder manipular el libro mayor distribuido de todas las transacciones de una criptomoneda.

En el caso de Bitcoin, el ataque de consenso a la mayoría podría ocurrir ante la presencia de mineros malintencionados en la red, aunque sería necesario que contaran con mucho poder computacional para lograr perpetuarlo (Sayeed & Marco-Gisbert, 2019).

Pseudonimato

El pseudonimato hace referencia a tener una identidad encubierta (Möser, 2013). En sistemas como Bitcoin, los usuarios utilizan los *hashes* de las claves públicas de los usuarios o nodos en la red. De esta forma, al interactuar con el sistema utilizando este mecanismo, no se puede saber realmente quién es la persona o entidad que está realizando dicha transacción.

La dirección que utiliza el usuario se puede ver como una pseudoidentidad, lo que se considera una propiedad de privacidad para proteger el nombre real el usuario o entidad que está realizando la transacción.

Los usuarios en Bitcoin, por ejemplo, pueden generar tantos pares de claves como deseen, como si estuvieran creando múltiples cuentas bancarias. No obstante, el estudio de Y. Li et al. (2019) muestra que existe riesgo de revelar la información de identidad en sistemas de criptomonedas como Bitcoin utilizando algunos algoritmos de análisis de transacciones, tales como algoritmos de *clustering*, teoría de grafos y análisis de datos.

4. Estado del arte de soluciones de correo electrónico sobre *blockchain*

Dados los requisitos de seguridad deseados en una solución de correo electrónico integral, los profesionales encuentran problemas a la hora de abordar soluciones de correo electrónico sobre *blockchain*, ya que por defecto esos requisitos no son considerados en su especificación inicial. En apartados anteriores, se hace referencia a las numerosas aplicaciones de *blockchain* en diferentes dominios (Pournader et al., 2020), pero a pesar de ello, muchas empresas no se arriesgan a la hora de abordar un proyecto con una tecnología donde los aspectos de seguridad están en estudio.

Se identifican en la literatura científica diferentes enfoques para abordar este problema (Bernal Bernabe et al., 2019; Taylor, Dargahi, Dehghantanha, Parizi, & Choo, 2020; Vassilakis, 2019). Existen varias propuestas comerciales disponibles en el mercado que ofrecen correo seguro sobre *blockchain*, pero no suelen aportar documentación sobre cómo abordan los problemas de seguridad ni los detalles técnicos de su construcción.

Mailchain (<https://mailchain.xyz>) es una solución comercial distribuida basada en *blockchain* de Ethereum. Está focalizado en el envío de mensajes cifrados con una infraestructura propietaria, sin una arquitectura definida y nada integrable con los sistemas de mensajería y los protocolos de correo electrónico actuales. El funcionamiento es simple, ya que los mensajes son enviados utilizando el mismo mecanismo que las transacciones en una red de *blockchain* de Ethereum. Los participantes se identifican por medio de su clave pública y, para realizar la

comunicación, previamente el remitente y destinatario deben haberse intercambiado las claves.

En Mailchain, el mensaje es cifrado con la clave pública del destinatario y almacenado en una localización fuera de *blockchain* (*offchain*), proporcionado al destinatario un *hash* del mensaje para propósitos de integridad y un enlace que es almacenado en el sobre del mensaje. Una transacción es creada y firmada incluyendo el sobre con toda la información. Para acceder al mensaje, el destinatario accede a la transacción, decodifica y descifra el mensaje descargado del enlace que ha sido proporcionado en el sobre. Es una solución cerrada y aparentemente es un proyecto olvidado.

Otra propuesta que se analizó es **Ledgermail** (<https://ledgermail.io>). Según sus creadores, es una solución comercial de correo electrónico descentralizado basada en *blockchain*. Es un sistema de pago, sin documentación *online* sobre su arquitectura y funcionamiento, pero se puede leer en su sitio *web* que los mensajes cifrados son almacenados en un libro mayor distribuido bajo el algoritmo de consenso DPoS. Los correos electrónicos son cifrados utilizando la clave pública del destinatario, siendo el único que puede leerlos al recibirlos.

Los creadores de Ledgermail ofrecen un sistema con resistencia a la manipulación de los mensajes, irrompible por terceros, un cifrado de mensajes y descentralización. Comentan que el sistema de identificación de usuario se fundamenta en direcciones de correo basadas en dominio y ofrecen integración con Gmail y Office 365. Es un proyecto activo, pero es imposible evaluarlo profundamente sin la documentación técnica disponible, aunque se puede concluir que es una solución de almacenamiento en la nube.

Cryptamail (<http://www.cryptamail.com>) está descrito en su página *web* como un sistema descentralizado, donde los mensajes son almacenados en una *blockchain* bajo el algoritmo de consenso PoS de Nxt, una plataforma de *blockchain* que fue desarrollada desde cero. Según los datos que se tienen, el correo se transmitía de igual forma que se transmiten las transacciones en Bitcoin. Utilizan su propio sistema de nomenclatura para las direcciones y es un proyecto abandonado en el año 2014.

Dmail (<https://dmail.co>) es una propuesta de correo electrónico desarrollada bajo EOS (<https://eos.io>), otra de las plataformas de *blockchain* diseñada para el desarrollo de aplicaciones empresariales basadas en tecnología *blockchain*. EOS es una plataforma muy sólida y, en el momento de escribir esta memoria, está resolviendo problemas de gobernanza y su centralización está siendo cuestionada debido a que el proceso de consenso es realizado por 21 nodos productores de bloques bajo el algoritmo de DPoS. Dmail utiliza un token para la firma de los correos y no tiene ninguna pasarela a servidores basados en SMTP. Hay muy poca

información técnica sobre su arquitectura y por ahora nos ha sido imposible participar en la versión beta.

Flowingmail (<http://flowingmail.com>) es otra de las propuestas comerciales de correo electrónico basada en tecnología distribuida. Los autores lo presentan como un protocolo descentralizado, seguro y cifrado. El principal componente del protocolo es una tabla de *hash* distribuida (DHT) basada en Kademlia (implementación conocida es la red eMule utilizada para compartir ficheros de forma distribuida) (Maymounkov & Mazières, 2002), que es la red donde se almacenan los correos mientras están en tránsito y los certificados de los participantes (Hassanzadeh-Nazarabadi, Küpçü, & Özkasap, 2021). Los clientes son nodos del DHT, almacenan los correos electrónicos de forma fragmentada y no todos los nodos almacenan los bloques. Comentan que tienen una pasarela para SMTP. Está bajo desarrollo y no hay documentación disponible.

Blockchain ha sido utilizada en muchas propuestas como una **extensión al correo electrónico tradicional** para realizar actividades como la validación de remitentes desconocidos (Dennis, 2019), la validación de correos para verificar integridad y saber si el correo electrónico no ha sido falsificado (Varghese, 2019), el uso de un pago con criptomonedas para poder hacer envíos de correo electrónico (Sheikh & Banday, 2021), la utilización de sistemas de reputación descentralizado sobre *blockchain* para evitar correo no deseado (Kannan & Jebakumar, 2020) o el uso de *blockchain* para el envío de correo certificado sin un TTP (Hinarejos & Ferrer-Gomila, 2020). Este tipo de trabajo focalizan el uso de *blockchain* como un módulo adicional al sistema tradicional de correo electrónico que presta un servicio para mitigar algunos de los problemas tratados en apartados anteriores.

Otro tipo de propuestas se centran en proporcionar una **solución integral** utilizando *blockchain* como el servicio central de correo electrónico. Una de las primeras propuesta analizadas empleaba como identificador de los participantes dentro de la comunicación de correo los mismos identificadores utilizados para realizar transacciones en *blockchain* (Khacef & Pujolle, 2019). La primera actividad que realiza el usuario es un proceso de registro y una autenticación mutua entre los participantes antes de iniciar cualquier comunicación. De esta forma, solo se pueden comunicar los usuarios validados por los contratos inteligentes en *blockchain*. El problema de esta propuesta es que se centra solamente en la autenticación de los usuarios, pero no especifica cómo se desarrolla la comunicación de los mensajes de correo ni cómo se resuelven los problemas de integración.

Otra propuesta más elaborada hace uso de *blockchain* para almacenar los mensajes utilizando un servidor propietario denominado **SteemCaller** (Menegay, Salyers, & College, 2019). Este servidor permite gestionar tanto correo SMTP como correo sobre *blockchain*, enrutándolo de acuerdo con unas reglas predefinidas. A causa de las limitaciones para enviar grandes cantidades de información (como las que suele requerir un mensaje de correo electrónico), el correo es segmentado y los fragmentos de correo son enviados con diferentes transacciones que luego son reensambladas por el destinatario del correo. El problema que presentan este tipo de soluciones es que no son de fácil adopción, ya que implican modificar los servidores de correo, y el almacenamiento de los correos electrónicos se realiza en *blockchain*.

El último trabajo analizado en este apartado es el presentado por Chamadoira-González et al. (2020) en el que proponen reemplazar los protocolos de correo electrónico por **contratos inteligentes**. Según los autores, el sistema propuesto es un servicio de correo electrónico certificado que gestiona y almacena datos personales de forma segura, previniendo los problemas tradicionales más conocidos relacionados con el correo electrónico (*spam*, suplantación de identidad, phishing, etc.), garantizando el no repudio y auditando las transacciones a través de un sistema. La arquitectura de este sistema se centra en los contratos inteligentes (desarrollados en Ethereum) como módulo central, que es quien gestiona las reglas del protocolo, facilitando el envío, recepción y autorización de envíos para los participantes. El sistema cuenta con clientes *web* que se conectan a un servidor *web* que, a su vez, se conecta al nodo de *blockchain*, a un servidor de base de datos que almacena la información de los usuarios con sus claves y al sistema IPFS para almacenar los mensajes cifrados. Es un trabajo experimental y ya no hay actividad del código publicado.

La Tabla 12 resume los trabajos analizados y presenta una comparativa entre las soluciones y propuestas estudiadas en esta sección. Esta tabla está desarrollada bajo aspectos teóricos, de los que se pueden concluir algunos datos relevantes:

- ❶ El aspecto de la usabilidad es un tema poco abordado, ya que la seguridad implementada en la mayoría de las soluciones no es transparente y las herramientas o soluciones necesitan demasiadas configuraciones o conocimientos para su uso.
- ❷ La integración también es un aspecto poco considerado. Solo dos propuestas la consideran y el resto solo comenta que en el futuro será añadido, pero no especifican detalles de cómo será esa integración.

Tabla 12. Comparativa de propuestas de correo electrónico basado en blockchain.

Soluciones	Aspectos analizados									Estado
	Confidencialidad	Integridad	No repudio	Autenticación	Disponibilidad	Control	Auditoría	Usabilidad	Integración	
Mailchain	Media	Alta	Alto	Baja	Media	Bajo	Baja	Baja	Baja	Olvidado
Ledgermail	Media	Media	Medio	Alta	Media	Bajo	Baja	Media	Media	Activo
Cryptamail	Media	Media	Medio	Baja	Media	Bajo	Baja	Media	Baja	Olvidado
Dmail	Media	Media	Medio	Alta	Media	Bajo	Baja	Media	Baja	Activo
Flowingmail	Media	Media	Medio	Baja	Media	Bajo	Baja	Media	Media	Bajo desarrollo
Khacef	Baja	Baja	Bajo	Alta	Alta	Bajo	Baja	Baja	Baja	Sin determinar
Menegay	Alta	Media	Medio	Alta	Media	Bajo	Baja	Media	Media	Sin determinar
González	Alta	Alta	Alto	Media	Alta	Bajo	Baja	Baja	Baja	Bajo desarrollo
Blockmail	Alta	Alta	Alto	Alta	Alta	Alto	Alta	Alta	Alta	Bajo desarrollo

Fuente: Elaboración propia.

Se debe considerar que algunas propuestas están en desarrollo y al parecer la arquitectura no se ha diseñado en su totalidad, sino que se van definiendo aspectos a medida que se avanza, lo que hace que no se puedan valorar con claridad todos los aspectos.

De todas las propuestas analizadas, la de Chamadoira-González et al. (2020) es la teóricamente más completa a nivel de seguridad, pero no considera todos los aspectos deseados para una solución de correo electrónico integral. Además, no trata aspectos de integración, lo que puede afectar su adopción. Es una solución en desarrollo y no está completamente integrada. Por lo tanto, tiene módulos que deben ser cuidadosamente diseñados y desarrollados debido a que utilizan TLS que, como se ha comentado en apartados anteriores, presenta vulnerabilidades y es compleja de adoptar.

V. CONCLUSIONES SOBRE LAS SOLUCIONES DE CORREO ELECTRÓNICO Y JUSTIFICACIÓN DEL ESTUDIO

- ❏ A lo largo de la sección anterior se discutieron las aproximaciones utilizadas para abordar el problema de la seguridad del correo electrónico. La mayoría de las soluciones analizadas abordan el problema desde una visión muy particular o específica.
- ❏ Algunas aproximaciones se focalizan solamente en la confidencialidad, con el objetivo de dar un servicio adicional, ya sea para dar solución a un ataque como MIT o simplemente como un valor añadido para proteger la información que circula por el correo y evitar fugas o agujeros de seguridad. Otro tipo de aproximaciones intentan, ya generado el problema (como el caso del correo no deseado), incorporar procesos adicionales de filtrado con el objetivo de evitar que esos correos lleguen a su destinatario.
- ❏ Todas las aproximaciones sin excepción trabajan sobre el correo ya generado y ninguna sobre la creación misma del correo, que debería de seguir unas reglas estrictas para proteger al remitente. El correo no deseado no solo es un problema para el destinatario, sino también para el remitente, quien puede ser suplantado para que se envíe correo no deseado a sus contactos. Se puede ver como el remitente es utilizado como medio para realizar muchas infracciones, no siendo en una primera instancia el afectado por el ataque, pero sí por sus consecuencias.
- ❏ Un ejemplo de esta casuística es el envío de correo a un destinatario con datos comprometidos por parte de un remitente que ha sido suplantado o está bajo control de una *botnet*. En este caso el remitente desconoce lo que pasa, no es el afectado directo, pero puede tener problemas legales indirectos por enviar información comprometida.
- ❏ La seguridad del correo debe ser abordada desde la creación del correo hasta que llega a su destino y es leído y almacenado por el destinatario. Hay diferentes tipologías de usuarios con diferentes necesidades que utilizan la misma tecnología y protocolos para comunicarse. Hay usuarios que requieren por comodidad medidas mínimas de seguridad y otros necesitan bastantes medidas, políticas y protocolos de seguridad en su entorno. Aunque todos ellos siguen los mismos protocolos de correo, se han tenido que tomar medidas adicionales para abordar la seguridad, lo que lleva a concluir la necesidad de contar con un protocolo que les proporcione seguridad a todos estos diferentes tipos de usuarios con sus distintos niveles de seguridad.
- ❏ En vista de que todos los usuarios utilizan los mismos protocolos, parece razonable pensar que todos deberían tener integradas las mismas reglas de seguridad. Aunque cada usuario tiene implementados los protocolos en sus servidores con un estándar muy definido (como es SMTP, POP o IMAP), hoy en día cada uno implementa sus propias medidas de

seguridad. Esto nos lleva a la conclusión de que el protocolo está de alguna manera centralizado (a través del estándar) pero la seguridad está distribuida y no estandarizada.

- ❏ Reconociendo estas propiedades del correo electrónico, donde un protocolo puede ser visto como una regla que está distribuida pero consensuada entre todos (el estándar), algunos autores han propuesto soluciones de seguridad utilizando la tecnología *blockchain*, como se ha descrito previamente. Esas propuestas siguen la misma línea y visión que las soluciones tradicionales, pero además tienen objetivos adicionales como eliminar la centralización propia del correo electrónico y hacer del correo electrónico una herramienta de comunicación totalmente confidencial mediante la utilización de técnicas criptográficas.
- ❏ El uso de la tecnología *blockchain* con ese enfoque de correo electrónico totalmente integrado en un libro mayor genera muchos retos y la filosofía de este trabajo considera que no es el apropiado dado que la naturaleza del correo electrónico propiamente dicho no se asemeja a un libro mayor, donde las transacciones se almacenan perpetuamente y es necesario el histórico para determinar el estado actual del sistema. Es probable que por esta razón las soluciones de correo electrónico que se han analizado y estudiado en este capítulo no han sido adoptadas en gran medida o, en el peor de los casos, terminan siendo soluciones centralizadas que predicen utilizar la tecnología *blockchain* como un mensaje comercial.
- ❏ En la presente investigación se propone utilizar *blockchain* en una dirección diferente. Como se ha comentado, los protocolos se pueden ver como una regla distribuida y consensuada por todos al ser definida en un estándar. Por este motivo se plantea utilizar la tecnología *blockchain* para definir los protocolos de correo electrónico seguro, añadiendo reglas de seguridad distribuidas. De esta manera, cualquier usuario que desee utilizar el servicio de correo electrónico deberá regirse por esas reglas distribuidas y no podrá realizar ninguna acción fuera de ellas, pues los protocolos de consenso utilizados no permitirán que se ejecute ninguna transacción que incumpla los contratos especificados por los usuarios.
- ❏ En esta propuesta, los protocolos de seguridad de correo electrónico son contratos inteligentes y estos contratos se rigen por los parámetros y configuraciones definidos por los usuarios que participan en el sistema, siendo los validadores de las reglas quienes determinan si una operación puede o no ser realizada.
- ❏ Con este enfoque, las reglas que rigen el envío y recepción de correo electrónico son consensuados por todos o parte de los nodos que participan en el sistema y el nivel de seguridad asociado a cada una de las operaciones es definido por el usuario. De esta forma,

en el sistema podrían coexistir dos usuarios con exigencia o niveles de seguridad diferentes y la comunicación o no entre ellos será definida por las reglas que determinen cada uno de ellos. Lo que se determina como inmutable y que debe llevar un registro histórico junto con las configuraciones y permisos son los protocolos de correo electrónico.

- ☐ En esta tesis se considera que el mensaje de correo electrónico no debería tener un registro inmutable asociado, a no ser que lo determine el usuario por las razones que sean. En este punto, se aleja de las soluciones donde el correo electrónico siempre es almacenado en una *blockchain*, ya que se estimó no apropiado y alejado de su naturaleza. Lo que hace a esta propuesta innovadora es que utiliza las propiedades de la tecnología *blockchain* en favor de la seguridad del correo electrónico y propone una arquitectura flexible y apropiada para las diferentes tipologías de usuarios y sus necesidades.
- ☐ En este trabajo, se considera como un aporte importante que el enfoque de seguridad para tratar los problemas de correo debería abordar otros aspectos de seguridad que no son tenidos en cuenta en las soluciones de seguridad actuales. Por esta razón, en el desarrollo de la propuesta descrita a lo largo del capítulo 3 son consideradas las siguientes propiedades de seguridad: confidencialidad, integridad, no repudio, autenticación, disponibilidad, control, auditoria, usabilidad e integración.

Capítulo 3

Desarrollo de la Contribución y Discusiones

I. INTRODUCCIÓN

Este capítulo está dedicado al desarrollo y presentación de los resultados de esta investigación. Se ha organizado esta sección en varias partes, que en conjunto componen esta contribución. En el **apartado II** se presenta una serie de **definiciones** y conceptos que se irán utilizando a lo largo del desarrollo de este capítulo. En el **apartado III** se definen los **requisitos** identificados que deben ser alcanzados para que una solución de correo electrónico sea considerada como segura.

En el **apartado IV** se muestra detalladamente la **arquitectura propuesta** que da solución a los requisitos anteriormente expuestos. Se describen todos los **protocolos** de seguridad que deben ser implementados y en el **apartado V** (Tabla 15) se **resumen** las cuestiones claves propuestas en torno a cada uno de ellos.

En el **apartado VI**, se detalla la **prueba de concepto** con la descripción de las tecnologías utilizadas para validar la arquitectura propuesta y los protocolos implementados. Se muestra en una tabla resumen los algoritmos y tecnologías utilizados para la protección de los datos sensibles presentes que se manejan en torno a la arquitectura. Se incluye una descripción de las pruebas realizadas para la verificación de los requisitos junto con los resultados obtenidos.

En el **apartado VII**, se comentan algunas **cuestiones adicionales** que se consideran claves en este trabajo. Finalmente, en el **apartado VIII**, se escribe sobre las **limitaciones** que se han encontrado durante el desarrollo de esta investigación y las **fortalezas** del presente estudio y, en el **apartado IX**, se marca el camino a seguir para **futuros trabajos** en torno a esta investigación.

II. CONCEPTOS Y DEFINICIONES PRELIMINARES

Se desarrollarán los conceptos teóricos básicos que serán utilizados a lo largo de este capítulo. Las siguientes definiciones facilitan la comprensión de la arquitectura y los protocolos propuestos y en general de todos los procesos descritos en esta sección.

Correo electrónico

Comunicación escrita creada por uno o varios autores y enviada por un remitente a uno o varios destinatarios con el objetivo de hacerle llegar información en formato texto (denominado mensaje) y con ficheros adjuntos opcionales, con un objetivo específico conocido o gestionado por los autores. Más formalmente un correo (Ce), es una tupla $(CeId, Pk, X(RC+, Sk), X(Mc, Sk), X(Ac*, Sk), X(Cc, Sk), Ds, SR)$ en la que:

- @ **CeId:** Es el identificador único del correo. Es un valor resumen o *hash* que identifica de manera única cada correo.
- @ **Pk:** Es la clave pública del remitente o un identificador único, que será utilizado dependiendo del nivel de confidencialidad que proporcione la implementación.
- @ **RC+:** Es la relación de comunicación. Es una tupla $(RCOpk, RCrol, RCDpk, RCcf, Sign(Mc, RCOsk))$. El símbolo **(+)** significa que puede ser una o más tuplas, lo que indica que el correo electrónico puede estar dirigido a más de un destinatario y ser enviado por más de un originador.
- @ **RCOpk:** Es el originador.
- @ **RCrol:** Es el papel que juega el originador en la comunicación, pudiendo ser autor, remitente o autorizador. El último componente de la tupla es *Sign*, que es la firma que crea el originador del mensaje (*Mc*), indicando con ello que autoriza el mensaje con el rol indicado.
- @ **RCDpk:** Es el destinatario.
- @ **RCcf:** Es el registro de seguridad o control relacionado con el *RC*, es controlado por el originador y permite indicar aspectos de seguridad para tener en cuenta específicamente entre originador y destinatario, más que sobre el correo en general. Este registro debe ser verificado por el cliente y por el servidor asociado a la cuenta. Si el usuario originador ha establecido que justo de esa cuenta destinataria requiere confirmación de lectura, lo especifica en este registro. Podría darse el caso de que el resto de los destinatarios no

requieran un aspecto de control específico, entonces no se especifica en el registro de control asociado al correo (SR).

- @ **Sign:** Es la firma digital.
- @ **Mc:** Es el mensaje, definido como un conjunto de caracteres en texto plano. La implementación podría utilizar las Extensiones Multipropósito de Correo de Internet (MIME) (Borenstein & Freed, 1993), pero se deja a decisión del implementador.
- @ **RC_{osk}:** Es la clave privada del originador.
- @ **Ac*:** Es el conjunto de ficheros adjuntos al correo, que puede ser ninguno o muchos.
- @ **Cc:** Es un registro que indica las características del correo como son la fecha de envío, el asunto o comentario y otros campos que el usuario desee incluir para tener una referencia del correo.
- @ **X:** Es el algoritmo de cifrado utilizado.
- @ **Sk:** Es la clave secreta utilizada por el cifrador.
- @ **Ds:** Es la firma digital del correo, generada por la cuenta que envía el correo. Esto se genera solamente cuando el correo está listo para ser enviado. Puede suceder que existan muchas cuentas involucradas en el mensaje, entonces se debe esperar a que todos firmen (*Sign* en *RC+*) para poder ser enviado.
- @ **SR:** Es el registro de seguridad y control específico del correo. Permite indicar si el correo es certificado, si es necesario una notificación (entrega, borrado o lectura), si requiere confirmación de lectura, si requiere autorización de borrado o si requiere autorización de reenvío.

Un correo electrónico pasa por diferentes estados y dichos estados son controlados por cada uno de los componentes por donde transita. El modelo de correo propuesto en este trabajo define de forma general cómo debe ser implementado el correo y los aspectos de estructura y seguridad que debe tener en consideración a la hora de manipular un mensaje dentro de una comunicación; pero diferentes implementaciones de dicha arquitectura podrían utilizar tecnologías diferentes para su desarrollo.

Un ejemplo claro de esto es que las relaciones de comunicación (entre remitente y destinatario) deberían quedar ocultas para cualquiera que de forma mal intencionada accediera al correo. En la propuesta se declara que las relaciones de comunicación deberían estar cifradas (indicado con la X en la tupla), pero podrían utilizarse diferentes esquemas de cifrado para implementar dicho requerimiento.

Usuario (U)

Es la persona o institución física que tiene bajo su control una o más cuentas y posee un identificador único en el sistema, que le permite realizar las acciones comunes para la gestión de correo electrónico asociadas a las cuentas. Formalmente un usuario U es una tupla ($Upk, Usk, Uact+, Upd$) donde:

-  **Upk:** Es la identificación única del usuario asociada a la clave pública generada cuando se crea la cuenta.
-  **Usk:** Es la clave privada asociada al usuario que le permite realizar la gestión segura de las cuentas por medio del cifrado y firma digital.
-  **Uact+:** Es la cuenta (*act*) asociada al usuario. El símbolo **(+)** indica que puede tener asociada una o más cuentas.
-  **Upd:** Es un registro que contiene los datos personales (*pd*) del usuario y los certificados de persona física necesarios para alguna gestión de identificación.

Usuario Corporativo (UE)

Es un usuario que representa una empresa o grupo y que tiene bajo su control y supervisión una serie de usuarios. Este usuario se diferencia de un usuario personal porque puede crear otros usuarios y gestionarlos, mientras que un usuario personal no. Un usuario corporativo (UE) se define como una tupla ($U+, UEcf$) en donde:

-  **U+:** Es una lista usuarios distribuidos con tres roles muy diferenciados: principal o administrativo (*UEp*), delegado (*UEd*) y supervisado (*UEs*).
-  **UEcf:** Es el registro de configuración que contiene todos los detalles de seguridad y control de la cuenta corporativa, las cuentas delegadas y las cuentas supervisadas.

Usuario Personal (UP)

Un usuario personal (UP) es una persona física que se registra en el sistema como usuario y que representa una única entidad que utiliza todas las funciones de correo electrónico seguro. UP y UE realizan todas las actividades de generación y consumo de correo electrónico y son los encargados de modificar el estado de BBM en relación con el contenido del correo. BBM considera la información de UP como confidencial y es él mismo quien determina el control y acceso a la misma, a diferencia de UE que tiene el control de todas las cuentas de la que es propietario.

Cuenta

Es la representación digital del usuario en el sistema. Todas las comunicaciones se realizan entre cuentas y son contenedores de buzones donde se almacenan los correos. Una cuenta se define como una tupla $(U, Cpk, Csk, Ub+, Ucf)$ en donde:

- ☞ **U:** Es el usuario propietario de la cuenta.
- ☞ **Cpk:** Es la clave pública asociada a la cuenta y utilizada para las operaciones a nivel de la cuenta.
- ☞ **Csk:** Es la clave privada asociada a la cuenta que le permite realizar la gestión segura sobre los buzones por medio del cifrado y firma digital.
- ☞ **Ub+:** Es el buzón asociado a la cuenta. El símbolo (+) indica que puede tener asociado uno o más buzones.
- ☞ **Ucf:** Es un registro de configuración donde se almacenan todos los detalles de seguridad de la cuenta.

Autor

Es el papel que puede desempeñar un usuario en una relación de comunicación RC . Es el responsable de la información contenida en el correo (Mc) y lo refleja en la comunicación firmando ($Sign$) el mensaje y los adjuntos (Ac) en cada correo en el que participa. En una comunicación de correo electrónico puede haber más de un autor y es requerido que cada uno de ellos firme el correo autorizando su contenido.

Remitente

Es el usuario responsable de enviar un correo electrónico (*Ce*) y puede tener el papel de autor si es el único originador de la comunicación. Los correos tienen un único remitente, a diferencia de los demás roles, que pueden ser desempeñados por diferentes participantes.

Destinatario

Es el usuario al que se le dirige la comunicación de correo (*Ce*) y está autorizado a la recepción y acceso de este. Ningún otro usuario, distinto al destinatario, puede tener acceso de lectura al correo recibido, ya que es el que está en posesión de la clave privada (*Sk*) utilizada para descifrar el correo. El destinatario es el único participante que actúa de forma pasiva en el envío, pero una vez recibido el correo, puede tener una participación dentro del ciclo de estados por los que pasa el mensaje.

Contacto (Ct)

Un contacto es una tupla que relaciona dos cuentas, el originador (O) y el destinatario (D). Se representa como una tupla (Cid, O, D, TS, Exp, St, TA, AD, Ct, Rjc) en donde:

- ☒ **Cid:** Es el identificador único del contacto. Ese valor es generado por BBM y se asigna al contacto cuando este es creado.
- ☒ **O:** Es la cuenta del usuario que solicita el contacto, denominado originador.
- ☒ **D:** Es la cuenta del usuario al que se le ha solicitado el contacto, denominado destinatario.
- ☒ **TS:** Hace referencia a una marca de tiempo (*TimeStamp*) que indica el momento exacto (fecha y hora) en el que se ha solicitado dicho contacto.
- ☒ **Exp:** Hace referencia a una fecha de expiración que puede ser asignada y modificada solo por el destinatario. Indica la fecha en la que el contacto cambia de estado y pasa a un estado “no aceptado”. Después de dicha fecha, desde la cuenta originadora no se podrán enviar ninguna comunicación de correo al destinatario.
- ☒ **St:** Es el estado del contacto y determina el tipo de comunicación que puede ser realizada entre las dos cuentas.
- ☒ **TA:** Es el token de autorización. Almacena la firma del destinatario indicado que ha aceptado el contacto. Sin ese token no es posible realizar ningún tipo de comunicación entre las cuentas.
- ☒ **AD:** Son los datos adicionales que pueden ser requeridos por el destinatario para la aceptación del contacto.
- ☒ **Ct:** Es el tipo de contacto establecido. Puede tener dos valores *Personal* o *Bussiness*, indicando que puede ser un contacto personal o profesional.

- ✉ **RJc:** Es el código de rechazo o aceptación. Si el contacto es rechazado, indica por medio de un código la razón del rechazo y la posible indicación de más información que debe ser suministrada por el originador en el campo AD.

Actor

Un actor es una entidad externa o interna que demanda u ofrece algún tipo de funcionalidad en el sistema BBM. En BBM existen actores como son los usuarios, los proveedores, los servidores, el integrador y la red *blockchain*, entre otros, que serán detallados en la descripción de la arquitectura. Todos estos demandan u ofrecen algún tipo de función que es regulada por el sistema, toda la actividad que se desarrolle dentro del entorno de BBM está bajo el control y supervisión de la red y de los protocolos.

Cada vez que un actor participa en la ejecución de una actividad, su estado puede cambiar, por lo tanto, si los actores no quedan en un estado válido, los cambios realizados durante la ejecución de la actividad deben ser revertidos o simplemente no aceptados. En caso contrario, los cambios serán almacenados y la máquina de estados de BBM junto con sus actores y componentes, quedarán en un estado válido.

Proveedor de almacenamiento

El proveedor es un actor externo (puede ser una organización o un usuario en particular) que proporciona algún servicio al sistema, en el caso de BBM, proporciona servicios de almacenamiento de correo. Este actor provee un servidor que participa de una red de almacenamiento distribuido para tales fines y son los usuarios del sistema quienes hacen uso de ese servicio.

Buzón

Es un recipiente o contenedor donde se almacenan los correos y que se puede identificar de forma única. Una cuenta puede gestionar más de un buzón. Cuando se envía un correo, realmente se está enviando a un buzón de una cuenta propiedad de un usuario.

Formalmente, un buzón B se define como una tupla (Bid, Bce^*, Bcf) donde:

-  **Bid:** Es el identificador único del buzón.
-  **Bce*:** Es la lista de correos almacenados en el buzón. En esta propuesta un correo electrónico no está estrechamente relacionado con un buzón, ya que este es simplemente un contenedor. Un correo podría ser relocalizado en otro buzón sin realmente afectar para nada su contenido. En los sistemas tradicionales de correo electrónico un correo y su buzón son indivisibles, lo que complica mucho la gestión del correo cuando se tienen varios buzones.
-  **Bcf:** Es el registro de configuración donde se almacena toda la información de seguridad del buzón.

Protocolo distribuido

Es un conjunto de reglas formales que se utilizan como guía para realizar una serie de actividades, pero que es validado por un grupo de participantes y ejecutado en una red distribuida. En el contexto de esta propuesta, es un contrato inteligente almacenado en una *blockchain* en el que se definen las reglas que deben seguir los participantes para lograr o completar un objetivo en el sistema.

Por ejemplo, para registrarse en el sistema o enviar un correo, se ejecuta o se sigue un protocolo distribuido y éste valida las reglas en consenso para lograr la seguridad deseada. Formalmente un protocolo distribuido es una máquina de estados donde la ejecución de una transición es consensuada por los nodos participantes. Los actores y nodos de la *blockchain* determinan el estado final de BBM. Se define como $PD = (PDe+, PDso+, PDst+, PDr, PDac)$ donde:

-  **PDe+:** Es un grupo de entidades participantes cuyo cambio de estado depende de la ejecución del protocolo.
-  **PDso:** Es el estado inicial de las entidades participantes.
-  **PDst:** Es un grupo de estados por los que pueden pasar las entidades participantes.
-  **PDr:** Es una lista de transiciones cuya ejecución depende de unas reglas muy definidas.
-  **PDac:** Es un algoritmo de consenso donde varios nodos verifican si se cumplen las reglas establecidas y se ponen de acuerdo para certificar la transición.

En esta propuesta existen varias entidades cuyo comportamiento puede ser modelado como una máquina de estados como son:

- 🖥 El correo electrónico (Ce).
- 🖥 El usuario (U).
- 🖥 Las cuentas (C).
- 🖥 Los buzones (B).
- 🖥 Los servidores (S).
- 🖥 BBM en sí misma se comporta como una máquina de estados.

III. LISTA DE REQUISITOS PARA LA GESTIÓN SEGURA DE CORREO ELECTRÓNICO

Una aplicación que proporcione servicios de gestión de correo electrónico es intensiva en interacción con los usuarios. Por lo tanto, hay que definir los requisitos funcionales del servicio que le dan valor al usuario y los aspectos de seguridad que le dan calidad a los procesos que ejecuta y a los datos que almacena. A lo largo del desarrollo de este trabajo, se han identificado una serie de requisitos que debe cumplir una solución de correo electrónico integral para lograr asegurar los aspectos de seguridad y uso por parte de los usuarios. Estos requisitos sirven de base para la definición de la arquitectura propuesta y sus protocolos de comunicación.

El cumplimiento de estos requisitos tiene como objetivo final alcanzar los objetivos generales y específicos de esta investigación. Se han definido estos requisitos basándonos en las necesidades de los usuarios y los aspectos de seguridad que deben ser considerados a la hora de proporcionar una solución de correo electrónico seguro: confidencialidad, integridad, no repudio, autenticación, disponibilidad, control, auditoría, usabilidad e integración.

Debido a la naturaleza del correo electrónico, centrada en el usuario, es necesario diferenciar los requisitos funcionales de los usuarios cuando utilizan una aplicación de correo electrónico, que suelen ser las capacidades esenciales para que un usuario pueda operar normalmente con el correo, de los requisitos no funcionales, como la seguridad, que son los atributos deseados cuando se implementa un requisito funcional específico. Por ejemplo, un usuario espera que, al utilizar el correo electrónico, pueda “enviar un correo”, eso es un requisito funcional y básico; pero para que el usuario pueda “enviar un correo confidencial”, surge un requisito de seguridad adicional no funcional que le proporciona calidad al requisito funcional “enviar un correo”.

La Tabla 13 muestra un resumen de todos los requisitos de seguridad que se han identificado, junto con la proporción de los aspectos considerados. Se puede ver en la tabla que el control es uno de los aspectos más requeridos, seguido de la confidencialidad y el no repudio. La autenticación y los aspectos de integración, aunque son importantes, están considerados en menos proporción. Cada una de las filas en la tabla representa un requisito, las columnas representan los aspectos considerados en cada requisito y la última columna hace referencia a qué componente está específicamente dirigido.

Tabla 13. Lista de requisitos identificados con los aspectos de seguridad considerados en cada uno de ellos.

Aspectos considerados											
Requisitos	Confidencialidad	Integridad	No repudio	Autenticación	Disponibilidad	Control	Auditoría	Usabilidad	Integración	Total Aspectos	Aplica a
REQ001	X	X	X	X	X	X	X	X	X	9	General
REQ002	X	X								2	Buzón
REQ003						X				1	Buzón
REQ004					X	X				2	Buzón
REQ005						X		X		2	Buzón
REQ006					X					1	Buzón
REQ007	X					X				2	Usuario
REQ008	X					X				2	Usuario
REQ009						X				1	Usuario
REQ010						X		X		2	Usuario
REQ011						X				1	Correo
REQ012						X	X			2	Correo
REQ013						X				1	Correo
REQ014				X						1	Usuario
REQ015						X				1	Usuario
REQ016				X						1	Correo
REQ017			X							1	Correo
REQ018						X				1	Correo
REQ019			X			X				2	Correo
REQ020			X			X				2	Correo
REQ021			X							1	Correo
REQ022			X			X				2	Correo
REQ023	X					X				2	Buzón
REQ024	X	X				X				3	Correo
REQ025	X	X				X				3	Correo
REQ026								X	X	2	Correo
REQ027						X		X	X	3	Correo
REQ028				X						1	Usuario
REQ029								X		1	Buzón
REQ030	X					X		X		3	Correo
REQ031								X	X	2	Correo
REQ032								X		1	Correo
REQ033						X				1	Cuentas
REQ034						X				1	Cuentas
REQ035						X				1	Cuentas
REQ036						X				1	Cuentas
REQ037						X	X	X		3	Cuentas
REQ038	X	X			X	X				4	Cuentas
REQ039	X					X				2	Correo
REQ040						X	X			2	Correo
REQ041				X				X		2	Cuentas
REQ042	X	X					X			3	Usuario
REQ043	X	X	X		X	X	X			6	Correo
REQ044	X	X	X		X	X	X			6	Correo
Total	13	8	8	5	6	31	7	11	4	93	
%	13,98 %	8,60 %	8,60 %	5,38 %	6,45 %	33,33 %	7,53 %	11,83 %	4,30 %	100,00%	

Fuente: Elaboración propia.

Los requisitos descritos aquí van en dirección de dotar a los requisitos funcionales de las características de seguridad analizadas y descritas en este trabajo. Los requisitos funcionales básicos se incluirán para facilitar el entendimiento de los requisitos no funcionales de seguridad y se omitirán aquellos requisitos básicos que no requieran aspectos de seguridad claves. Los siguientes apartados contienen una descripción de los requisitos identificados en este trabajo bajo el enfoque especificado en este párrafo y una explicación de cómo se aborda la solución con la propuesta.

REQ001. Es necesario contar con una solución de correo electrónico seguro.

Descripción:

Considerado el principal requisito y el más alto dentro de la jerarquía de requisitos presentados. Los demás requisitos se desprenden de este, con el objetivo de sumar entre todos para dar cumplimiento al requisito principal. Basado en los objetivos de esta investigación se puede describir este primer requisito como: “Se requiere una solución de correo electrónico segura basada en una arquitectura flexible, integrable y de fácil adopción por parte de los usuarios para dar solución al problema de la seguridad del correo electrónico proporcionando servicios de confidencialidad, integridad, disponibilidad, no repudio, autenticación, control, auditoría, usabilidad e integración”.

Es importante poder cumplir el objetivo principal para que la solución sea:

- **Flexible:** La solución debe facilitar la configuración de las características de seguridad, de forma que el **sistema** facilite al usuario adoptar una actitud despreocupada o paranoica en su gestión.
- **Integrable:** Asumiendo que la realidad es que una solución sea adoptada progresivamente, es necesario dotar al sistema para que pueda comunicarse con otros usuarios que utilicen otras soluciones convencionales.
- **Fácil adopción:** La usabilidad es un problema a la hora de abordar una solución de seguridad en el contexto del correo electrónico. Una solución compleja, por muy buena que sea, si exige demasiados conocimientos técnicos al usuario y complica la operativa usual a la que están acostumbrados los usuarios, tiende a no utilizarse o utilizarse por un conjunto de usuarios muy específico.

Propuesta:

Además de estas características generales, se persigue que la solución de respuesta a los problemas de seguridad del correo electrónico analizados en este trabajo a través de los siguientes servicios: confidencialidad, integridad, disponibilidad, no repudio, autenticación, control, auditoría.

- **Confidencialidad:** Se propone que la confidencialidad sea abordada en varios niveles. A nivel de la cuenta o usuario, es deseable que solamente las personas autorizadas puedan tener conocimiento acerca de los detalles de usuario y una vez esa autorización sea

revocada, ese conocimiento desaparezca. Ese conocimiento debería ser temporal hasta el momento de la revocación. A nivel de transporte, es deseable una confidencialidad para los participantes en la que solamente las personas o entidades externas autorizadas puedan ver qué usuarios están participando en el transporte del correo electrónico. Por último, a nivel de mensaje, se requiere que solamente el remitente y destinatarios sean los conocedores del contenido del correo electrónico, característica en la que más se enfocan las soluciones que ofrecen confidencialidad.

- **Integridad:** Se propone que la integridad sea abordada a nivel del mensaje, la comunicación, la configuración de las cuentas y los buzones para evitar su manipulación.
- **No repudio:** Se propone que además de proporcionar la garantía de que los participantes de la comunicación de correo electrónico no puedan negar su participación, se ofrezca la garantía de que los participantes no puedan negar haber realizado algunas acciones como son la lectura, recepción o borrado de los mensajes que gestiona.
- **Autenticación:** Esta propiedad garantiza que el autor de la comunicación es quien dice ser y no otro. Se propone que la autenticación dentro del sistema de correo se realice de dos maneras. Una de ellas es que el usuario se autentifica cuando puede demostrar que posee las credenciales para realizar todas las tareas en el sistema (algo que sabe). La otra forma es que el usuario pueda certificar que es quien dice ser y no una persona falsificando una identidad, de forma que entre dos usuarios que deseen realizar comunicaciones de correo electrónico, se pueda exigir en algunos casos, la certificación física por parte de un tercero de confianza (algo que tiene).
- **Disponibilidad:** Los ataques contra la disponibilidad del sistema (ataques DoS) son muy frecuentes en las plataformas centralizadas y en especial, como se ha demostrado en la sección de motivación de este trabajo, muy costosos para el caso de los servicios de correo electrónico. Se propone un sistema que esté preparado para este tipo de ataques o que para los atacantes sea muy costoso realizarlos. Las *blockchain* son más difíciles de destruir que las arquitecturas convencionales, no obstante, no están exentas de los ataques DDoS y, por lo tanto, sigue siendo necesario implementar algunas medidas como los sistemas de alta disponibilidad.
- **Control:** Se entiende como control a la capacidad que tiene un usuario para definir cuándo, cómo y quiénes pueden acceder a sus correos. En las diferentes propuestas analizadas en el capítulo 2 y las comparativas desarrolladas entre ellas, los mecanismos de control estaban ausentes. La presente propuesta consiste en implementar esta funcionalidad con mecanismos distribuidos y seguros que permitan definir controles de acceso a los correos y sea una funcionalidad gestionada por el mismo usuario.

- **Auditoría:** Otra de las características olvidadas en las soluciones de correo electrónico es la posibilidad de poder trazar los correos. Las opciones incluidas en la mayoría de las soluciones pasan por gestionar hilos de correo para hacer seguimiento a los correos similares o que han sido generados por reenvíos de correos anteriores. Otra característica deseada es poder recibir notificaciones de algunas acciones como son la recepción, la entrega, el borrado, etc., que en algunas ocasiones es necesario.

Disponer de una solución que proporcione control, no repudio y además trazabilidad de algunas acciones, proporciona un gran abanico de posibilidades para realizar auditorías por parte de las empresas, a los cuales les interesa, ya que el correo electrónico es un activo de la organización y su mal uso podría acarrear consecuencias negativas a la organización. Quizás para un usuario en particular algunas de estas características no le sean de interés, pero la solución que se propone está dirigida para la implementación de sistemas de correo que abarquen las diferentes tipologías de usuario.

REQ002. Un buzón debería solamente ser accedido por el usuario propietario o por un usuario autorizado.

Descripción:

Este requisito está enfocado en proporcionar confidencialidad a nivel del contenido de los correos almacenados en el buzón propiedad de un usuario. El buzón debe estar lo suficientemente protegido y en caso de que un tercero no autorizado se apropiara de él no debería poder acceder a la información que contiene y si la modificara, debería poderse identificar dicha modificación. Acceder al buzón de forma no autorizada puede tener los siguientes objetivos: copiar información confidencial, alterar contenido, borrar correos, apropiarse del buzón permanentemente, utilizar el buzón para enviar correos no autorizados sin ser identificado y secuestro del buzón, entre otros.

Propuesta:

En esta propuesta los buzones almacenan los correos cifrados con las credenciales del usuario, de manera que solamente el que está en posesión de las credenciales puede acceder y gestionar el buzón. Si un usuario no autorizado se apropiara del buzón, no podría acceder a los correos almacenados, a menos que esté en posesión de las credenciales para descifrar los correos.

Se diferencia el buzón como contenedor destino de los correos del buzón local al que accede el usuario para la gestión. En dicho contenedor, se puede tener o no una copia de los correos y no todos los correos estarán cifrados a petición del usuario. Se pretende que el usuario tenga control total sobre este aspecto, de forma que un usuario pueda determinar el alcance de la confidencialidad a nivel del buzón, pudiendo seleccionar los correos que la necesiten.

REQ003. Un usuario debería gestionar sus buzones sin intervención de un tercero, pudiendo incluso modificar de forma transparente para los remitentes, el destino de los correos que recibe.

Descripción:

Las soluciones de correo electrónico existentes hoy en día gestionan sus propias cuentas utilizando como nomenclatura para el identificador de la cuenta un nombre de dominio, representando los nombres de los buzones como “nombrebuzon@dominio”. El problema de este esquema es que complica la gestión, ya que al final un usuario que está en posesión de muchas cuentas, con diferentes dominios, tiene todo su correo disperso entre diferentes compañías. Cada compañía proporciona diferentes servicios que no son controlados por el usuario, lo que hace difícil controlar los aspectos de seguridad del correo y de su información por parte de los propietarios.

Este requisito exige que los buzones sean controlados totalmente por el usuario y ningún tercero tenga control sobre el contenido, los datos de propiedad asociados a ellos y su localización. Se requiere que un usuario pueda cambiar la localización de un buzón sin necesidad de comunicar a los contactos que envían correo a dicho buzón.

Propuesta:

Se propone un sistema en el que los datos del usuario, la localización de los buzones y su almacenamiento pueda ser gestionado por terceros, pero sin comprometer el control y la seguridad de los mismo. Se plantea separar la cuenta de usuario de los buzones. La cuenta del usuario contendrá los datos de configuración, permisos, contactos y aspectos de control de los buzones. Un usuario podría tener varios buzones y asociar un servicio de almacenamiento a cada uno. Como la cuenta está separada del buzón, el usuario podría cambiar de proveedor y modificar su localización sin afectar a sus contactos. Para los contactos dicho cambio sería transparente.

REQ004. El usuario debería gestionar la localización del buzón sin afectar los servicios que tiene asociados.

Descripción:

Como se ha comentado en el REQ002, el buzón puede ser gestionado por un tercero, pero no debería controlarlo. Actualmente, si se tiene una cuenta de Gmail, por ejemplo, todos los correos almacenados en ese buzón no pueden ser fácilmente transportados a otro proveedor y si se decide cambiar de compañía, las opciones disponibles son:

- ➡ Realizar una copia de seguridad para no perder la información.
- ➡ Notificar a todos nuestros contactos de cambio.
- ➡ Cambiar tarjetas de presentación si se tienen esos datos en la tarjeta.
- ➡ Modificar la *web* si se presta un servicio y se reciben notificaciones.
- ➡ Modificar todos los servicios en Internet que tienen asociada esa cuenta.
- ➡ Perder los hilos de correo ya que no se podrán continuar en otra cuenta, que tendrá un nombramiento totalmente diferente, entre otras cosas.

En conclusión, se requiere que la localización del buzón pueda modificarse sin afectar la gestión y los servicios activos asociados al correo.

Propuesta:

Se propone que el usuario tenga disponible varias opciones de ubicación para el buzón. Los buzones pueden estar de forma local, en un directorio compartido o en un servidor de un tercero que preste el servicio de alojamiento de buzón. Independientemente del tipo de buzón, el acceso debe seguir los protocolos de almacenamiento descritos en esta propuesta. La localización del buzón estaría desligada completamente del contenido del correo, pudiendo el usuario seguir gestionando el correo sin verse afectado por un cambio de proveedor.

REQ005. Un usuario debería poder modificar la ubicación de sus buzones sin autorización de terceros.

Descripción:

Para un usuario, tener control total sobre un buzón implica que en cualquier momento pueda cambiar la ubicación de este sin necesidad de requerir una autorización del tercero donde lo tiene localizado. El proceso para realizar el cambio debería ser amigable y transparente para el usuario, exigiendo muy poco conocimiento y participación de su parte. Esta característica permitiría al usuario gran flexibilidad a la hora de gestionar sus correos.

Propuesta:

Se propone un sistema donde el usuario pueda cambiar la configuración de su buzón y de forma automática, siguiendo los protocolos del sistema, la plataforma realice los cambios sin afectar al usuario, a sus contactos o, en caso de tener alojado el buzón en un tercero, no requerir permiso o autorización de su parte.

REQ006. Los buzones y su contenido deberían estar accesibles el usuario cuando lo necesite.

Descripción:

En condiciones normales (por ejemplo, si no ha habido un bloqueo del buzón) un usuario debería poder acceder a los correos que ha recibido y debería tener el buzón disponible cuando lo requiera. La disponibilidad del correo es esencial para los individuos y para las empresas que lo utilizan para la gestión diaria de sus actividades.

Propuesta:

Se propone un sistema basado en la distribución de los buzones. El uso de buzones controlados por el usuario y distribuidos facilita esa labor y, dependiendo de la necesidad del usuario y de los riesgos de su información, debería poder contar con un sistema que le proporcionara al usuario la opción de seleccionar o no una alta disponibilidad.

REQ007. Un usuario destinatario de un correo debería poder autorizar a los usuarios remitentes antes de recibir cualquier correo.

Descripción:

Los usuarios deberían recibir correo de remitentes autorizados. Este requisito está enfocado en atacar el problema del correo no deseado. Los protocolos de seguridad deberían evitar que un usuario no autorizado envíe correo electrónico a otro usuario sin existir una autorización de parte del destinatario. Este requisito también declara la necesidad de un protocolo de autorización previo al intercambio de correos.

Propuesta:

Se propone un sistema de autorización entre usuarios para poder realizar comunicaciones de correo. Un usuario si lo desea podría deshabilitar esa característica o determinar niveles de autorización. Se formulan varios niveles de autorización. En el primer nivel, denominado básico, se hace uso de un protocolo de autorización donde la confirmación de los usuarios se realiza sin ningún tipo de intercambio de información adicional. En el siguiente nivel, denominado medio, un usuario podría solicitar más información a la otra cuenta, como, por ejemplo, el nombre completo o una identificación simple. El último nivel, denominado alto, consiste en requerir verificación de identidad al usuario que hace la solicitud, y este debe acreditar su identidad, por ejemplo, suministrando un certificado de persona física.

REQ008. Es necesario que los usuarios puedan controlar a qué usuarios pueden enviarles correo electrónico.

Descripción:

Cuando un usuario envía un correo, debería poder exigir que el usuario destinatario tenga que autorizarlo. En este caso la intención de autorización viene de parte del remitente y no del destinatario, a diferencia del caso del REQ007. Un remitente debería de tener la posibilidad de configurar si puede o no enviar correos a una cuenta abierta.

Propuesta:

Se propone que un usuario pueda configurar la forma general de gestionar las comunicaciones autorizadas. Esto se deberá establecer en un registro de configuración al que todas las entidades de gestión deberían acceder y consultar cuando están ejecutando algún protocolo que requiera esa información. En dicho registro debería de haber un campo indicando que el usuario desea o no que todos los envíos sean autorizados por el usuario destinatario.

REQ009. Un usuario debería tener flexibilidad a la hora de gestionar las autorizaciones de los remitentes, incluso pudiendo autorizarlos a todos o ninguno.

Descripción:

Basado en el REQ007 y el REQ008, pero de forma más general, este requisito indica que un usuario debería poder abrir su cuenta para que cualquier remitente pudiera enviarle correo sin previa autorización o, por el contrario, que solamente lo hicieran usuarios autorizados.

Propuesta:

Se propone que además de configurar los niveles de autorización para todos los remitentes, un usuario pueda determinar de forma general y específica cómo los remitentes pueden enviarle el correo. Se plantean dos formas de alcanzar este requisito. La primera es creando contactos de un nivel básico y con autorización automática (sin intervención humana), lo cual permite que a todos los que hagan una solicitud de contacto, se les conceda la autorización. La otra forma es utilizar el registro de configuración de seguridad con un campo que indique que la cuenta está abierta y, por lo tanto, no sea necesario verificar autorizaciones.

REQ010. Es necesario que los usuarios tengan la opción de autorizar mutuamente a los remitentes.

Descripción:

La autorización mutua consiste en solicitar autorización de envío de correo, enviando en la misma solicitud su propia autorización. Este requisito está orientado a los usuarios que deseen al mismo tiempo los requisitos REQ007 y REQ008 dentro del mismo proceso de autorización y no dos procesos separados.

Propuesta:

Este requisito se puede alcanzar de dos formas: la primera, que cuando el usuario solicite autorización a un destinatario, pueda adjuntar su propia autorización para recibir correo; la otra forma es el uso de un registro de seguridad donde el usuario pueda indicar si desea o no que todas las solicitudes sean solicitudes con autorización mutua.

REQ011. El sistema debería ofrecer la opción de asegurar la entrega de los correos enviados por un usuario cuando este considere oportuno.

Descripción:

Este requisito tiene como objetivo el no repudio por parte del destinatario, ya que ofrece al usuario la posibilidad de que el destinatario no pueda negar que ha recibido el correo. De igual forma, es un mecanismo de control, ya que para algunos usuarios esta opción le daría el control total sobre el estado de algunos correos. También, para ciertos usuarios corporativos, permite realizar seguimiento del correo electrónico que gestionan sus empleados para verificar si se están siguiendo los procesos de negocio.

Propuesta:

Se propone que un usuario pueda activar la notificación de entrega cuando deposita el correo o en todas las comunicaciones activando la opción a través de un campo en el registro de seguridad asociado al usuario.

REQ012. Debe existir la opción de activar notificaciones de lectura y eliminación de los correos electrónicos.

Descripción:

El seguimiento de los correos por parte del usuario es una característica de seguridad y control. Seguridad porque permite que el usuario se proteja de cualquier daño al no estar informado de las acciones sobre el correo que supervisa y gestiona. Es una característica de control porque el usuario puede activar o no las notificaciones a su elección, dependiendo de sus necesidades. En el caso de estas dos notificaciones, el destinatario debe tomar parte activa en la notificación y la propuesta debe garantizar que dicha notificación no dependa de la voluntad del destinatario.

Propuesta:

Se propone que esta característica se pueda activar para comunicaciones en concreto o para todas las comunicaciones. Para notificaciones en concreto, el usuario activaría la opción cuando deposita el correo. Para las notificaciones en general, se debería activar dicha opción en el registro de seguridad del usuario. En cuanto al proceso de notificación, es necesario involucrar dentro de la solución a los agentes de transporte y entrega de correo, que deberían responsabilizarse de dicha tarea cuando ejecuten los protocolos de envío y entrega de correo.

REQ013. El usuario debería tener la capacidad de eliminar un correo que esté bajo su propiedad o supervisión sin violar reglas de seguridad.

Descripción:

La eliminación de un correo electrónico puede violar las reglas de seguridad de una organización si se realiza sin el consentimiento de los propietarios y, por ello, en algunos casos, esta acción debería ser realizada con autorización. La eliminación no autorizada de información es un riesgo a la que están expuestas las organizaciones y el ataque sobre un sistema de correo electrónico puede estar dirigido con ese objetivo. Es por esa razón que se consideró que este requisito es indispensable para proporcionar una seguridad integral. La eliminación de un correo debería ser autorizado por el usuario que lo creó si él lo considera

correcto y el sistema no debería permitir el borrado deliberado de correos sin autorización previa, o aceptarlo siguiendo unas reglas predefinidas.

Propuesta:

Se propone que el borrado siempre requiera autorización y sea controlado por los usuarios y que exista la posibilidad de solicitar autorización a un remitente si el destinatario quiere eliminar el correo. Para alcanzar este objetivo deben involucrarse los agentes de almacenamiento y entrega de correo dentro del esquema de seguridad de esta propuesta. Un agente de almacenamiento de correo debería de estar informado si el usuario requiere por temas de seguridad la no eliminación del correo y, por consiguiente, considera que es su responsabilidad no eliminarlo.

REQ014. El usuario debería tener la opción de verificar la identidad de otro usuario.

Descripción:

Este requisito es una especialización del REQ007. Una de las razones del correo no deseado es, como su nombre lo indica, que un usuario no desea que ese remitente desconocido le envíe correos. Verificar la identidad es un requisito necesario y obligatorio en muchos casos y el sistema debería proporcionar la opción cuando sea requerido por parte de un usuario. En caso de un usuario de empresa, su identificación podría realizarse por medio de algún tipo de vinculación con la organización.

Propuesta:

Se propone seleccionar el nivel más alto de autorización que es la verificación de identidad. Este requisito se describe por separado porque para poder realizar esa verificación será necesario el uso de un protocolo independiente basado en verificación de certificados de persona física. En este caso, la idea es que la entidad certificadora asocie el identificador utilizado en las comunicaciones con la identidad de la persona que la usa. Un usuario corporativo debería de solicitar dicho certificado a sus empleados para que legalmente estén vinculados con la organización. El usuario corporativo debería identificarse con un certificado digital de empresa o certificado de representante cuando otro usuario le solicite dicho certificado.

REQ015. Un usuario debería poder revocar cualquier autorización de contacto con otro usuario.

Descripción:

Un usuario debería poder anular cualquier autorización de contacto realizada o tener la posibilidad de extinguir la autorización de forma automática entre las dos partes. Mientras la revocación exista, ninguno de los participantes debería tener contacto. Esto significa que es necesario realizar una nueva solicitud de contacto si se desea una nueva comunicación.

Propuesta:

Se propone que, en el momento de la autorización del contacto, se pueda añadir un periodo de expiración para que los agentes responsables de la comunicación puedan consultarlo y hacerlo efectivo cuando se cumpla. El contacto debería quedar revocado en el momento de cumplirse la fecha de expiración o debería poderse cambiar de forma manual a estado “desactivado”. Una revocación deja al usuario revocador protegido de posibles futuras comunicaciones.

REQ016. Un usuario no debería poder enviar correo electrónico en nombre de otro usuario.

Descripción:

No se debería permitir la suplantación de identidad dentro del sistema. Un usuario no debería poder realizar ninguna acción en el sistema con su identificador en nombre de otro usuario. Para poder realizar cualquier actividad en nombre de otro usuario, tendría que ser autorizado o estar en posesión de las credenciales para hacerlo. El sistema debería proporcionar los medios y las herramientas para no permitir este tipo de actividad.

Propuesta:

Se propone que en todas las acciones permitidas por el sistema el usuario que las realiza deba demostrar que es quien dice que es. Se utilizará un esquema de clave pública donde el usuario deberá demostrar que está en posesión de la clave privada en cada acción que realiza en el sistema. No será posible suplantar a otro usuario si no se está en posesión de sus claves privadas. Un usuario no podrá negar la participación en ninguna comunicación y podrá tener la seguridad que controla todas las acciones de su usuario siempre y cuando no pierda las claves privadas.

REQ017. El destinatario de un correo debería tener la certeza de que el remitente no ha sido suplantado.

Descripción:

Este requisito está muy relacionado con el REQ016, pero tiene un matiz diferente. Que el sistema proporcione al usuario la confianza de que nadie suplantarán su identidad utilizando su identificador para sus fines maliciosos, no significa que el destinatario no necesite la certeza de quien ha sido el remitente. Si el usuario tiene la seguridad de que nadie puede suplantar su identidad, debe tener la certeza de que el otro usuario confía de la misma manera en el sistema.

Propuesta:

Para proporcionar las mismas garantías a todos los participantes, se propone que toda comunicación haga uso de mecanismos criptográficos como la firma digital y el cifrado, de tal forma que el remitente de una comunicación de correo pueda estar seguro de que el remitente sea el que ha realizado dicha comunicación.

REQ018. Un usuario debería tener la posibilidad de activar una opción para que el destinatario de un correo pueda leerlo si confirma antes su recepción.

Descripción:

Esta característica proporciona un nivel adicional de seguridad a una comunicación de correo. Es una característica de control porque el usuario remitente es quien determina el flujo del

correo basado en los estados por los que puede pasar un correo y, además, se asegura que el usuario que recibe y lee el correo es la persona a la que se lo ha enviado.

Propuesta:

Se propone que, cuando se envíe este tipo de correos, se envíe una doble comunicación. El destinatario firmará la primera comunicación, contestará al sistema enviando esa notificación firmada y el sistema le pasará la segunda comunicación con el contenido del correo. La primera comunicación contendrá los datos del remitente y una descripción de lo que se le envía. Al ser firmada esta primera comunicación, el sistema procederá a enviarle la segunda comunicación que es el correo completo.

REQ019. Un usuario debería tener la opción de delegar el envío de un correo a otro usuario sin perder la autoría.

Descripción:

El contenido de un correo debería siempre tener autoría. Es imprescindible que se diferencie el autor del remitente, aunque muchas veces sea el mismo. Un correo debería poderse enviar en nombre de otro, pero debería poder constatarse que el remitente y el autor están de acuerdo con el envío y su contenido.

Propuesta:

Se propone como solución al problema que un correo siempre esté asociado a su autor a través de su firma, lo que significa que no sería posible hablar de correo sin autor. Otro usuario podría copiar el contenido y crear un nuevo correo, pero como no podría firmar el autor original, ese correo sería un nuevo correo y, por lo tanto, no sería responsable el autor original de su contenido.

REQ020. Un usuario debería poder enviar un correo cuyo contenido esté autorizado por un tercero.

Descripción:

En la misma línea que el REQ019, un remitente podría enviar un correo en nombre de otro, como en el caso de una empresa que delega el envío de correo a un empleado en particular, pero es requerido que el usuario que autoriza ese envío quede reflejado en la comunicación. Un usuario podría, en caso de confiar en un tercero, autorizar todas las comunicaciones que el usuario hiciera en su nombre, pero también debería de tener la opción de solamente autorizar ciertas comunicaciones, en cuyo caso tendría que intervenir autorizando solo esos envíos.

Propuesta:

Algunos tipos de usuarios, en especial los corporativos, podrán tener un usuario que gestione su cuenta. El usuario gestor podría activar que todas las comunicaciones que realiza un usuario bajo su supervisión requieran su autorización. Se propone dos formas de alcanzar este requisito. La primera forma es que el usuario gestor cree una firma de autorización con caducidad. La otra forma es que el gestor firme cada correo individual cuando aparezca como autorizador.

En la primera forma, el gestor creará un ítem criptográfico firmado por él con fecha de caducidad y lo asignará a un usuario en particular, quien podrá realizar todos los envíos con su autorización hasta una determinada fecha. Por ejemplo, este sería un caso de uso del personal directivo de una empresa que delega todos los envíos de correo realizados en su nombre al personal que está bajo su cargo.

En la segunda forma, si no se utiliza ítem criptográfico, el gestor recibirá una petición de autorización por cada correo en la que aparezca como autorizado y deberá firmarlo para que pueda ser enviado.

REQ021. Los originadores (remitentes, autores y autorizadores) no deberían poder negar su participación en el rol que juegan dentro de la comunicación de correo.

Descripción:

Los otros requisitos relacionados con la autorización y declaración manifiesta de autoría o envío por parte de un usuario están relacionados con el no repudio. Estos casos hacen referencia a no repudio de autor y no repudio de envío. El requisito aquí expuesto es general, se espera del sistema que un participante, independiente del rol que tomó en la comunicación, no pueda negar su participación con dicho rol.

Propuesta:

Esta propuesta expande los roles de los usuarios como se venían utilizando en las comunicaciones de correo electrónico convencionales. El usuario simplemente era la persona que enviaba o recibía el correo y había otros participantes que, aunque actuaban de alguna forma en la comunicación, permanecían ocultos y pasivos frente a ella y, en cualquier momento, al no quedar ese rol reflejado, podían negar su participación. Esta propuesta no deja lugar a dudas del rol que tomo un participante en la comunicación. Cada participante deberá aparecer y puede ser reflejado en la comunicación dejando su firma y confirmación reflejada, y además puede ser verificado por el destinatario.

REQ022. Un usuario debería poder enviar un correo a varios destinatarios que previamente le hayan autorizado.

Descripción:

En caso de enviar correo a un grupo de usuarios, como una lista de distribución, es necesario que todos los integrantes del grupo sean contactos entre sí. Este requisito es necesario debido a que, si algunos participantes de la lista no son contactos, se podría enviar por error un correo a un destinatario no autorizado y sobrepasar los controles de seguridad del sistema.

Propuesta:

Se propone que la seguridad del sistema sea integral y que no haya conflictos cuando, en un grupo, los usuarios configuren niveles de seguridad diferentes. El sistema debe garantizar la integridad de las reglas para evitar fugas o agujeros de seguridad y, en el caso de un grupo de usuarios, el control debería ser total, ya que un usuario podría no tener ya acceso a la lista y el sistema debería tomar cuenta de ello. Todos los participantes del sistema utilizan el mismo protocolo, que está distribuido pero cuya integridad es regulada por los nodos participantes. Si el protocolo de seguridad está distribuido, todos los participantes en la comunicación hablarán en el mismo idioma y no podrá ser violada ninguna regla que ocasione que la seguridad del sistema quede comprometida.

REQ023. Un usuario debería conservar la localización de sus buzones de forma privada.

Descripción:

El buzón es el contenedor de los correos y es donde se almacenan los correos recibidos y enviados por el usuario. Uno de los problemas identificados en este trabajo es que la localización de los buzones es pública y para los atacantes es muy fácil dirigir sus ataques contra una institución o un usuario en particular. Una dirección de correo ya indica al atacante donde está almacenado su buzón y, en muchos casos (como se recomienda), la dirección ya hace mención del propietario o dueño del buzón.

Propuesta:

En esta propuesta, el remitente nunca envía el correo al destinatario, sino que lo deposita en su propio buzón y le indica al usuario donde está el buzón para que descargue el correo. El usuario remitente puede utilizar sistemas como IPFS o servicios de almacenamiento en la nube para dejar sus mensajes. El usuario destinatario simplemente recibirá la indicación de la localización del mensaje y procederá a su descarga. Para cumplir con este requisito, los datos del servidor se almacenan cifrados para que solo el usuario pueda interpretarlo y, dentro de la red de proveedores de almacenamiento, se hace uso de un protocolo donde los servidores intercambian sus localizaciones.

REQ024. Un usuario como autor de un correo debería poder controlar el reenvío que se realiza de ese correo por parte de un tercero.

Descripción:

Este requisito es bastante estricto a nivel de seguridad. Es una opción deseable que el reenvío de un correo necesite autorización debido a la protección de derechos de autor. Normalmente es algo usual reenviar correos de otros usuarios sin pedir permiso al autor, olvidando que el autor original no había dirigido inicialmente el correo al nuevo destinatario. Hay que recordar que los correos electrónicos tienen que cumplir con la normativa de RGPD (Reglamento General de Protección de Datos) y LOPD (Ley Orgánica de Protección de Datos) y que la información que se envía en un correo es confidencial y debería de permanecer de esa forma a menos que el remitente autorice su divulgación.

Propuesta:

En esta propuesta el reenvío está permitido, pero es necesario que se ejecute siguiendo un estricto control de seguridad bajo un protocolo específico. Para poder reenviar un correo, es necesario solicitar el consentimiento del reenvío del correo a los autores originales y también las verificaciones de todos los contactos con los nuevos destinatarios. En caso de no optar por el reenvío, es necesario crear un correo con un nuevo contenido.

REQ025. Los ficheros que se adjuntan a un correo electrónico deberían ser tratados como contenido del correo.

Descripción:

Los ficheros adjuntos a un correo deberían ser tratados con las mismas reglas y restricciones de seguridad aplicadas al contenido del correo y, por lo tanto, no deberían poder ser reenviados o manipulados sin autorización de su autor dentro del contexto del sistema.

Propuesta:

En esta propuesta, el fichero adjunto a un correo hace parte del mensaje mismo del correo y no son tratados de forma independiente. Está fuera del alcance de este trabajo la identificación y

manipulación de ficheros individuales y es por ello por lo que en esta propuesta el fichero adjunto se trata como parte del mensaje. Cuando un usuario deposita un correo, todo el contenido será cifrado y, en consecuencia, los ficheros adjuntos se cifrarán junto con el contenido, quedando al final integrados dentro del mensaje del correo. Esto significa que solo el usuario que pueda descifrar el mensaje sabrá si vienen o no ficheros adjuntos dentro del correo.

REQ026. Desde el sistema, un usuario debería poder enviar un correo electrónico SMTP sin perder los aspectos de seguridad del sistema.

Descripción:

Un usuario debería poder enviar un correo desde el sistema a otros usuarios que utilicen soluciones SMTP, pero sin perder la seguridad del sistema. Como un requisito de usabilidad e integración, es necesario proporcionar una solución para que el sistema pueda interactuar con correo convencional SMTP y progresivamente pueda adoptarse la nueva solución.

Propuesta:

Este requisito es uno de los más difíciles de conseguir debido a la poca seguridad de la que goza el correo electrónico, como se ha descrito en capítulos anteriores en este trabajo. Se propone la utilización de una puerta de enlace que sirve como traductor de los dos sistemas. Un usuario de un sistema convencional debería de enlazar el sistema que esté utilizando y realizar una traducción o trazado de las operaciones entre los dos sistemas.

La idea es utilizar el sistema tradicional como uno de sus buzones, pero toda la ejecución de las reglas de seguridad se debería de ejecutar dentro del sistema que se propone. Al enviar un correo a un destinatario que utiliza correo SMTP, la puerta de enlace resolvería la traducción. Aunque por temas de compatibilidad los usuarios exigen este requisito para poder continuar sus comunicaciones, realmente los servidores SMTP deberían de empezar a ser menos utilizados por la poca seguridad que ofrecen.

REQ027. Un usuario debería definir los aspectos de seguridad deseados cuando envía un correo convencional SMTP.

Descripción:

Siguiendo con el REQ026, este requisito exige la flexibilidad del sistema, proporcionando al usuario la opción de que él mismo considere el nivel de seguridad cuando se comunica con servicios convencionales de correo electrónico SMTP.

Propuesta:

El usuario podría definir un esquema despreocupado, donde no le interese conservar la seguridad con este tipo de sistemas, o un esquema paranoico que obligue a los destinatarios a integrarse al sistema utilizando las reglas de seguridad del sistema y las puertas de enlace si quiere conservar sus antiguas cuentas, o crearse una nueva cuenta y olvidarse del antiguo servicio de correo SMTP. En esta propuesta, se considera que recibir correo desde una cuenta SMTP solo es seguro si el usuario que envía el correo es también usuario dentro del contexto de esta solución.

REQ028. Es necesario que el sistema proporcione mecanismos de autenticación para que el usuario pueda operar en el sistema.

Descripción:

El sistema debería proporcionar mecanismos de autenticación que permitan confirmar que el usuario que interactúa con el sistema es quien dice ser. Durante el proceso de autenticación el usuario debería proporcionar pruebas para que el sistema pueda verificar que es el propietario de la cuenta.

Propuesta:

Se propone el uso de la firma digital para autenticar todas las operaciones. Durante el registro, al usuario se le proporcionará un par de claves, pública y privada, y un identificador asociado a la clave pública. Todas las operaciones que realice dentro del sistema necesitarán del uso de una firma realizada con la clave privada y de la asociación del identificador con la clave pública.

Al ejecutar cualquier proceso dentro del sistema, el sistema comprobará que la firma solicitada es válida y que se corresponde con la clave pública y el identificador registrados en el sistema.

REQ029. Los usuarios deberían poder realizar algunas operaciones seguras sin estar conectados a Internet.

Descripción:

La solución debería permitir realizar las operaciones del sistema de forma segura cuando no se esté en línea, aunque queden a la espera de poder ser verificadas o validadas.

Propuesta:

Todos los datos que utilizan los protocolos seguros en esta propuesta pasan por ser validados por un procedimiento de consenso en la red. Si un usuario ha revocado la autorización de otro, dicha actualización debería ser actualizada en todos los nodos participantes. Mientras un nodo está fuera de línea, el nodo podría no estar actualizado. Si el usuario envía un correo mientras está sin conexión, debería poder realizar la operación, pero quedará pendiente de ser confirmada por la red.

En el caso de una operación de envío de correo, el usuario podría crear el correo cumpliendo todas las reglas de seguridad si tiene acceso a un nodo de la red, pero no podría depositarlo hasta que tenga conexión. La presente especificación indica que, en caso de ser permitidas las operaciones sin conexión, es la plataforma distribuida al final quien determina si son válidas o no esas operaciones.

REQ030. Los usuarios deberían poder configurar la visibilidad del correo cuando acceden con sus aplicaciones.

Descripción:

Normalmente el correo que se descarga en los ordenadores y al que se accede con algún MUA, como Outlook o ThunderBird, no solicita ningún tipo de autenticación. Hay usuarios que protegen los correos mediante alguna contraseña para evitar que una persona con la que compartan el equipo lea las conversaciones. Muchos sistemas utilizan el cifrado para proteger

los correos, pero su protección finaliza cuando los correos llegan y son descargados por este tipo de aplicaciones. Es necesario que el sistema proporcione algún tipo de seguridad o control para minimizar los riesgos cuando se trabaja en local.

Propuesta:

Se propone que el usuario pueda determinar la visibilidad de los correos descargados, pudiendo decidir si mostrará la información cifrada o no en la aplicación MUA que utiliza y qué tipo de información. Un usuario podría tener todos los datos cifrados y solo mostrar en texto plano los asuntos cuando accede con un MUA, pero sin revelar los participantes. Podría dejar en texto plano todos los correos y navegar entre ellos por facilidad o, si necesita un alto grado de confidencialidad, podría decidir no revelar ninguna información y descifrar a demanda lo que él considere que necesita. Todo dependerá de los requisitos de seguridad que requiera el rol que desempeña.

REQ031. Un usuario debería poder analizar el contenido de un correo o fichero recibido con alguna herramienta externa para comprobar si no es malicioso.

Descripción:

Es un requisito adicional de integración que un usuario pueda pasar el contenido de un correo ya descifrado por alguna herramienta de verificación como un antivirus. Aunque muchos de los aspectos de seguridad quedarían anulados al permitir que una herramienta de un tercero tenga acceso al contenido o los ficheros, siguiendo unas normas sencillas, esto no debería ser un problema si el que controla las herramientas es el mismo propietario de los correos y no un tercero como ocurre con los servidores actualmente.

Propuesta:

Se propone como solución que la implementación pueda generar un fichero que pueda ser verificado por la herramienta que tenga a disposición el usuario. No se recomienda el acceso de una herramienta al contenido de los buzones por razones de seguridad, pero debido a la flexibilidad de esta propuesta, se deja en manos del implementador dicha utilidad y que el usuario pueda tener la opción de utilizarla de forma interna o externa como se ha indicado.

REQ032. Los usuarios deberían disponer de una solución que no les afecte el rendimiento de la gestión del correo electrónico debido a los procesos de seguridad involucrados.

Descripción:

Uno de los problemas con los que se enfrentan los usuarios a la hora de adoptar una solución de seguridad es la usabilidad de ésta. Puede ser porque no tengan los conocimientos técnicos que exige la herramienta o porque tengan que esperar más tiempo de lo habitual para poder realizar cualquier proceso y, al final, desactiven la herramienta y continúen sin ella.

Propuesta:

Hoy en día el proceso de cifrado y descifrado utilizando, por ejemplo, AES, es muy rápido. Se propone que, aunque un sistema de correo electrónico seguro tiene una alta exigencia de carga en procesos de firma, cifrado y descifrado, se tengan en cuenta el rendimiento de la aplicación si, por ejemplo, se van a realizar procesos intensivos de descifrado de cientos de correos, como es el caso de un cambio de la visibilidad de los mensajes dentro del MUA, y se deseen descifrar todos sus correos.

REQ033. El sistema debería de proporcionar a un usuario la opción de crear varias cuentas bajo su supervisión, pero manejadas por otros usuarios.

Descripción:

Los protocolos de correo tradicional no proporcionan soluciones empresariales a nivel de gestión y control. El control recae tradicionalmente sobre herramientas comerciales que han sido desarrolladas sobre los protocolos de correo. Lo que se suele hacer es que una empresa tenga un dominio, instale un servidor de correo bajo ese dominio y cree tantas cuentas como empleados necesite para su gestión. Hoy en día, es necesario tener control sobre las cuentas corporativas, no solo a través de auditorías de correo, sino también cuando alguna fuga de seguridad haya sido denunciada. Es necesario que la solución propuesta proporcione a los administradores soluciones de seguridad y control para las cuentas que están bajo su supervisión.

Propuesta:

En esta propuesta, el sistema permite a un usuario controlar varias cuentas gestionadas por otros usuarios. Este tipo de usuario podrá tener el papel de delegado y controlará todas las operaciones realizadas sobre las cuentas que supervisa. En este caso, un usuario como propietario legal de todas las cuentas delegará su gestión a otros usuarios, pero conservará el control sobre ellas. Se propone que un usuario pueda crear tantas cuentas como necesite y tenerlas asociadas en su configuración, almacenando las claves públicas y privadas de todas ellas. A este usuario dentro de esta solución se denomina usuario corporativo.

REQ034. Un usuario debería poder bloquear, eliminar, modificar o cambiar el acceso a las cuentas que tiene bajo su supervisión.

Descripción:

Este requisito está basado en el requisito REQ033. Lo que se especifica en este requisito es qué tipo de control se debería tener sobre las cuentas que un usuario tiene bajo su supervisión. Es necesario que el sistema permita a un usuario supervisar las opciones de bloqueo, eliminación y acceso de las cuentas supervisadas.

Propuesta:

En esta propuesta, aunque un usuario supervisado tengas las credenciales de acceso a una cuenta, esta pueda ser bloqueada, eliminada o modificada por el usuario gestor. Se plantea utilizar un tipo especial de credenciales denominado cuentas de gestión que permitirá el control de cuentas supervisadas bajo el esquema de claves privada y públicas.

REQ035. Un usuario corporativo debería poder determinar si es necesaria la autorización para cualquier acción sobre las cuentas que tiene bajo su supervisión.

Descripción:

Las operaciones que pueden realizar los usuarios supervisados deberían poder ser controladas por el usuario gestor. Un usuario supervisado podría ser bloqueado por el usuario gestor para realizar operaciones como la recepción, eliminación o envío de correo. Debería también ofrecer la posibilidad de que dichas operaciones se puedan realizar, pero solo bajo la supervisión del usuario gestor.

Propuesta:

Los protocolos que se proponen proporcionan las reglas que son necesarias para brindar estas características de control y seguridad. Para ejecutar cualquier operación, el protocolo debe determinar si la cuenta puede ejecutar libremente una operación o si es necesaria la autorización de un tercero, en cuyo caso se ejecuta un protocolo de autorización. La solución planteada en esta tesis es utilizar algunos campos adicionales en el registro de configuración de seguridad donde puedan activarse este tipo de controles. Con el registro de configuración, un usuario corporativo podría, por ejemplo, configurar una cuenta solo para recepción de correo, controlar el horario de gestión de los mensajes o el bloqueo total de la cuenta.

REQ036. Un usuario corporativo debería tener la opción de delegar la gestión de todas o algunas de sus cuentas a otro usuario bajo su supervisión.

Descripción:

Los usuarios corporativos, al poder gestionar varias cuentas, deberían también poder delegar dicha gestión. La delegación de control no significa que necesariamente se pierda permanentemente el control, sino que dicha gestión puede ser realizada por otro usuario, aunque puede ser cancelada en cualquier momento. Este usuario delegado debería seguir estando bajo la supervisión del usuario gestor, pero con potestad sobre las cuentas supervisadas que han sido asignadas a su cargo.

Propuesta:

Se propone que la delegación se realice por medio de registros de seguridad distribuidos. Cuando un usuario administrador cree un usuario corporativo, se le asignará el usuario gestor de la cuenta y el registro de seguridad asociado. Ese registro solo puede ser modificado por el gestor donde se indicará las opciones de control que el gestor considere oportunas. El gestor del usuario corporativo podrá ser modificado por el administrador de las cuentas y solamente el nuevo gestor podrá modificar el registro.

REQ037. El sistema debería de permitir el acceso al contenido de los buzones por parte de un tercero autorizado.

Descripción:

Un usuario debería poder dar acceso de lectura a otro usuario para ver el contenido de los correos almacenados en un buzón de su propiedad. El sistema debería proporcionar un mecanismo para que, de forma temporal, el contenido de un buzón fuera compartido por más de un usuario.

Propuesta:

Se propone que el sistema permita por medio de un protocolo de concesión de acceso dar a un tercero legalmente identificado acceso a un buzón determinado sin vulnerar la seguridad del sistema. Existen casuísticas corporativas donde este tipo de acceso puede ser utilizado, como el acceso concedido a un trabajador para que tenga acceso de lectura a un buzón para leer notificaciones o a un buzón de quejas al que tenga acceso un grupo de usuarios para su resolución.

Se proponen dos soluciones para cumplir a este requisito. El primero es crear un buzón de solo lectura donde las credenciales de acceso se le entreguen al tercero y todos los correos de interés vayan en copia a ese buzón. La otra alternativa es utilizar el registro de la cuenta y asociarle una cuenta de *backup* donde todos los correos entrantes y salientes sean direccionados a esa cuenta.

REQ038. El sistema debe proporcionar la opción de bloquear una cuenta y recuperar toda su información en caso de extravío de las credenciales de acceso.

Descripción:

Este requisito es muy común y de fácil resolución en los sistemas tradicionales de correo electrónico. La pérdida de las credenciales de acceso facilita a un atacante el acceso a todo el sistema. Puede darse el caso de la pérdida u olvido de las credenciales de acceso y el sistema debería garantizar una solución a esta problemática. Dado que el control debería de tenerlo el usuario, este debería tener las garantías de que el sistema es capaz de recuperarse en este aspecto.

Propuesta:

Esta propuesta está basada en la utilización de claves privadas y públicas como la piedra angular en la que se basa toda la seguridad del sistema. Si la seguridad del buzón está relacionada directamente con la posesión de la clave privada y por alguna razón se pierde, el sistema debe proporcionar un mecanismo para recuperar la información. Partiendo de que la clave privada no se puede cambiar, ya que significaría la creación de un nuevo usuario, se puede utilizar una frase semilla (*seed phrase*) para la generación de la clave y su recuperación. Esa frase semilla debe guardarse en un lugar seguro para evitar su pérdida por una avería del ordenador o por cualquier circunstancia no deseada.

Se propone un proceso de bloqueo de la cuenta y recuperación en caso de perder la clave privada. En cualquier momento un usuario puede crear una clave de recuperación, esto es, una cuenta de bloqueo paralela que tendrá la copia total de la cuenta original. La cuenta con la copia nunca accederá a la clave privada que se almacenará utilizando una frase semilla. La clave privada (si lo desea) y la frase semilla serán almacenadas fuera del sistema y solo tendrá sentido su uso en caso de pérdida. Si por alguna razón el usuario pierde las credenciales, deberá arrancar el sistema con la cuenta de bloqueo y podrá acceder a toda la información. Debido a la utilización de las cuentas de gestión, las credenciales de una cuenta siempre se pueden generar y, en caso de robo, desde la cuenta de gestión se puede bloquear la cuenta para que no sea utilizada.

REQ039. El sistema debería permitir al usuario determinar el grado de visibilidad entre los destinatarios cuando se envíe un correo a varios usuarios diferentes.

Descripción:

Cuando se envía un correo electrónico a varios destinatarios, se debe considerar si se debe revelar la RC remitente-destinatario a los demás participantes de la comunicación. El sistema debería ofrecer garantías para que esa visibilidad fuera controlada. En los sistemas tradicionales de correo electrónico se suele utilizar la copia oculta, que es una solución a este requisito y, por consiguiente, debe ser incorporado en el nuevo sistema.

Propuesta:

Se propone que una comunicación de correo puede tener varias RC y el usuario debería tener la opción de modificar su visibilidad entre los diferentes RC para conservar la confidencialidad entre los diferentes destinatarios.

REQ040. Es necesario que el sistema notifique e invalide las operaciones relacionadas con modificaciones no autorizadas de los datos involucrados en la comunicación de correo electrónico.

Descripción:

El mensaje, los participantes y los adjuntos podrían ser intencionadamente modificados. Por lo tanto, el sistema debería invalidar cualquier acción relacionada con una alteración de datos no autorizada.

Propuesta:

En esta propuesta, todas las operaciones que los usuarios ejecutan requieren de la validación de un protocolo descentralizado. En el caso del envío de correo, los protocolos utilizados en este trabajo no permiten la modificación de los datos y, si los datos son modificados, dichas modificaciones pueden ser fácilmente identificadas e invalidadas. Los protocolos pueden notificar a los interesados si se produce cualquier violación de seguridad.

REQ041. Un usuario del sistema debería tener la opción de activar o no la autenticación en cada operación o una autenticación global.

Descripción:

Un usuario podría requerir extremar la seguridad de las acciones que realiza y requerir que el sistema le solicite credenciales para autenticar cada operación que realice. Otro usuario podría determinar que no es necesario y el sistema no le debería solicitar credenciales. El sistema debería permitir los dos casos.

Propuesta:

Esta propuesta no permite que el usuario realice una operación sin demostrar que es realmente el propietario de la cuenta, ya que la mayoría de las operaciones requieren de la firma y del acceso a sus credenciales. Una implementación de esta propuesta podría tener un acceso con contraseña a las claves privadas y dejar al usuario la opción de utilizar las claves sin su intervención. Pero por razones de seguridad, no se aconseja dicha solución, debido a que las claves privadas deberían ser manipuladas fuera de línea. Este requisito se desea sobre todo por razones de usabilidad, ya que el uso continuado de una solución de alta seguridad requiere de la solicitud continuada de la clave privada y, por ende, la participación del usuario.

REQ042. El sistema no debería de proporcionar ninguna información personal de un usuario a un tercero sin autorización y no debería ser requerida para realizar ninguna operación.

Descripción:

En los sistemas tradicionales de correo electrónico, donde la información personal es requerida para realizar un registro u otra operación en la plataforma que proporciona el servicio, esto genera muchos problemas de seguridad, además de violaciones de confidencialidad y robos de datos personales. La información del usuario no debería estar expuesta a un tercero y, en caso de cualquier modificación de los registros de información y configuración del usuario, debe ser factible identificar dicha alteración. Es necesario que el

sistema proporcione las notificaciones pertinentes en caso de cualquier intento de alteración de la información del usuario.

Propuesta:

Se propone que la información personal o de identificación esté almacenada y gestionada por el usuario fuera del sistema y no deba ser requerida por un tercero nunca, a excepción de una solicitud de demostración de identidad que, en tal caso, debería ser gestionada por un tercero de confianza distribuido, como una CA o por cualquier sistema de identificación externo.

REQ043. El sistema debería garantizar el cumplimiento de todos los requisitos de seguridad mencionados anteriormente durante el envío de correo.

Descripción:

Durante el envío del correo, todos los participantes involucrados en su transporte deberían garantizar que los requisitos de seguridad se cumplan. El envío de correo debería poder satisfacer las necesidades tanto de un usuario que extreme la seguridad como de otro que no le dé importancia a la misma.

Propuesta:

Se propone que el protocolo de envío de correo sea flexible para que los diferentes tipos de usuarios puedan hacer uso de él sin afectarse entre ellos. Todos los integrantes responsables del transporte del correo deben tener acceso a los registros de seguridad gestionados por los protocolos y, por lo tanto, pueden velar por su cumplimiento. Si uno de los responsables viola alguna de las reglas de seguridad, puede ser expulsado del sistema, ya que los usuarios serían notificados de dicho incumplimiento y la confianza en dicho responsable decaería.

REQ044. El sistema debería garantizar el cumplimiento de todos los requisitos de seguridad mencionados anteriormente durante la recepción de correo.

Descripción:

De igual manera que el requisito REQ043, todos los requisitos y aspectos de seguridad discutidos en esta sección deberían de ser respetados cuando se recibe correo electrónico. La recepción de correo debería poder satisfacer tanto las necesidades de un usuario que extreme la seguridad como de otro que no la estime importante.

Propuesta:

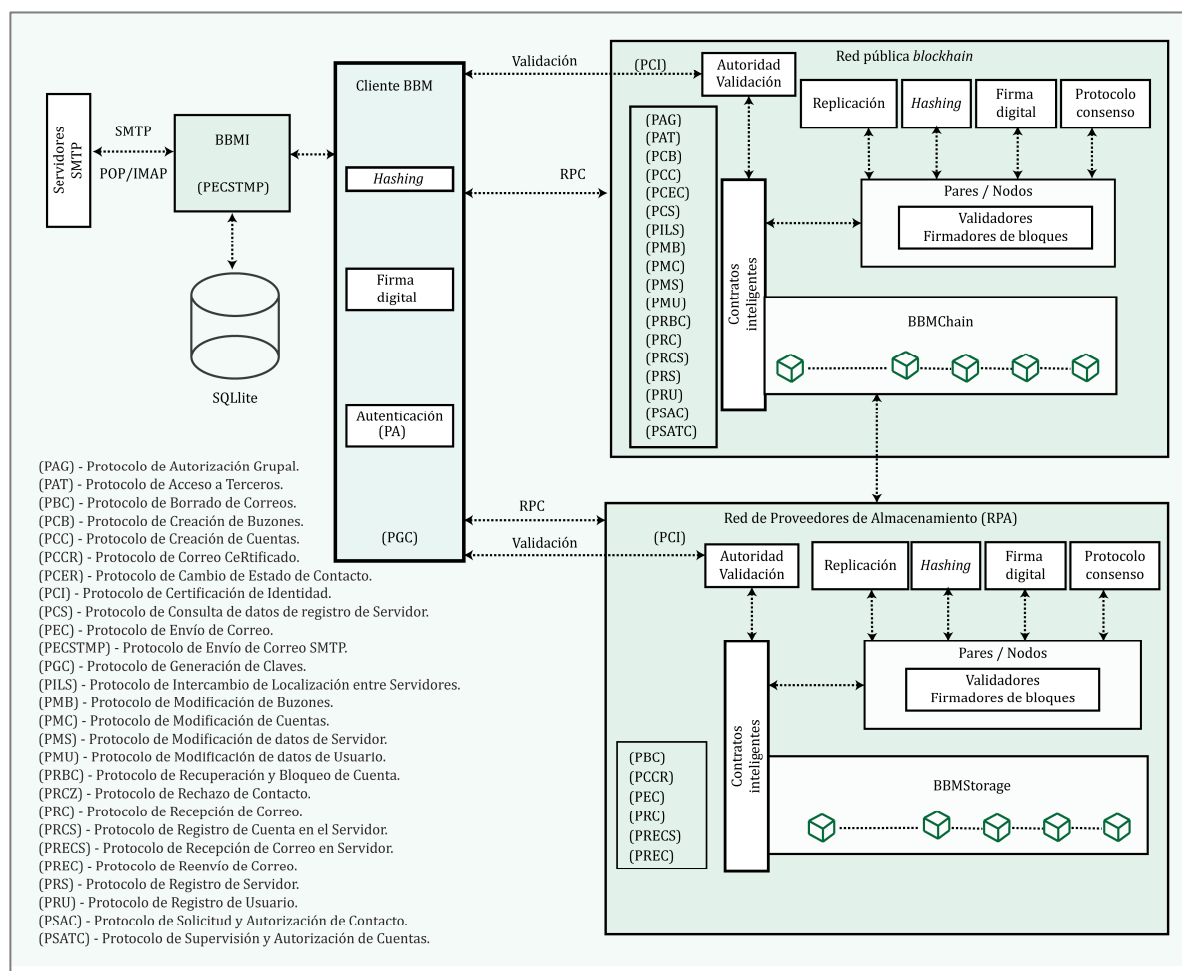
Se propone que el protocolo de recepción de correo sea flexible para que los diferentes tipos de usuarios puedan hacer uso de él sin afectarse entre ellos. Como se explicó en el REQ043, los protocolos de seguridad que se proponen en este trabajo garantizan el cumplimiento de los requisitos de seguridad de todos los participantes.

IV. ARQUITECTURA DE BBM (BLOCKCHAIN BASED MAIL)

1. Introducción

La arquitectura presentada en este trabajo responde como solución a los requisitos presentados en el apartado anterior. Está compuesta por una serie de módulos que se comunican utilizando una serie de protocolos distribuidos. En esta sección se describirán cada uno de esos componentes, su funcionalidad y el modelo de datos subyacente utilizado para posteriormente en la próxima sección detallar cada uno de los protocolos. La arquitectura propuesta (Figura 43) hace uso de una tecnología híbrida de *blockchain* privada y pública con un componente adicional para la interoperabilidad entre ambas. Se ha denominado a la solución BBM (Blockchain Based Mail) y, a partir de ahora, se utilizará ese nombre para hacer referencia a la solución.

Figura 43. Arquitectura de BBM.



Fuente: Elaboración propia.

2. Componentes de la arquitectura BBM

2.1. Red de Proveedores de Almacenamiento (RPA)

La Red de Proveedores de Almacenamiento (RPA) es la red de almacenamiento de correo electrónico y tiene como objetivo asegurar la disponibilidad del correo y su gestión. El sistema de correo electrónico es un sistema de uso intensivo por los usuarios y es por esa razón que en la presente propuesta el almacenamiento del correo se realiza fuera de la red de nodos de BBM.

La red la componen una serie de servidores privados que prestan sus servicios de forma independiente, registrándose en la red BBM y siguiendo los protocolos de seguridad proporcionados por la red. Esta independencia permite diferentes implementaciones de cada servicio de almacenamiento, pudiendo ser una red privada de nodos basada en alguna tecnología que proporcione disponibilidad.

Hoy en día muchas soluciones proporcionan alta disponibilidad y están preparadas para resistir ataques DOS, por lo que pueden ser utilizadas para el almacenamiento dentro de BBM, pero siguiendo las reglas propuestas en esta especificación. En la prueba de concepto se utilizó MongoDB con las opciones de fragmentación (*sharding*) y replicación de nodos para proporcionar un servicio de almacenamiento básico que cumpliera con la especificación de los protocolos. En una implementación de BBM se puede utilizar un servicio de replicación para cada cuenta, que consiste en un sistema de copia de seguridad de los correos basado en la tecnología IPFS o cualquier otra tecnología de almacenamiento de ficheros distribuida, usado en caso de bloqueo, borrado o inactividad de un servidor.

2.2. Servidor

Dentro del RPA, el servidor es un actor del sistema BBM. Su objetivo principal es proporcionar acceso al contenido de los correos electrónicos de los usuarios y proporcionar el servicio de alta disponibilidad para los mismos. Un servidor debe registrarse en BBM como proveedor de almacenamiento. La actividad de los servidores está regulada por los protocolos distribuidos y hacen parte del estado del sistema BBM. Los servidores participan en las transiciones de la máquina de estados BBM, que se relaciona con el estado de todo el sistema.

Los servidores almacenan correos electrónicos depositados por los usuarios, pero nunca envían correo a otros servidores. Un servidor debe reclamar el correo que los usuarios han depositado en otros servidores dirigido a él mismo. El sistema BBM regula la actividad de los servidores. Los servidores son los encargados de proporcionar confianza al usuario y a los demás servidores con los que interactúan. El usuario deposita la confianza en el servidor y el

estado del servidor va a depender de esa confianza. El servidor puede ser desconectado del sistema si la seguridad que suministra se ve comprometida y la confianza que depositan los usuarios en dicho servidor decae.

Un servidor acepta peticiones de otros servidores registrados en BBM. Ante cualquier petición, el servidor que la recibe verifica en BBM si lo que le están solicitando cumple con las reglas de seguridad del sistema. Los servidores generan advertencias sobre el comportamiento de un servidor en forma de notificaciones si son violadas las reglas dentro de BBM, lo que puede afectar la clasificación de servidor dentro del sistema. Si un servidor realiza una petición que no cumple con un protocolo y persiste en su actividad, el sistema BBM podrá banearlo, lo que llevará a que los usuarios no se registren y los que están registrados decidan abandonarlo. Todos los servidores conforman la RPA, que puede ser implementada ya sea utilizando una red de almacenamiento *blockchain* privada, un servicio basado en DHT, servidores con replicas o servicios de fragmentación como, por ejemplo, MongoDB.

2.3. Blockchain de BBM (BBMChain)

Es uno de los componentes más críticos de BBM. Es la *blockchain* donde se almacena toda la información del usuario, de los protocolos de seguridad y la información sobre la RPA de BBM. Tanto los usuarios como los servidores se registran en BBM a través de protocolos de registro y es aquí donde se almacena el identificador único para cada uno de ellos.

Su principal función es gestionar y almacenar de forma segura la información que proporciona el usuario y los servidores en el registro. Participa en la ejecución de los protocolos de seguridad proporcionando toda la información de seguridad que necesitan para su correcta ejecución. No permite que un tercero pueda manipular la información que tienen almacenada sin su autorización.

Debido a que cliente BBM puede ser implementado sobre una *blockchain* pública, son necesarios mecanismos adicionales para salvaguardar la información y dar cumplimiento a los requisitos, aspectos de seguridad y objetivos de este trabajo. En la descripción de los protocolos se aclarará cómo se logran esos objetivos y en la descripción de la prueba de concepto, cómo pueden ser implementados.

2.4. Generador de claves (KG)

Es un proceso que puede ser ejecutado en línea o fuera de línea, dependiendo de la seguridad deseada por el usuario, y tiene como objetivo la generación de las claves privada y pública. El generador de claves (KG) puede estar integrado dentro del cliente BBM o puede ser un proceso independiente que siga la especificación de la arquitectura.

2.5. Cliente BBM

Es el MUA del sistema que permite al usuario interactuar y realizar todas las tareas dentro de BBM. Desde el cliente BBM, el usuario puede ejecutar por ejemplo el proceso KG dentro del protocolo de registro, registrar la clave pública generada en KG, gestionar el registro para ser almacenado en BBM, almacenar de forma segura la clave privada y procesar los mensajes de error en caso de cualquier fallo. El cliente BBM es la interfaz del usuario, y es la herramienta utilizada para interactuar con todos los protocolos proporcionados por BBM.

Son cinco las funciones principales que deben ser implementadas en este cliente:

-  Gestión de usuarios.
-  Gestión de cuentas
-  Gestión de buzones.
-  Gestión de correo.
-  Controles de seguridad.

El cliente BBM es un nodo que pertenece a una red distribuida. Dado que BBM posee una arquitectura híbrida, el cliente BBM se comunica con una o varias *blockchain* con funcionalidades y objetivos distintos que se describirán en este capítulo. Es por ello por lo que se considera que el cliente BBM es un nodo multi cadena o “*multi-chain node*”.

2.6. Almacenamiento BBM (BBMStorage)

Es el sistema donde se almacena el contenido del correo electrónico. Este sistema puede ser implementado como una *blockchain* privada o como un servicio de almacenamiento distribuido (como sistema externo a BBM). El componente del sistema BBM que realiza y desempeña el rol de servidor es el responsable y encargado de proporcionar BBMStorage a todos los usuarios de BBM.

Todos los correos son almacenados de forma cifrada, pero hay información que solo pueden interpretar los servidores depositarios. BBMStorage hace parte de RPA y son los servidores de

RPA los encargados de interactuar con este componente, que puede ser implementado utilizando diferentes tecnologías de alta disponibilidad.

2.7. Integrador BBM (BBMI)

Hace referencia al sistema que tiene como objetivo realizar las actividades de puente para las tareas de integración con otros protocolos de correo electrónico. Es un componente que debe ser implementado con el único objetivo de ofrecer comunicación bidireccional a otros servidores, como, por ejemplo, los servidores SMTP.

Todos los mensajes que pasan por BBMI deben ser regulados por BBM, lo que significa que la entrada y salida de correo electrónico debe cumplir con los mismos requisitos de seguridad que ofrece BBM. Protocolos como SMTP deben someterse a ciertas restricciones que serán explicadas cuando se describa el protocolo utilizado para interactuar con BBMI.

3. Especificación y descripción de la arquitectura BBM

3.1. Diagrama de contexto del sistema

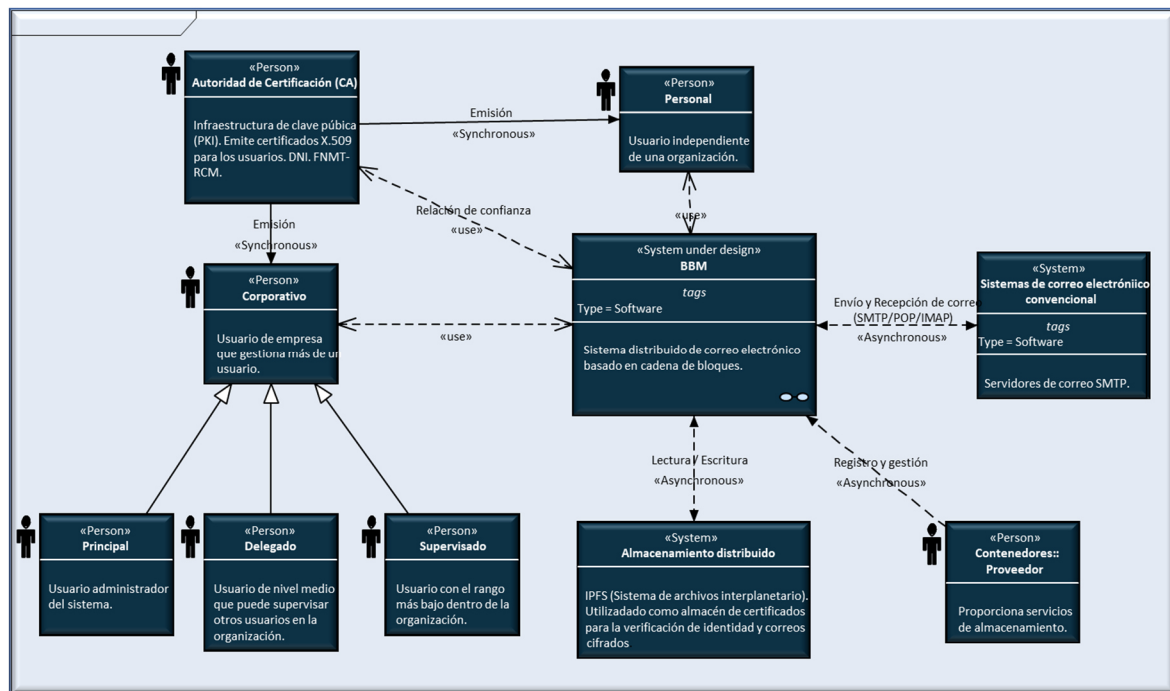
El objetivo de este diagrama es mostrar de forma muy simplificada la arquitectura propuesta desde el punto de vista de los usuarios y los otros sistemas con los que interactúa. En la Figura 44 se puede observar los usuarios y tipos de usuarios que hacen uso del sistema y los otros sistemas con los que opcionalmente podría comunicarse, tales como los sistemas de correo convencionales. Un detalle importante es que no hay una comunicación directa entre los usuarios y los sistemas de correo SMTP, lo que indica que para que un correo de esas características llegue al usuario debe pasar por BBM y, por lo tanto, debe cumplir las reglas establecidas por la arquitectura.

Se distinguen dos tipos de usuarios muy diferenciados, el usuario personal y el usuario corporativo que fueron ya definidos. La implementación mínima de la arquitectura incluiría usuarios de tipo personal, ya que el usuario corporativo es el que requiere el uso de más controles y protocolos debido a las jerarquías definidas por defecto en una organización o empresa.

En el diagrama solamente se ha definido la relación de uso entre los usuarios y BBM para simplificar el sistema. Otro usuario externo es la entidad certificadora, que juega el papel de verificador de identidad cuando es requerido por un usuario, como se explicará dentro de los protocolos de seguridad sobre las relaciones de comunicación o contactos. El objetivo de este

actor es emitir un certificado X.509 para que un usuario pueda utilizarlo en caso de que otro usuario requiera la verificación de su identidad para aceptar la solicitud de contacto.

Figura 44. Diagrama de contexto BBM.



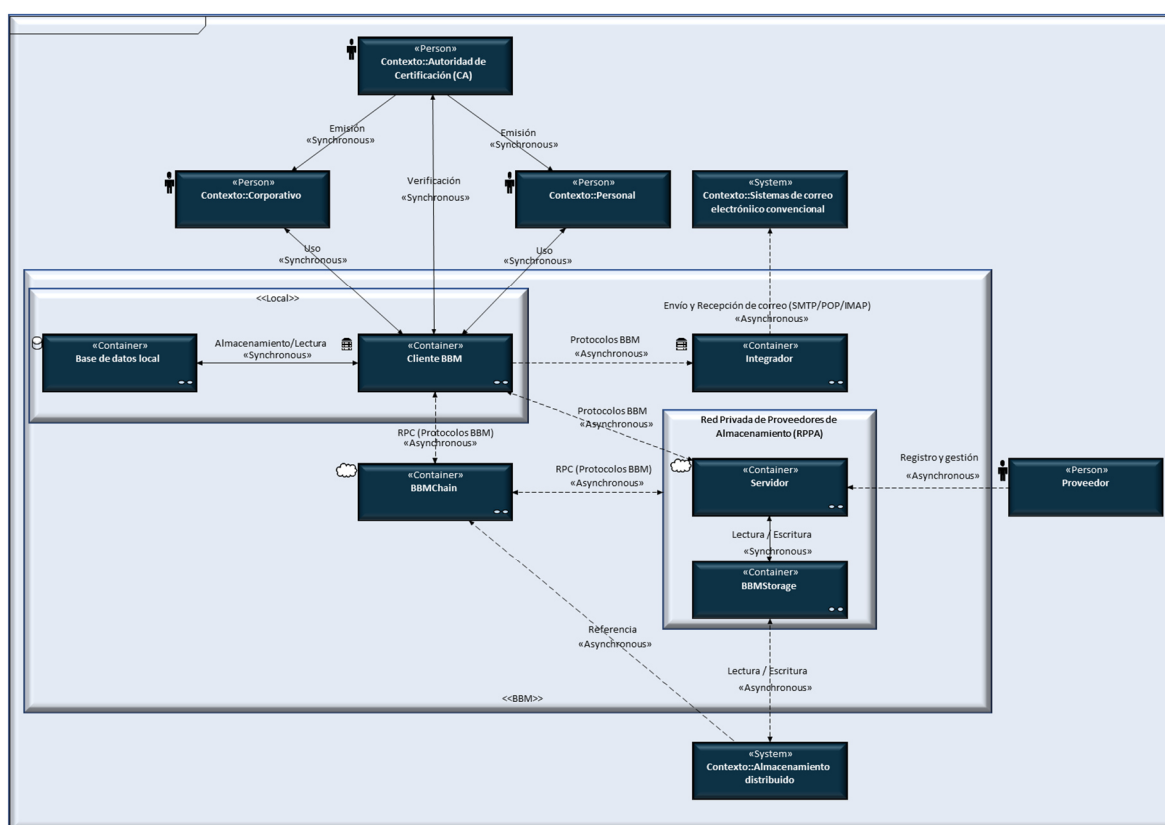
Fuente: Elaboración propia.

En el gráfico también se puede observar un sistema externo para almacenamiento distribuido de ficheros, por ejemplo, podría utilizarse IPFS (Baumgart & Mies, 2007), Storj (Wilkinson et al., 2016) o FileCoin (Benet & Greco, 2017). Dicho sistema es utilizado como repositorio distribuido para almacenar tanto adjuntos de ficheros y correos cifrados, como también certificados públicos (X.509). El uso de estos sistemas junto con *blockchain* ha sido ampliamente estudiado (H. Huang, Lin, Zheng, Zheng, & Bian, 2020) y son muchas las aplicaciones que pueden darse a este tipo de soluciones.

3.2. Diagrama de contenedores

Con el objetivo de facilitar el entendimiento de la arquitectura y proporcionar más detalle, se ha diseñado el diagrama de la Figura 45, donde se hace un *zoom* del diagrama de contexto para comprender los componentes y sus relaciones dentro de BBM. En la parte superior del diagrama se puede ver los actores externos al sistema. Mientras que en el diagrama de contexto (Figura 44) se podía ver la emisión de los certificados por parte de la autoridad certificadora (a muy alto nivel), en el diagrama de contenedores (Figura 45) se observa como la validación se realiza desde un componente interno de BBM (el cliente BBM).

Figura 45. Diagrama de contenedores de BBM.



Fuente: Elaboración propia.

Los usuarios interactúan con BBM haciendo uso del cliente BBM, que hace las veces de agente de usuario como se explicó en apartados anteriores. Este cliente se ha dibujado como un contenedor ya que puede ser un cliente móvil, de escritorio o *web*. Esto va a depender de la implementación que se realice de la arquitectura. Aquí se puede apreciar como ese cliente se encuentra en un contexto local para el usuario y, por ello, hace uso de una base de datos local donde se almacena el correo al que el usuario tiene acceso cuando no está conectado al sistema o donde descarga los correos recibidos.

En el diagrama se puede ver como los sistemas de correo electrónico convencionales (servidores SMTP, por ejemplo) pueden interactuar con esta arquitectura y es donde el integrador o BBMI ejecuta su trabajo de integración. BBMI podría ser una aplicación local o externa al cliente BBM y podría estar alojada como un servicio en la nube o un servicio local que hace de puente recibiendo el correo y ejecutando los protocolos BBM directamente con el cliente BBM. Independiente de su implementación, se propone que todo correo que pase por BBMI debe cumplir las reglas de seguridad de BBM como cualquier otro correo. En el apartado

donde se describen los protocolos relacionados con esta comunicación se aclarará cómo se deben realizar esas tareas y las posibles implementaciones.

Otro contenedor mostrado en el diagrama es BBMChain que, como se explicó anteriormente, puede ser implementado como una *blockchain* pública o privada, pudiendo ser utilizada para su implementación Ethereum, Hyperledger, entre otras. Lo importante que se quiere resaltar en este diagrama es cómo este contenedor interactúa con otros componentes de la arquitectura. Directamente el cliente BBM hace uso de los protocolos de BBM para realizar actividades como el registro, la gestión de contactos, registro del servidor, entre otras actividades administrativas. BBMChain actúa como un repositorio administrativo y no como un contenedor de correos.

Se ha explicado que la RPA es la red que se encarga de las labores de almacenamiento y es por eso por lo que se traza esa línea de comunicación entre RPA y BBMChain, para dar a entender que el proceso de almacenamiento y lectura sigue las reglas administrativas almacenadas en BBMChain. En el diagrama de contenedores, la línea que discurre desde el sistema de almacenamiento distribuido hasta BBMChain indica que se hace uso del almacenamiento distribuido en algunos protocolos, más precisamente los relacionados con la gestión de los contactos.

Por último, se puede observar en el diagrama como RPA aparece como un subsistema que interactúa con el cliente BBM y BBMChain. Esta representación indica la relación con los servidores de RPA, que son los que gestionan las peticiones de almacenamiento y obtención del correo. Estos servidores gestionan el almacenamiento del correo interactuando con BBMStorage, pudiendo en algunos casos interactuar con un sistema de almacenamiento distribuido.

3.3. Diagramas de componentes

A continuación, se detallarán cada uno de los contenedores y sus componentes para visualizar como interactúan entre ellos antes de especificar como deberían de realizarse esas comunicaciones y los protocolos utilizados en cada una de ellas. Se describirán los contenedores más importantes en los que se necesita detallar algunos aspectos. El contenedor de base de datos local y BBMStorage son sistemas de gestión de bases de datos y no necesitan una explicación adicional.

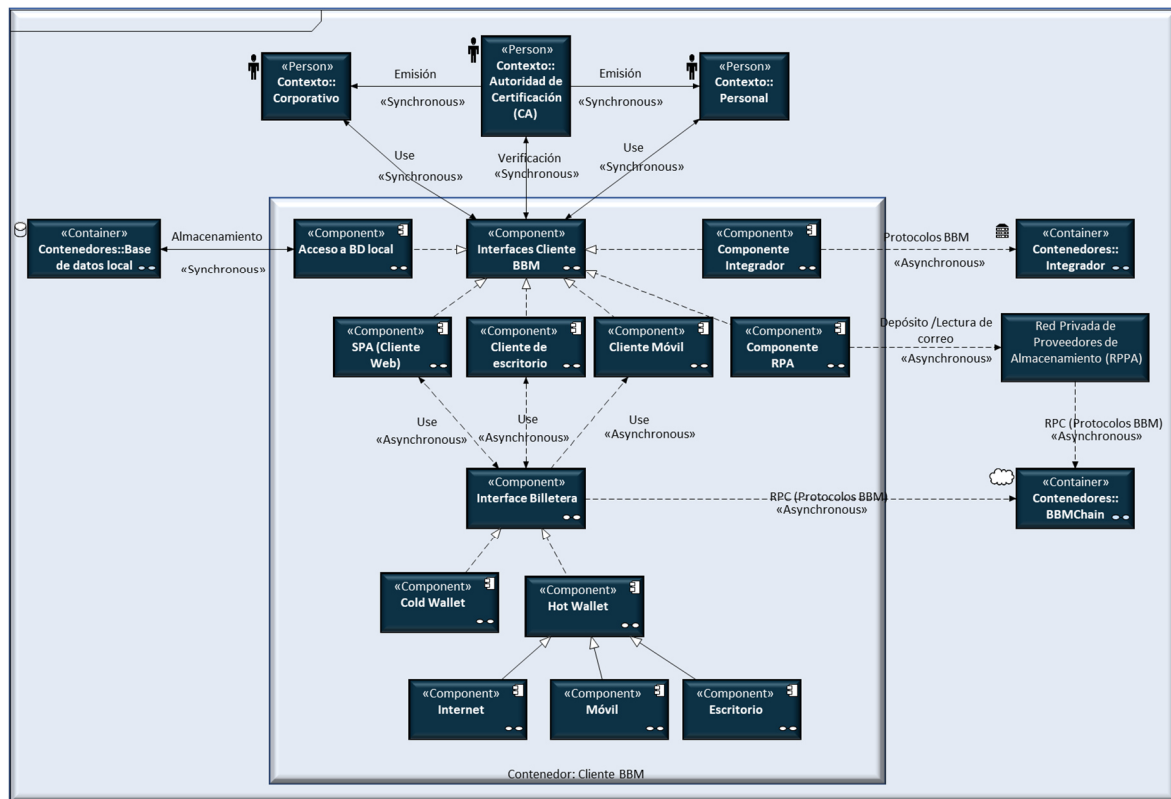
a. Diagrama de componentes para el contenedor Cliente BBM

Este contenedor es la herramienta que sirve de interfaz entre los usuarios y lo demás contenedores. En este apartado se describirán sus componentes y los diferentes tipos de relaciones entre ellos. Es un diagrama de más bajo nivel que los dos anteriores (el de contexto y de contenedores), necesario para explicar más detalladamente la arquitectura y, de esta forma, facilitar su implementación. La Figura 46 muestra los principales componentes que contiene el cliente BBM.

El primer componente que se describirá es “Interfaces Cliente BBM”. Estas interfaces definen todos los métodos que deben ser implementados por los diferentes tipos de clientes que pueden ser utilizados por los usuarios, así como por otros componentes presentes en el contenedor. Cada uno de estos diferentes tipos de clientes son representados en el diagrama como componentes de alto nivel. El “componente integrador” es el encargado de interactuar con BBMI para velar por el cumplimiento de los protocolos de seguridad en BBM y hace uso de todos los protocolos BBM relacionados con la recepción y envío de correo a servidores convencionales.

El componente “Acceso a BD local” permite el almacenamiento seguro de los correos y de la información administrativa necesaria para poder trabajar fuera de línea. Este componente puede ser implementado como un acceso a una base de datos local como SQLite (Bhosale, Patil, & Patil, 2015), como un acceso a un servidor de ficheros donde cada correo es un fichero independiente o como una base de datos JSON utilizando MongoDB (Chauhan, 2019) de forma local. En esta tesis, se propone utilizar JSON (Bray, 2014) como formato para el almacenamiento ya que dentro de la propuesta en la arquitectura es el formato utilizado mayoritariamente por todos los protocolos. Este componente es el encargado de proporcionar todos los métodos para el almacenamiento y lectura de los objetos (correos, información de gestión, etc.) que van a ser almacenados en la base de datos local.

Figura 46. Diagrama de componentes de cliente BBM.



Fuente: Elaboración propia.

El diagrama muestra tres tipos de clientes (*web*, escritorio y móvil). Estos componentes implementan las “Interfaces Cliente BBM” y son los encargados de proporcionar toda la funcionalidad de cliente de correo. Estos clientes deben implementar y hacer uso de los métodos de gestión de correo y administración del usuario, las cuentas y los buzones.

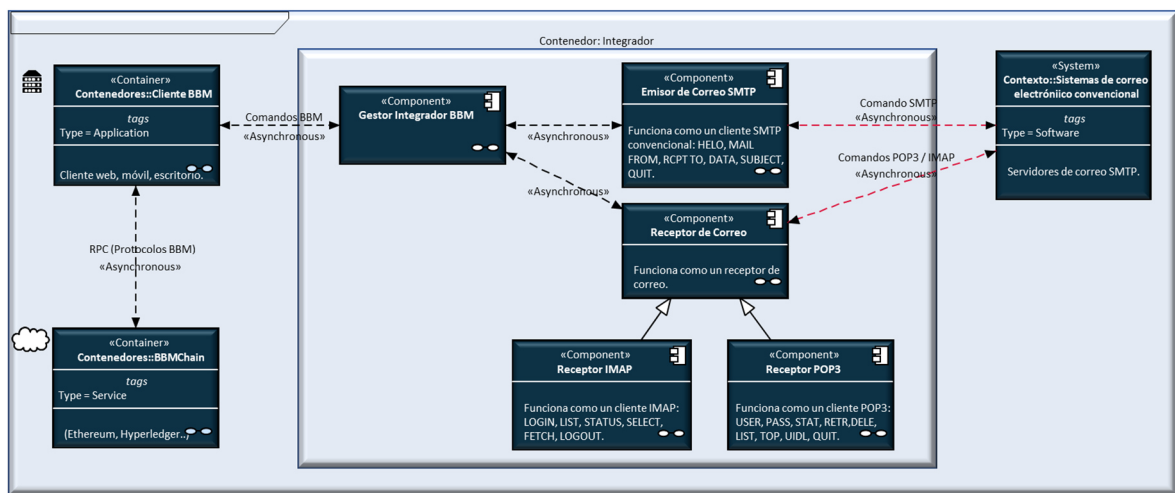
La comunicación con BBMChain se realiza utilizando el componente “Interfaces de Billetera”. Este componente proporciona la seguridad para las claves privadas, pudiendo utilizar billeteras físicas o virtuales dependiendo del cliente que sea implementado. Como se ha explicado anteriormente, las claves privadas deben ser almacenadas de una forma segura en la Blockchain y son este tipo de billeteras las que proporcionan dicha seguridad, siendo las billeteras *offline* (*Cold Wallets*) las más seguras debido a que no necesitan estar en línea para realizar las operaciones de confirmación de transacciones (Jokić, 2019).

La implementación del componente de “Acceso a BD local” se realiza solo cuando se implementan clientes de escritorio o móviles. Con los clientes *web*, el acceso a los correos debe realizarse directamente haciendo uso del componente RPA, que es el encargado de realizar las peticiones utilizando los protocolos de BBM relacionados con la comunicación a la RPA.

b. Diagrama de componentes del contenedor Integrador

El objetivo de este contenedor es la gestión del correo SMTP entrante y saliente. Es el puente de BBM para aplicar las reglas de seguridad a través de protocolos SMTP y BBM adaptados para dicha tarea. Este contenedor puede ser implementado como un servidor SMTP que siga las reglas de BBM, pero también puede ser implementado como un servicio que proporcione una capa adicional de seguridad bajo las reglas de la arquitectura BBM a otros servidores SMTP que ya estén en servicio. La Figura 47 muestra los principales componentes que debería utilizar este contenedor.

Figura 47. Diagrama de componentes del integrador.



Fuente: Elaboración propia.

La implementación de este componente debe seguir la arquitectura diseñada en este apartado. Los componentes “emisores y receptores de correo” son los encargados de realizar la gestión del correo entre los sistemas de correo electrónico convencional, utilizando para ellos los protocolos SMTP, POP e IMAP, respectivamente. Para implementar esta funcionalidad como un servidor SMTP/POP/IMAP, sería necesario reemplazar por completo la operativa interna del correo convencional añadiéndole el componente “Gestor Integrador de BBM”, que es quien se entiende a través de protocolos de integración con los sistemas SMTP.

Dada las características de esta propuesta, se espera que, con la adopción de este tipo de soluciones, no sea necesario utilizar los protocolos SMTP, POP e IMAP. No es recomendable entonces desde la perspectiva adoptada en este trabajo realizar una implementación completa de estos protocolos añadiéndole el componente gestor a un servidor SMTP.

A nivel de esta arquitectura, el componente más importante de este contenedor es el gestor integrador BBM, ya que es el que trabaja junto con el cliente BBM para la ejecución de los

protocolos de integración que validan en BBMChain todas las reglas de seguridad para la recepción y envío de correo.

El uso de un Integrador elimina dentro del contexto BBM las vulnerabilidades encontradas en los protocolos de correo, ya descritas en apartados anteriores. Pero dentro del contexto del correo convencional, esas vulnerabilidades siguen existiendo, con la excepción del correo saliente y entrante SMTP que siga las reglas de BBM. Para aclarar mejor este aspecto, se puede observar en la Figura 48 los dos contextos a los que nos referimos.

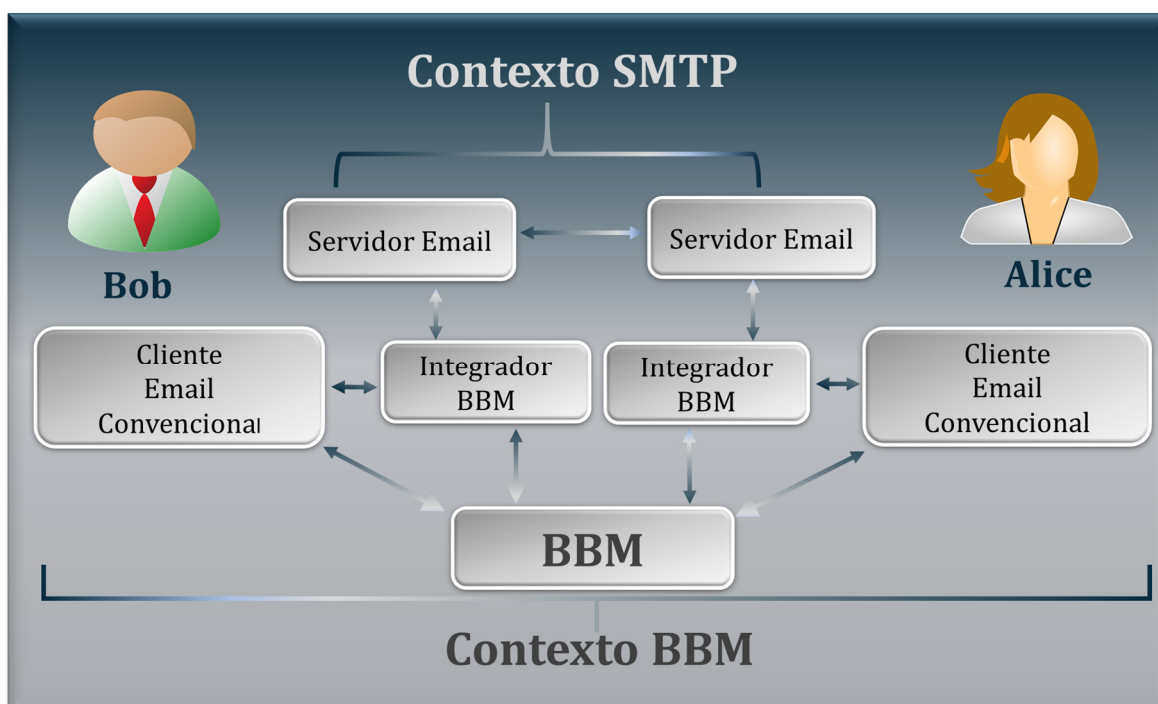
Figura 48. Flujo de correo desde SMTP y BBM.



Fuente: Elaboración propia.

Dado que dentro del contexto SMTP las vulnerabilidades siguen existiendo (no es el objetivo de este trabajo proponer una solución de seguridad sobre SMTP), para lograr el objetivo de la integración es necesario que solo un subconjunto de correos (los aceptados por las reglas de BBM) pueda llegar desde SMTP a BBM. El usuario que utiliza clientes de correo electrónico convencionales puede encontrarse, si sigue las reglas de BBM, que muchos de sus correos no llegan a su destino y son rechazados.

La arquitectura propuesta no descarta la posibilidad del uso de envíos entre servidores SMTP por BBM, pero debería darse solamente dentro del periodo de adopción de la solución. El esquema de dicha propuesta de integración puede verse en la Figura 49. Se ha considerado esa propuesta para futuros trabajos, ya que el objetivo de este trabajo es sentar las bases de la arquitectura y no proponer BBM como una solución a los protocolos de correo convencionales.

Figura 49. Utilización de BBM como un servicio adicional al correo electrónico SMTP.

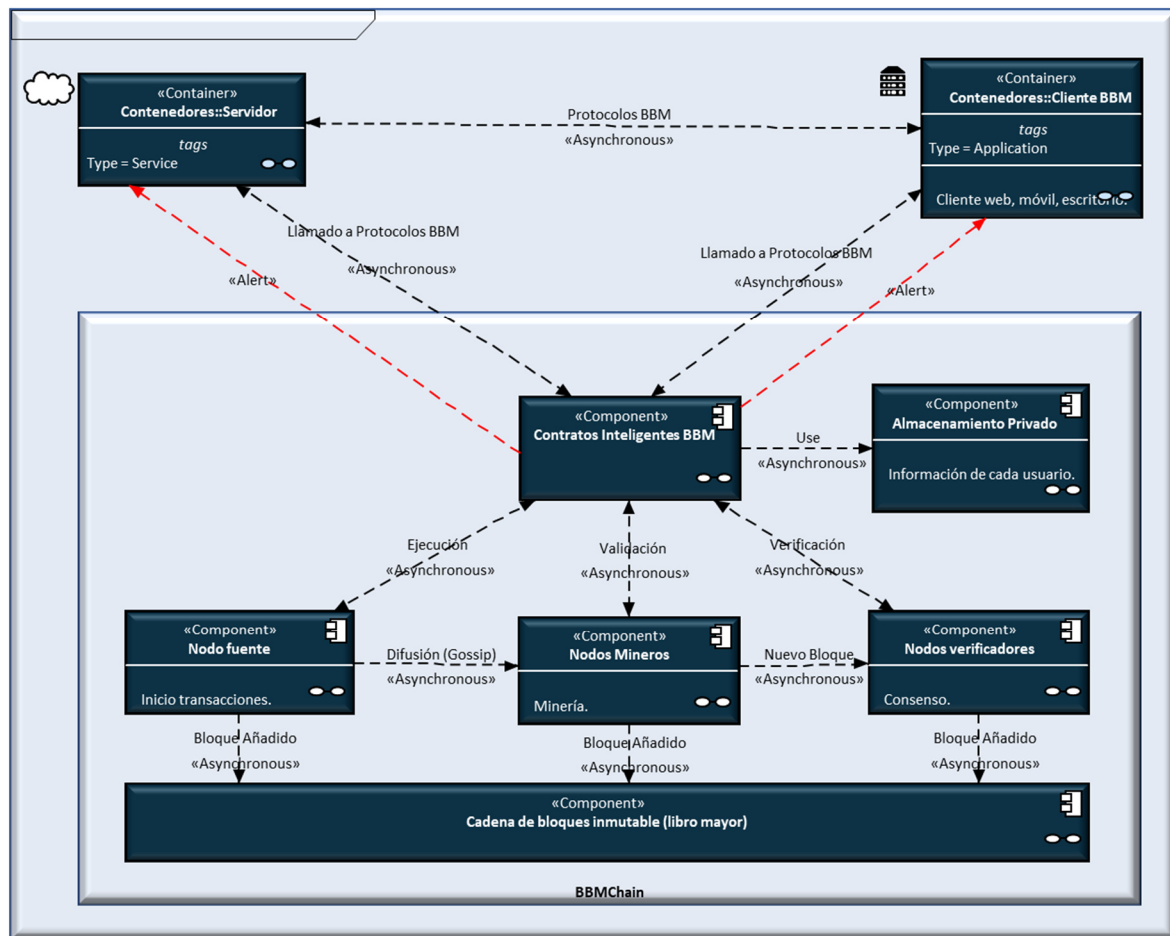
Fuente: Elaboración propia.

Se puede observar en la figura que la comunicación se inicia desde un cliente de email convencional, el cual se comunica con el BBMI, que es el encargado de filtrar todo el correo aplicando las reglas BBM. Esto significa que, en este caso Bob, configurará las reglas en BBM y el integrador será el encargado de validarlas, tanto para el correo entrante como saliente hasta Alice. Este es un esquema general, ya que los servidores en BBM funcionan de forma muy diferente a los servidores SMTP.

c. Diagrama de componentes del contenedor BBMChain

Este contenedor es el corazón (*core*) de la arquitectura BBM y es donde se almacenan todas las reglas administrativas que permiten a un usuario gestionar sus cuentas, contactos, buzones y reglas de control utilizadas por los protocolos de BBM. La Figura 50 muestra los componentes que se encuentran en BBMChain. Se puede ver que este contenedor se comunica con otros dos contenedores, el contenedor Servidor y el contenedor Cliente BBM.

Figura 50. Diagrama de componentes de BBMChain.



Fuente: Elaboración propia.

BBMChain tiene una arquitectura genérica de una *blockchain* que proporciona servicios de contratos inteligentes, replicación, *hashing*, firma digital y protocolos de consenso. Cada nodo que participa de la red gestiona una copia del libro mayor donde se registran todas las transacciones de forma permanente y consensuada. Este contenedor no es más que una implementación específica de Ethereum, Hyperledger o EOS, por mencionar algunas. En la prueba de concepto desarrollada en este trabajo, se utilizó Ethereum como contenedor BBMChain, pero otra implementación podría perfectamente utilizar otra *blockchain* pública conocida o alguna privada desarrollada expresamente para este tipo de solución.

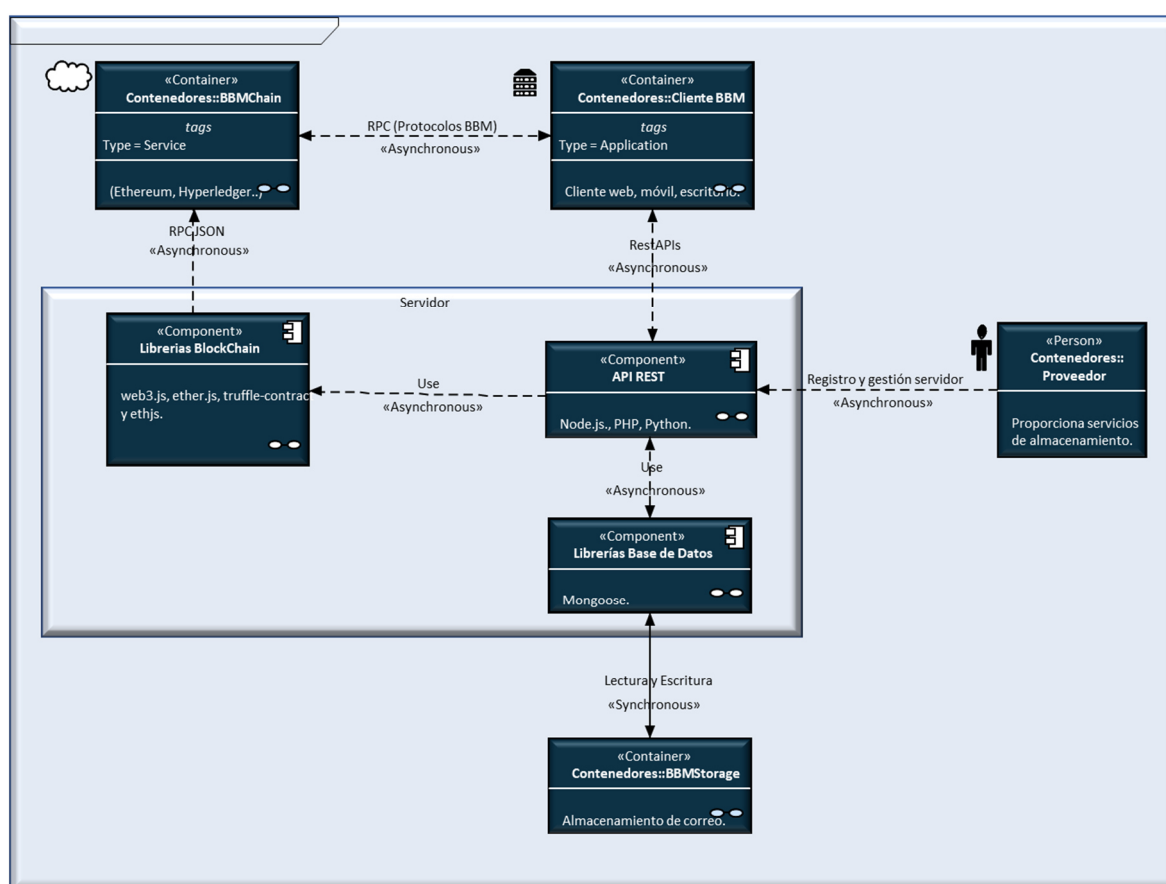
BBMChain almacena los protocolos distribuidos de BBM en forma de contratos inteligentes, las reglas de control y validación más importantes, la información de los usuarios, cuentas, contactos y buzones de forma descentralizada y segura. Estos protocolos involucran la participación de muchos componentes que se describirán en la sección sobre la parte dinámica de la arquitectura. En la Figura 50 también se puede observar cómo los servidores ejecutan los

protocolos BBM a través de los contratos inteligentes, para gestionar sus datos y las cuentas de los usuarios registrados en ellos.

d. Diagrama de componentes del contenedor Servidor

El servidor hace parte de la RPA y es el contenedor que hacer de interfaz con los proveedores de almacenamiento. Hay tres componentes principales de este contenedor: las librerías de *blockchain*, la Interface de programación de aplicaciones (API) de Transferencia de estado representacional (REST) (Doglio, 2018) y las librerías de bases de datos. En la Figura 51, se puede ver cómo están relacionados estos componentes dentro del contenedor.

Figura 51. Diagrama de componentes del servidor.



Fuente: Elaboración propia.

Un cliente BBM realiza las llamadas al Servidor utilizando las API REST. Estas hacen uso de los protocolos BBM por medio de librerías de *blockchain*, como puede ser web3.js (W.-M. Lee, 2019; Panda & Satapathy, 2021), utilizada para la prueba de concepto de BBM.

El proveedor puede configurar y gestionar el servidor utilizando el componente de las API REST, almacenando la información de gestión en BBMChain y toda la información de los

correos en BBMStorage. Todas las llamadas al gestor de base de datos utilizan sus respectivas librerías. En la prueba de concepto desarrollada en este trabajo se ha utilizado Mongoose (Mardan, 2018) para el acceso a la base de datos MongoDB (Chauhan, 2019).

El proveedor es el usuario o empresa que proporcionará el servicio de almacenamiento. Este actor hace uso de las API REST para realizar el registro y las gestiones administrativas en BBMChain. Esta es la arquitectura necesaria para que un proveedor pueda implementar los servicios de almacenamiento y unirse a RPA para estar disponible a los usuarios en BBM.

4. Aspectos dinámicos de la arquitectura BBM

Los diagramas anteriormente descritos muestran los aspectos estáticos de la arquitectura BBM. Los aspectos dinámicos de BBM se basan en un conjunto de protocolos que describen cómo, quién y cuándo se deben ejecutar las acciones que le dan consistencia a la arquitectura y el comportamiento esperado. En esta sección se expondrán todos los protocolos utilizados dentro de BBM, centrándonos en los aspectos de seguridad de estos. Cada uno de los protocolos puede ser utilizado dentro del contexto de la aplicación que se le dé a la arquitectura.

El objetivo de este conjunto de protocolos es proporcionar todos los aspectos operativos y de seguridad que deben ser llevados a cabo para que BBM cumpla con los requisitos especificados en las secciones anteriores y se cumplan los objetivos de este trabajo. Los protocolos los ejecutan los actores del sistema de forma interactiva. Algunos protocolos se ejecutan automáticamente con cierta periodicidad, como puede ser el protocolo que utiliza un cliente BBM para leer el correo almacenado en un servidor, o el protocolo que utilizan los servidores para comunicarse.

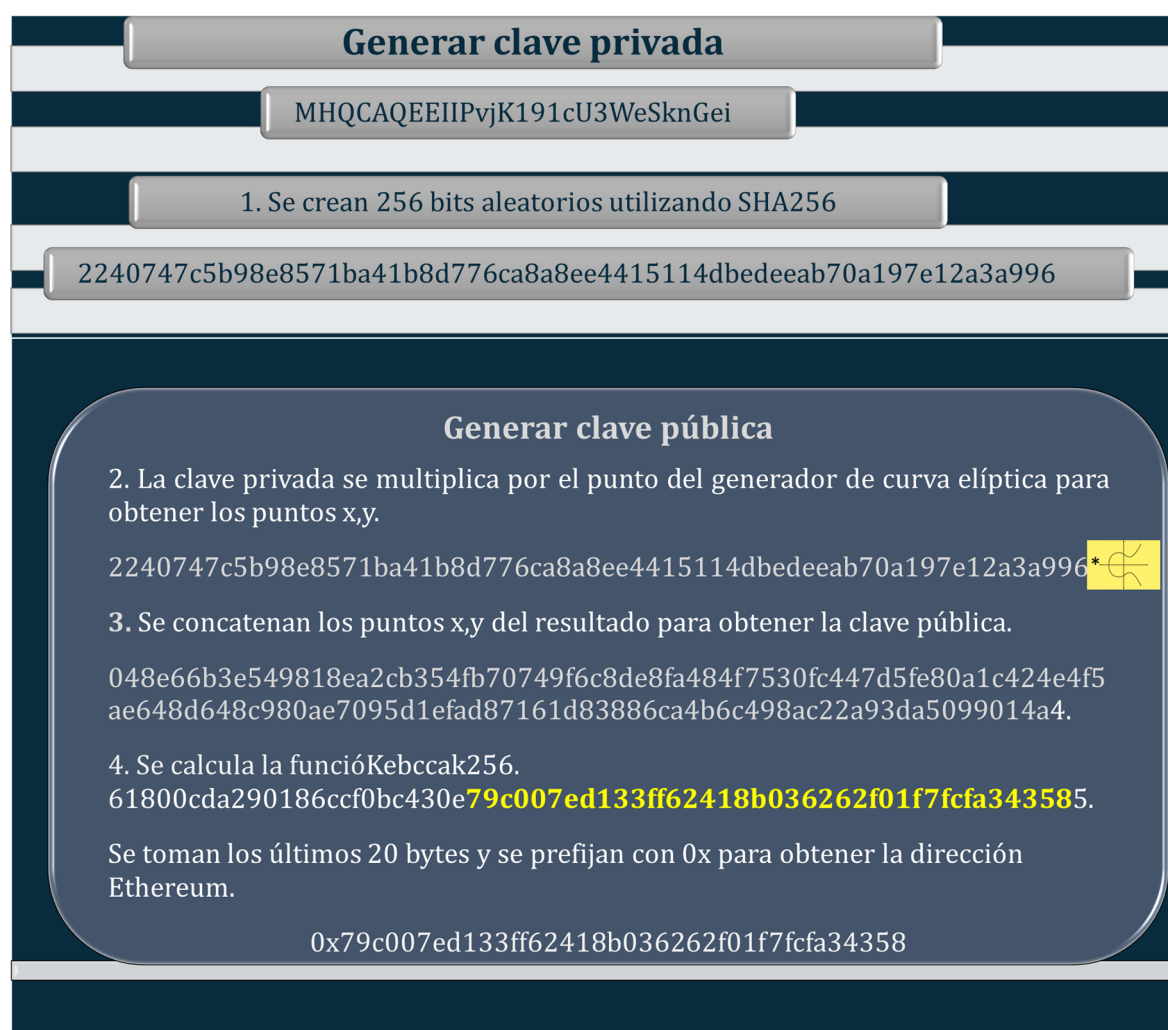
Independientemente de quién ejecute un protocolo, hay muchos elementos comunes entre ellos y, por esta razón, la especificación debe ser lo más general posible remarcando los aspectos que diferencian un protocolo de otro. Un aspecto común que comparten los protocolos es la operativa de almacenamiento y consulta de información en BBMChain, con el cual todos los protocolos interactúan. Los aspectos comunes se explicarán una sola vez y cuando se necesite incorporar la explicación en otro protocolo, se hará una referencia a él con el objetivo de simplificar.

4.1. Protocolo de Generación de Claves (PGC)

Para realizar cualquier operación dentro de BBM, un usuario necesita contar con una clave pública y una privada. Tanto los actores, usuarios y proveedores, deberán estar en posesión de dichas claves para interactuar con los componentes de BBM. Algunos actores necesitarán más de un par de claves, como los usuarios que pueden utilizar otras claves para identificar ciertas entidades como las cuentas o los buzones.

El proceso para obtener las claves debe seguir un protocolo específico que normalmente está especificado en la documentación de *blockchain* utilizada. Para la prueba de concepto, se utilizó Ethereum. Se explicará a continuación el proceso, así como los aspectos de seguridad que se recomiendan para el uso y almacenamiento de dichas claves en esta arquitectura. Con el propósito de ayudar a una mejor comprensión del procedimiento utilizado en la generación de las claves, la Figura 52 muestra el proceso de forma gráfica.

Figura 52. Proceso de generación de dirección Ethereum.



Fuente: Elaboración propia.

Se consideran lo siguiente con respecto a las claves de la figura anterior:

- Se debe crear una clave privada aleatoria usando la función SHA256.
- Las claves privadas se generan como 256 bits aleatorios, que son 64 caracteres (hexadecimales) o 32 bytes.
- Las claves públicas de Ethereum (128 caracteres / 64 bytes) se crean utilizando el algoritmo ECDSA.
- Ethereum usa secp256k1 para generar claves públicas.
- La clave pública es un punto en la curva elíptica. Tiene coordenadas x e y que se utilizan para crear la dirección Ethereum.
- La dirección Ethereum se genera aplicando la función *hash* de Ethereum keccak256, utilizando las coordenadas x e y para crear la dirección a partir de los últimos 20 bytes del resultado.
- Por último, se retorna la clave pública (Pk) y la dirección Ethereum y se almacena la clave privada de forma segura.

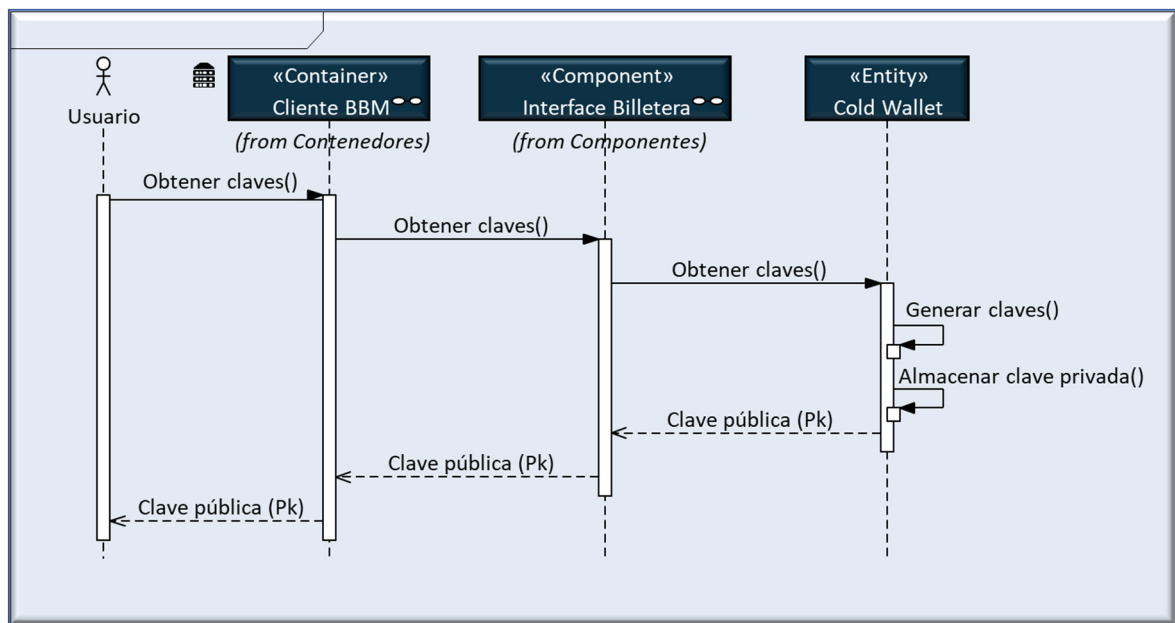
Este proceso se debe realizar en lo posible “fuera de línea” y desde un dispositivo seguro, para que la generación de las claves se realice de forma confidencial. Aunque el cliente BBM debería facilitar esta opción por temas de usabilidad, lo ideal es que esto se realice de forma externa, ya que es un proceso crítico dentro de BBM.

El objetivo de este proceso es garantizar que nadie pueda copiar la clave privada generada. El robo o pérdida de la clave privada significaría la pérdida de la cuenta en *blockchain*, pero en el contexto de BBM esto tiene una lectura diferente, ya que el usuario utiliza la clave como identificador para generar su registro, pero el control de usuario se realiza desde BBM.

Para solventar el problema de la pérdida de las claves, se propone un protocolo de recuperación del usuario que será explicado más adelante y una estructura de usuario basado en varias claves que pueden ser generadas utilizando derivación de claves por semilla (Pieter, 2012). Las claves generadas por semilla pueden sufrir ataques de escalada (Das, Erwig, Faust, Loss, & Riahi, 2021; C. I. Fan, Tseng, Su, Hsu, & Kikuchi, 2019), pero en BBM eso no significa perder el usuario. La única forma de perder el usuario es perder la clave y no poder regenerarla a partir de la semilla. La gestión y almacenamiento seguro de la clave es un tema que está fuera del alcance de esta propuesta.

La Figura 53 muestra el Protocolo de Generación de Claves (PGC). Un usuario solicita la generación de las claves al cliente BBM, este interactúa con la “*interface Billetera*” que hace de interfaz entre el cliente BBM y la billetera utilizada. La billetera debe generar el par de claves pública y privada, almacenar la clave privada de forma segura y retornar al cliente la clave pública. El objetivo del protocolo es asegurar la confidencialidad de la clave privada, lo que solo es logrado si se generan las claves y se mantiene el uso de la clave privada fuera de línea, como por ejemplo en las “Cold Wallet”.

Figura 53. Diagrama del protocolo PGC.



Fuente: Elaboración propia.

El protocolo PGC realiza los siguientes pasos:

Protocolo PGC
1. El usuario accede al cliente BBM. 2. Ejecuta la opción “generar claves” que llama a una función de la Billetera. 3. La función es ejecutada por el generador de claves quien genera un par de claves, pk y sk. 4. ColdWallet almacena sk de forma segura utilizando el formato UTC/JSON wallet. por ejemplo: <pre> { "version":3, "id":"aeda57e9-4373-4789-aef0-36be970b5ff8", "address":"272e87c2ebb3f5047c69ba37ef06920d4ef2e3ef", "crypto": { "ciphertext":"884c95c9b9524c4eefda15128bda74c3e5c4151d70e17d4646507e52a1360cdc", "cipherparams":{"iv":"c22d1a1fbeb4b66e63511e184041fcf8"}, "cipher":"aes-128-ctr", "kdf":"scrypt", "kdfparams":{ "dklen":32, "salt":"5647c5791bc63824ec7a4ea98a0b752e93cdf1cfb1a2dea4d425f480d8bfba52", "n":8192, "r":8, "p":1}, "mac":"ba77917f37ad0bcbad898b9d44da62cc4c8561cdc260c5d7fd3c64b99bfacfd6" } } </pre> 5. Se retorna al cliente BMM la clave pk para ser mostrada al usuario.

Normalmente sk es almacenada en los denominados entornos de seguridad personal (PSE). Dichos entornos protegen a las claves privadas de acceso no autorizado y algunos proporcionan ciertas funcionalidades como la firma y el cifrado fuera de línea. Una clave privada puede ser almacenada de forma segura y estandarizada utilizando, por ejemplo, el formato PKCS#12 (Moriarty, Nystrom, Parkinson, Rusch, & Scott, 2014), que es utilizado por muchos navegadores y sistemas operativos para almacenarlas.

Algunas aplicaciones utilizan su propio gestor de almacenamiento, como es el caso del sistema operativo Windows por medio de la aplicación certmgr.msc, o bien podría utilizarse tecnología

hardware como las tarjetas inteligentes (Smart Card), como es el caso de FINREAD. Este último ejecuta aplicaciones Java o dispositivos más complejos como un módulo de seguridad de hardware (HSM) (Boireau, 2018), que son dispositivos optimizados para operaciones criptográficas, pero sin las limitaciones de tamaño de una tarjeta inteligente.

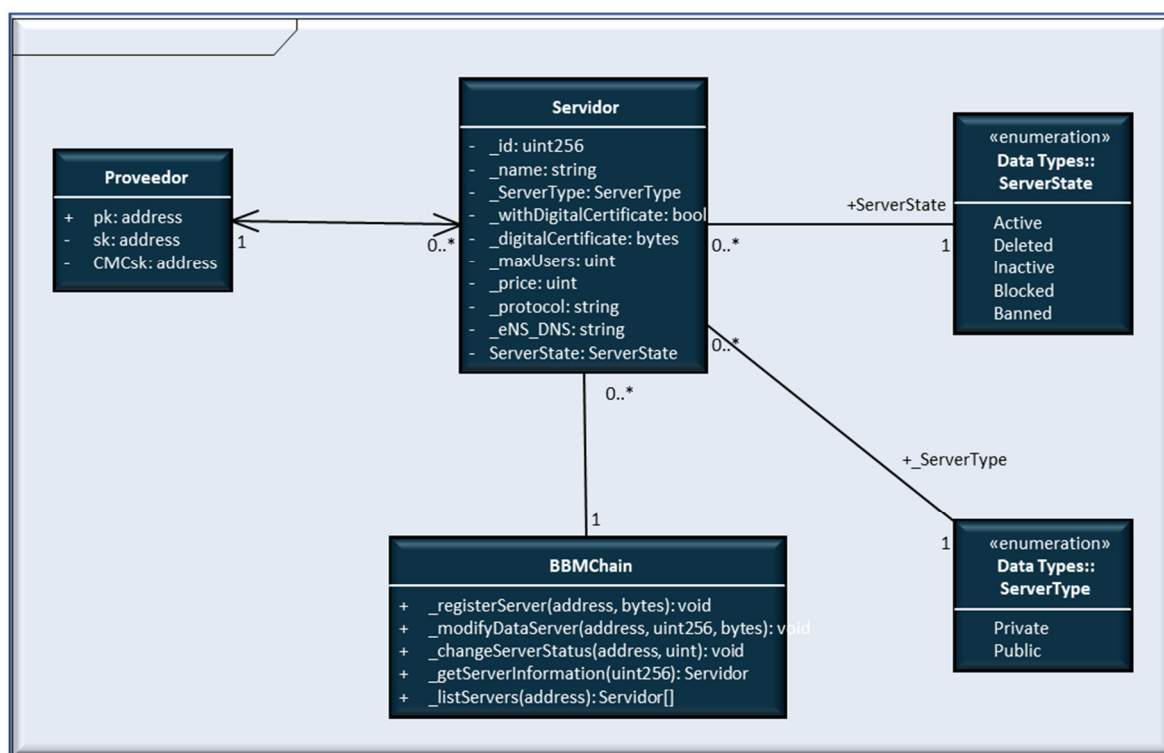
En esta tesis, se recomienda utilizar un PSE que pueda ejecutar las operaciones de generación de claves dentro del mismo dispositivo y, si es posible, también el proceso de cifrado, para una mayor seguridad de las claves. El proceso de autenticación para acceder a las claves debería ser proporcionado por el PSE utilizado.

4.2. Protocolo de Registro de Servidor (PRS)

Para publicar los servicios de almacenamiento en BBM, un proveedor debe registrar un servidor. El proceso de registro almacenará datos identificativos del servidor en BBMChain de manera única, para que el servidor pueda ofrecer sus servicios y estén disponibles para los usuarios de BBM. La identificación del servidor consistirá en un identificador generado en el momento del registro y un identificador que indique su localización.

La Figura 54 muestra el diagrama de clases las entidades involucradas en el registro del servidor, sus campos y métodos más importantes relativos al registro.

Figura 54. Diagrama de clases del servidor.



Fuente: Elaboración propia.

Se puede observar que un proveedor debe estar en posesión de dos claves, una privada (sk) y una pública (pk). La clave pública es la clave asociada al servidor y debe ser generada según el protocolo PGC. Dicha clave es la utilizada por el proveedor para el registro de un servidor. Si el proveedor quiere registrar otro servidor, necesitaría un nuevo par de claves.

BBMChain es utilizado para almacenar parte de la lógica de este protocolo a través de un contrato inteligente, que expone una serie de operaciones o métodos, entre ellos el registro del servidor. Este método requiere la clave pública del servidor (pk) y los datos que serán almacenados en BBMChain. La ejecución exitosa de este protocolo dará como resultado la creación de un registro que contendrá los siguientes datos:

-  **_name:** El nombre del servidor.
-  **_ServerType:** El tipo de servidor. Puede tener los valores “privado” o “público”. Un servidor privado acepta solamente a los usuarios registrados y no aparece disponible para el resto de los usuarios. Si es público, cualquier usuario podría registrarse.
-  **_withDigitalCertificate:** Es un valor booleano que indica si el servidor está en posesión de un certificado digital.
-  **_digitalCertificate:** Almacena la referencia al certificado digital, que normalmente está localizado externamente.
-  **_maxUsers:** Número máximo de usuarios que aceptará el servidor.
-  **_price:** Precio del registro, en caso de estar utilizando un servidor de pago.
-  **_protocol:** Protocolo utilizado para la conexión y depósito de los correos.
-  **_eNS_DNS:** Localización del servidor por DNS centralizado o descentralizado.
-  **_serverState:** Indica el estado del servidor.

El registro del servidor debe realizarse sin poner en peligro la confidencialidad de la clave privada del servidor (sk). Para ello, lo primero que debe hacer un proveedor es presentar la clave pública del servidor a registrar (Pk) ante BBM (utilizando un cliente BBM), el cual iniciará un proceso sencillo de autenticación, que normalmente será realizado por la billetera donde se almacenan las claves como se ha comentado anteriormente. De forma práctica, en la prueba de concepto se utiliza la billetera Metamask (Aaron Davis, 2015) y cuando se conecta al cliente BBM, verifica que el usuario se ha conectado a su billetera. Si no lo ha hecho, solicita la contraseña creada anteriormente cuando se generaron las claves para poder realizar la conexión.

Una vez realizado el proceso de autenticación, desde el cliente BBM, el proveedor podrá proceder desde allí al proceso de registro del servidor. El proveedor debe proporcionar los datos que se solicitan desde el cliente BBM, comprobarlos y validar que todo esté correcto antes de enviarlos al contenedor “servidor” de BBM. Este contenedor, como se explicó anteriormente, expone a través de diferentes API una serie de servicios, entre ellos el servicio de registro de servidores.

Una vez recibidos estos datos, el contenedor “servidor” crea una transacción utilizando algunas librerías de *blockchain* y solicita la firma por parte del proveedor. El proveedor, haciendo uso de la clave privada del servidor, firma la transacción y retorna todos los datos. La transacción es enviada a BBMChain y se queda a la espera que sea confirmada. Una vez la transacción es confirmada, el contenedor “servidor” almacena los datos en su repositorio BBMStorage y envía al proveedor el resultado del proceso.

Un proveedor debe proporcionar toda la infraestructura para dar el servicio a un servidor, como fue explicado anteriormente cuando se describió el contenedor “servidor”. Antes de registrar el servidor, el proveedor debería tener ya disponible para los usuarios un servidor BBM con servicio de almacenamiento de alta disponibilidad, conexión a BBMChain y un servidor con las API REST para la comunicación con los usuarios y con el mismo proveedor para realizar tareas como la ejecución de este protocolo.

De ahora en adelante, se utilizará el termino servidor para hacer referencia al contenedor “servidor” que proporciona los servicios de almacenamiento y enrutamiento de los correos. Este servidor tiene un registro almacenado en BBMChain y proporciona servicios para los usuarios y para la gestión del servidor por parte del proveedor propietario.

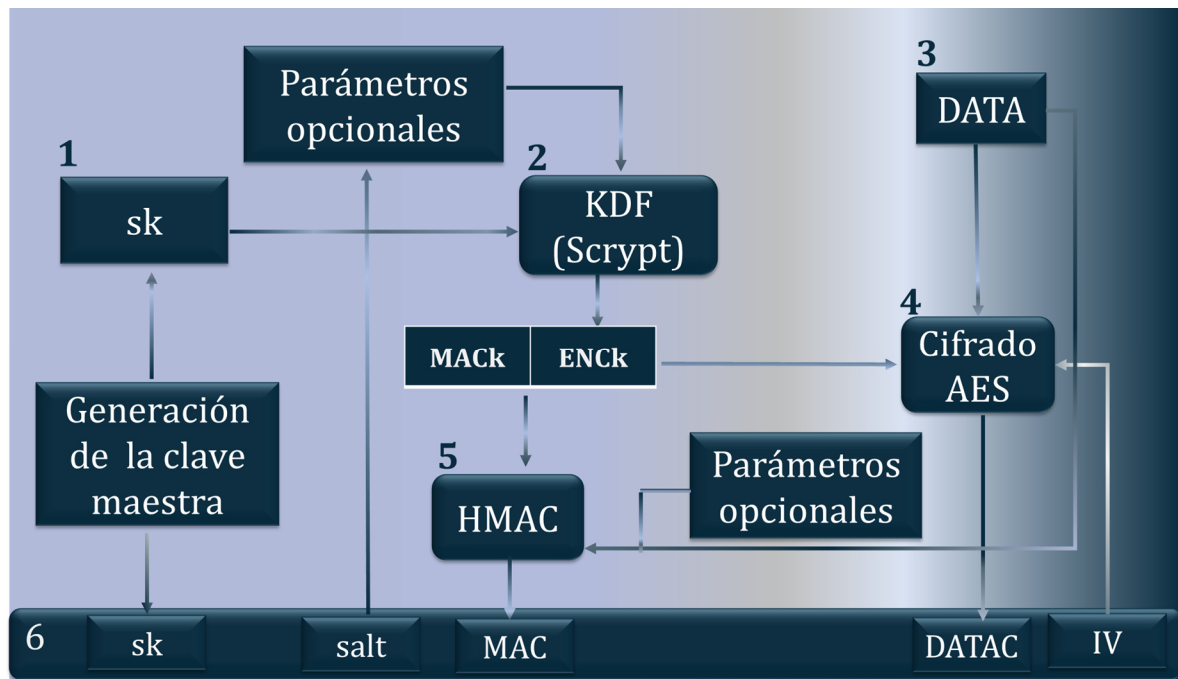
El conjunto de mensajes que se intercambian entre los diferentes actores cuando se ejecuta cualquier protocolo en BBM puede generar mensajes de error o excepciones en caso de que el comportamiento del protocolo no sea el esperado. El servidor debería de gestionar todas las excepciones, como, por ejemplo, las originadas por las librerías de *blockchain* que esté utilizando. La gestión de esas excepciones podría ocasionar la interrupción de la ejecución de un protocolo, como, por ejemplo, si la transacción no puede firmarse.

Cualquier problema que se identifique con un nodo de BBMChain o con BBMStorage debería de tratarse, ya que al ser un protocolo que se ejecuta de forma distribuida y asíncrona es necesario que todos los actores queden en un estado consistente. Se simplificará la gestión de errores o excepciones en los diagramas que se utilicen para representar la ejecución de un protocolo, con el objetivo de mostrar los elementos más relevantes del protocolo y facilitar su entendimiento.

La seguridad del Protocolo de Registro de Servidor (PRS) se sustenta en la confidencialidad de la clave privada, el uso de la firma de forma segura y la integridad de los datos almacenados dentro de BBMChain con la validación y confirmación de la transacción realizada haciendo uso de un contrato inteligente. Hay tres partes críticas dentro de este protocolo que pueden comprometer su seguridad. La primera es la firma, que, con el uso de un monedero externo o de hardware para firmar, minimiza la exposición de la clave privada, brindando confidencialidad al proceso.

La segunda es el cifrado de los datos en caso de ser necesario, que debería de realizarse sin exponer la clave privada o haciendo uso de un esquema simétrico de cifrado. El algoritmo de cifrado de datos (ACD) será utilizado para almacenar datos dentro de BBMChain en todos los protocolos que lo requieran y se indicará cuando sea necesario. La seguridad que proporciona se basa en la generación de una clave maestra que es almacenada en la billetera, el uso de una función de derivación de claves (KDF) como Scrypt (Alwen, Chen, Pietrzak, Reyzin, & Tessaro, 2017; Hatzivasilis, 2017; Percival, 2009) y el uso de un algoritmo simétrico AES (Ako, 2017) con un Código de autenticación de mensajes basado en resumen (HMAC) (Krawczyk, Bellare, & Canetti, 1997) para la verificación de la integridad de los datos cifrados. El algoritmo ACD que se propone en BBM para almacenar cualquier información en BBMChain se muestra en la Figura 55 y se describe posteriormente.

Figura 55. Algoritmo de cifrado de datos (AES + Scrypt + HMAC).



Fuente: Elaboración propia.

Algoritmo ACD (AES + Scrypt + HMAC)

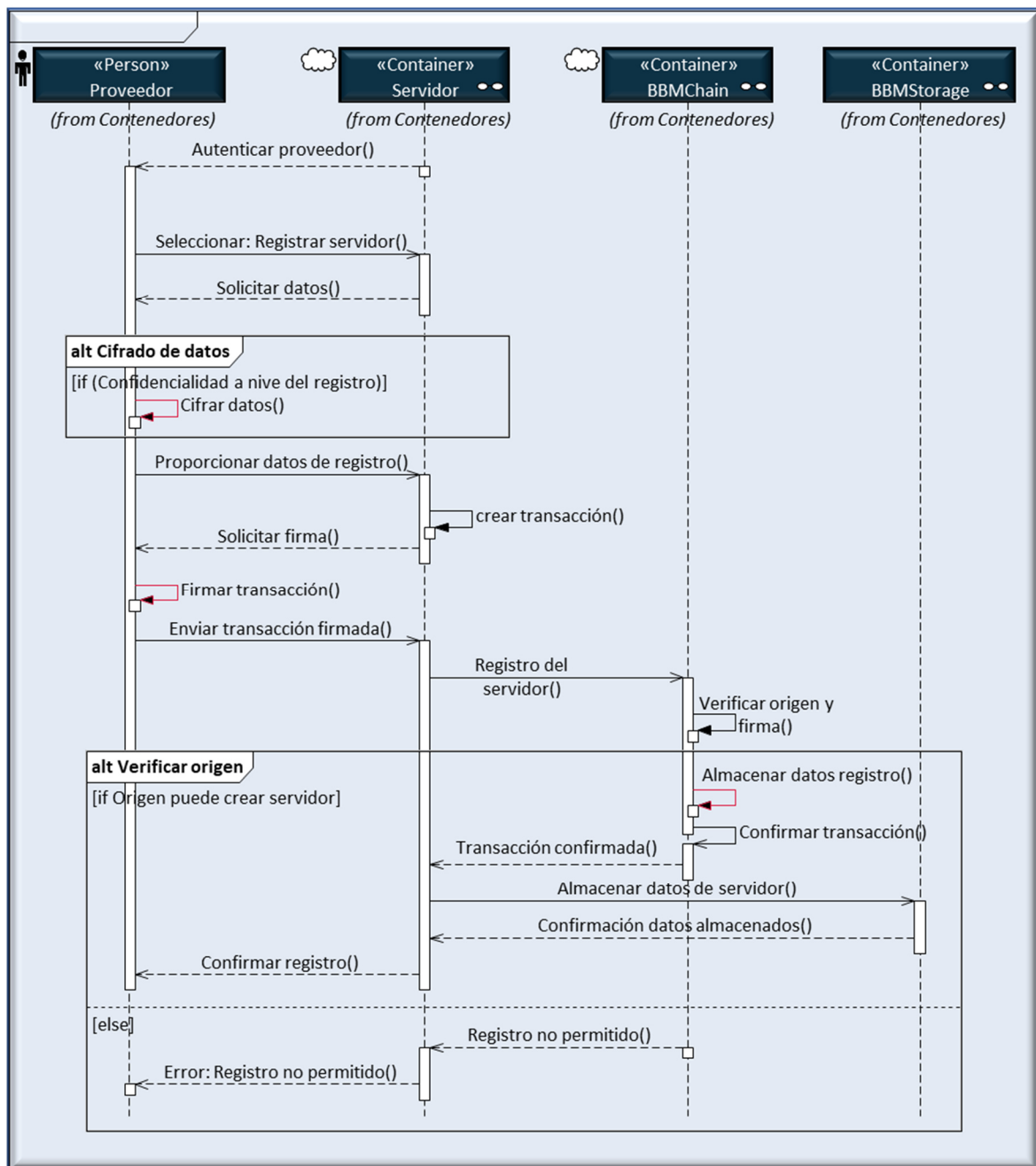
1. Generar un clave sk (privada) y almacenarla como clave maestra de cifrado CMCsk en la Billetera BBM.
2. Utilizando una KDF como Scrypt, generar una clave $k = (\text{CMCsk}, \text{salt}, 32, 2^{20}, 8, 1)$ donde salt es un valor aleatorio de mínimo 16 bytes y la longitud de la clave deseada es 32.
3. Inicializar DATA con los datos a cifrar. Los datos estarán en formato JSON, por ejemplo, el registro del servidor tiene el siguiente formato:
SR:

```
{
  "_name": "Srv01",
  "_ServerType": "Public",
  "_withDigitalCertificate": "1",
  "_digitalCertificate":
    "https://ipfs.io/ipfs/QmTRQ4G25aCDD2GqWr2sS5u1UQYB7tjlgwiDAK4ReMy5op?filename=CertificadoServidor.cer",
  "_maxUsers": "2000",
  "price": "0",
  "_protocol": "HHTP",
  "_eNS_DNS": "BBM.eth"
}
```
4. Cifrar DATA utilizando el cifrador aes-128-ctr con los parámetros $k[16..31]$ (últimos 128 bits de k) y el vector de inicialización IV (un valor aleatorio de 128 bits) y obtener RSC que son los datos cifrados. Se generan DATA + IV.
5. Se calcula la MAC utilizando $k[0..15]$ (primeros 128 bits de k) como clave utilizando código de autenticación integrado HMAC y el mensaje DATA.
6. Se almacena salt, MAC + IV en la billetera BBM y DATA se almacena en BBMChain.

La tercera parte crítica del protocolo PRS es el almacenamiento del registro dentro de *blockchain*, como es mencionado en el paso 6 del algoritmo. El contrato inteligente y los mecanismos de validación y confirmación por parte del consenso realizado por los nodos de *blockchain* proporciona la integridad necesaria para que el registro que se almacene sea el mismo que el proveedor ha firmado y, en caso contrario, sea rechazada la transacción. De esta forma, queda garantizada la integridad de los registros y la información que BBM almacena en BBMChain.

Después de explicar los aspectos técnicos y de seguridad de este protocolo, se pasará a describirlo. La Figura 56 muestra un diagrama de secuencia de los pasos a ejecutar con este protocolo en BBM a nivel de contenedores.

Figura 56. Diagrama del protocolo PRS.



Fuente: Elaboración propia.

En dicho diagrama se pueden ver señaladas en rojo las partes críticas comentadas anteriormente. Los aspectos para resaltar de la figura son los siguientes:

- ✎ El fragmento denominado “Cifrado de datos” hace referencia a la posibilidad que el servidor almacene los datos del registro cifrados. Este proceso debería de realizarse según el algoritmo ACD, explicado previamente en esta sección.
- ✎ Durante el proceso de “Firmar transacción”, se accede a la clave privada y se realiza la firma de forma segura.
- ✎ Se puede ver un proceso denominado “Verificar origen y firma”. Este proceso comprueba que la firma proporcionada en el momento del registro sea correcta y que el usuario que crea la transacción sea el firmante. Por otro lado, el origen se refiere a la clave pública utilizada en el registro. La verificación de origen es la comprobación de que esa clave no ha sido utilizada para el registro de otro servidor y que con esa clave puede ejecutar el contrato inteligente. Se verá esta misma comprobación en otros contratos inteligentes de otros protocolos y la comprobación que se realice va a depender del contexto del protocolo en donde se esté utilizando.
- ✎ “Almacenar datos de registro” se refiere al almacenamiento de los datos en BBMChain, que es una tarea que se realiza en un nodo local, debe ser replicada a todos los nodos y confirmada por *blockchain*.
- ✎ “Confirmar transacción” es un proceso asíncrono. Cuando es finalizado, envía una notificación al servidor indicando el resultado.
- ✎ “Almacenar datos en servidor” es un proceso que se ejecuta cuando se recibe la confirmación de la transacción en BBMChain y se refiere al almacenamiento de los registros internos del servidor en BBMStorage.
- ✎ En el diagrama se ve reflejado el tratamiento de algunas excepciones, pero en posteriores gráficos no se incluyen para darle más simplificada al modelo. Aquí se ha incluido solo a título informativo.

Si una implementación de BBM específica decide utilizar los registros del servidor cifrados (Fragmento “Cifrado de datos” en la gráfica), es importante que defina qué datos específicamente van a estar cifrados y cómo va a gestionar la visibilidad del servidor a los usuarios, ya que estos necesitarán tener acceso al registro del servidor cuando lo requieran.

4.3. Protocolo de Consulta de datos de registro de Servidor (PCS)

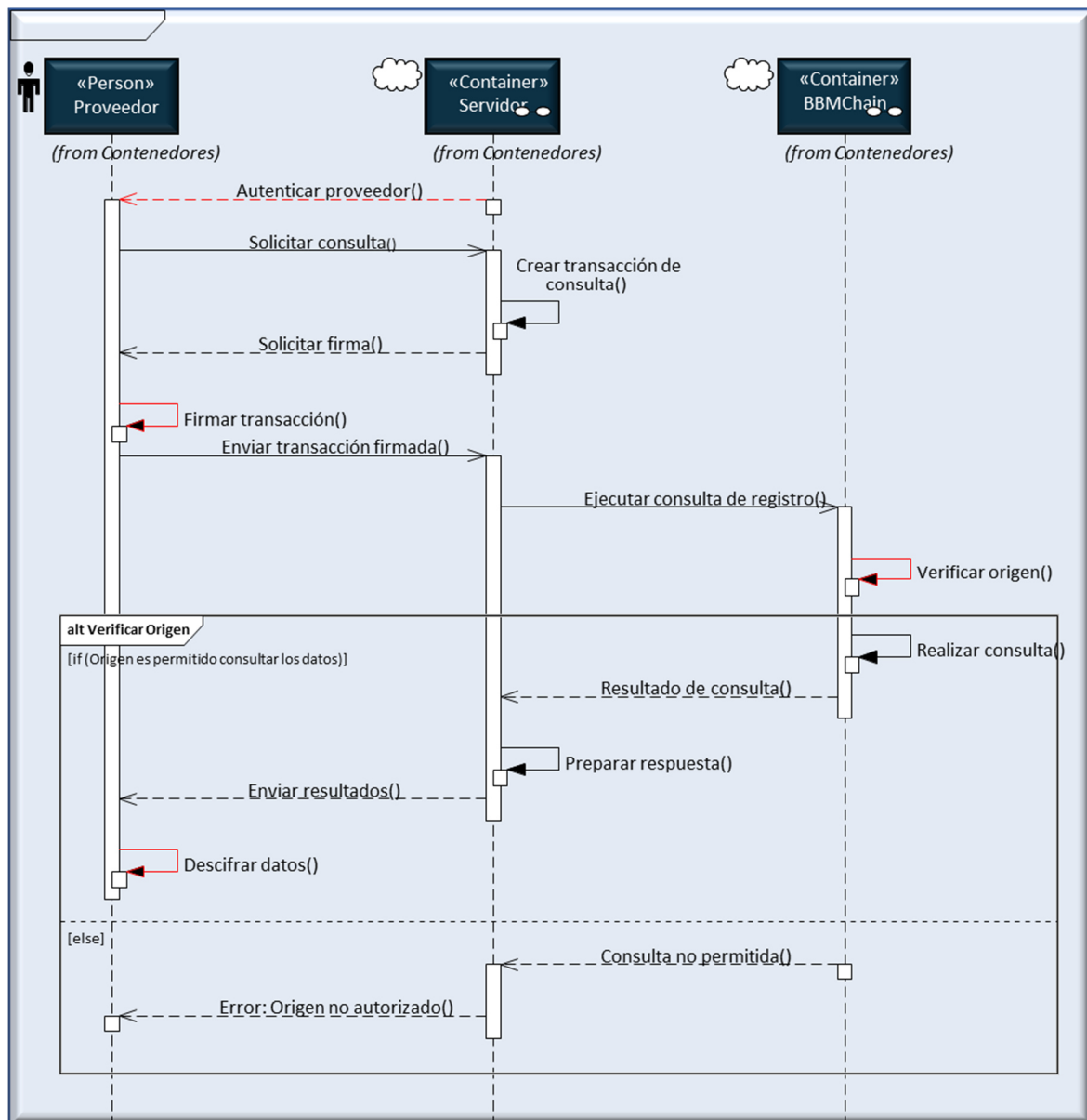
Consultar datos en general en BBMChain puede estar sujeto a restricciones dependiendo del objetivo perseguido y de la casuística de los datos que se estén manejando. En el caso de los datos de registro del servidor, el proveedor podría elegir dos tipos de restricciones de acceso para lograr la confidencialidad:

- Ø El registro puede ser consultado por cualquier usuario.
- Ø El registro puede ser consultado solamente por usuarios de BBM.

En el caso de una *blockchain* basada en permisos, los datos solo pueden ser accedidos por usuarios debidamente registrados e identificados, por lo tanto, no es necesario ocultar el registro, ya que los datos pueden ser accedidos solamente por los usuarios registrados. En las cadenas públicas, como Ethereum, los datos que son almacenados en un contrato inteligente pueden ser consultados por cualquiera, incluso ajenos a la red, ya que son públicos; da igual si la transacción requiere de una firma, ese tipo de control no restringe el acceso a los datos, solamente la ejecución del contrato inteligente. Si se utiliza BBM en una cadena pública, es necesario cifrar los datos como se propone con el protocolo PRS.

En caso de no optar por la confidencialidad del registro del servidor, se pueden utilizar contratos inteligentes con funciones de consulta que no requieren una firma (tipo *view* en Ethereum, por ejemplo) y prescindir del cifrado de los datos. El Protocolo de Consulta de datos de registro de Servidor (PCS) en BBM se muestra en la Figura 57 y deberá ser ejecutado por el proveedor propietario del servidor.

Figura 57. Diagrama del protocolo PCS.



Fuente: Elaboración propia.

Los aspectos para resaltar de la figura son los siguientes:

- ✎ “Crear transacción de consulta” es el proceso que implica la preparación de la transacción de consulta por parte del servidor. “Solicitar consulta” deberá suministrar la clave pública con la que se realizó el registro del servidor.
- ✎ “Solicitar firma” y “Firmar transacción” es una petición y proceso opcional que solo aplica para algunas *blockchain*. En caso de no ser necesaria, no se implementaría.
- ✎ “Ejecutar consulta de registro” es la petición de ejecución del contrato inteligente de consulta de datos del servidor. Debe suministrar la transacción de consulta firmada.

- ✎ “Verificar origen” en el contexto de este protocolo debe comprobar que la clave pública asociada a la transacción recibida haya sido registrada para algún servidor y pueda realizar la consulta.
- ✎ “Realizar la consulta” accedería a el registro del servidor almacenado en BBMChain. El registro estará asociado a la clave pública que fue utilizada para el registro del servidor.
- ✎ “Descifrar datos” es un proceso critico que se ejecuta dentro del contexto del cliente utilizado por el proveedor. En el diagrama aparece el proveedor para simplificar, pero en la práctica hace uso de un cliente *web* que al recibir los datos cifrados los proporciona a la billetera y es en dicho entorno donde se produce el descifrado. Esto es necesario porque en la billetera es donde se han almacena la clave y los demás valores requeridos en el descifrado AES: salt, MAC y IV. El proceso de descifrado es el proceso inverso descrito en el algoritmo ACD (Figura 55).

El cifrado de los datos de registro del servidor es una decisión que recae sobre el proveedor y el implementador de BBM. En una implementación de BBM sobre una *blockchain* privada, los datos de registros podrían estar almacenados sin cifrar, ya que solos los usuarios registrados en BBM podrían consultarlos. En el caso de la cadena de datos pública, cuando se haga uso del cifrado, los datos de registro solamente podrán ser consultados por el usuario proveedor propietario del registro del servidor.

El uso de un esquema de cifrado simétrico para lograr la confidencialidad en BBM dentro de una *blockchain* pública es apropiado cuando un usuario, como el caso de un proveedor, desea almacenar datos confidenciales que posteriormente solo él pueda consultar. Todos los protocolos de seguridad en BBM están constantemente consultando datos en BBMChain y son varios los actores que hacen uso de ellos. Si los datos que van a ser consultados son sensibles, es necesario dotar a BBMChain de una solución que ofrezca confidencialidad a dichos datos.

Se ha abordado este reto en BBM pensando en una implementación de BBM en una *blockchain* pública. Se proponen dos esquemas de cifrado integrados que solucionan el problema de la confidencialidad en BBM:

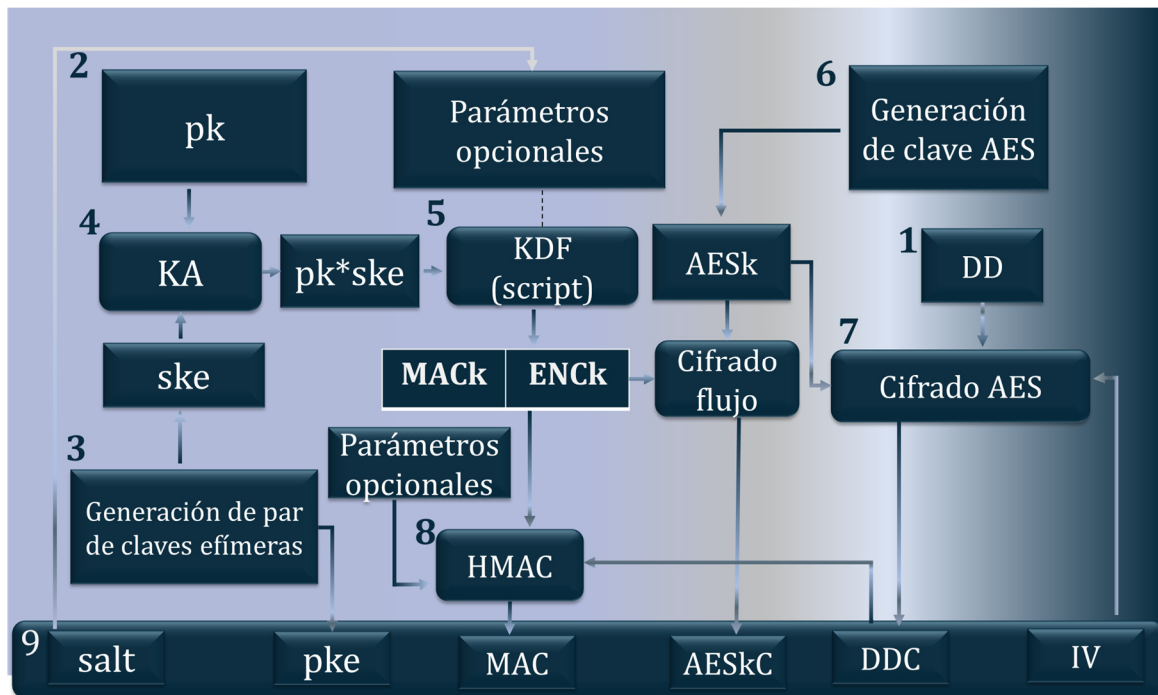
1. Cuando los datos almacenados en BBMChain estén dirigidos para el mismo usuario que los almacenó, el esquema de cifrado utilizado es el algoritmo ACD (Figura 55).
2. Cuando los datos que se almacenan en BBMChain están dirigidos a otro usuario, es necesario utilizar el algoritmo de cifrado para datos compartidos (ACDC).

Se hará referencia al algoritmo ACDC en otros protocolos de BBM, que se irá desarrollando cuando sea necesario. El algoritmo se basa en la criptografía de curva elíptica (ECC) (Oswald,

2005) utilizando el esquema de cifrado integrado de curva elíptica (ECIES) (Oswald, 2005) con cifrado AES, logrando la confidencialidad deseada sin comprometer las claves ni los datos de interés. En ningún momento se ha hecho referencia explícita a cómo se intercambian las claves públicas, que es lo que se explicará en cada uno de los protocolos siguientes cuando sea necesario hacer uso de ACDC. El algoritmo ACDC se desarrolla en los siguientes nueve pasos y se muestra en la Figura 58.

Algoritmo ACDC (ECC + AES-128-CTR + HMAC)	
1.	Inicializar DD (datos del destinatario) con los datos que se deseen cifrar. Los datos estarán en formato JSON siguiendo el formato seleccionado para cada caso.
2.	Obtener la clave pk (pública) del destinatario.
3.	Generar un nuevo par de claves efímeras pke (pública) y ske (privada) donde $pke = G \cdot ske$ y G es el punto generador de la curva elíptica utilizada.
4.	Utilizando una función de acuerdo de claves, calcular la clave de cifrado sECCKey a partir de pk y ske ($pk * ske$).
5.	Utilizando la función de derivación de claves Scrypt, generar una clave $k = (sECCKey, salt, 32, 2^{20}, 8, 1)$ donde salt es un valor aleatorio de mínimo 16 bytes y la longitud de la clave deseada es 256 bits. Se obtiene MACk (los primeros 128 bits) y ENCK (los últimos 128 bits).
6.	Generar una clave secreta AESk y cifrarla utilizando un cifrador de flujo con la clave ENCK, esto genera AESkC, que es la clave cifrada.
7.	Cifrar DD utilizando el cifrador aes-128-ctr con la clave AESk(128 bits) y el vector de inicialización IV (un valor aleatorio de 128 bits). Se obtiene DDC que son los datos cifrados.
8.	Calcular MAC utilizando HMAC (MACk, DDC, SHA-256).
9.	Enviar DDC + salt + pke + MAC + IV + AESkC al destinatario.

Figura 58. Cifrado en BBM utilizando la clave pública del destinatario basado en cifrado híbrido ECIES.

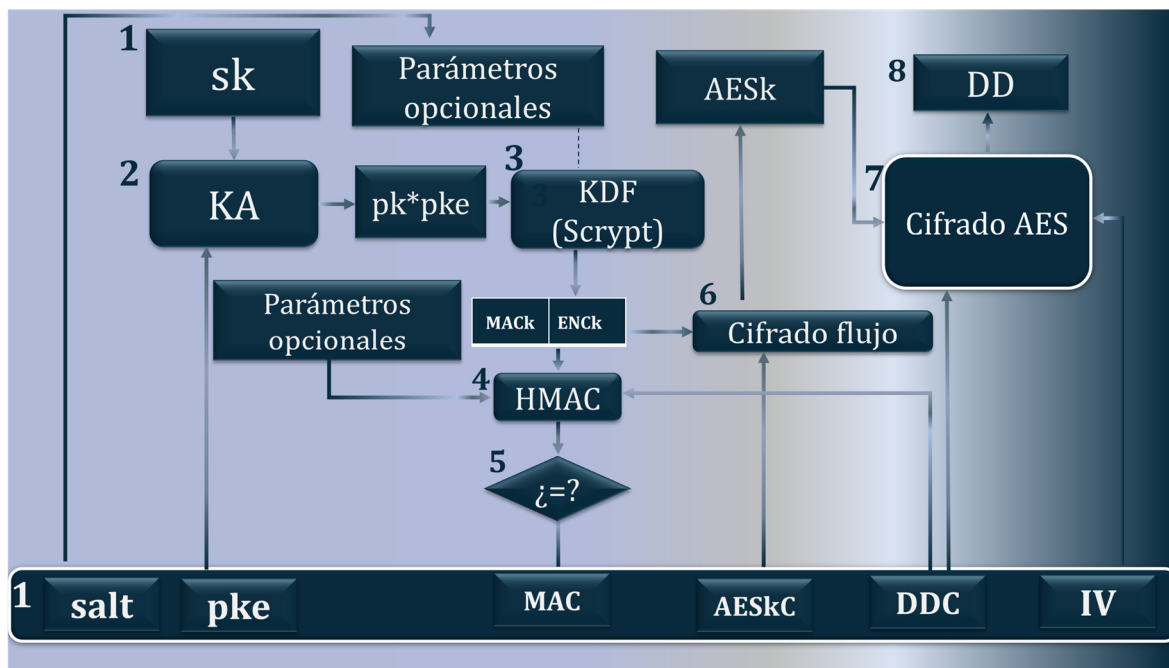


Fuente: Elaboración propia.

En cuanto al algoritmo de descifrado para datos compartidos (ADDC), sigue el esquema de la Figura 59 y se da en ocho pasos:

Algoritmo ADDC (ECC + AES-128-CTR + HMAC)

1. Obtener DDC + salt + pke + MAC + IV + AESkC y la clave privada sk del destinatario.
2. Utilizando una función de acuerdo de claves, calcular la clave de cifrado sECCKey a partir de sk y pke (pke * sk).
3. Utilizando la función de derivación de claves scrypt, generar una clave k = (sECCKey, salt, 32, 2^{20} , 8, 1). Se obtiene MACk (los primeros 128 bits) y ENck (los últimos 128 bits).
4. Calcular MACt utilizando HMAC (MACk, DDC, SHA-256).
5. Verificar que la MACt sea igual a MAC.
6. Descifrar AESkC utilizando un cifrador de flujo con la clave ENck, esto genera AESk, que es la clave utilizada para descifrar.
7. Descifrar DDC utilizando el cifrador aes-128-ctr con la clave AESk(128 bits) y el vector de inicialización IV.
8. Se obtiene DD que son los datos originales.

Figura 59. Descifrado en BBM utilizando la clave privada del destinatario basado en cifrado híbrido ECIES.

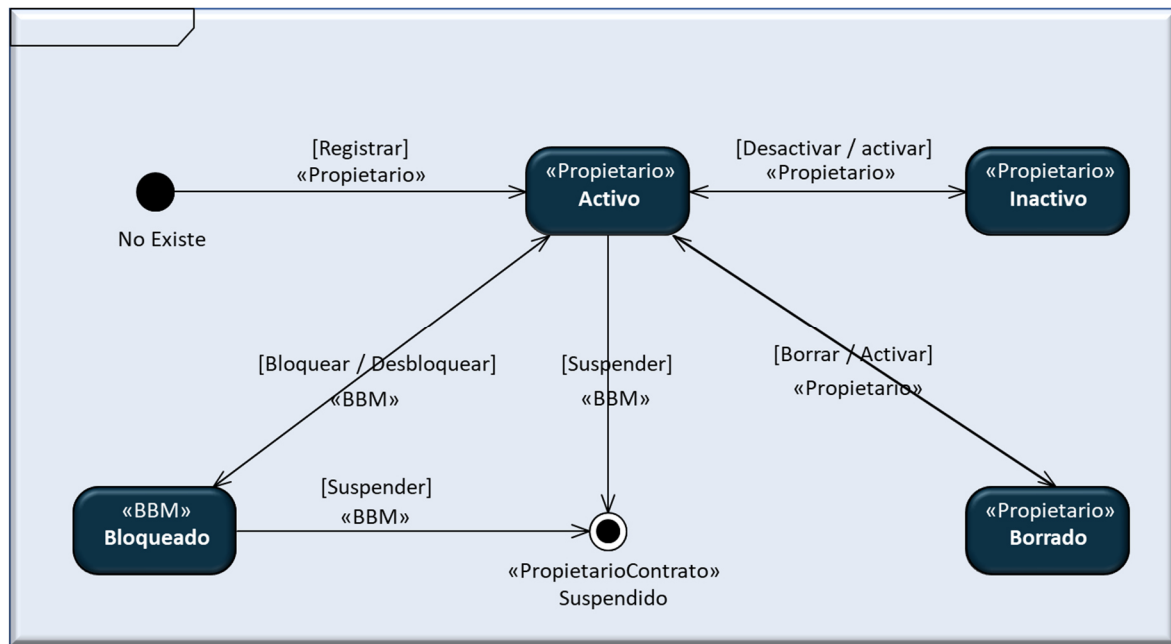
Fuente: Elaboración propia.

4.4. Protocolo de Modificación de datos de Servidor (PMS)

Modificar los datos del servidor es una tarea de gestión en BBM. La gestión de un servidor es una actividad que realiza el proveedor propietario del servidor y una de las tareas de gestión es la modificación del registro del servidor. Uno de los campos objetivo de modificación es el campo de estado (**_serverState**) y su importancia radica en que es el que define el estado del servidor. Si en la ejecución de un protocolo es necesario consultar los datos del servidor, este es el campo que consultará.

Los servidores tienen una representación lógica dentro de BBMChain, el registro, y es el estado del servidor el que determina la disponibilidad y operativa del servidor. La Figura 60 muestra una máquina de estados donde se pueden ver las diferentes transiciones por las que puede pasar un servidor dentro de BBM.

Figura 60. Máquina de estados del servidor.



Fuente: Elaboración propia.

Con respecto a esta máquina de estados, se considera lo siguiente:

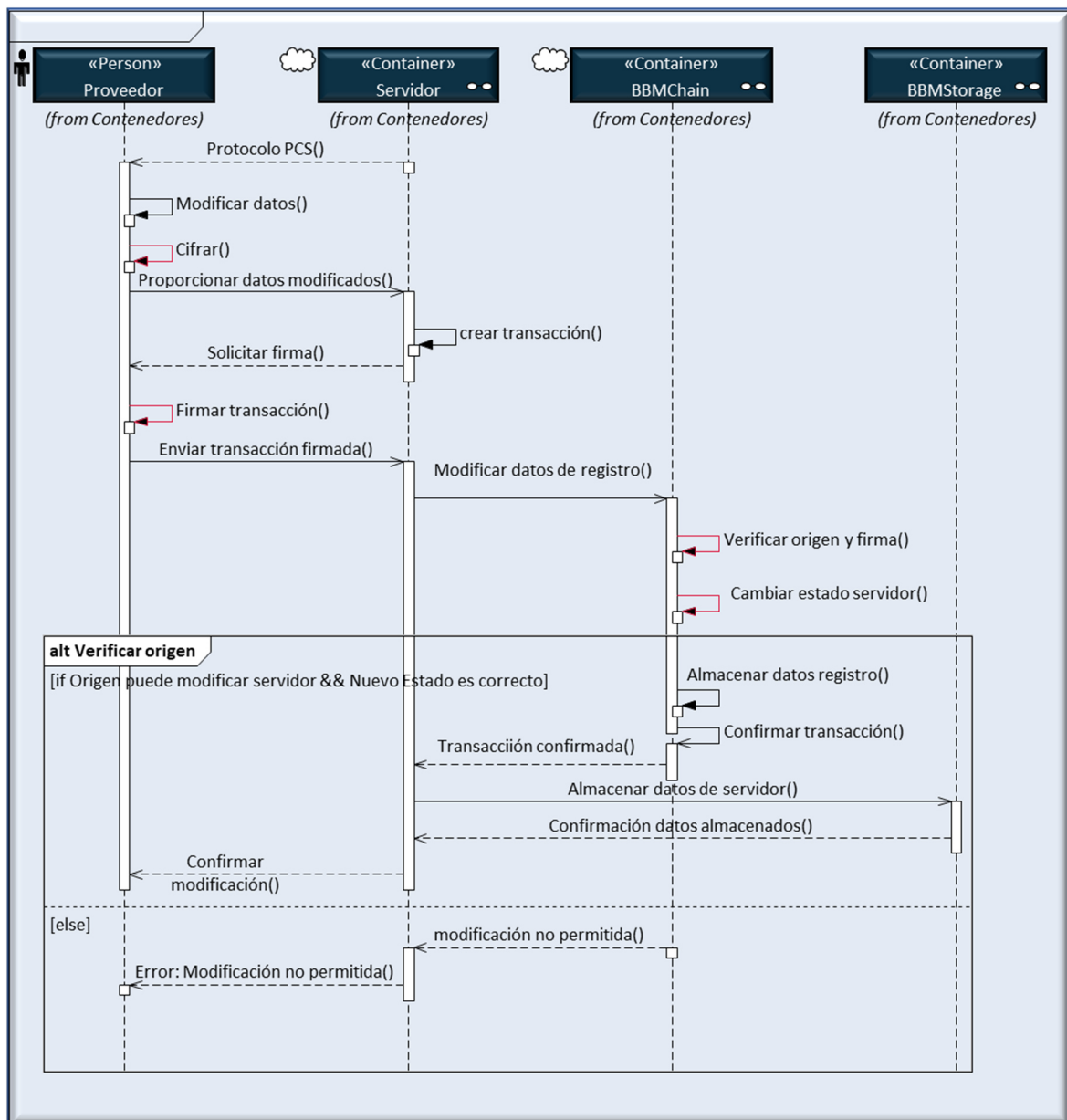
- ✿ Al finalizar el registro de un servidor (protocolo PRS), este queda en estado “activo”. Es en este estado donde el servidor aparece como disponible para que los usuarios puedan registrarse. Mientras el servidor esté activo, los usuarios podrán depositar los correos y consultarlos.
- ✿ El servidor podría ser desactivado (estado “inactivo”) y activado por el propietario para realizar labores de mantenimiento o cualquier otra actividad que requiera. Este proceso no debería de afectar la disponibilidad del servicio para los usuarios que se han registrado ya que, de lo contrario, los usuarios muy probablemente dejarían el servicio y se registrarían en otro servidor. Es importante que el proveedor proporcione una solución a este problema. Cuando se ha creado el servidor, mientras se ajusta y no hay usuarios registrados, podría cambiarse a estado “Inactivo” para tales actividades.
- ✿ Un servidor podría pasar al estado “borrado” si el propietario lo solicitara. Ningún usuario podría registrarse y solo se puede borrar si no hay usuarios registrados en él. El proveedor podría reactivarlo y arrancar de nuevo el servicio.
- ✿ BBM podría suspender un servidor en caso de incumplir las reglas de seguridad, en cuyo caso todos los usuarios deberían migrar sus cuentas a otro servidor. Ese proceso no debería de afectar a los usuarios dado que los usuarios tienen control sobre los correos y siempre

deberían tener una copia local en el cliente BBM y el servicio de replicación de BBM activado. Debido a que la verificación de las reglas de seguridad se realiza por diferentes actores, todos los actores que participan en un protocolo pueden regularse entre ellos y denunciar el incumplimiento de alguna regla. Es de este mecanismo del que se vale BBM para suspender un servidor.

En BBMChain se debe implementar la máquina de estados. Esto se puede realizar por medio de una función de un contrato inteligente que gestione el estado del registro. La máquina de estados que se propone aquí es la implementación mínima que un implementador debería desarrollar, pero podría ser extendida si se actualiza esta especificación. La gestión del servidor siempre involucra consultar el registro del servidor para su posterior modificación y es aquí donde es necesario la ejecución del protocolo PCS visto anteriormente. Se pasará a la descripción de cómo debería ser diseñado un protocolo de modificación de datos en BBM, en este caso la modificación de los datos del servidor, pero este esquema aplica perfectamente a otros protocolos incluidos en esta propuesta.

El Protocolo de Modificación de datos de Servidor (PMS) sigue un esquema muy parecido al protocolo PRS, exceptuando por la consulta previa que se realiza con el protocolo PCS descrito anteriormente. En la Figura 61 se puede apreciar de forma simplificada un diagrama de secuencia que muestra cómo se desarrollan cada uno de los pasos en la ejecución de este protocolo. Hay muchos aspectos de validación que se omiten en el modelo, pero que un implementador de BBM debe considerarlos durante su implementación.

Figura 61. Diagrama del protocolo PMS.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PMS son los siguientes:

- ✎ El proceso “Modificar datos” aparece en el diagrama después de la ejecución de PCS. Se parte del hecho de que se ha producido una consulta de los datos del servidor y se realizan cambios al registro.
- ✎ “Cifrar” se ha comentado anteriormente en el protocolo PRS. Como el estado del servidor es un dato público y, además, el contrato inteligente que implementa la máquina de estados debe poder leer esa información y modificarla, no debería estar cifrada. Esto aplica a cualquier campo que otro contrato inteligente necesite procesar o interpretar. El cliente

BBM debería realizar cualquier validación sobre los datos antes de cifrarlos y almacenarlos en BBMChain.

- ✎ Hasta “Verificar origen y firma” este protocolo sigue los mismos procesos y validaciones que PRS.
- ✎ “Cambiar estado servidor” es el proceso más crítico debido a la ejecución de la máquina de estados que debe realizar las diferentes verificaciones antes de hacer las transiciones de estado comentadas.
- ✎ Las partes de seguridad críticas del protocolo PCS coinciden con las expuestas en PRS, como son el cifrado, la firma y la verificación de origen.

Los nodos validadores en BBMChain ejecutan los contratos para incluir los resultados definitivamente en los bloques de BBMChain y confirmar la transacción. El nuevo estado del servidor quedará reflejado cuando todos los nodos participantes se actualicen. Por lo tanto, el estado del servidor es mantenido de forma distribuida por todos los nodos y su integridad asegurada gracias a BBMChain.

4.5. Protocolo de Registro de Usuario (PRU)

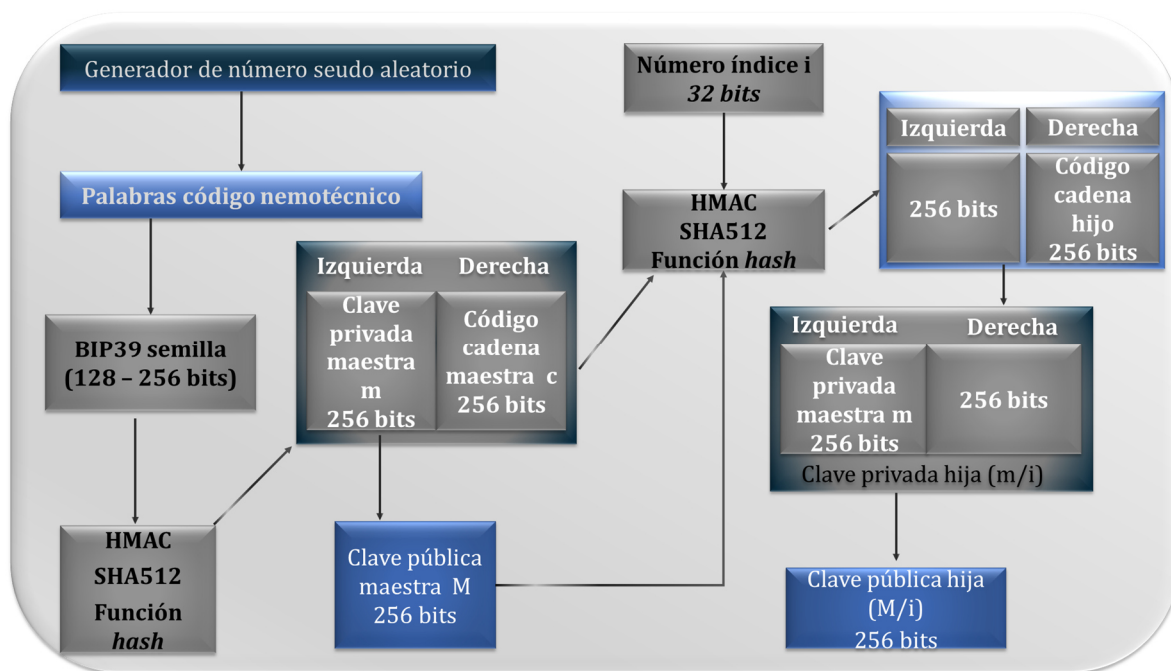
Los usuarios son uno de los actores más importantes dentro de BBM, ya que son los encargados de crear las cuentas de los remitentes y destinatarios de los correos. Dentro de BBM, un usuario es identificado por un valor numérico único y puede tener asociado un nombre. Aunque el usuario tiene asociado un ID, este está asociado a una o más claves, que pueden ser obtenidas siguiendo el protocolo PGC o por medio de claves generadas de billeteras determinísticas jerárquicas (Das et al., 2021). Antes de describir el Protocolo de Registro de Usuario (PRU), se detallarán los aspectos técnicos del uso de las diferentes claves en el contexto del usuario.

Se utilizará pk (clave pública) y sk (clave secreta o privada) para hacer referencia a las respectivas claves del usuario que desea registrarse. Estas claves se generan utilizando el protocolo PGC y se almacenan en la billetera como claves maestras. Para registrar un usuario en BBM es necesario estar en posesión de un par de claves pk y sk. No serán estas claves las utilizadas para el envío de correo, sino que su uso se limitará a aspectos administrativos en BBM. Serán tratadas como las claves maestras para la generación del de las claves necesarias para la creación de las diferentes estructuras utilizadas por los usuarios, como los identificadores de usuario, cuentas, buzones y contactos. La clave sk se denominará “**m**” y la clave pk se denominará “**M**” cuando se haga referencia al algoritmo de generación de claves.

Las claves generadas a partir de las claves maestras tendrán el objetivo de lograr confidencialidad en algunas operaciones realizadas en BBM. Se persigue que, cuando un usuario se registre en un servidor, no sea posible relacionar la cuenta de registro con el usuario y solamente los contactos tengan conocimiento de esa comunicación. De igual manera, se pretende lograr que, cuando un tercero no autorizado tenga acceso a un correo, no sea posible leer su contenido, conocer sus contactos y tampoco saber qué usuarios se están realmente comunicando.

Un usuario se registrará generando un nuevo par de claves a partir de la clave m , llamemos $(m/0)$ y $(M/0)$ a las claves generadas en un primer nivel de generación. La clave $(M/0)$ será utilizada para el registro del usuario y $(m/0)$ para las firmas o el cifrado cuando se requiera un algoritmo de cifrado. El procedimiento para generar las claves en BBM se basa en la Propuesta de mejora de Bitcoin (BIP) número 32 (Pieter, 2012), que describe cómo se generan las claves jerárquicas deterministas (Figura 62).

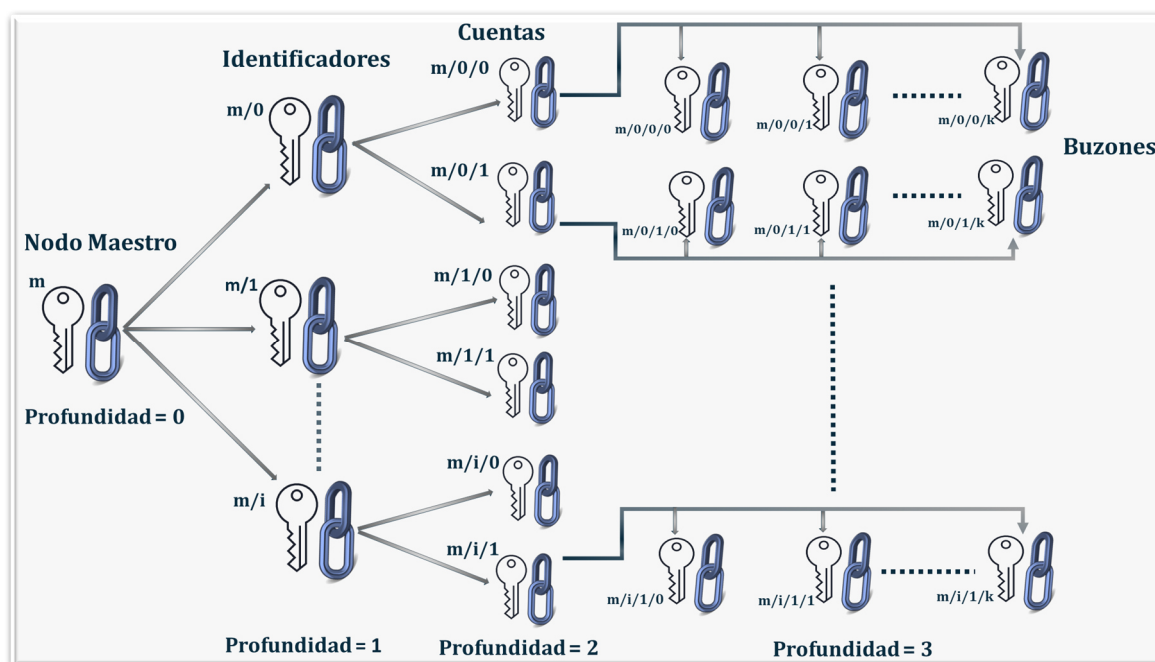
Figura 62. Generación de las claves.



Fuente: Elaboración propia a partir de datos de Pieter (2012).

Para poder generar más cuentas, se deben derivar más claves a partir de m y se obtienen $(m/1)$, $(m/2)$... (m/n) . Para poder organizar las claves en BBM, se utiliza un esquema basado en BIP44 (Palatinus & Marek, 2014), pero con el siguiente formato: Nodo Maestro – Identificadores – Cuentas – Buzones. La correspondencia con BIP44 (Figura 63) sigue la estructura:

Nodo Maestro (m) → Identificadores (m/i) → Cuentas $(m/i/z)$ → Buzones $(m/i/z/k)$.

Figura 63. Estructura de claves en BBM.

Fuente: Elaboración propia.

Con esta estructura se organizan en BBM los identificadores de usuario en nivel de profundidad= 1, las cuentas en profundidad= 2 y los buzones en profundidad= 3. A partir de la clave m , un usuario podría crear varios identificadores de usuarios, varias cuentas y en cada cuenta varios buzones. Esta estructura le permite, por ejemplo, a una empresa, organizar las cuentas en una jerarquía organizada.

A medida que se van creando las claves, se deben almacenar en la billetera donde asociando el uso que se le ha dado. Por ejemplo, las claves $(m/0)$ y $(M/0)$ se deben asociar al identificador generado cuando se crea el usuario. Se deberían crear solamente en BBM claves privadas extendidas, es decir, claves a partir de la clave privada padre. Esto se debe a que las claves utilizadas se necesitan para operar sobre BBMChain y serán utilizadas para firmar y cifrar datos.

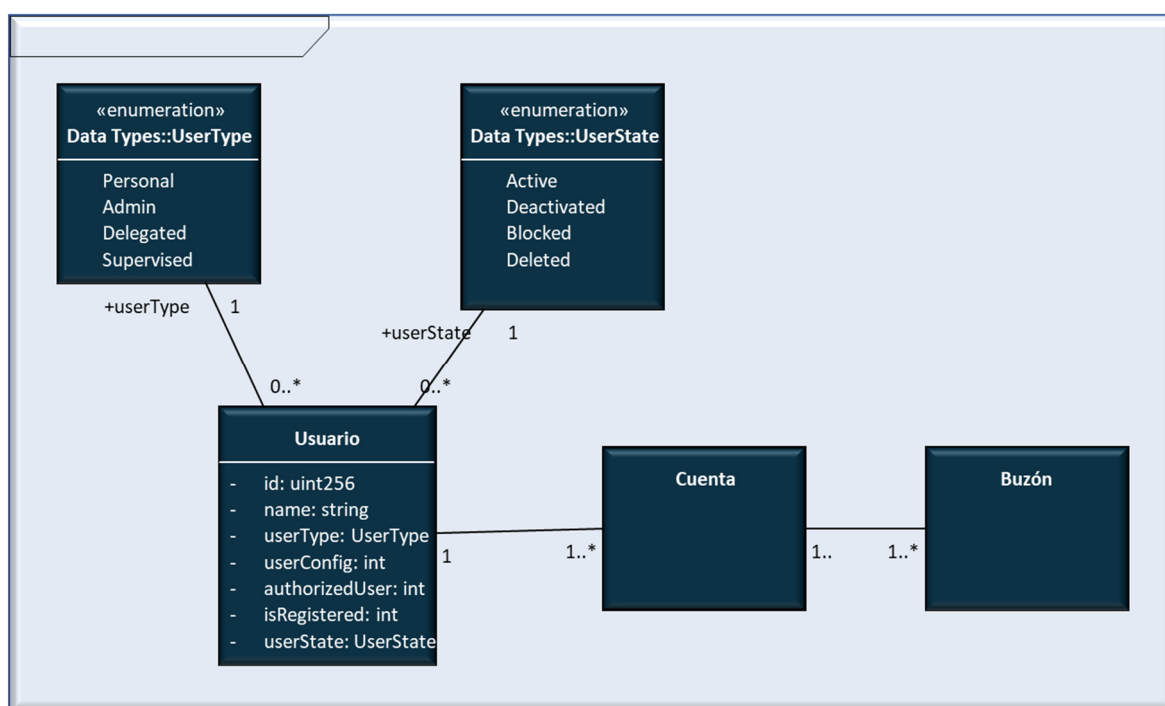
Una vez definidas los aspectos técnicos del protocolo, se procede a su especificación. Un usuario debe registrarse en BBM para poder hacer uso de los servicios del correo seguro distribuido. Para el registro de usuario, es necesario suministrar los siguientes datos:

- 📄 **_id:** El identificador del usuario, que de forma única representa al usuario en BBM. Este identificador es la clave pública (M/n) .
- 📄 **_name:** El nombre que el usuario desea publicar en BBM.

- 
_userType: Puede tener los valores: personal, admin, delegado y supervisado. El valor “personal” es asociado cuando se crea una cuenta personal y los valores “admin”, “delegado” y “supervisado” son utilizados cuando se crean una cuenta corporativa. Los usuarios delegado y supervisado solo pueden ser creados por un usuario admin. Admin sería la cuenta **(m/0)** y el resto de usuario que se desprenden de esa rama serían los usuarios delegados y supervisados, diferenciándose estos entre sí por los datos de su registro de configuración y los otros campos del registro. El usuario delegado es un usuario que gestiona otros usuarios y puede gestionar otros delegados, pudiéndose crear una estructura jerárquica adecuada para cada corporación.
- 
_userConfig: El registro de configuración **Ucf**, donde se almacenan los detalles de seguridad asociados al usuario.
- 
_authorizedUser: Aplicable en el caso de los usuarios corporativos. Indica el usuario delegado que supervisa a este usuario. En el caso de los delegados, puede ser otro delegado o el administrador. Para el caso de una cuenta personal, sería la misma cuenta.
- 
_mainUser: Aplicable en el caso de los usuarios corporativos. Indica el usuario administrador **(M/0)** de la corporación que tiene total autonomía y potestad sobre los demás usuarios.
- 
_isRegistered: Indica si el usuario está registrado o no. Esto tiene sentido dentro del entorno de los contratos inteligentes y sirve para indicar si una clave ha sido utilizada con anterioridad en otra cuenta.
- 
_userState: Indica el estado del usuario y es modificado de acuerdo con la ejecución de la máquina de estados en BBMChain. Los estados básicos son: activado, desactivado, bloqueado, borrado y suspendido. Cada uno de estos estados se relacionan con la disponibilidad del usuario y las acciones que podrá ejecutar.

La Figura 64 muestra un diagrama de clases simplificado donde se puede observar los detalles de la clase “usuario” y la relación con otras entidades de BBM. Un usuario puede tener más de una cuenta y una cuenta puede tener más de un buzón. Para proporcionar confidencialidad a la comunicación dentro de BBM es necesario que cada una de estas entidades esté relacionada de manera oculta, de tal forma que no sea posible relacionarlos entre sí. Al menos la relación de usuarios con cuentas no debería ser deducida por un tercero no autorizado observando los datos en BBMChain.

Figura 64. Diagrama de clases del usuario.



Fuente: Elaboración propia.

Debido a la naturaleza distribuida de esta solución, los protocolos en BBM deben acceder a algunos campos presentes en el registro del usuario sin comprometer la confidencialidad, por ejemplo, divulgando la relación jerárquica entre los usuarios. Si el implementador de BBM desea implementar la opción de cifrado para el registro de usuario, entonces debería de suministrar mecanismos que faciliten luego el transporte de correo, como hacer llegar los registros de configuración de seguridad a los servidores para que verifiquen las reglas de cada usuario. Se propone que esa información se almacene en BBMChain no cifrada y de esa forma se utilicen los registros de seguridad (***_userConfig***) para su consulta durante la ejecución de los protocolos. En caso utilizar el registro cifrado, los registros de configuración no estarían distribuidos y eso afectaría la seguridad debido a que cualquier cambio no estaría disponible para todos los actores y participantes y las reglas de validación serían muy costosas.

Los campos ***_mainUser*** y ***_authorizedUser*** son los que más preocupan, ya que con dichos campos sería posible conocer toda la jerarquía de usuarios de una organización. Un tercero podría saber que usuario está bajo la supervisión de otro y quién es el usuario administrador, y realizar ataques más específicos. Estos campos son necesarios a la hora de validar los cambios en el registro de seguridad en BBMChain por medio de los contratos inteligentes. Por lo tanto, deben estar en texto claro y no cifrados.

Una de las limitaciones al utilizar *blockchain* para BBM está reflejada en este aspecto, ya que el libro mayor muestra las transacciones y las direcciones involucradas. Si una dirección en *blockchain* representa un usuario en BBM que es un usuario corporativo, todos los movimientos que realice en *blockchain* sobre otros usuarios (por ejemplo, cambiar su registro de seguridad) puede ser fácilmente trazado y, por ello, se puede establecer las relaciones entre las cuentas y deducir cuál es la estructura de direcciones corporativa. No obstante, los datos personales del usuario nunca son almacenados en *blockchain* y, por lo tanto, solo dicha estructura pública se utilizaría por razones de trazabilidad, que es la base para la seguridad en la que se sustenta BBM.

Después de estudiar y analizar diferentes alternativas para abordar este problema, se concluye que se pueden presentar tres tipos de soluciones:

1. Estructura pública: Dentro de BBMChain se almacenaría toda la estructura (sin datos personales) de usuarios, cuentas y contactos, y un tercero podría deducir la estructura intrínseca de toda esa información. BBMChain se comportaría como el universo de las cuentas de correo donde los usuarios podrían realizar sus búsquedas y solicitar contactos dentro de ella. Esto sería ideal implementarlo en una *blockchain* donde las solicitudes de contacto requieran un coste y eso intimide a que se envíen solicitudes aleatorias a contactos o ataques de denegación de servicio.

En esta solución, un usuario cualquiera podría buscar aleatoriamente direcciones para solicitar contactos o, sabiendo una dirección específica (una dirección que un usuario haya publicado en su página *web*) de una organización, conocer toda la jerarquía de direcciones y utilizarlo para algún propósito. Si se implementará BBM con este esquema dentro de una *blockchain* privada, el riesgo sería mínimo y sería una solución muy sencilla. La estructura corporativa aquí propuesta está sujeta a mejoras y no es indispensable que una organización se ciña a ella.

2. Estructura semipública: En este tipo de solución, BBMChain almacenaría la estructura de usuarios y cuentas, pero la estructura de contactos se ocultaría de forma que sea imposible ver las relaciones de comunicación entre las cuentas. A diferencia de la estructura pública, aquí no es posible ver quiénes son los contactos de las otras cuentas, y solo los servidores donde los usuarios confían y registran sus cuentas podrían conocerlas cuando intercambian correos. Esto se verá más detalladamente cuando se describan los protocolos relacionados con los contactos. De igual forma que en la estructura pública, los contactos son almacenados en BBMChain, pero no es posible conocer las relaciones entre los usuarios y los contactos.

3. Estructura privada: En este tipo de solución, BBMChain almacenaría toda la estructura de usuario, cuentas y contactos, pero no sería posible trazar la relación entre usuarios y cuentas. Debido a que BBMChain almacena la estructura de los registros y las máquinas de estado, los protocolos no podrían consultar los registros. Esta solución es más compleja de implementar y requiere de otros tipos de soluciones que están bajo estudio por la comunidad científica, como son la aplicación de firmas en anillo para ocultar direcciones, el uso de pruebas de conocimiento cero para verificar ciertos datos sin revelarlos o el cifrado homomórfico que permitirá poder procesar los datos sin tener que descifrarlos.

La pregunta que surge aquí es: ¿cuál es la razón para que sea necesaria la utilización de las tecnologías mencionadas anteriormente?

En las *blockchains* convencionales, cuando un usuario realiza un llamado a un contrato inteligente, suele suministrar su clave pública y su firma. Esto deja una traza dentro del libro mayor y cualquiera puede consultarlo. Si BBM ha sido implementado en una *blockchain* convencional, significa que el usuario BBM creado queda relacionado con la dirección que se utilizó permanentemente. Si un usuario aleatorio crea varias cuentas y utiliza para ellas diferentes claves, no es posible relacionar los datos de la cuenta con el usuario, ya que ha utilizado claves diferentes a la proporcionada para la creación del usuario.

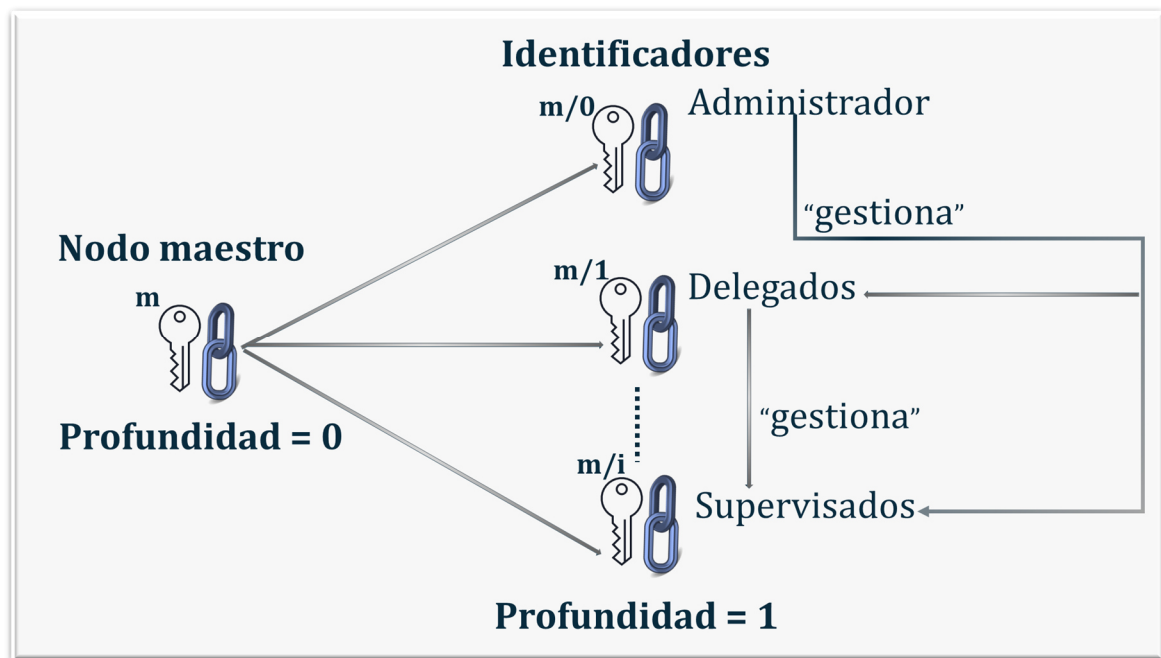
El problema surge con los usuarios corporativos o usuarios que tienen relaciones de dependencia, como los propuestos en este trabajo. Un usuario supervisado que almacena su registro de configuración de seguridad no puede modificar dicho registro, ya que este solo puede ser modificado por un usuario superior. Si utiliza un contrato inteligente para la modificación del registro, debe de establecerse algún tipo de relación para permitir solo las modificaciones de usuarios autorizados, por ejemplo, el campo `_authorizedUser`. El contrato inteligente debe poder leer ese dato, por lo tanto, hay que buscar una forma de ocultarlo y que el contrato inteligente pueda verificarlo. Es deseado entonces poder verificar ese dato sin revelarlo.

En conclusión, el uso de diferentes claves para la creación de usuario y cuentas no soluciona el problema de la revelación de la estructura organizacional del correo en BBM. Para poder ocultar las claves y que las relaciones de jerárquicas no sean deducibles, en este trabajo se plantea utilizar dos claves para los usuarios de forma que una clave sea utilizada como identificador y la otra, llamada “de gestión”, pueda ser utilizada para las modificaciones de seguridad del registro. El enfoque de esta tesis para abordar este problema se basa en el cifrado simétrico con intercambio de claves dentro de BBMChain para el descifrado y el uso de claves

de gestión generadas a partir de las jerarquías de claves. Este enfoque será explicado antes de abordar el protocolo de registro.

Se ha explicado que los identificadores en BBM tienen la forma (m/i) . Para un usuario personal, el usuario autorizado o administrador es el mismo, por lo que no hay ninguna jerarquía que ocultar. Con las cuentas, independientemente del tipo de usuario, sí es necesario ocultar la relación y se utilizará el mismo esquema desarrollado aquí. Para los usuarios corporativos, varios usuarios (m/i) pertenecen a una estructura administrativa en la que el usuario administrador $(m/0)$ es el usuario de nivel superior y el resto de los usuarios están bajo su supervisión, incluidos algunos usuarios delegados que podrán gestionar a otros usuarios supervisados. La Figura 65 presenta un esbozo del esquema propuesto.

Figura 65. Estructura jerárquica de los usuarios corporativos en BBM.



Fuente: Elaboración propia.

Si el usuario administrador (**admin**) va a gestionar el registro de seguridad de un usuario de nivel inferior (**usuinf**), que está almacenado en BBMChain, es necesario que esa modificación requiera la clave de **admin** y no la de **usuinf**. Para poder lograr esto, se verifica que el usuario **admin** que gestiona al usuario y que es almacenado en el campo **_authorizedUser**, sea el usuario que está realizando el llamado a la función de modificación, de lo contrario, no se puede realizar dicho cambio. Esta verificación se debería de realizar dentro del proceso “Verificar origen” en BBMChain, como se ha comentado en anteriores protocolos.

La transacción que se ejecuta para realizar un cambio administrativo contiene la clave que realizó el llamado (clave pública de **admin**) y los datos de la clave del usuario a modificar (clave pública de **usuinf**) y estos datos se almacenan en BBMChain. Si se realiza una búsqueda de la transacción se pueden identificar los usuarios bajo la supervisión de **admin** y, por lo tanto, toda la estructura corporativa. Es necesario que solamente **admin** pueda modificar al usuario **usuinf**, pero **admin** no podrá utilizar su clave pública (M/0), ya que quedaría al descubierto la relación.

Se propone ocultar la clave pública de **admin** utilizando una nueva clave que **admin** pueda generar fácilmente y que sirva como clave de gestión. Un usuario supervisado o delegado puede ser gestionado por otro usuario que esté bajo la supervisión de **admin** y, si hay una modificación del usuario gestor, será necesario generar una nueva clave de gestión y la anular la anterior. Esto lleva a declarar estas claves como “temporales a la gestión” y debe ser tenido en cuenta en el algoritmo de generación de esas claves.

El algoritmo de generación de claves de gestión (AGCG) que permite ocultar las relaciones entre las claves de usuarios y se presenta a continuación:

Algoritmo AGCG

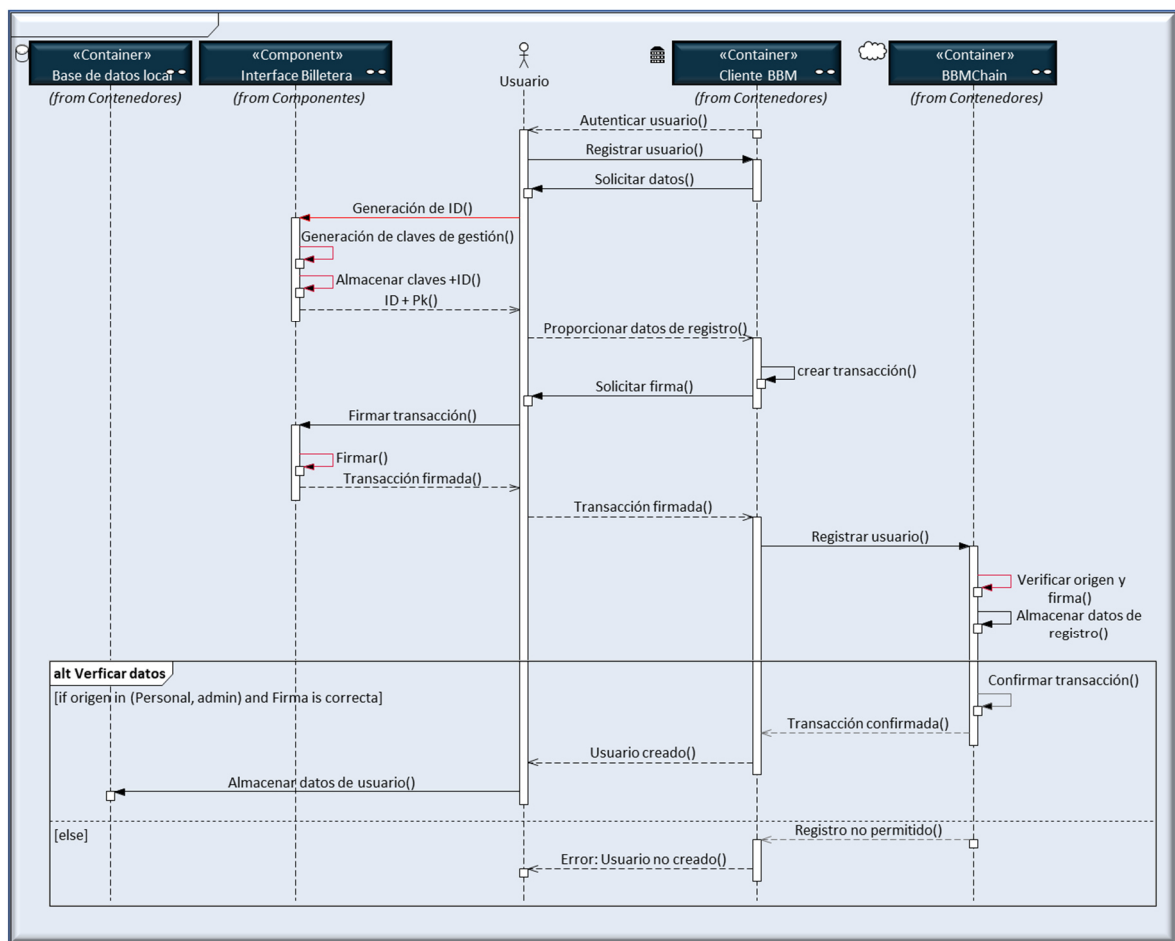
1. Obtener la clave privada $S_{(m/0)}$ de admin y la clave privada $S_{(m/i)}$ del usuario a gestionar.
2. Utilizando una función de generación de claves secundarias (CKD) proporcionar los parámetros $S_{(m/0)}$, $S_{(m/i)}$ y un índice I de 8 bytes, que permite generar 2^{32} claves de gestión. La función retorna una clave privada secundaria.
3. La función CKD hace uso de una función $R = \text{HMAC-SHA512}(\text{Key} = S_{(m/i)}, \text{Data} = \text{ser256}(S_{(m/0)}) \parallel \text{ser32}(I))$, donde ser32 serializa un entero i sin signo de 32 bits como una secuencia de 4 bytes dejando el byte más significativo primero y ser256 serializa el entero $S_{(m/0)}$ como una secuencia de 32 bytes, el byte más significativo primero.
4. El resultado anterior se divide en dos secuencias de 32 bytes. RL (izquierda) and RR (derecha).
5. La clave de gestión se calcula como $\text{parse256}(\text{RL}) + S_{(m/0)} \pmod{n}$, donde parse256 interpreta una secuencia de 32 bytes como un número de 256 bits, siendo el byte más significativo el primero y n el parámetro recomendado para la curva secp256k1 (Brown & Research, 2010).

Este algoritmo se basa en el esquema de generación de las claves (Figura 62), pero se utiliza (m/i) como código de cadena maestro para generar la clave. AGCG se utiliza en protocolos posteriores para la gestión de cuentas y contactos y es la solución de confidencialidad que se propone en BBM para ocultar la relación entre las diferentes claves que representan usuarios, cuentas y contactos. Toda esta información se almacena en BBMChain y está expuesta a cualquier usuario que consulte la red en *blockchain* públicas o a todos los usuarios de *blockchain* privadas.

El protocolo PRU hace uso de todos los aspectos técnicos y criptográficos discutidos anteriormente. La Figura 66 muestra un diagrama de secuencia de las acciones realizadas en BBM para lograr el registro de un usuario. Los aspectos para resaltar son los siguientes:

- ✎ “Autenticar usuario” se realiza desde el cliente BBM interactuando con la billetera. Para simplificar, se utiliza solamente este mensaje.
- ✎ “Registrar usuario” es la petición realizada al cliente BBM para iniciar el registro. El cliente BBM solicitará los datos para que el usuario proporcione la información deseada.
- ✎ La relación que se muestra desde “Usuario” y la “Interface Billetera” representa la interacción que realizaría el usuario frente a su billetera, pero en la práctica el cliente BBM es quien interactúa con la billetera cuando es necesario.
- ✎ “Generación de ID” ejecuta el algoritmo de generación de claves. En el caso de usuarios corporativos, este protocolo lo ejecutaría el “admin”, ya que para generar las claves debe tener acceso a la clave (m/0).
- ✎ “Generación de claves de gestión” ejecuta el algoritmo AGCG para obtener la clave de gestión. Esta clave es la que se utilizará para los campos ***_authorizedUser*** y ***_mainUser***.
- ✎ Con las dos claves generadas se calcularán las claves públicas. Todas las claves se almacenarán en la billetera y se retornarán las claves públicas al usuario para el registro.
- ✎ “Proporcionar datos de registro” enviará todo el registro cumplimentado por el usuario.
- ✎ “Firmar” se realizará con la clave (m/i) generada para el usuario.
- ✎ “Verificar origen y firma” verificará que no se haya creado anteriormente un usuario con la clave (m/i), que la clave de gestión sea diferente de la clave (m/i) para cuentas corporativas y que la firma sea correcta.
- ✎ “Almacenar datos de registro” almacenará el registro en BBMChain y el sistema quedará en espera de finalizar el protocolo al ser confirmada la transacción.

Figura 66. Diagrama del protocolo PRU.



Fuente: Elaboración propia.

En cuanto a los aspectos críticos de seguridad de PRU, se comenta lo siguiente:

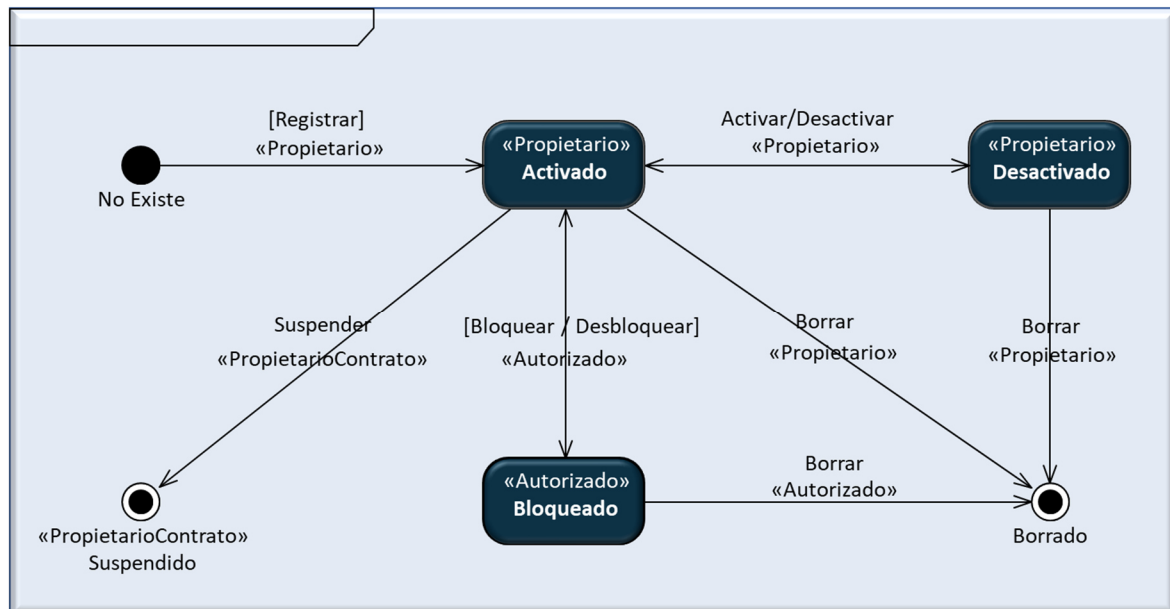
- Δ La generación de las claves del usuario y la cuenta de gestión se realizan bajo la seguridad de la billetera.
- Δ El protocolo garantiza la confidencialidad de los datos de registro con respecto a la relación entre el usuario administrador y el usuario creado.
- Δ El protocolo garantiza la integridad del registro de usuario gracias al uso de BBMChain. El registro no puede ser modificado por un tercero que no esté en posesión de la clave de gestión utilizada en la creación.

4.6. Protocolo de Modificación de datos de Usuario (PMU)

La modificación de los datos de un usuario es una tarea de gestión que está a cargo de un usuario gestor, como se comentó en el protocolo PRU. Los usuarios administradores y personales se gestionan con la misma clave utilizada en la creación y no utilizan cuenta de gestión. El campo *_authorizedUser* y *_mainUser* hacen referencia a la misma clave de creación con estos dos tipos usuarios. El resto de los usuarios serán gestionado por un usuario administrador o por un usuario delegado haciendo uso de una clave de gestión.

Un usuario en BBMChain se comporta como una máquina de estados que regula a través de unas reglas muy específicas dentro de BBMChain la modificación del registro de un usuario. La Figura 67 muestra la máquina de estados del usuario, cuyo estado explícitamente se almacena en el campo *_UserState* del registro.

Figura 67. Diagrama de estados del usuario.



Fuente: Elaboración propia.

Con respecto a esta máquina de estados, se considera lo siguiente:

- ✎ Solamente el usuario que ha creado el usuario (el propietario asignado a campo *_mainUser*) y el usuario autorizado (*_authorizedUser*) pueden modificar el estado del registro de usuario.
- ✎ Cuando se realiza el registro de un usuario (protocolo PRU), el usuario queda en estado “activado”. Este estado, el usuario puede realizar las acciones de depósito, recepción y gestión de correo, siempre y cuando el registro de configuración lo permita (campo *_userConfig* del registro).

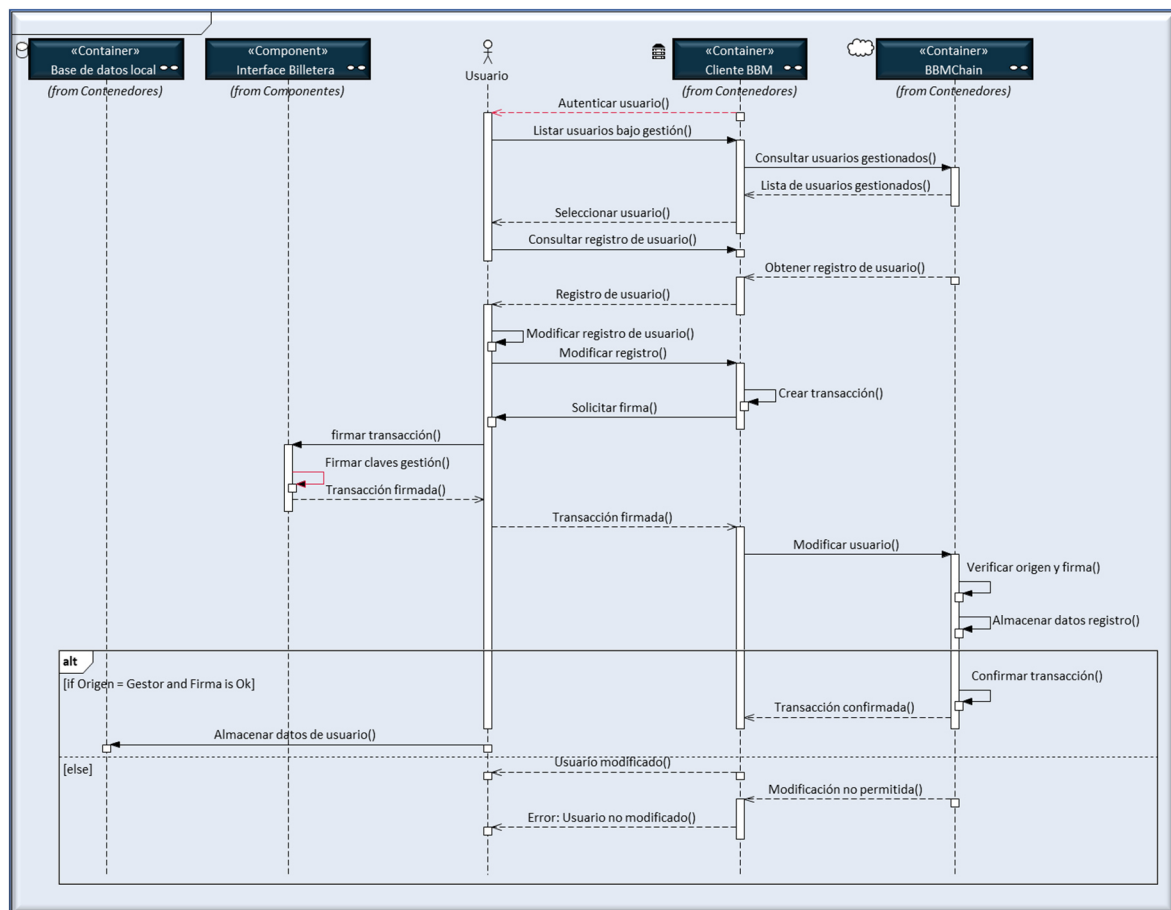
- ✎ El estado “bloqueado” es un estado de suspensión de actividades temporal. Es aplicado por un usuario autorizado para realizar algún tipo de gestión administrativa en la que el usuario puede ser desbloqueado para reanudar su actividad.
- ✎ El estado “borrado” indica que ese usuario no podrá realizar ninguna actividad más. Un usuario puede pasar a este estado desde el estado “activado” o “desactivado” por decisión del propietario. Solo el autorizado puede borrar un usuario que se encuentre en estado “bloqueado”.
- ✎ En BBM, se propone el estado “suspendido”. Este estado podría ser activado por el “propietario del contrato” que representa al sistema BBM. Esto significa que BBM debería de forma automática suspender a los usuarios por diversos motivos, por ejemplo, basado en algún tipo de clasificación. En una implementación de BBM haciendo uso de una *blockchain* pública, podría ser realizado por un conjunto de reglas de suspensión acordadas por algún tipo de gobernanza. En una implementación de BBM haciendo uso de una *blockchain* privada, donde la gobernanza está perfectamente establecida, las reglas de suspensión podrían ser fácilmente impuestas.

Todo este proceso de modificación se realiza dentro del contexto de un contrato inteligente asociado al registro de usuario y almacenado en BBMChain. La Figura 68 muestra el diagrama de secuencia donde se puede ver como interactúan los diferentes actores en el Protocolo de Modificación de datos de Usuario (PMU). Algunos procesos representados en el diagrama han sido explicados anteriormente y no se mencionarán más en la descripción de los posteriores protocolos a menos que sea relevante. Los aspectos para resaltar son los siguientes:

- ✎ Para poder realizar cualquier modificación del registro de usuario, es necesario realizar una consulta previa. El registro de usuario no está cifrado y, por lo tanto, no es necesario especificar un protocolo de consulta aparte a diferencia del protocolo PCS, donde fue necesario especificarlo debido a la posibilidad del cifrado de datos para el registro de servidor. Por esa razón, la consulta de los datos del registro de usuario solo se ilustra como un mensaje dentro del protocolo PMU.
- ✎ La modificación del registro de usuario en BBMChain solo puede ser realizado por el usuario autorizado o el administrador. Por este motivo, es necesario realizar una consulta de los usuarios gestionados (“Listar usuarios bajo gestión”) y seleccionar uno para edición (“Seleccionar usuario”). La consulta utilizará la clave de gestión y el contrato inteligente buscará todos los usuarios que tengan en el campo *_authorizedUser* o *_mainUser* dicha clave almacenada.

- ✎ “Consultar registro de usuario” consultará el registro del usuario en BBMChain con la clave del usuario seleccionado.
- ✎ La modificación del registro seguirá el mismo esquema que la modificación de cualquier otro registro en BBMChain, siguiendo los mismos pasos: modificación del registro, creación de la transacción, firma de la transacción y envío de datos. La firma será realizada con la clave de gestión en el caso de este registro.
- ✎ “Verificar origen y firma” en el contexto del registro de usuario comprobará que la clave de gestión sea la que realiza la modificación, comparando la clave de gestión con el campo ***_authorizedUser*** o ***_mainUser*** y validando que la firma sea correcta.

Figura 68. Diagrama del protocolo PMU.



Fuente: Elaboración propia.

Los aspectos críticos de seguridad de PMU más relevantes son los siguientes:

- △ Al leer el registro no es posible relacionar el gestor con el usuario que realiza la gestión.
- △ No es posible que un tercero no autorizado manipule el registro de seguridad, ya que los datos son enviados firmados con la clave de gestión y BBMChain comprueba que la firma sea correcta.
- △ El usuario gestor no puede negar que es quien realizó la modificación del registro, ya que es el único que posee la clave de gestión.
- △ Se asegura la disponibilidad del registro de usuario, debido a que es almacenado en BBMChain.
- △ El usuario no puede modificar su registro debido a que se requiere la clave privada de gestión para firmar. El gestor puede controlar todos los aspectos de seguridad configurados en el registro del usuario (campo *_userConfig*).
- △ La modificación del campo *_authorizedUser* solo puede ser realizada por el usuario *admin* (*mainUser*) y esto es verificado en el contrato inteligente en “Verificar origen y firma”. En caso de modificar el campo *_authorizedUser*, el usuario *admin* seleccionaría otro usuario gestor de la jerarquía, generaría otra clave de gestión y modificaría el valor de dicho campo. El gestor anterior no podría realizar ninguna acción sobre el usuario.

El campo *_userConfig*, que guarda la configuración de seguridad del usuario, puede ser modificado. Dicho registro tiene una configuración especial. Almacena un valor entero de 40 bytes, lo que permite que puedan almacenarse 40 variables utilizando un esquema de configuración basado en los bits posicionales. Esta configuración permite optimizar el almacenamiento, que suele estar muy restringido en las soluciones de *blockchain*. El valor de registro contiene el bit menos significativo a la derecha, con el índice indicando la posición de la variable. Esto significa que la posición parte del bit menos significativo, a la derecha, al bit más significativo, a la izquierda. Si está establecido a verdadero (true) tiene el valor 1 y la opción está activada, en caso contrario está desactivada y tiene el valor 0. Se han definido inicialmente 19 valores de configuración (Tabla 14). Se verá cómo el registro de configuración es consultado a medida que se va avanzando en la explicación del resto de protocolos.

Tabla 14. Registro de seguridad de usuario.

Índice	Configuración	Descripción de la configuración
0	ContactsAuth	☞ Si está activado, todos los contactos deberán ser autorizados por el gestor.
1	OnlySenderAuth	☞ Cuando está activado, indica que el usuario solo recibirá comunicaciones de contactos autorizados en su cuenta. ☞ En caso de no estar activado podría recibir correo de cualquier usuario, la cuenta estará abierta.
2	MutualAuthorization	☞ Indica si todas las comunicaciones deberían ser autorizadas por los dos usuarios (emisor - receptor). ☞ Asegura que todas las comunicaciones son enviadas y recibidas de cuentas conocidas.
3	CertifiedMail	☞ Permite activar el acuse de recibo.
4	DeliveryNotification	☞ Indica si se activan las notificaciones de entrega. ☞ Activado por defecto si está activado CertifiedMail.
5	DeleteNotificacion	☞ Indica si se activan las notificaciones de borrado.
6	ReadNotification	☞ Indica si se activan las notificaciones de lectura.
7	DeleteAuth	☞ Indica si está autorizado el borrado de correo.
8	RequireCertificate	☞ Indica si los contactos requieren un certificado para ser autorizados.
9	OnlyReceiverAuth	☞ Activado se bloquean todos los envíos no autorizados por el destinatario. Desactivado, el usuario podrá enviar correo a cualquier usuario, por ejemplo, a cuentas abiertas.
10	LetSend	☞ Indica si se puede enviar correo.
11	LetDelete	☞ Indica si se puede borrar correo.
12	LetReSend	☞ Indica si se puede reenviar correo.
13	LetBlock	☞ Indica si se puede bloquear a otro usuario.
14	RequiredAuthforAll	☞ Indica si todas las acciones requieren autorización por parte del gestor.
15	ReqSendAuth	☞ Indica si se requiere autorización para enviar correo.
16	ReqDelAuth	☞ Indica si se requiere autorización para borrar correo.
17	ReqBlockAut	☞ Indica si se requiere autorización para bloquear correo.
18	ReqReSendAuth	☞ Indica si se requiere autorización para reenviar correo.

Fuente: Elaboración propia.

4.7. Protocolo de Creación de Cuentas (PCC)

Las cuentas son las entidades más importantes de BBM ya que son el origen y destino de los mensajes o correos electrónicos. El proceso de creación de los identificadores para las cuentas sigue el mismo esquema de la creación de los identificadores para los usuarios, excepto que con las cuentas se utiliza la clave privada del usuario para generar el identificador de la cuenta (Figura 62), según se ha explicado anteriormente bajo el siguiente esquema:

Nodo Maestro (m) → Identificadores (m/i) → Cuentas (m/i/z) → Buzones (m/i/z/k).

Un usuario puede tener muchas cuentas asociadas. Para preservar la confidencialidad dentro de BBMChain y ocultar la relación entre el usuario y sus cuentas, el uso de la estructura jerárquica de claves para las cuentas y las claves de gestión es lo apropiado. Con este esquema, no es posible consultar BBMChain y listar todas las cuentas de un usuario sino se está en posesión de la clave privada del usuario propietario de la cuenta para generar la clave privada de gestión.

El registro de una cuenta en BBMChain requiere de los siguientes valores:

- 📄 **_id:** El identificador único de la cuenta en BBM. Es la clave pública (M/i/z) generada a partir del usuario (m/i).
- 📄 **_name:** Es un alias de la clave pública (M/i/z). Es una cadena de caracteres que permite de forma más amigable identificar la cuenta.
- 📄 **_ucf:** Es el registro de configuración de la cuenta y su valor es heredado del registro de configuración del usuario.
- 📄 **_owner:** Es el usuario propietario de la cuenta.
- 📄 **_accountState:** Indica el estado de la cuenta y su valor es modificado de acuerdo con la ejecución de la máquina de estados en BBMChain. Los estados utilizados son: activado, desactivado, bloqueado, borrado y suspendido. Cada uno de estos estados se relacionan con la disponibilidad de la cuenta y las acciones que se podrán ejecutar con ella.
- 📄 **_mainUser:** Aplicable en el caso de los usuarios corporativos, indica el usuario *admin* de la corporación.

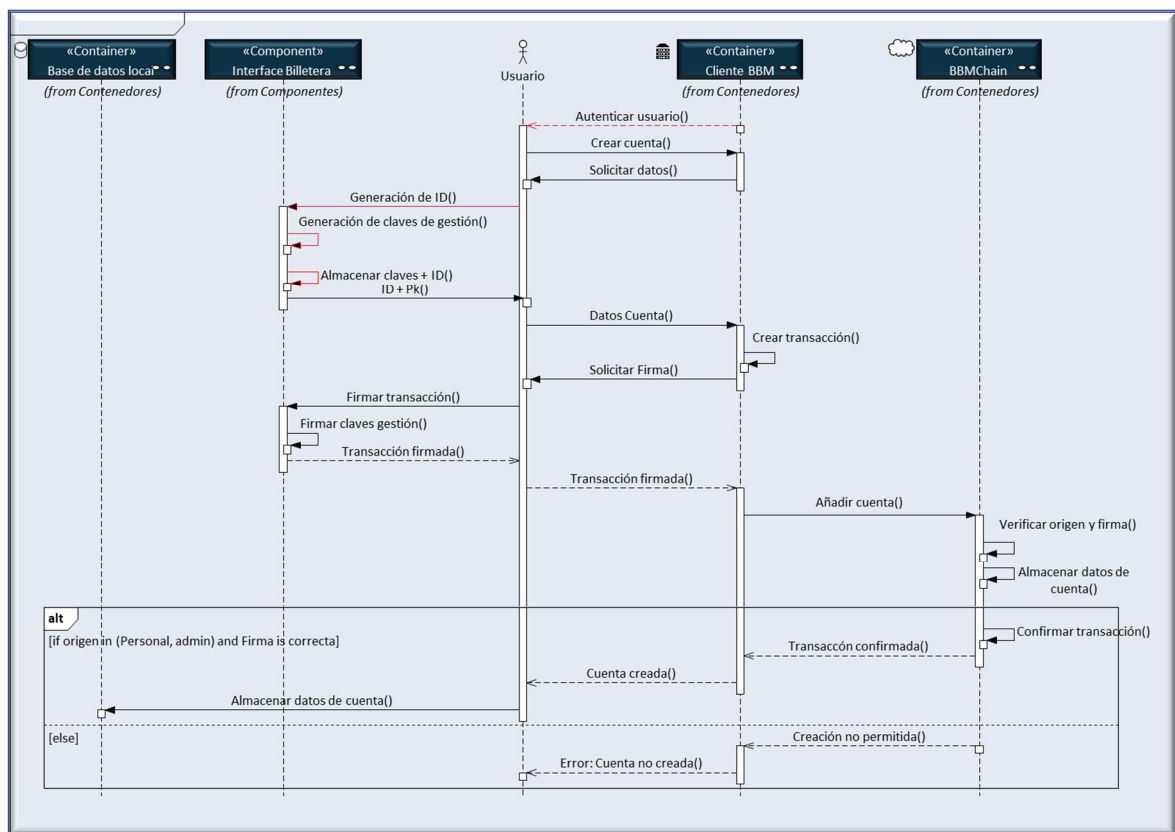
El campo **_owner** debe ser tratado como el campo **_authorizedUser** del registro de usuario y el campo **_mainUser** de la misma forma que en el registro de usuario. Debe almacenar la clave de gestión de la cuenta, que es generada con el algoritmo AGCG, pero la entrada es la clave privada (m/i/z) de la cuenta a gestionar en lugar de la clave privada del usuario.

La clave de gestión permite manipular el registro **_ucf** de la misma forma que se hace con el registro de seguridad **_userConfig**. En principio la cuenta debería tener una copia del registro **_userConfig** y este debería ser modificado solamente por el gestor del usuario. Si el usuario está en posesión de la clave de gestión de la cuenta, podrá modificarlo. Cuando se crea el registro de la cuenta, los valores de los campos **_ucf** y **_mainUser** deben ser proporcionados por el gestor que está creando la cuenta.

El campo **_owner** solo podrá ser modificado por el usuario **admin** en el caso de los usuarios corporativos. Con los usuarios personales solamente un usuario tiene control sobre todas sus cuentas. Es imprescindible que la creación de cuentas la realice el usuario **admin** y no otro usuario. Si el usuario **admin** entrega las claves de gestión a un gestor mal intencionado antes de crear las cuentas, el gestor podría almacenar en **owner** una clave diferente a la de **admin** y se perdería el control de la cuenta. La Figura 69 muestra un diagrama de secuencia del Protocolo de Creación de Cuentas (PCC). Se observa que se sigue el mismo patrón que con la creación de los usuarios. Los aspectos críticos de seguridad más destacados son los siguientes:

- △ Al leer el registro no es posible relacionar a qué usuario pertenece la cuenta.
- △ No es posible que un tercero no autorizado manipule el registro de seguridad, ya que los datos son enviados firmados con la clave de gestión y BBMChain comprueba que la firma sea correcta.
- △ El usuario gestor no puede negar que es quien realizó la creación del registro, ya que es el único que posee la clave de gestión.
- △ Se asegura la disponibilidad del registro de la cuenta, debido a que es almacenado en BBMChain.
- △ Las claves de la cuenta se utilizan para la gestión de correo, pero no para la gestión administrativa, que corre a cargo del poseedor de la clave de gestión. El gestor puede controlar todos los aspectos de seguridad configurados en el registro de la cuenta (campo **_ucf**).

Figura 69. Diagrama del protocolo PCC.



Fuente: Elaboración propia.

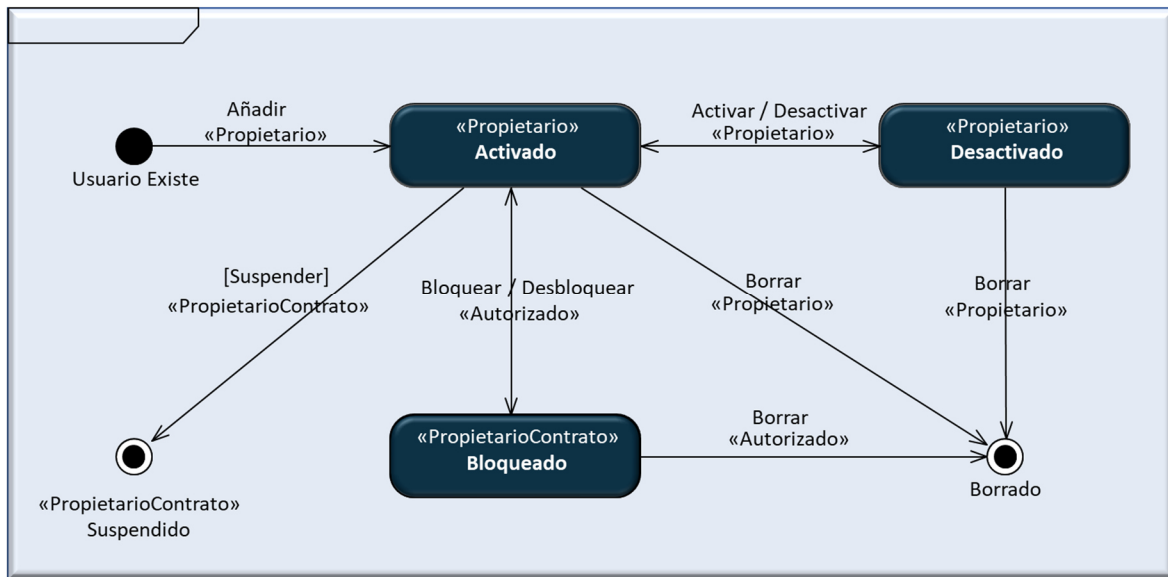
4.8. Protocolo de Modificación de Cuentas (PMC)

La modificación de la cuenta sigue el mismo patrón que la modificación de los usuarios, ya que en ambos casos se utilizan las cuentas de gestión. En este punto es importante tener en cuenta que quien modifica la cuenta es un gestor y debe estar en posesión de las claves de gestión. Si el administrador ha modificado al gestor del usuario propietario de las cuentas, también deberá modificar las claves de gestión de cuentas y pasarlas al nuevo gestor.

La razón de estas restricciones de gestión se debe principalmente a que las cuentas son las originarias y destinatarias de los correos y, por lo tanto, deben tener un registro de seguridad asociado. Se ha visto que el campo *_cfg* del registro de la cuenta se basa en el campo *_userConfig* del registro del usuario, pero los servidores no pueden consultar los registros de seguridad de los usuarios debido a que desconocen a qué usuarios pertenecen las cuentas. Por motivos de confidencialidad, ha sido necesario que el registro de seguridad se almacene con el registro de la cuenta y que solamente los gestores puedan modificarlo.

Las cuentas se comportan como una máquina de estados en BBMChain, donde las transiciones de un estado a otro requieren la validación de una serie de prerequisites. Este tipo de comportamiento ya se ha comentado con los usuarios y las cuentas. En el registro de la cuenta el campo `_accountState` almacena el estado de la cuenta. Las transiciones de estado son reguladas por la máquina de estados de la cuenta (Figura 70) que está almacenada en BBMChain.

Figura 70. Diagrama de máquina de estados de la cuenta.



Fuente: Elaboración propia.

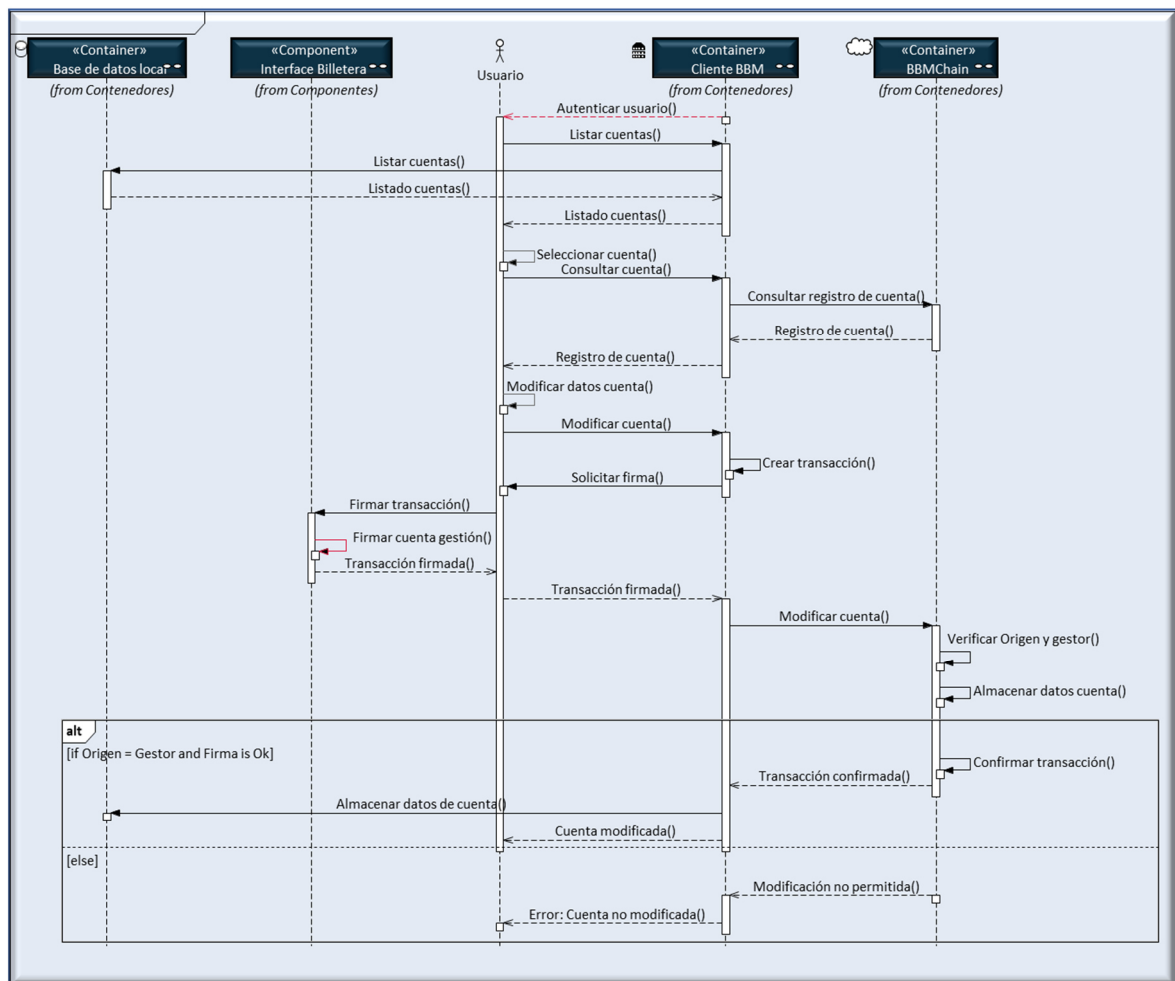
Los estados y usuarios involucrados en la máquina de estados de la cuenta siguen la filosofía de la máquina de estados del usuario. Con respecto a la máquina de estados de la cuenta hay que tener presente las siguientes cuestiones:

- ❄ Todos los estados de las cuentas deberían estar sincronizados con los estados de los usuarios y esa responsabilidad recae sobre el cliente BBM que debería realizar dicha validación. Cuando un gestor modifica el estado de un usuario, esa modificación tiene que replicarse en todas las cuentas, pero no al contrario. Los cambios de estado de una cuenta no afectan el estado del usuario.
- ❄ Los protocolos de seguridad relacionados con el correo consultarán el estado de la máquina de las cuentas y nunca la de los usuarios.
- ❄ La sincronización entre las dos máquinas (cuentas y usuarios) es una tarea administrativa que no afecta la ejecución de las máquinas, sino que puede retrasar un cambio de estado si falla la sincronización.

- ✿ La interpretación de los estados en las cuentas sigue la misma interpretación que los estados de usuarios.

El Protocolo de Modificación de Cuentas (PMC) sigue el esquema de modificación de los registros en BBMChain cuando se utilizan cuentas de gestión como en el caso de los usuarios. La Figura 71 muestra el diagrama de secuencia de este protocolo.

Figura 71. Diagrama del protocolo PMC.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PMC son los siguientes:

- ✿ “Listar cuentas” aparece como una petición a la “Base de datos local”. Aunque en otros diagramas este tipo de consulta no se ha reflejado, en la práctica, los clientes BBM deben consultar su base de datos, ya que ésta se sincroniza en el arranque del cliente.
- ✿ “Firmar cuenta de gestión” indica como en otros casos que la modificación será realizada con la clave de gestión, exceptuando el caso donde se modifica el campo **_owner**, que requiere la clave del usuario **mainUser**.

👉 “Verificar origen y firma” comprobará que la clave utilizada se corresponda con la clave autorizada para realizar las modificaciones. También lanzará la máquina de estados para comprobar los prerrequisitos si hay una solicitud de cambio de estado de la cuenta.

En cuanto a los aspectos críticos de seguridad de PMC, se resalta los siguientes:

- △ No es posible que un tercero no autorizado manipule el registro de seguridad, ya que los datos son enviados firmados con la clave de gestión y BBMChain comprueba que la firma sea correcta. Se asegura la integridad del registro de cuenta.
- △ El usuario gestor no puede negar que es quien realizó la modificación del registro, ya que es el único que posee la clave de gestión.
- △ La modificación del campo **_owner** solo puede ser realizada por el usuario **admin (mainUser)** y esto se comprueba en BBMChain en “Verificar origen y gestor”. Para modificar el campo **_owner**, el usuario **admin** deberá generar otra clave de gestión a partir de la clave de otro usuario delegado y modificar el campo. El gestor anterior no podría realizar ninguna acción sobre la cuenta.

4.9. Protocolo de Creación de Buzones (PCB)

Una cuenta representa un emisor o receptor de correo. Un correo enviado a una cuenta es enviado internamente en BBM a un buzón. El buzón representa la localización lógica de un servidor. Para que una cuenta pueda ser utilizada para gestionar correo es necesario crear y asociar un buzón a ella. La información del buzón es almacenada en un registro de buzón en BBMChain. Una cuenta puede tener más de un buzón para garantizar la disponibilidad y distribuir las comunicaciones de correo a destinos diferentes.

Como se ha visto anteriormente con los usuarios y las cuentas, el proceso de creación de los identificadores para los buzones sigue el mismo esquema de generación de claves (Figura 62), excepto que para generar el identificador de los buzones se utiliza la clave privada de la cuenta. El esquema siguiente es el utilizado para los buzones:

Nodo Maestro (m) → Identificadores (m/i) → Cuentas (m/i/z) → Buzones (m/i/z/k).

El uso de la estructura jerárquica de claves para los buzones y las claves de gestión, permiten ocultar la relación y el control entre las cuentas y sus buzones. En el caso de los buzones, puede no ser necesario ocultar dicha relación y tampoco es necesario el uso de las cuentas de gestión. Se deja a consideración de la implementación específica de BBM o a las necesidades de ésta el uso de cuentas de gestión a este nivel.

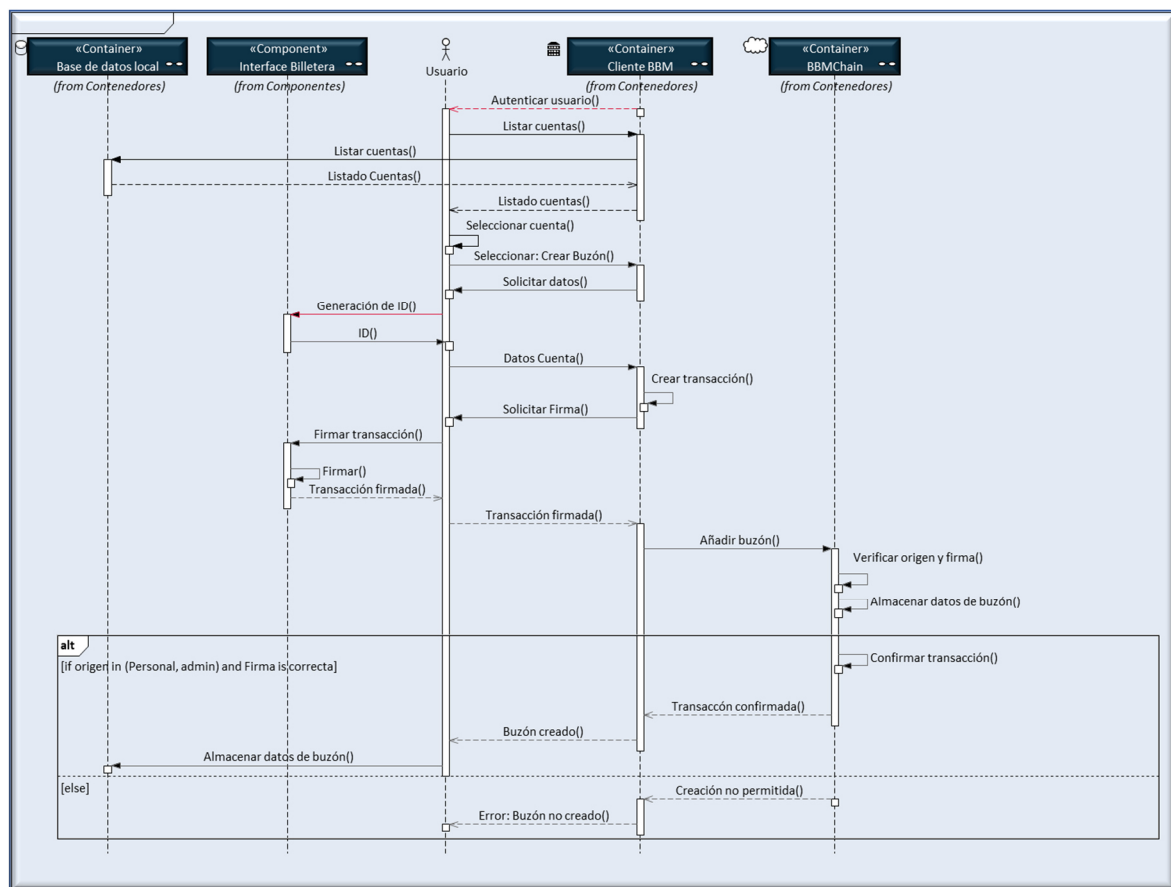
Las cuentas de gestión permiten ocultar la relación entre dos estructuras fuertemente acopladas que se almacenan en una *blockchain* pública, pero el uso con los buzones no está recomendado debido a que el objetivo es la localización del correo y es necesario que esta información esté disponible para que los servidores localicen los buzones de las cuentas.

Los campos necesarios para la creación de un buzón almacenado en BBMChain son:

-  **_id**: El identificador único del buzón en BBM. Es la clave pública (M/i/z/k).
-  **_name**: Es un alias de la clave pública (M/i/z/k). Es una cadena de caracteres que permite de forma más amigable identificar al buzón.
-  **_bcf**: El registro de configuración y seguridad del buzón. En esta primera versión de los protocolos se ha utilizado, se deja para futuras actualizaciones.
-  **_BRPrivateLocalization**: Indica la localización física del buzón, que es el servidor donde se depositarán los correos. Este campo lo modificará el servidor almacenando los datos de localización de forma cifrada para la cuenta registrada.
-  **_accountId**: Almacena el Id de la cuenta asociada al buzón.
-  **_MailboxState**: Indica el estado del buzón y su valor es modificado por el gestor de la cuenta propietaria.
-  **_ServerSignature**: Almacena la firma del servidor de los datos del identificador del buzón (*id*) y el identificador de la cuenta (*accountId*). Sirve para comprobar que el servidor acepta gestionar correos en este buzón para la cuenta específica.

La creación del buzón está estrechamente relacionada con el registro de la cuenta en el servidor. Este registro permite asociar el servidor en el campo **_BRPrivateLocalization** y esto es realizado haciendo uso del Protocolo de Registro de Cuenta en el Servidor (PRCS). La Figura 72 muestra un diagrama de secuencia del Protocolo de Creación de Buzones (PCB).

Figura 72. Diagrama del protocolo PCB.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PCB son los siguientes:

- 👉 Se puede ver en la figura anterior que la creación sigue el esquema utilizado para la modificación de un registro en BBMChain.
- 👉 La creación del buzón se realiza con la clave de gestión de la cuenta.
- 👉 “Listado cuentas” muestra las cuentas asociadas al usuario autenticado. Esa información se encuentra almacenada en la Base de datos local.
- 👉 “Seleccionar cuenta” indica la selección de una de las cuentas listadas.
- 👉 “Crear buzón” indica al sistema que solicite los datos para la creación del buzón en la cuenta seleccionada.
- 👉 “Firmar” deberá firmar la transacción con la clave privada de gestión de la cuenta. El gestor tiene la clave privada de esa cuenta en su billetera.
- 👉 “Añadir buzón” envía el registro creado y firmado a BBMChain.

- ✎ “Verificar origen y forma” valida que la firma sea realizada con la clave de gestión de la cuenta, es decir la clave almacenada en `_owner`.
- ✎ Al validarse el registro, “Almacenar datos de buzón” almacena el registro en BBMChain y queda a la espera de confirmación por parte de los nodos validadores.
- ✎ El buzón no podrá ser utilizado hasta que se ha realizado el registro del buzón en un servidor disponible en BBM, que es cuando se asocia la localización física.

En lo relativo a los aspectos críticos de seguridad de PCB, destaca lo siguiente:

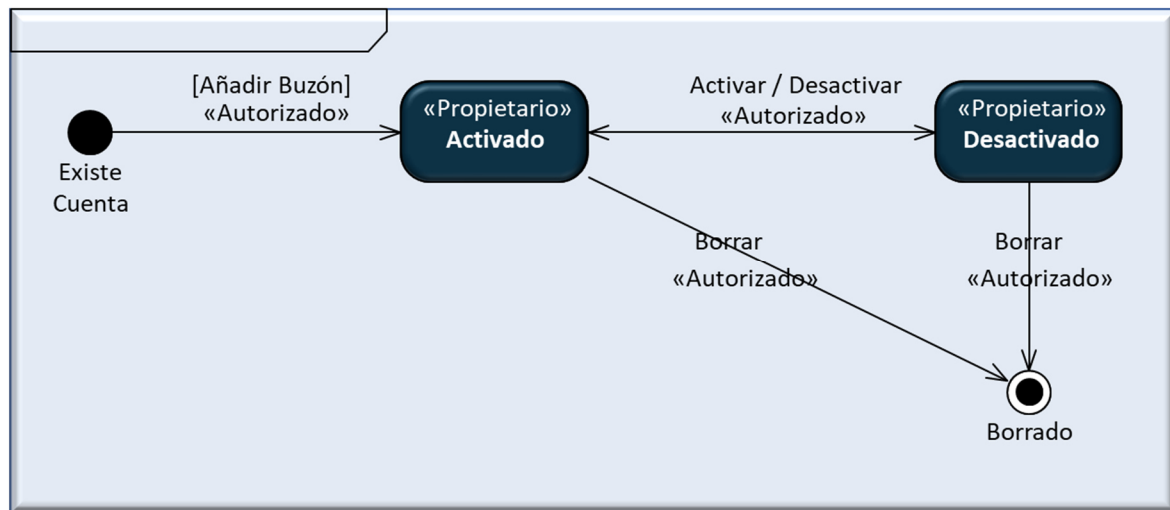
- ⚠ Accediendo al registro del buzón no es posible saber la localización física del buzón. Esa información solo la conoce el poseedor de la clave privada del buzón.
- ⚠ No es posible que un tercero no autorizado manipule el registro del buzón cuando los datos están siendo transportados y almacenados, ya que estos son enviados firmados con la clave de gestión y BBMChain comprueba que la firma sea correcta. Se asegura la integridad del registro del buzón.
- ⚠ Se garantiza la disponibilidad del registro del buzón gracias a BBMChain.
- ⚠ El gestor poseedor de la clave de gestión no puede negar la creación del buzón.

4.10. Protocolo de Modificación de Buzón (PMB)

La modificación del buzón sigue el patrón que se ha estado utilizando para modificar los registros almacenados en BBM. En el caso específico de los buzones, uno de los campos de interés a modificar por parte del gestor es el estado del buzón (`_MailboxState`), que sigue una máquina de estados cuyo comportamiento es controlado por BBMChain. La Figura 73 muestra el diagrama de estados asociado a los buzones de las cuentas, en el que hay que tener en cuenta:

- ⚙ Los buzones solamente pueden estar en tres estados: activado, desactivado y borrado.
- ⚙ El gestor de la cuenta es el único autorizado para cambiar el estado del buzón.
- ⚙ El estado “borrado” es un estado no reversible, deja al buzón inutilizado.
- ⚙ Cuando un buzón se encuentra “desactivado”, no puede recibir correo.
- ⚙ Los estados “bloqueado” y “suspendido” de la cuenta dejan al buzón en estado “desactivado”. Como hay visibilidad entre la cuenta y su buzón, este proceso se puede ejecutar directamente en BBMChain.
- ⚙ El estado “borrado” de la cuenta deja al buzón en estado “borrado”.

Figura 73. Diagrama de estados del buzón.



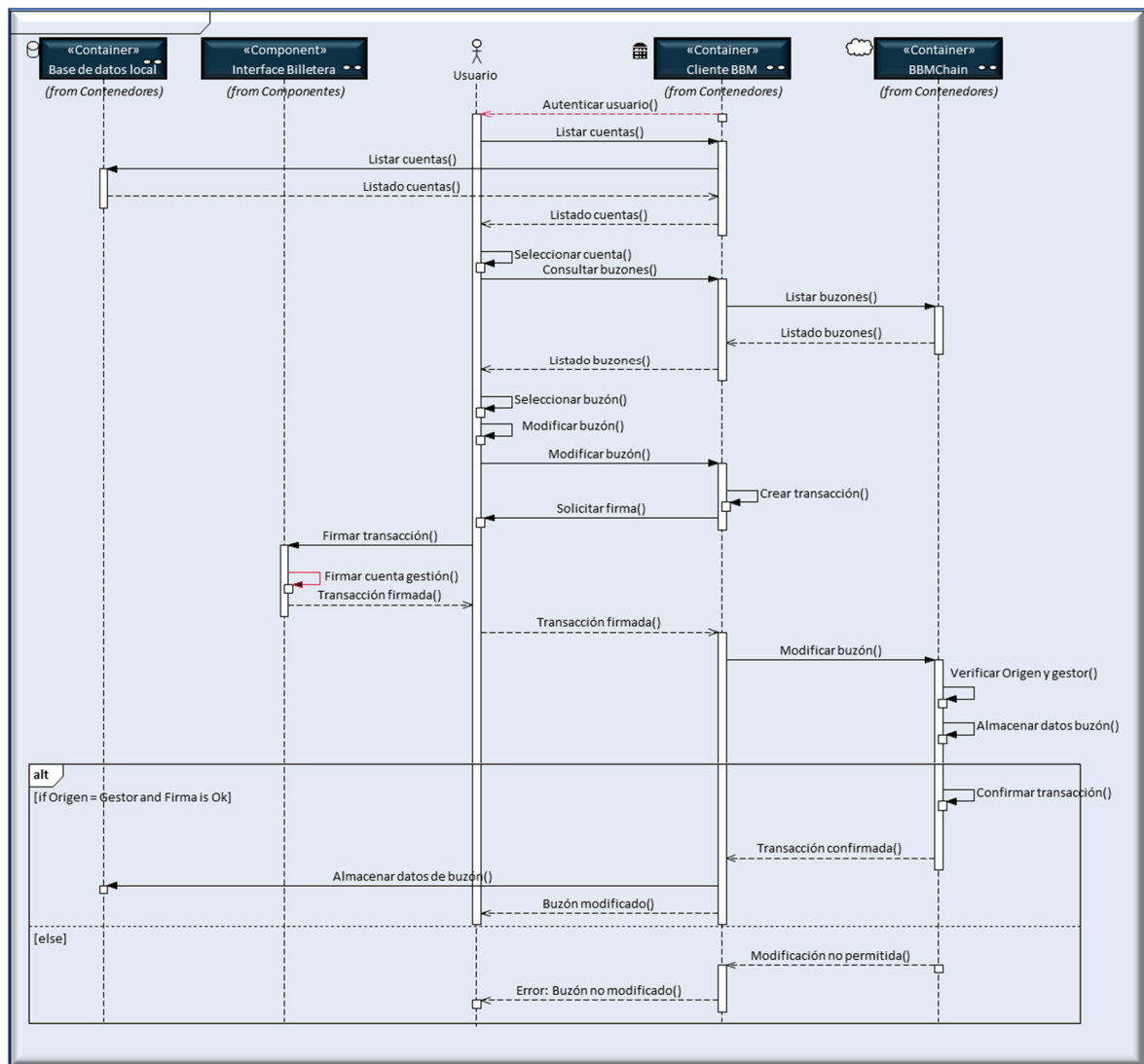
Fuente: Elaboración propia.

Los campos *_BRPrivateLocalization* y *_ServerSignature* pueden ser modificados por el usuario gestor, pero no es aconsejable, excepto cuando se desee desligar la cuenta de un servidor por algún motivo de fuerza mayor. Estos dos campos deberían ser solamente modificados por un servidor utilizando los protocolos de BBM propuestos para dicho fin.

La Figura 74 muestra el diagrama de secuencia de los diferentes mensajes que se intercambian los actores durante la ejecución del Protocolo de Modificación de Buzón (PMB). Los aspectos para resaltar son los siguientes:

- ✎ La autenticación se realiza con la clave de gestión de la cuenta.
- ✎ “Seleccionar cuenta” selecciona una de las cuentas previamente consultadas.
- ✎ “Consultar buzones” envía una consulta a BBMChain para obtener los buzones creados para la cuenta seleccionada.
- ✎ “Modificar buzón” envía los datos del registro del buzón modificados y firmados con la clave de gestión de la cuenta.
- ✎ “Verificar origen y gestor” realiza las validaciones respectivas de la firma y de la clave de gestión utilizada durante la creación de la transacción. También se lanza la máquina de estados del buzón para validar las precondiciones en caso de alguna modificación de estado.

Figura 74. Diagrama del protocolo PMB.



Fuente: Elaboración propia.

4.11. Protocolo de Registro de Cuenta en el Servidor (PRCS)

La modificación del campo *_BRPrivateLocalization* del registro del buzón requiere un protocolo especial aparte para su modificación. Una vez el usuario ha creado una cuenta y le ha asociado un buzón, es necesario registrar la cuenta en un servidor para que todos los correos sean almacenados en su localización. El registro implica el envío de una solicitud al servidor que es almacenada en BBMChain.

Los datos solicitados para ese registro de la solicitud son:

📄 **_MailBoxid:** Identificador del buzón. Es la clave pública del buzón que realiza la petición del registro.

📄 **_ST:** El estado de la solicitud. Puede tener los valores “solicitado”, “rechazado” y “aceptado”. Cuando se envía el registro el valor de este campo se inicializa a “solicitado”.

📄 **_ServerPk:** Clave pública del servidor al que se le realiza la solicitud.

📄 **_ServerSignature:** Firma del servidor de los datos del identificador del buzón (*_MailBoxid*), la clave pública del servidor (*_ServerPk*) y el estado de la solicitud (*_ST*).

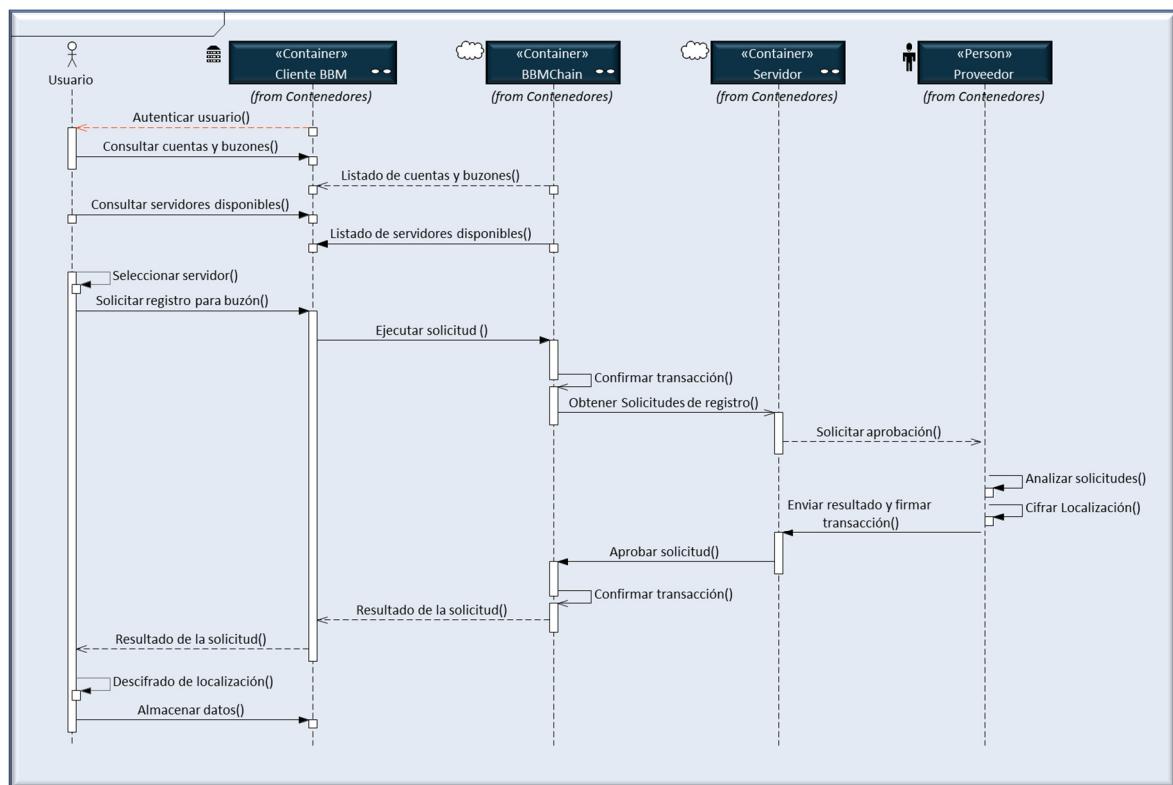
Dicha confirmación es realizada por el proveedor propietario del servidor firmando la solicitud y almacenando la firma en el campo **_ServerSignature** de la solicitud. En la confirmación se modifica el registro del buzón con los datos del servidor en el campo **_BRPrivateLocalization** y la firma del campo **_ServerSignature** del registro del buzón. Los datos del servidor se almacenarán cifrados para que la cuenta registrada sea la única que pueda interpretarlos. Este campo puede ser modificado por el servidor en cualquier momento y las firmas deben renovarse cuando se realice algún cambio.

Si los datos del servidor no son públicos y están cifrados, los usuarios no tendrán acceso a ellos hasta que se confirma el registro. Los datos son almacenados en el mismo registro de forma cifrada para que el usuario pueda descifrarlos y desde el cliente se puedan realizar las peticiones al servidor. Los servidores para intercambiar correo deben comunicarse entre ellos y es esencial que conozcan sus localizaciones.

Cuando un servidor se comunica por primera vez con otro servidor para recibir correo, le debe enviar una solicitud de contacto para intercambiar sus respectivas localizaciones por medio del Protocolo de Intercambio de Localización entre Servidores (PILS).

El protocolo PRCS modifica el registro del buzón añadiendo la localización y la firma de confirmación por parte del servidor cuando el proveedor acepte el registro. Este proceso es totalmente asíncrono. La Figura 75 muestra un diagrama donde se pueden ver los pasos que debe ejecutar un usuario para registrar una cuenta en un servidor y el proceso de confirmación realizado para parte del proveedor. Con el objetivo de simplificar el diagrama, se han eliminado algunos contenedores (Interface billetera, la Base de datos local y BBMStorage) y mensajes de intercambio realizados con ellos.

Figura 75. Diagrama del protocolo PRCS.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PRCS son los siguientes:

- ✎ La autenticación se realiza con la clave de gestión de la cuenta.
- ✎ “Listado de cuentas y buzones” es un conjunto de mensajes y procesos simplificados que tiene como objetivo la selección de un buzón para registrar.
- ✎ “Listado de servidores disponibles” es una consulta que se realiza a los registros de los servidores almacenados en BBMChain.
- ✎ “Ejecutar solicitud” envía la solicitud de registro cumplimentada y firmada con la clave de gestión.
- ✎ “Confirmar transacción” es un proceso simplificado en el que se realizan las validaciones del registro. Se debe verificar que la clave utilizada en la firma del registro del buzón se corresponda con la clave de gestión de la cuenta (campo **_owner** del registro de la cuenta **_accountId**) propietaria del buzón (**MailBoxId**).
- ✎ “Solicitar Aprobación” es un proceso asíncrono que se ha utilizado aquí para representar la espera de la aprobación de la solicitud. El servidor del proveedor debe de estar activo

esperando cualquier solicitud para comunicarla al proveedor. No se representa aquí el proceso de aprobación “detrás” de BBM que dependerá de cada implementación.

- ✎ “Analizar solicitudes” será el proceso que leerá la solicitud y determinará por un medio externo a BBM dicha aprobación.
- ✎ “Cifrar Localización” utilizará el algoritmo CDC con la clave pública del buzón como entrada para que solo el poseedor de la clave privada del buzón pueda interpretarla.
- ✎ “Enviar resultado y firmar transacción” firmará el registro con la clave privada del servidor y enviará a BBMChain la confirmación.
- ✎ “Confirmar transacción” es una función del contrato inteligente que actualizará los campos *_BRPrivateLocalization* y *_ServerSignature* del registro del buzón y el campo *_ServerSignature* del registro de la solicitud.
- ✎ “Resultado de la solicitud” es un proceso asíncrono de la notificación que el cliente BBM recibirá cuando el registro de la solicitud y el registro del buzón se actualicen.

En cuanto a los aspectos críticos de seguridad del protocolo PRCS, se comenta lo siguiente:

- Δ La solicitud de registro sólo puede ser realizada por el poseedor de la clave de gestión de la cuenta del buzón *MailBoxid* que aparece en el registro. No es posible registrar en un servidor un buzón que no sea de propiedad del registrador.
- Δ No es posible suplantar la identidad del servidor sin tener en posesión la clave privada del servidor.
- Δ No es posible modificar en el registro la localización del servidor sin estar en posesión de la clave privada del servidor. La integridad del registro de la solicitud y el registro del buzón están aseguradas.
- Δ La localización del buzón no podrá ser conocida accediendo al registro de la solicitud. Solo al poseedor de la clave privada del buzón será revelada dicha información.

4.12. Protocolo de Solicitud y Autorización de Contacto (PSAC)

En BBM, la comunicación segura entre usuarios se basa en la creación de contactos autorizada entre cuentas. Los contactos son solicitados y almacenados en BBMChain. Un contacto, como se describió inicialmente en este capítulo, es una relación entre dos cuentas: una originadora, que es quien solicita el contacto, y otra destinataria, que es la que autoriza al originador a que le envíe correo, pero no a la inversa.

Si se hace uso de una *blockchain* privada y se decide que los contactos pueden ser consultados por cualquier usuario, solo es necesario crear la estructura de contactos y almacenarla en texto plano en *blockchain*. Como este comportamiento no es el deseado en la mayoría de los casos, en BBM se propone una solución a este problema. Lograr la confidencialidad en los contactos creados en BBMChain y evitar que puedan ser reveladas las relaciones entre las cuentas, es el objetivo del Protocolo de Solicitud y Autorización de Contacto (PSAC).

Utilizando el mismo mecanismo de jerarquía de cuentas que en BBMChain se trata un contacto como una entidad más que tiene asociada un par de claves. Se propone el uso de jerarquía de claves añadiendo una profundidad más. El esquema utilizado para generar las claves es la siguiente:

Nodo Maestro (m) → Identificadores (m/i) → Cuentas (m/i/z) → Buzones (m/i/z/k) →
Contactos (m/i/z/0/x)

Las claves se generan a partir de la clave del primer buzón (buzón principal con $k=0$) y por cada nuevo contacto es necesario que se genere una nueva clave. En caso de necesitar más contactos (que es poco probable) se pasaría al siguiente buzón como clave de generación. El siguiente paso es definir la creación de las claves de gestión de los contactos, que debe seguir el algoritmo de generación de claves de gestión, pero se utilizará la clave privada $S(m/i/z/0/x)$ del contacto de la cuenta a gestionar.

BBM permite flexibilidad con respecto a la gestión de contactos. La generación de la clave de gestión del contacto utiliza por defecto la clave de gestión de la cuenta, lo que significa que solamente el gestor puede crear los contactos de una cuenta determinada. Si se utiliza la clave de gestión del usuario, solamente el usuario **admin** podrá crear los contactos. Este esquema permite un gran control por parte de los administradores de un sistema en un entorno corporativo.

Si se desea permitir que los usuarios puedan crear sus propios contactos, entonces bastará con que el esquema utilizado para generar la clave de gestión del contacto parta de la clave privada del usuario. Todo esto va a depender del esquema de políticas de seguridad utilizado en un entorno corporativo. El proceso de generación de claves en su totalidad debe estar oculto para el usuario y la billetera es la encargada de que sea transparente para el usuario.

La clave de gestión del contacto será la clave utilizada para la creación del contacto. A continuación, se definirá el contenido de la solicitud de contacto y cómo debe ser configurada para evitar que, una vez realizada la solicitud, sea imposible que el resto de los usuarios sepan la cuenta originadora del contacto. La cuenta destinataria debe ser informada de la solicitud,

por lo tanto, uno de los datos que contiene el registro es la clave pública de la cuenta destinataria de la solicitud y ésta es la única que aparece legible para que el destinatario sea informado del contacto.

Para la creación del contacto, es necesario proporcionar a BBMChain los siguientes datos:

-  **_Cid:** Es la clave pública del contacto (M/i/z/0/x) generada.
-  **_RCO:** Es la clave pública firmada de la cuenta del originador. Por motivos de confidencialidad, este campo debe permanecer cifrado en BBMChain. Los datos cifrados van dirigidos al destinatario.
-  **_RCD:** Es la clave pública de la cuenta del destinatario. Aparece en texto plano para que pueda ser consultado fácilmente.
-  **_TS:** Hace referencia a una marca de tiempo (*TimeStamp*) que indica el momento exacto (fecha y hora) en el que se ha solicitado dicho contacto. La transacción creada en BBMChain contiene esta información.
-  **_Exp:** Este valor hace referencia a una fecha de expiración que puede ser asignada y modificada solo por el destinatario. Indica la fecha a partir de la cual el contacto cambia y no es válido. Una vez expirado el contacto, desde la cuenta originadora no se podrán enviar ningún correo al destinatario.
-  **_St:** Almacena el estado del contacto y determina el tipo de comunicación que puede ser realizada entre las dos cuentas.
-  **_TA:** Es el *token* de autorización. Almacena el *hash* de la firma del *token* (*RCO*, *_RCD*, *_EXP*) realizada por el destinatario: $Hash(Firma(Token))$ y **Token** = *RC* + *Exp* + *RCD*.
-  **_AD:** Son los datos adicionales que pueden ser requeridos por el destinatario para la aceptación del contacto. Estos datos deben estar cifrados para el destinatario. En caso de requerir un certificado, puede almacenar la referencia al mismo (por ejemplo, con IPFS), que debe estar cifrada.
-  **_SRCD:** Es la firma del campo *_TA* realizada por el destinatario que debe ser verificada en BBMChain. Esta firma es la aprobación del contacto: *Firma(_TA)*
-  **_TokenSig:** Es la firma del *token* cifrada. Contiene la firma del *token* cifrada que ha sido resumida en el campo *_TA*: *Cifrar (_RCO, Firma (Token))*.
-  **_Ct** es el tipo de contacto establecido. Puede tener dos valores *Personal* o *Business*, indicando que puede ser un contacto personal o corporativo.

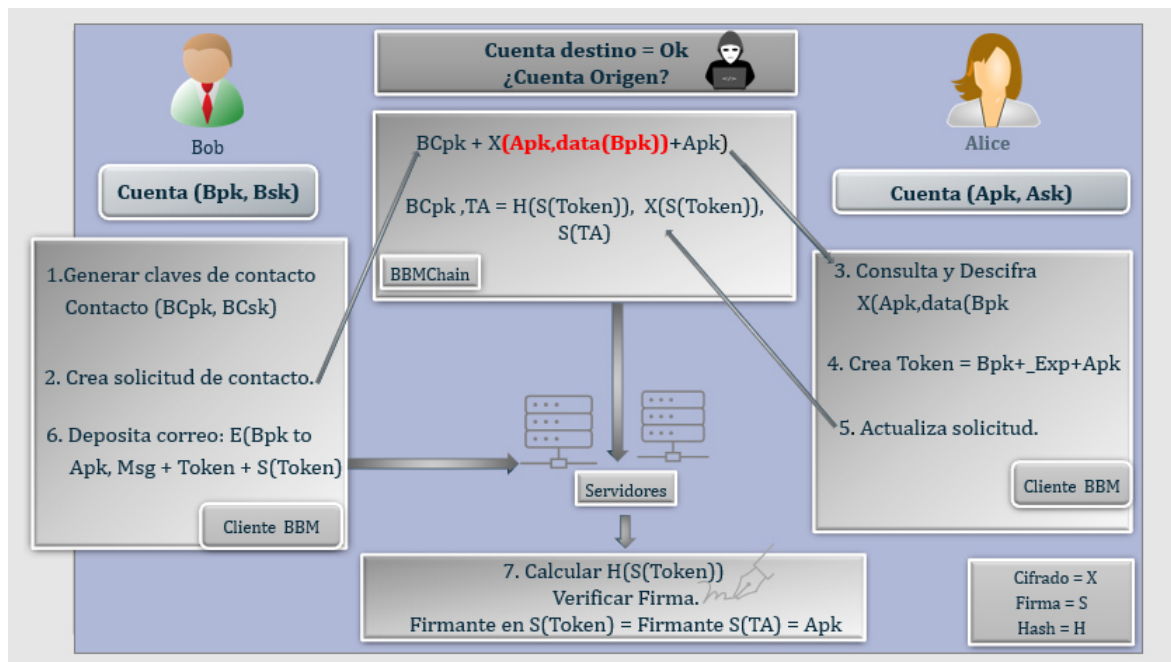
-  **_Rjc:** Es el código de rechazo o aceptación. Si el contacto es rechazado, indica por medio de un código, la razón del rechazo y la posible solicitud de más información que debe ser suministrada por el originador en el campo **_AD**.
-  **_Server:** Es el servidor donde se depositará el correo para el destinatario. Este valor será añadido por la cuenta que solicita el contacto y es la clave pública del servidor asociado a la cuenta.
-  **_owner** y **_mainUser:** Son los dos campos utilizados para el control de seguridad y ya han sido explicados en el protocolo PCC.

Aunque el campo **_RCO** está cifrado y contiene la clave firmada de la cuenta del originador, se mencionará de ahora en adelante a **_RCO** cuando se haga referencia a la clave del originador. Cuando este registro ha sido almacenado, el destinatario puede saber que la solicitud está dirigida a su cuenta, ya que el campo **_RCD** está en texto plano. El originador recibe una estructura denominada **Token** cuyo contenido solo es conocido por el originador y el destinatario. El **Token** hace uso de dos firmas con dos objetivos diferentes: el primer objetivo es suministrar al originador una prueba de su confirmación, el segundo es que la validez del *token* pueda ser verificada a nivel de **BBMChain** sin exponer la confidencialidad del contacto.

_TokenSig es utilizado para evitar que pueda ser deducido el originador a partir de la firma. Si se conoce la firma del **Token**, **_Exp** y **_RCD**, un atacante podría intentar con los datos de las claves públicas de las cuentas comprobar la firma para diferentes combinaciones de *Tokens* hasta que uno de los **Tokens** coincida y revelar el contacto. Si la firma está cifrada, se evita este tipo de ataque.

El originador puede presentar la firma del **Token**, que es recibida en el campo **_TokenSig** y los valores del contacto (**_RCO+RCD+Exp**) a un tercero (por ejemplo, un servidor). Este tercero calcula el *hash* de la firma y verifica si existe un contacto solicitado a **_RCD** comparando ese *hash* con el campo **_TA**. Si coincide y además puede comprobar la firma en **_SRCD**, el contacto de **_RCO** a **_RCD** está autorizado por el destinatario. El firmante de **_SRCD** debe coincidir con el firmante de la firma del **Token** entregada por el originador. La Figura 76 representa de forma esquemática estos conceptos.

Figura 76. Esquema de solicitud de contacto.

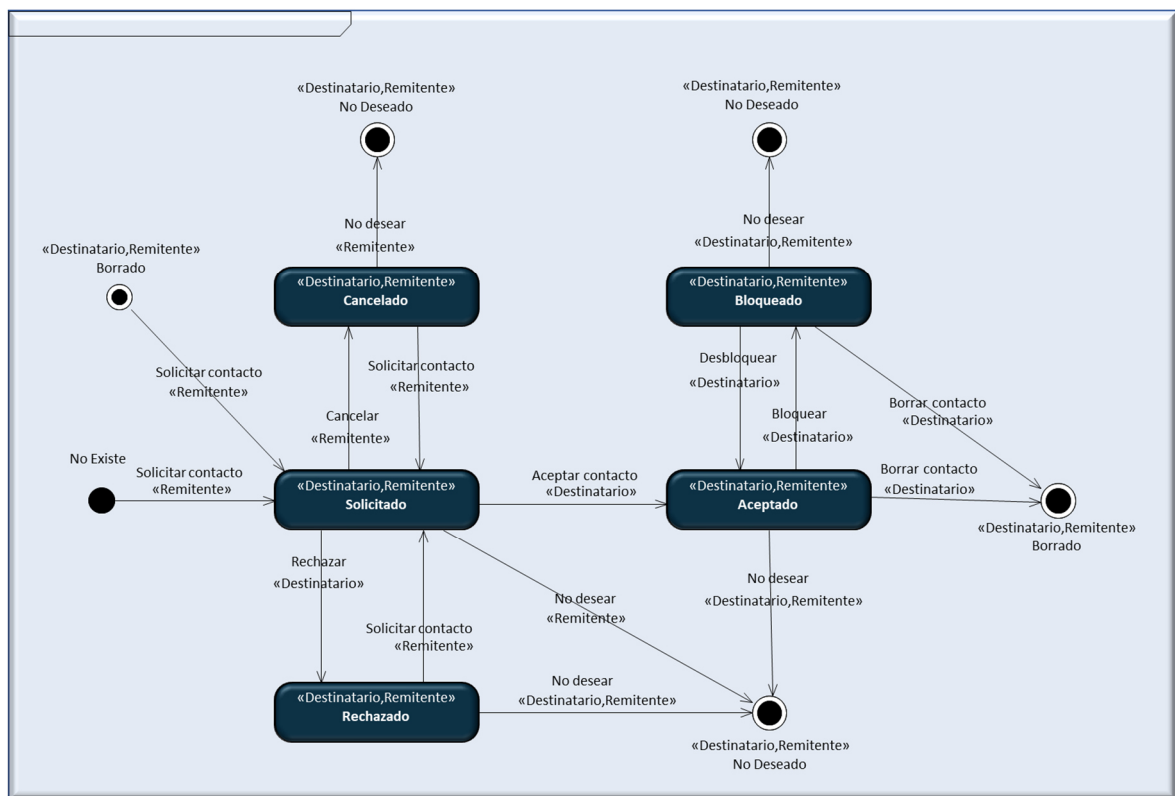


Fuente: Elaboración propia.

En la figura anterior se puede ver como el registro de contacto puede ser leído e interpretado por Alice, pero ningún otro usuario o un atacante podría interpretarlo. Los servidores representan el tercero, que cuando reciben un depósito de correo, pueden verificar si es válido comparando las firmas como se ha explicado en el párrafo anterior.

El campo **_ST** describe los estados de un contacto y, como descrito con otras entidades, el estado de los contactos también es controlado por una máquina de estados almacenada en BBMChain. La Figura 77 muestra un diagrama de máquina de estados donde se pueden ver las transiciones y los eventos o acciones que las activan.

Figura 77. Diagrama de máquina de estados del contacto.



Fuente: Elaboración propia.

Con respecto a la máquina de estados del contacto, destaca:

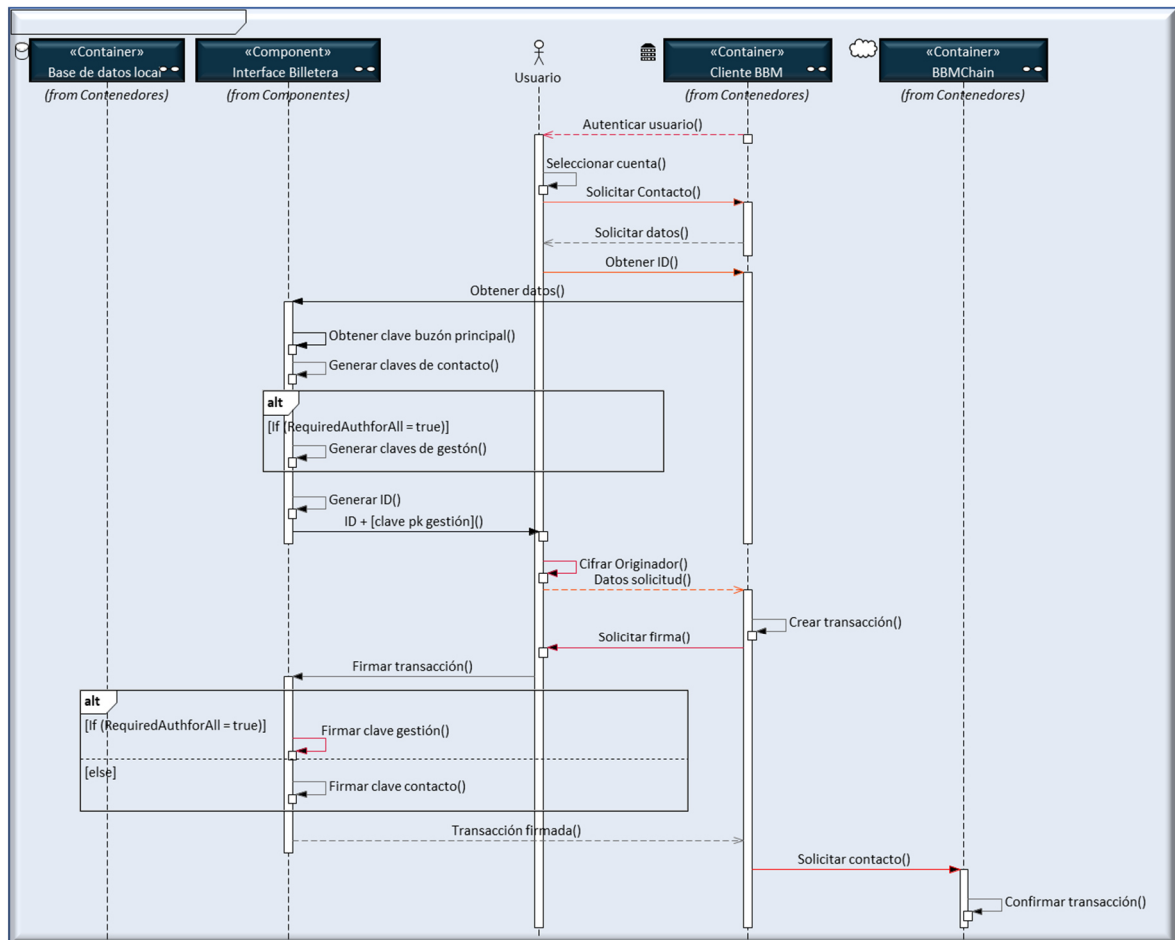
- ❖ Los contactos pueden estar en los siguientes estados: solicitado, aceptado, cancelado, bloqueado, rechazado, borrado y no deseado.
- ❖ Cuando se crea un contacto, el estado inicial es “solicitado”. Se observa que esta transición es iniciada por el originador. En este estado, el contacto es visible para los destinatarios y en este momento se debería verificar para analizar las solicitudes.
- ❖ Un originador podría cancelar el contacto mientras está en estado “solicitado”. El contacto en estado “cancelado” no puede ser manipulado por el destinatario, solamente por el remitente quien podría cancelarlo definitivamente pasando a “no deseado” y podría activarlo de nuevo a “solicitado”.
- ❖ Un contacto solo puede ser “rechazado” si está en estado “solicitado” y ese cambio solo lo puede realizar el destinatario. Es un estado que indica que el destinatario está a la espera de más información. El campo ***R/c*** puede ser consultado por el originador para saber la razón del rechazo y, en caso de necesitar adjuntar más información para el destinatario, el

originador modificaría el campo **_AD**. El remitente podría pasar el contacto a “solicitado” y quedar a la espera del destinatario.

- ✿ El destinatario es el único que podría pasar el contacto de estado “solicitado” a “aceptado”. Un contacto en estado “aceptado” puede ser utilizado como relación de comunicación en los correos.
- ✿ El estado “bloqueado” indica que el destinatario por alguna razón deja al contacto inutilizado. Es un estado de investigación donde el destinatario podría pasarlo a “borrado” o “no deseado”.
- ✿ Un contacto “borrado” es un estado temporal que le da al originador la posibilidad de poder solicitar de nuevo contacto.
- ✿ Se puede observar que un contacto puede pasar a “no deseado” desde casi cualquier estado, excepto desde “borrado”. Este estado indica que el contacto no podrá ser utilizado nunca y habría que crear un nuevo contacto si se desea poder intercambiar correo con dicha cuenta destinataria.

Una vez explicados los elementos que sirven de base para el protocolo PSAC, se puede describir el protocolo a nivel de componentes para comprender cómo debería ser implementado. El protocolo parte del hecho de que el originador conoce la clave pública del destinatario, la cual puede haber sido entregada o publicada en algún canal, red social o alguna página *web* de interés para el originador. En BBM, como las cuentas son públicas, un usuario podría consultar y estar interesado en tener contacto con alguna cuenta publicada. La Figura 78 y la Figura 79 muestran un diagrama de secuencia todos los pasos utilizados por el originador y el destinatario respectivamente para exitosamente solicitar y autorizar un contacto.

Figura 78. Diagrama de protocolo PSAC (rol originador).



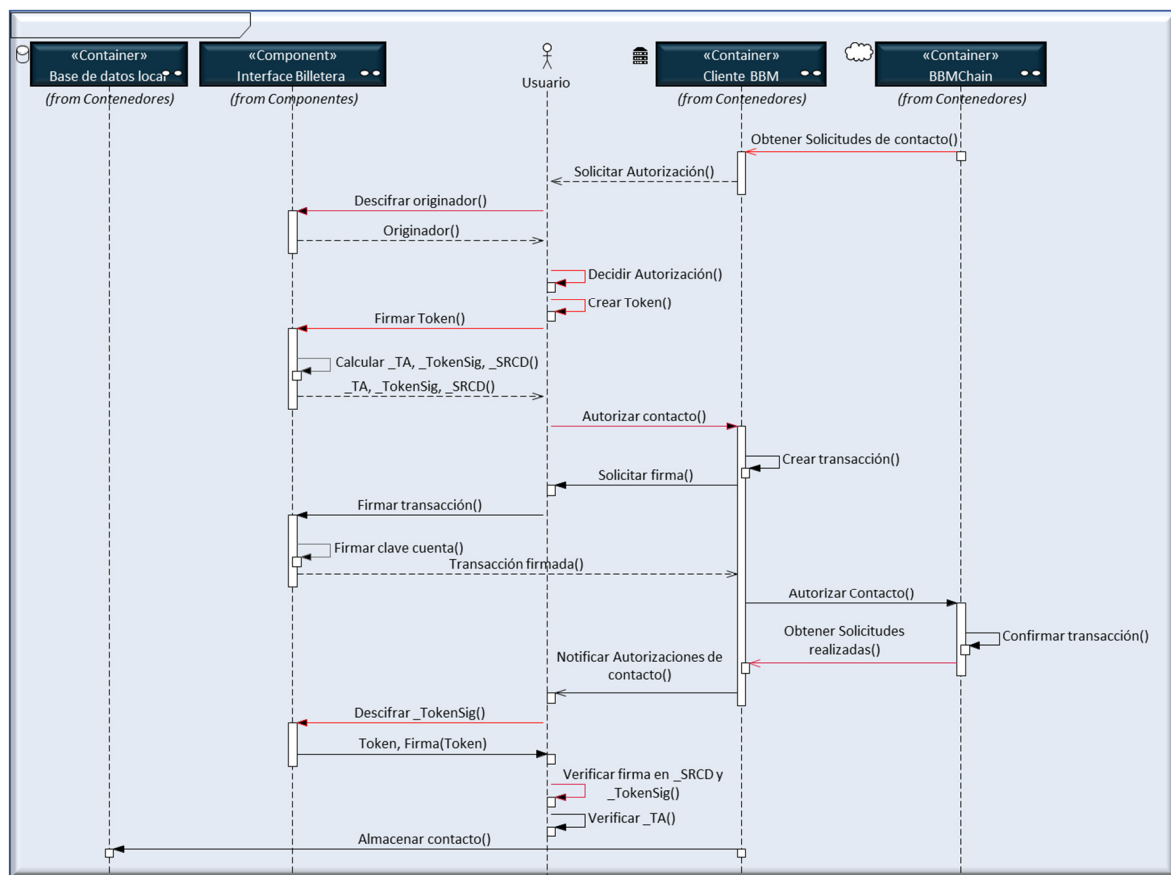
Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PSAC (rol originador, Figura 78) son los siguientes:

- ✎ En el diagrama se describe el protocolo mostrando como el actor usuario juega los dos roles de solicitante y autorizador. Se han integrado los dos roles para simplificar el diagrama y el protocolo.
- ✎ “Autenticar usuario” es realizado con las claves de la cuenta. Si se están creando los contactos en un entorno más restringido (con el parámetro RequiredAuthforALL del registro de seguridad del usuario activado), la creación de contactos la realizaría el gestor.
- ✎ “Solicitar datos” inicia el proceso de creación del contacto.
- ✎ “Obtener ID” indica al cliente BBM que genere los identificadores desde la billetera. Se deberían de generar la clave de contacto (**m/i/z/0/x**) y la clave de gestión si fuera necesaria. Esto es indicado en el fragmento “alt” con “Generar claves de gestión.”
- ✎ “Cifrar originador” prepara el valor del campo **_RCO** cifrando la clave pública de la cuenta del originador y su firma.

- ✎ “Datos solicitud” envía el contacto cumplimentado incluyendo el valor del campo `_Server` de la cuenta originadora.
- ✎ “Solicitar firma” es llamado después de la creación de la transacción y, como se muestra en el diagrama, indica a la billetera que firme la transacción con la clave privada adecuada.
- ✎ “Solicitar contacto” envía el registro del contacto que queda a la espera de confirmación. Se han incluido por simplicidad las verificaciones de origen y cuentas que deben ser ejecutadas antes de la confirmación.

Figura 79. Diagrama de protocolo PSAC (rol destinatario).



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PSAC (rol destinatario, Figura 79) son los siguientes:

- ✎ “Obtener solicitudes de contacto” es representado como una notificación desde BBMChain que indica al usuario que ha recibido una petición de contacto. Desde aquí el usuario hará el papel de destinatario, teniendo que responder a “Solicitar autorización”.
- ✎ “Descifrar originador” toma el valor del campo `_RCO` y lo descifra para identificar quien ha realizado la petición y comprueba que la firma sea válida. Con la firma comprobada el

destinatario tiene la seguridad que el contacto ha sido enviado por el originador que aparece en **_RCO**.

- ✎ “Decidir autorización” es un proceso que ilustra la decisión que debe tomar el destinatario para aceptar o rechazar la petición.
- ✎ “Crear **Token**” indica que el destinatario asocia **_RCO+_RCD+_Exp**, indicando una fecha de expiración para el contacto que será almacenada en el campo **_Exp** del contacto.
- ✎ “Firmar **Token**” inicia el proceso de firma junto con la generación de los valores de **_TA**, **_TokenSig** y **_SRCD** que deben ser retornados para cumplimentar el registro.
- ✎ “Autorizar contacto” crear la transacción que debe ser firmada con la clave de la cuenta **_RCD**. Esta transacción llamará al contrato inteligente enviando los campos generados y actualizando el registro del contacto. Se deben realizar las validaciones y comprobaciones respectivas. Una de las comprobaciones a realizar dentro de BBMChain es que la firma **_SRCD** del campo **_TA** sea válida y el firmante sea **_RCD**.
- ✎ “Obtener solicitudes realizadas” es un proceso asíncrono que realiza el usuario con el rol de originador. Desde esta parte del diagrama se muestra el proceso de verificación de la solicitud por parte del originador para comprobar si ha sido aceptada su petición.
- ✎ “Descifrar **_TokenSig**” indica que el originador descifra el campo **_TokenSig** y obtiene la firma del **Token**.
- ✎ “Verificar firma en **_SRCD** y **_TokenSig**” indica el proceso de verificación de la firma del **token** para comprobar que **_RCD** es el autor de las dos firmas.
- ✎ “Verificar **_TA**” comprueba por último que el *hash* de la firma del **Token** sea válido.

En cuanto a los aspectos críticos de seguridad de PSAC, se remarca lo siguiente:

- △ Accediendo al contacto no es posible conocer qué cuentas están involucradas en la petición de contacto.
- △ El protocolo garantiza el control de los contactos para los usuarios corporativos que utilicen diferentes políticas de seguridad, desde muy restrictivas con el uso de las cuentas de gestión de alto nivel, hasta poco restrictivas permitiendo que los usuarios de más bajo nivel controle sus contactos.
- △ El uso del campo **_RCO** cifrado garantiza la privacidad de los contactos, haciéndolo visibles solo para los interesados o autorizados.

- △ Ocultar la firma dentro de **_TokenSig** dificulta a los atacantes descubrir las cuentas que están relacionadas en el contacto.
- △ Se garantiza la integridad del registro de contacto, pudiendo solo ser modificado por el originador y el destinatario cuando sea necesario.
- △ No es posible negar la participación de las dos cuentas en la creación del contacto puesto que las dos presentan sus firmas y, por lo tanto, son poseedores de las claves privadas correspondientes a las claves públicas asociadas en el contacto.

Los siguientes protocolos complementan al protocolo PSAC:

📁 **Protocolo de Rechazo de Contacto (PRCZ):** En la máquina de estados del contacto (Figura 77) el estado “rechazado” indica que el contacto queda a la espera de una respuesta por parte del originador. Debido a que se modifica el campo **_Rjc** del registro del contacto, es necesario definir los posibles códigos de rechazo a considerar para que el originador pueda suministrar la información solicitada.

Puede ocurrir que el destinatario solicite un certificado de identidad, que debe ser suministrado por el originador haciendo uso del campo **_AD**. Este campo podría contener un enlace a un fichero cifrado almacenado en un directorio compartido (IPFS, por ejemplo) que solo el destinatario pudiera interpretar.

Cuando un contacto se rechaza, el campo **_ST** pasa de “solicitado” a “rechazado”. Ambos estados son los únicos en los que se espera una actuación del destinatario y originador respectivamente y por eso requieren este protocolo como una especificación especial adicional. El Protocolo de Rechazo de Contacto (PRCZ) sigue el esquema de los protocolos que realizan cambios en BBMChain, por lo tanto, no se describirá en detalle su funcionamiento.

📁 **Protocolo de Cambio de Estado de Contacto (PCEC):** Como se observa en la Figura 77, los estados “solicitado” y “rechazado” exigen un tratamiento especial con diferentes protocolos, pero el resto de los estados pueden ser controlados con un Protocolo de Cambio de Estado de Contacto (PCEC). El cambio de estado gestionado por el destinatario solo puede ser realizado con la clave que aparece en el campo **_RCD** del registro de contacto. De la misma manera que el protocolo PRCZ, PCEC sigue el esquema de modificación de datos en BBMChain, que ya ha sido descrito y se puede deducir su funcionamiento a partir de dichos protocolos.

4.13. Protocolo de Intercambio de Localización entre Servidores (PILS)

Este protocolo permite la comunicación segura entre los servidores. En BBM, antes de que dos servidores intercambien información debe realizarse un acuerdo entre ellos. Bajo este concepto nace la RPA, que puede ser configurada e implementada de muchas formas diferentes, pero siempre cumpliendo dos objetivos. El primer objetivo es que debe asegurar la disponibilidad de la información almacenada y el segundo objetivo es que debe cumplir con los protocolos y filosofía de BBM.

Una forma de implementar ARP es por medio de una *blockchain* privada con autenticación donde los servidores deben ser invitados para dar servicios y solo entre ellos intercambian sus ubicaciones. En el caso de la red privada, no sería necesario la utilización del protocolo PILS, que es un protocolo que pretende generar confianza entre los servidores cuando ARP es implementado en una *blockchain* pública o como un grupo de servidores independientes que se unen a BBM para ofrecer sus servicios.

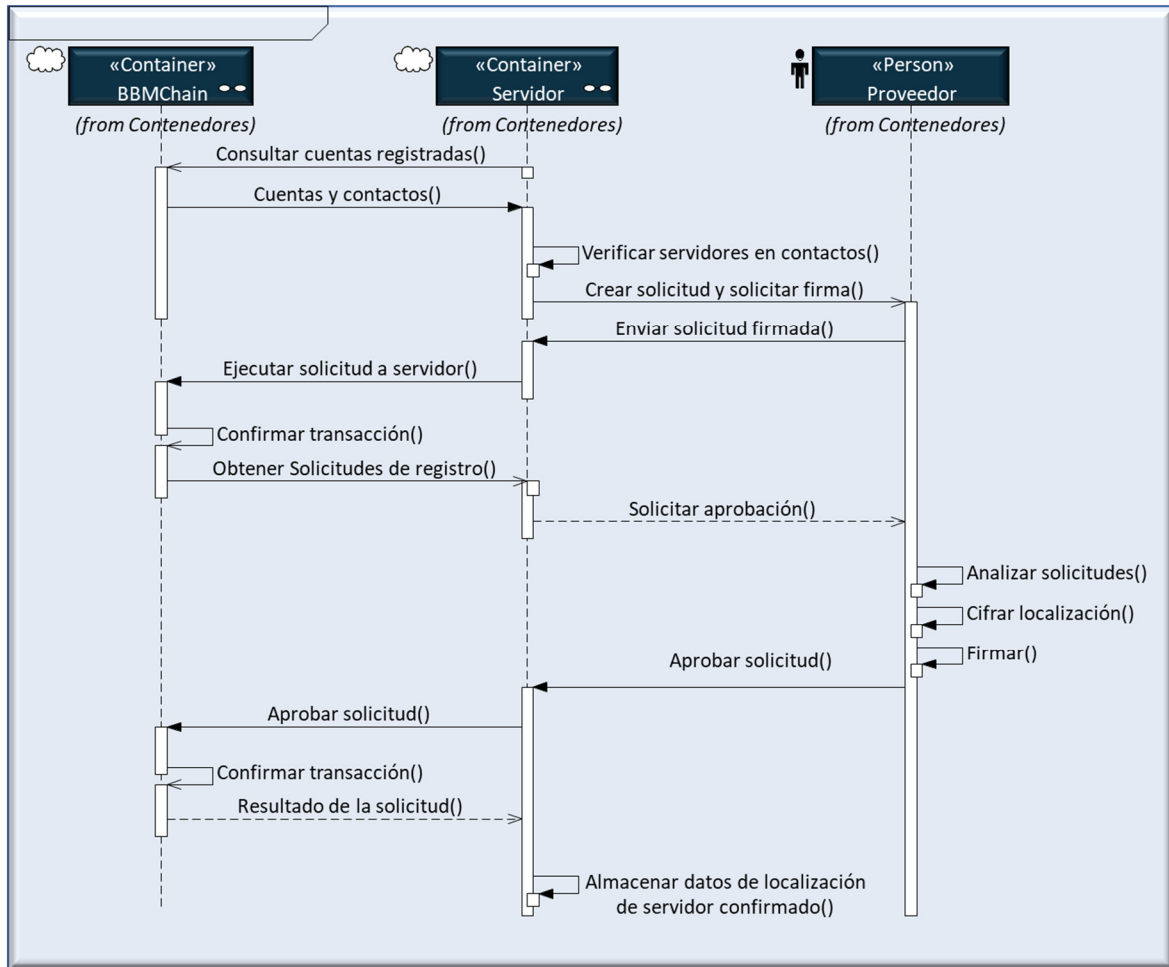
En PILS intervienen dos partes. Un servidor origen que realiza una petición de acuerdo con otro servidor que debe confirmar ese acuerdo. La petición de registro contiene la siguiente información:

- 📄 **_id_source:** Identificador del servidor que realiza la petición, al que se llamará “origen”. Esta es la clave pública utilizada por “origen” durante la ejecución del protocolo PRS.
- 📄 **_id_target:** Identificador del servidor al que se le realiza la petición, al que se llamará “destino”. Esta es la clave pública utilizada por “destino” durante la ejecución del protocolo PRS.
- 📄 **_location:** Localización cifrada por “destino” utilizando la clave pública de “origen”.
- 📄 **_Signature:** Firma realizada por “destino” para confirmar la petición, los datos que se firman es la tupla **_id_source+_id_target**.

Cada servidor deberá realizar la petición a los servidores con los que necesite comunicarse. Si fuese necesario una comunicación mutua, se necesitarían dos peticiones, una para cada servidor. Este protocolo es utilizado cuando una cuenta se registra en un servidor y la cuenta tiene una lista de contactos asociados. El servidor revisará cada contacto en el que la cuenta aparece como destinatario y consultará en el campo **_Server** la clave pública de servidor del originador. Con este dato, el servidor del destinatario del contacto realizará peticiones a los diferentes servidores para las comunicaciones futuras.

La razón de ser de este protocolo es debido a la necesidad de ocultar en lo posible la localización de los servidores a terceros que no pertenezcan al entorno de BBM y que solo sean conocidas por los participantes de la red. La Figura 80 muestra un diagrama de secuencia con los pasos realizados por los servidores para intercambiar sus respectivas localizaciones.

Figura 80. Diagrama de protocolo PILS.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PILS son los siguientes:

- ✎ En la figura previa se muestran los dos roles de “origen” y “destino” ejecutados por los servidores durante la ejecución de este protocolo.
- ✎ “Consultar cuentas registradas” es un proceso periódico que “origen” realiza para identificar las cuentas que están registradas y actualizar la información en BBMStorage.
- ✎ “Cuentas y contactos” retorna todos los contactos solicitados en las cuentas registradas del servidor.

- ✎ “Verificar servidores en contactos” consulta el campo ***_Server*** de los contactos para obtener todas las claves públicas de los servidores y verifica si hay alguna actualización de los registros con los servidores ya contactados con anterioridad.
- ✎ “Ejecutar solicitud a servidor” crea el registro de petición indicando las claves públicas de cada servidor en ***_id_source*** e ***_id_target***.
- ✎ “Obtener solicitudes de registro” es un proceso asíncrono que “destino” realiza para consultar que servidores le han realizado peticiones de conexión.
- ✎ “Solicitar Aprobación” indica que de alguna forma el proveedor debe analizar las solicitudes. Este proceso, aunque indica que debe participar el proveedor, se podría implementar de forma automática.
- ✎ “Cifrar localización” indica el proceso de cifrado que debe ser realizado para que “origen” pueda conocer la localización de “destino”. Ese valor será almacenado en el campo ***_location***.
- ✎ “Firmar” indica que se realiza la firma de las claves públicas de “origen” y “destino” y se almacena en el campo ***_Signature***.
- ✎ “Aprobar solicitud” actualiza el registro de la petición en BBMChain. Se debe verificar en BBMChain que la firma realizada en ***_Signature*** sea realizada por “destino”.
- ✎ “Resultados de la solicitud” indica que “origen” recibe la notificación de las aprobaciones y entonces almacena los datos en BBMStorage para su uso.

En cuanto a los aspectos críticos de seguridad de PILS, se comenta lo siguiente:

- ⚠ No es posible modificar el registro de petición si no se es poseedor de la clave de los servidores involucrados en la petición.
- ⚠ La localización de servidor no es visible públicamente en el registro, teniendo acceso a ella solamente el servidor “origen”.
- ⚠ No se puede modificar el campo ***_location*** si no se está en posesión de la clave privada del servidor “destino”.
- ⚠ No es posible realiza la petición en nombre de otro servidor. La suplantación de servidor no puede ser realizada.
- ⚠ Utilizando las claves públicas de los servidores en los contactos de las cuentas registradas asegura que solamente los servidores conozcan sus respectivas localizaciones.

Con el uso de una RPA independiente basado en una *blockchain* privada no sería necesario el uso de este protocolo, ya que los nodos participantes serían solamente servidores y estarían autenticados e identificados en la red y regulados por su propia gobernanza.

4.14. Protocolo de Envío de Correo (PEC)

El correo en BBM debe ser depositado en los servidores del originador para ser reclamado por los destinatarios. Cuando una cuenta es registrada en un servidor, este se convierte en el repositorio de confianza para gestionar todos los correos que se depositan. A diferencia de los servidores SMTP, la seguridad es regulada por BBM, gestionada por los actores encargados del transporte y controlada por los actores usuarios. Todos ellos trabajan coordinadamente para proporcionar seguridad en BBM.

Uno de los aspectos más diferenciadores de esta propuesta con respecto a las demás relacionadas con la seguridad del correo electrónico, es el tratamiento que los servidores dan al correo que almacenan. Los servidores ejecutan dos roles. El primero como depositarios esperando que el correo de sus cuentas registradas sea almacenado en el servidor. El segundo como solicitantes requiriendo el correo de sus cuentas registradas que aparecen como destinatarios en otros servidores. Dado estos dos roles, los servidores nunca envían correo a los destinatarios.

Para que un servidor basado en BBM acepte el depósito de correos, deben cumplirse los siguientes prerequisites:

1. El originador debe de estar registrado en el servidor (protocolo PRCS).
2. El destinatario tiene que ser contacto “autorizado” del originador (protocolo PSAC).
3. El originador debe presentar el **Token** firmado por el destinatario.
4. El destinatario debe ser una cuenta registrada en BBM (protocolo PRU).
5. El destinatario debe estar registrado en algún servidor “activado” (protocolo PRCS).

En caso de no cumplirse todos los requisitos anteriores, se debe rechazar el correo y el servidor debe notificar dicho rechazo o aceptación al originador. Una vez comprobados todos los prerequisites, el servidor también debe comprobar que con todos los servidores asociados a los destinatarios se haya ejecutado el protocolo PILS para el intercambio de localización entre los servidores. Una vez cumplidas todas las reglas, el servidor depositará el correo y quedará a la espera para entregarlo al destinatario cuando su servidor lo solicite.

A diferencia de los protocolos de correo SMTP de envío, el correo nunca llega al servidor destinatario con el Protocolo de Envío de Correo (PEC) y es necesario la ejecución de otros protocolos para que esto suceda. Debido a los altos requisitos de seguridad y control de BBM, este protocolo tiene una alta complejidad y por esa razón se exponen primero todos los aspectos técnicos para tener en cuenta antes de describirlo en detalle.

El correo en BBM no está almacenado en *blockchain* BBMChain, sino en RPA. Cada servidor BBM debe cumplir con la especificación de los protocolos BBM y una de las normas es que el correo debe ser almacenado siguiendo un formato acorde a los campos descritos en las definiciones a principio de este capítulo.

Los correos contienen una cabecera y un cuerpo. La cabecera va dirigida al servidor, que es quien puede leerla e interpretarla. El cuerpo solo puede ser interpretado por el destinatario, por lo tanto, estará cifrado. A nivel de protocolo, en esta propuesta el interés estará centrado en la cabecera, que son los datos que se procesarán hasta que el correo llegue al destinatario. La **cabecera** deberá contener los siguientes campos:

- ✎ **_Celd**: Valor resumen o *hash* que identifica de manera única cada correo. Es el valor *hash* del cuerpo del correo.
- ✎ **_CeldRef**: Valor *hash* del correo que cambia dependiendo del estado asociado al correo.
- ✎ **_CbHash**: Valor *hash* de la cabecera del correo.
- ✎ **_ServerPk**: Clave pública del servidor del remitente, que es el destino inicial del correo.
- ✎ **_MailBoxid**: Clave pública del remitente del correo. Se utilizará la clave pública asociada al buzón para consultar el registro de la cuenta en el servidor. Del registro obtener el valor del campo **_accountId** que es la clave pública de la cuenta remitente.
- ✎ **_ServerToken**: Es la firma del remitente de algunos campos de la cabecera y el cuerpo del mensaje en los que se incluye: **_Celd**, **_ServerPk**, **_MailBoxid**, **_SR**. Este campo servirá para que el servidor compruebe que la cuenta firmante es la cuenta registrada y es la que ha enviado ese correo.
- ✎ **_ServerSignature**: Es la firma realizada con la clave pública almacenada en **_ServerPk** del contenido del campo RC+.
- ✎ **_SR**: El registro de seguridad asociado a la cuenta del remitente. Este valor debería ser consultado por el servidor para comprobar las reglas de control especificadas en el registro de seguridad.

- ✎ **_ST**: Estado del correo. Los valores posibles son: creado, esperando firma (remitente, autor, autorizador), firmado (remitente, autor, autorizador), rechazado, depositado, entregado, recibido, leído y reenviado.
- ✎ **_RC+**: La relación de comunicación que está compuesta por todas las tuplas remitente y destinatario. Este campo estará cifrado con la clave pública del servidor (Figura 58). Cada RC contendrá los siguientes campos:
 - 📄 **RCpk**: Clave publica de la cuenta del originador.
 - 📄 **RCrol**: Rol que juega el originador en la comunicación, pudiendo ser: autor, remitente o autorizador.
 - 📄 **RCDpk**: Clave pública de la cuenta del destinatario.
 - 📄 **RCcf**: Registro de seguridad de la relación de comunicación específica entre originador y destinatario, más que sobre el correo en general. No se ha utilizado en esta primera versión de la especificación.
 - 📄 **Sign(Mc,RCsk)**: Es la firma que crea el originador del cuerpo del mensaje (Mc), indicando con ello que autoriza el mensaje con el rol indicado.
 - 📄 **S(Token)**: El **Token** y su firma son indispensables para validar el contacto. Este valor debe ser suministrado por cada participante cuando valida su participación. El remitente, como es el que crea el correo, incluye este valor en la creación del correo.
 - 📄 **_ToServer**: Es la clave pública del servidor que está asociado a la cuenta del destinatario (campo **_ServerPk** del registro de la cuenta del destinatario en el servidor).

Los más importantes relacionados con la seguridad del correo son los campos **_RC+**, **_SR**, **_ServerToken** y **_ServerSignature**. El campo **RC+** contiene información importante que debe ser consultada y verificada en **BBMChain** por parte del servidor depositario. En su forma más simple, un correo contendría solamente una relación de comunicación que se debería de validar contra un contacto asociado a la cuenta remitente dentro de **BBMChain**.

En cuanto al **cuerpo** del correo, este debe ser interpretado solamente por el destinatario y como mínimo contiene:

- ✉ **_Cc**: Es un registro que indica las características del correo, como la fecha de envío, el asunto y otros campos que el usuario desee incluir para tener una referencia del correo.

Esta información debería de ser solamente interpretada por el destinatario y es la información que primero descifran los clientes BBM para mostrar al usuario.

- ✉ **_Mc**: Es el mensaje cifrado utilizando la clave pública del destinatario (Figura 58).
- ✉ **_XMc**: Datos opcionales utilizados por el destinatario para descifrar el correo. El algoritmo ACDC es el utilizado en toda la propuesta, y en este campo se almacenan los datos opcionales para que el destinatario pueda descifrar.

El formato del mensaje cifrado en el campo **_Mc** no es definido en este trabajo y, como se comentó a principios del capítulo, puede ser utilizado MIME o cualquier otro formato desarrollado para este fin que incluya la posibilidad de añadir ficheros adjuntos al contenido del correo. Se propone que por rendimiento se utilice un sistema como IPFS y se añada la referencia dentro del cuerpo del correo. En tal caso el fichero adjunto debería ser cifrado para el destinatario y almacenado en IPFS.

Una de las tareas que realiza un servidor al recibir un correo después de validar e interpretar la cabecera, es descifrar las relaciones de comunicación para validar los contactos del remitente y los roles de cada uno de los participantes. **RC+** deberá contener al menos una relación de comunicación donde estará indicado el destinatario y el rol que juega el originador en el correo, es decir como mínimo contendría una **RC** con el rol de remitente. En BBM, un correo puede ser enviado a varios destinatarios y ser enviado por varios originadores, tomando estos diferentes roles (campo **_RCrol**) como se explica a continuación:

- 👤 **Remitente**: Solo puede ser un originador el que tome este rol y tiene la responsabilidad de enviar el correo desde su servidor. Todos los destinatarios deben ser contactos aceptados por los destinatarios. Solo puede haber un remitente por cada correo y es el que crea, cifra y firma el correo inicialmente.
- 👤 **Autor**: El autor aparece en el correo cuando el remitente es el que envía el correo y no es el autor del contenido como tal. Este rol es importante en un entorno corporativo donde el contenido de un correo y su envío puede acarrear problemas laborales, sobre todo evita el enviar un correo en nombre de otro sin su autorización.
- 👤 **Autorizador**: Este rol es una figura de control que evita el envío de correos sin autorización de un tercero. Al igual que el autor, tiene una función muy importante dentro de las organizaciones donde la fuga de información, errores o el envío de información clasificada es delicada y requiere un tercero para su verificación. Esto es muy útil para las organizaciones donde el contenido o los ficheros adjuntos tiene una sensibilidad muy alta y es necesario que cada correo pueda ser supervisado antes del envío.

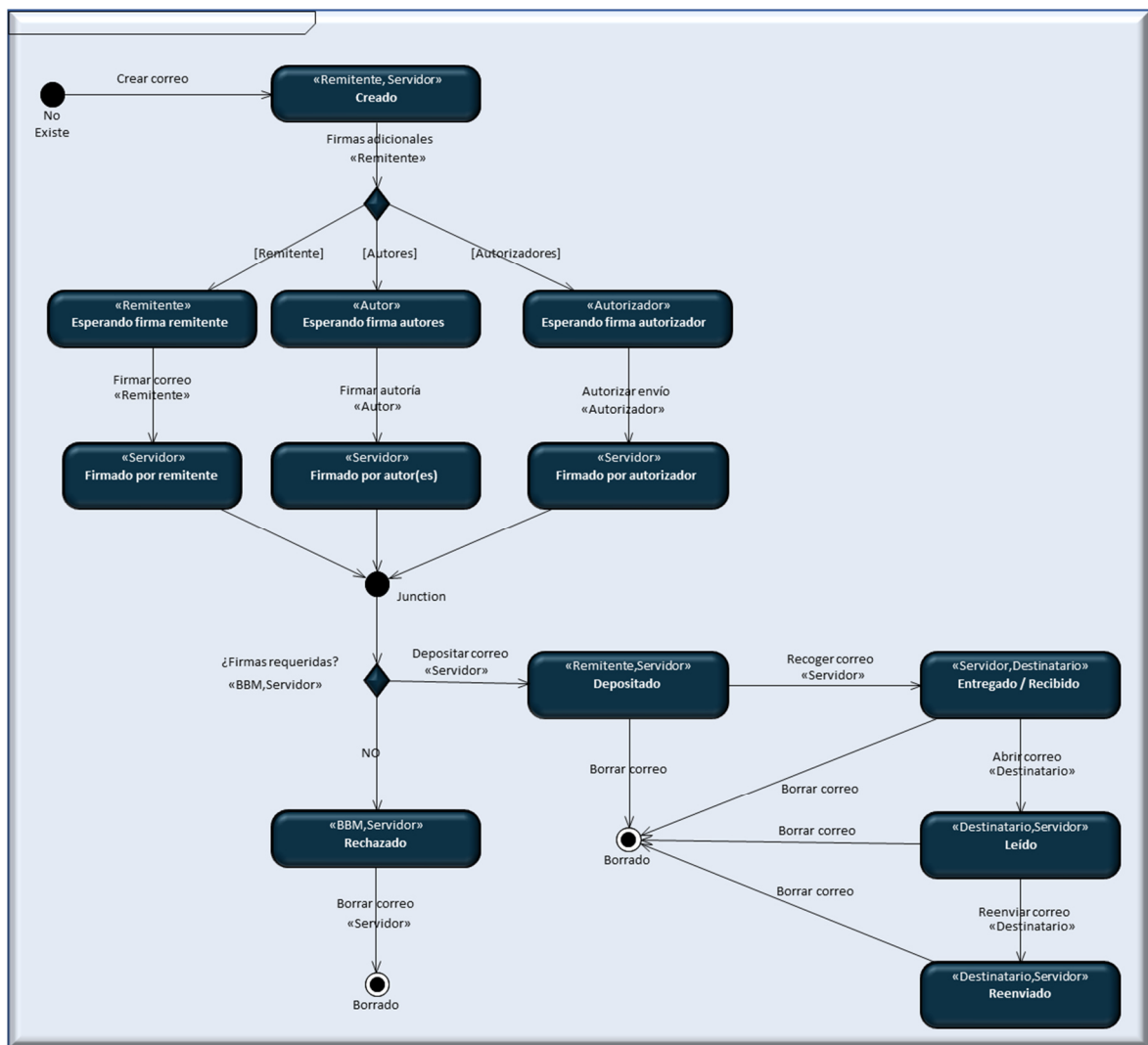
El campo *_RCcf* de la relación de comunicación, que está relacionado con los aspectos de control que requiere el originador sobre el correo mismo, indica de forma particular cuáles son los requisitos que este usuario impone sobre el correo que envía. Esta característica permite que varios originadores puedan exponer diferentes requisitos sobre el correo que se envía cuando varios de ellos participan en la comunicación.

Los controles de seguridad relacionados con los usuarios (Tabla 14) fueron descritos anteriormente y algunos de ellos aplican especialmente al correo. El servidor comparará los valores de este campo con los valores asociados a la cuenta para que no entren en conflicto y procederá según la configuración deseada. Dentro de la filosofía BBM, el cliente BBM con el servidor deben verificar si se cumplen, pero los servidores de los destinatarios también deberían verificar esos registros.

Al final, la seguridad la definen los usuarios y los participantes de todo el sistema velan por ella (servidores, usuarios y BBMChain) y si alguno no lo cumple, otro participante puede denunciar dicho comportamiento y, según la implementación, el sistema puede llegar a sancionar o disminuir su reputación si se utiliza un sistema de gamificación o *ranking* para los participantes.

El campo *_ST* almacena el valor del estado del correo. Para entender mejor como son los diferentes estados por los que pasa un correo y las acciones que pueden ser realizadas, se presentan en la Figura 81 las diferentes transiciones de estado del correo en un diagrama de máquina de estados.

Figura 81. Diagrama de máquina de estados del correo.



Fuente: Elaboración propia.

Con respecto a la máquina de estados del correo:

- ✿ El estado “creado” indica que el remitente ha creado un correo y que está a la espera de ser procesado por el servidor.
- ✿ “Esperando firma ...” aparece tres veces indicando que, una vez procesado el correo por el servidor, queda a la espera de la confirmación de los autorizadores y los autores, si hay incluido alguno. La firma del remitente ya está incluida cuando se creó el correo y ha sido verificada por el servidor durante el procesamiento del correo.
- ✿ “Firmado por ...” aparece también tres veces señalando que las cuentas que ejecutan esos tres roles han confirmado los correos y han modificado el campo ***Sign(Mc,RCsk)*** y el campo ***S(Token)***, indicando que están de acuerdo con el correo y su contenido.

- ✿ “Firmas requeridas” indica la comprobación que realiza el servidor para determinar si pasa el correo a estado “rechazado” o lo acepta pasando el correo al estado “depositado”.
- ✿ “Rechazado” es un estado de espera donde el servidor ha notificado al remitente la razón del rechazo y queda a la espera para ser eliminado.
- ✿ “Borrado” es el estado que indica que un correo es borrado y está a la espera de ser eliminado permanentemente.
- ✿ “Depositado” es el estado que indica que el correo está listo para ser entregado a los servidores de los destinatarios cuando los soliciten.
- ✿ “Entregado” es el estado que indica que los servidores de los remitentes los han reclamado y el servidor los ha entregado. En ese momento, quedarán como “entregados” en el servidor del remitente, pero en el servidor del destinatario aparecerán como “recibidos”.
- ✿ “Leído” es un estado que indica que el cliente BBM ha reclamado el correo y el servidor lo ha entregado.
- ✿ Si las notificaciones han sido activadas en el registro de la cuenta, es necesario que los servidores las realicen cuando se efectúen esos cambios de estado en el correo. En una implementación de BBM basada en ranking, la no realización de las notificaciones por parte del servidor podría generar una penalización.
- ✿ Hay que tener en cuenta que una notificación de lectura es muy diferente a una solución de correo certificado, ya que en este caso se requiere la firma del destinatario para recibir el correo, es decir, es una recepción del correo consciente y activa por parte del destinatario.

Se ha comentado previamente que cuando un correo está en el estado “creado” queda a la espera de que el correo sea procesado por el servidor. Antes del depósito, el servidor realiza una serie de comprobaciones para determinar la validez del correo y de todos los participantes. Todas estas validaciones se realizan consultando el estado de **BBMChain**:

1. El autorizador se debe corresponder con el gestor de la cuenta, el cual es el responsable absoluto del remitente y, por lo tanto, el único usuario que puede tener este papel. Esto puede ser fácilmente verificado, ya que el campo **_owner** del registro de la cuenta tiene almacenada su clave pública, así que la clave pública del autorizador debe coincidir con dicho campo.

2. El remitente y los autores deben tener como contacto a todos los destinatarios del correo. Cada uno de los participantes debe proporcionar ***S(Token)*** para validar el contacto, esto puede ser realizado en el momento que se realice la firma.
3. Todos los originadores son contactos del remitente. Aunque esta regla es lógica y de sentido común, hay que tenerla presente dentro del proceso de validación y verificar que sean contactos válidos. El remitente debe presentar en la cabecera todos los ***S(Token)*** que lo autorizan para poder incluirlos dentro de la comunicación.
4. El autorizador solo juega un papel de control, es un usuario de gestión y no es una cuenta, por lo que no tiene contactos.
5. El registro de seguridad ***_SR*** se debe corresponder con el registro de seguridad asociado a las diferentes cuentas originadoras. Este registro es opcional, ya que puede ser consultado directamente desde **BBMChain**, que es donde se almacena. En este trabajo se propone una configuración básica (Tabla 14) que es suficiente para los objetivos marcados en la tesis, pero que puede ser ampliada perfectamente para otros objetivos siguiendo la misma filosofía presentada aquí.
6. Todas las firmas del cuerpo del mensaje deben ser válidas. El servidor debe validar cada una de las firmas comparando la firma con la clave pública de cada uno de los originadores del correo.
7. Hay que comprobar si la cuenta destinataria puede recibir correo. El servidor puede consultar el registro de seguridad y ver si no hay algún control que impida la recepción de correo, por ejemplo, si está activado o no el valor **OnlySenderAuth**. Si este valor no está activado, eso indica que no es necesaria la validación de los contactos. El registro de seguridad, como se ha comentado anteriormente, permite gran flexibilidad para el control y gestión del correo y se pueden crear más parámetros para mejorar este aspecto.

El protocolo PEC describe el flujo que sigue el correo desde que el estado “creado” hasta el estado “depositado” del correo y debe hacer cumplir todas las reglas de seguridad marcadas por los originadores. La creación del correo implica al **remitente** los siguientes **pasos**:

1. Seleccionar la clave pública del buzón de la cuenta del remitente.
2. Obtener la clave pública del servidor del remitente. El valor ***_ServerPk*** del registro de la cuenta.
3. Crear correo con el formato correspondiente.

4. Añadir al campo **_Cc** un asunto con otros datos de interés para la búsqueda del correo en el cliente BBM.
5. Seleccionar de los contactos de la cuenta del destinatario, los autores y añadirlos en **_RC+**
6. En caso de ser necesaria la autorización de un gestor (opción **ReqSendAuth** debería estar activada en el registro de configuración de la cuenta), añadir la clave pública del autorizador a **_RC+** y cifrar **_RC+** con la clave pública del servidor.
7. Cifrar el correo y el campo **_CC** para el destinatario (Figura 58) y modificar el campo **_MC**, añadir los datos necesarios para el descifrado en el campo **_XMc**. Los campos **_Cc**, **_MC** y **_XMc** conformarían el cuerpo del correo.
8. Calcular resumen (*hash*) del cuerpo del correo (**_Celd**).
9. Crear cabecera y firmar (**_ServerToken**). Hay que recordar que los datos firmados por el remitente no tienen las firmas de los originadores (**Sign (Mc, RCsk)**) ni los **S(Token)** de seguridad del contacto con el destinatario que serán añadidos posteriormente por el servidor. Los únicos **S(Token)** que estarían inicialmente son los que posee el remitente: los contactos con el destinatario y con los otros originadores.
10. Cifrar el correo para cada uno de los autores y el autorizador (Figura 58) y firmarlo. Esto significa que, si el correo es enviado por dos autores y un autorizador, se deben añadir al cuerpo del correo cuatro correos cifrados: uno para destinatario, dos más para los autores y otro para el autorizador.

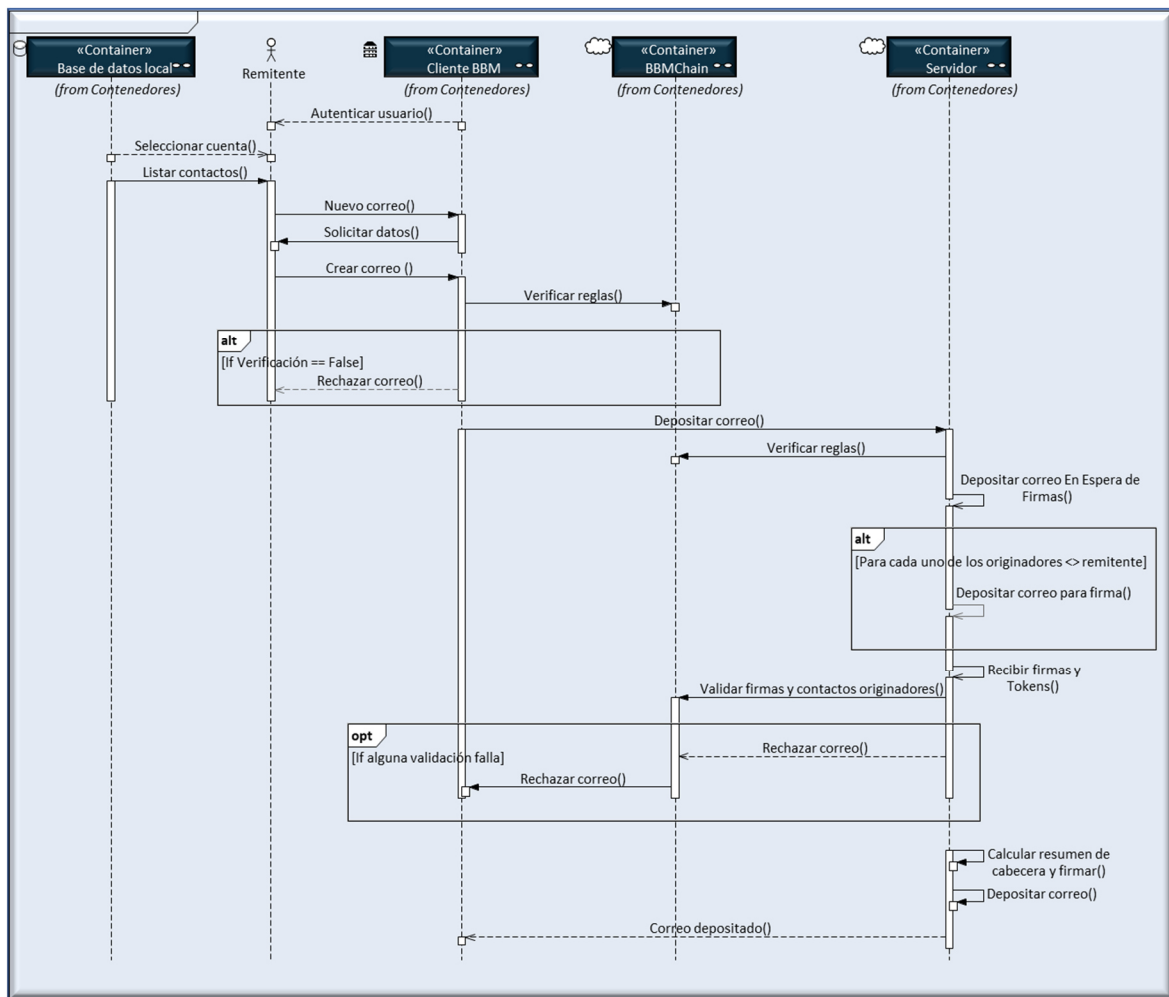
Estos correos creados por el remitente podrán ser depositados por el servidor después de realizar todas las comprobaciones. El **servidor** al recibir el correo en estado “creado” debe realizar una serie de acciones para poder depositar el correo y ponerlo en estado “depositado”. Dichas **acciones** son las siguientes:

1. Recibir el correo y obtener el remitente (**_MailBoxid**) y su firma (**_ServerToken**).
2. Comprobar que el remitente está registrado en el servidor, es decir, comprobar el registro del buzón donde aparece la firma del servidor (**_ServerSignature**).
3. Comprobar, en caso de que aplique, que el autorizador será el gestor de la cuenta (**_owner**).
4. Comprobar que los originadores sean contactos válidos del remitente (**_RC+**).
5. Comprobar los resúmenes del cuerpo y la cabecera del correo “creado” (**_Celd** y **_CbHash**).

6. Verificar la firma del remitente (*_ServerToken*).
7. Comprobar que el registro de seguridad *_SR* del correo se corresponda con el asociado a la cuenta del remitente.
8. Si todas las comprobaciones son correctas, se deberá depositar el correo con el estado “pendiente de firma”. Debe haber tantos correos como firmas de los originadores se necesiten. Todos los correos enviados como “pendiente de firma” deberán tener el valor del campo *_Celd* del correo original en el campo *_CeldRef*. En la práctica, suele utilizarse un valor *nonce* para generar un nuevo valor para el campo *_Celd* para estos correos de firma.
9. Cuando todos los originadores firmen el correo (*Sign (Mc, RCsk)*), adjunten los tokens (*S(Token)*) y el servidor reciba los correos, deberá comprobará que son correos en estado “pendiente de firma” y buscará el correo original identificado con el valor *_CeldRef*.
10. Se obtendrán las firmas y los tokens y se comprobará la validez de los contactos y sus firmas.
11. Si todas las comprobaciones son correctas, en el correo original se actualizan los campos *Sign (Mc, RCsk)* y *S(Token)* del campo *_RC+* de la cabecera y se cifra *RC+* para el destinatario con la clave pública del servidor asociado a la cuenta (campo *_ServerPk* de la solicitud del registro de la cuenta).
12. Se calcula el resumen de la cabecera (*_CbHash*) y se firma la cabecera (*_ServerSignature*).
13. Se eliminan todos los mensajes cifrados del mensaje original y se deposita el correo con la cabecera actualizada para que el servidor destino pueda interpretar *_RC+* y en el cuerpo del correo se añade solo los datos cifrados para el destinatario. En caso de ser más de un destinatario, se deberán crear tantos correos como destinatarios haya.

La Figura 82 muestra un diagrama de secuencia donde se pueden ver los mensajes y actores dentro de la arquitectura BBM que participan dentro del protocolo PEC y se ilustran de forma simplificada los procesos detallados anteriormente. Cabe resaltar el mensaje “depositar correo para firmar”, donde se observa que el protocolo queda realmente en espera de la recepción de las firmas y tokens por parte de los originadores. En el apartado correspondiente al protocolo de recepción de correo, se aclara en qué consiste esa espera y cómo se logra seguir el proceso exitosamente.

Figura 82. Diagrama del protocolo PEC.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PEC son los siguientes:

- ✎ “Autenticar usuario” es un proceso resumido que indica el uso de la clave del usuario para la autenticación.
- ✎ “Seleccionar cuenta” es un proceso resumido de la selección de una de las cuentas del usuario para el envío del correo.
- ✎ “Solicitar datos” es la respuesta del cliente para informarle al usuario que debe realizar todos los pasos de la actividad del remitente vistos anteriormente (paso 1 al 10).
- ✎ “Verificar reglas” desde el cliente BBM hacia BBMChain representa la preparación que el cliente BBM realiza antes de enviar los datos al servidor. En BBM, cada agente (cliente BBM, servidor, BBMChain) debe realizar las comprobaciones respectivas y, de esta forma, los agentes pueden identificar si un agente está fallando en los controles y denunciarlo en BBMChain.

- ✎ “Rechazar correo” en “alt” indica el fallo de alguna comprobación, pero este rechazo no cambia el estado, solo es llamado debido al incumplimiento de algún prerrequisito antes de ser creado cualquier correo.
- ✎ “Depositar correo” indica el envío del correo ya preparado en el cliente al servidor del remitente. Hay que recordar que el estado cuando el correo es recibido en el servidor es “creado”.
- ✎ “Verificar reglas” desde el servidor a BBMChain indica las comprobaciones que el servidor realiza cuando recibe un correo que han sido descritas anteriormente. (paso 1 al 7). Para simplificar se no representa en el diagrama el rechazo si la verificación falla, aunque es un cambio de estado que implica solamente una notificación al remitente.
- ✎ “Depositar correo en Espera de Firmas” representa el almacenamiento de correo original que queda a la espera de recibir todas las autorizaciones de los originadores (paso 8).
- ✎ “Depositar correo para firma” representa la preparación de los correos para los originadores que debe realizar el servidor (paso 8). En este momento el servidor queda a la espera de recibir las firmas de los originadores.
- ✎ “Recibir firmas y Tokens” indica que el servidor recibe las firmas y asocia dichos correos al correo original (paso 9).
- ✎ “Validar firmas y contactos originadores” representa las validaciones que el servidor realiza de la información recibida en los correos de firmas (paso 10).
- ✎ “Rechazar correo” en “opt” indica el cambio de estado en caso de alguna validación fallida. El correo se cambia a “rechazado” y se envía una notificación al remitente.
- ✎ “Calcular resumen de cabecera y firmar” representa la preparación del servidor del depósito del correo (pasos 11 y 12).
- ✎ “Depositar correo” representa el cambio de estado del correo a “depositado” y la finalización del protocolo PEC (paso 13).

Respecto a los aspectos críticos de seguridad de PEC, se enfatiza lo siguiente:

- △ No es posible modificar el correo en estado “creado” antes del depósito en el servidor sin ser identificado el cambio. La primera validación del servidor verifica la integridad del correo validando la firma del remitente (*_ServerToken*) y los valores de *hash* (*CeId* y *CbHash*).

- △ El contenido del correo (**MC**) solo puede interpretado por originadores y el destinatario. Los servidores desconocen el contenido y no tienen forma de acceder y leerlo. Un tercero no autorizado tampoco podría leerlo ni interpretarlo.
- △ Un tercero no autorizado que acceda al servidor no puede interpretar la cabecera del correo a menos que esté en posesión de las claves privadas del servidor.
- △ El remitente del correo no puede negar la creación del correo puesto que es el único poseedor de la clave privada de la cuenta y se puede comprobar por la firma que la ha utilizado.
- △ Los originadores no pueden negar su participación con el rol asignado ya que son los únicos poseedores de la clave privada de sus cuentas y por medio de la firma se puede comprobar su uso.
- △ Los autores no pueden negar ser los autores del correo, ya que han autorizado su envío.
- △ El autorizador no puede negar la autorización dado que, de igual manera que los otros originadores, han utilizado la clave privada de la cuenta que tienen en posesión.
- △ No es posible que una cuenta no registrada en un servidor deposite un correo, el servidor puede comprobar en BBMChain el registro y las firmas.
- △ No es posible enviar un correo a una cuenta destinataria no autorizada a menos que los registros de seguridad indiquen lo contrario. Tanto el cliente BBM como el servidor BBM deben realizar las comprobaciones antes del depósito.
- △ No es posible depositar un correo si en el registro de seguridad se ha indicado que no puede enviar correos. Tanto el cliente BBM como el servidor BBM rechazarían el envío.
- △ No es posible sin estar autorizado conocer que originadores y destinatarios están participando en un correo. Solamente los servidores depositarios y los originadores pueden interpretarlas.
- △ No es posible enviar correo no deseado, todo el correo es deseado.

Si solamente existe un destinatario y no hay autorizador, el protocolo PEC es muy simple, considerando que no es necesaria la espera de firmas y, después de la creación del correo, el servidor puede comprobar y depositar el correo directamente. Por simplicidad, en el diagrama no se muestran las excepciones, ya que son mensajes de error que surgen cuando algo falla y

que deben ser gestionados por cada uno de los componentes de la arquitectura que participan en el proceso.

4.15. Protocolo de Recepción de Correo en Servidor (PRECS)

La recepción del correo es otro de los procesos críticos dentro de BBM con motivo de que implica la comunicación entre servidores. El protocolo PEC, explicado anteriormente, permite el depósito del correo hasta el servidor del remitente. Diseñar el proceso para la distribución del correo nos brindó una oportunidad para cambiar la forma como se ha venido haciendo tradicionalmente la entrega del correo electrónico.

En lugar de que el servidor del remitente hiciera la labor de búsqueda y conexión a los servidores de los destinatarios, lo que se propone con BBM es que los servidores del remitente estén a la espera de las peticiones de los destinatarios, como los apartados postales del correo postal de los destinatarios. Esto significa que los servidores destinatarios tienen que ser los encargados de reclamar el correo a los servidores depositarios.

Para explicar mejor este enfoque es necesario tener en cuenta lo siguiente en BBM:

1. Una cuenta solo podrá recibir correo de sus contactos almacenados en BBMChain (protocolo PSAC).
2. Una cuenta debe registrarse en un servidor, por lo tanto, el servidor debe tener conocimiento de los servidores a los que debe realizar la petición de recogida de correo (protocolo PILS).
3. Las solicitudes de contacto van dirigidas a una cuenta y se almacena en el campo `_server` el servidor donde se depositan los correos del destinatario. Es ese campo el que debe consultar un servidor para realizar las peticiones de descarga de correo.

Bajo estos tres conceptos, un servidor conoce las cuentas registradas y, por lo tanto, los contactos asociados, entonces sabe dónde reclamar el correo, pero solo reclamará el correo dirigido expresamente a sus cuentas y cuyos contactos (y servidores) han sido autorizados por cada una de ellas.

Con el uso del protocolo PILS, los servidores establecen relaciones de confianza y pueden fácilmente autenticarse mutuamente. Si se hace uso de una RPA basada en una red *blockchain* privada, la autenticación se realiza dentro del ámbito de la red privada y es más segura debido a los controles de acceso más estrictos. Sabiendo que los servidores conocen donde deben recoger el correo y que entre los servidores hay confianza mutua, se pueden explicar los pasos que debe ejecutar un servidor para obtener el correo de las cuentas registradas en él.

Un servidor tiene acceso a todas las cuentas registradas y a los datos de los servidores asociados a sus contactos. Un servidor destinatario debe periódicamente verificar los servidores de las cuentas registradas para identificar los posibles servidores remitentes. Este proceso se puede optimizar utilizando mecanismos de notificaciones eficientes donde un servidor comunique a otro que tiene correo para recoger. Los datos que requiere el servidor destinatario para poder obtener el correo depositado son:

-  El campo ***_accountId*** de los buzones de las cuentas registradas en el servidor.
-  El registro de contactos tiene dos campos, el campo ***_RCD*** indica el destinatario y el campo ***_Server*** indica el servidor donde se el remitente almacena los correos para el destinatario.

Con esta información, el servidor de un destinatario puede solicitar al servidor del remitente el correo depositado para él, ya que conoce los servidores donde los contactos de sus cuentas le depositan el correo. El servidor deberá presentar, previa autenticación, el mensaje de solicitud de correo con los siguientes datos:

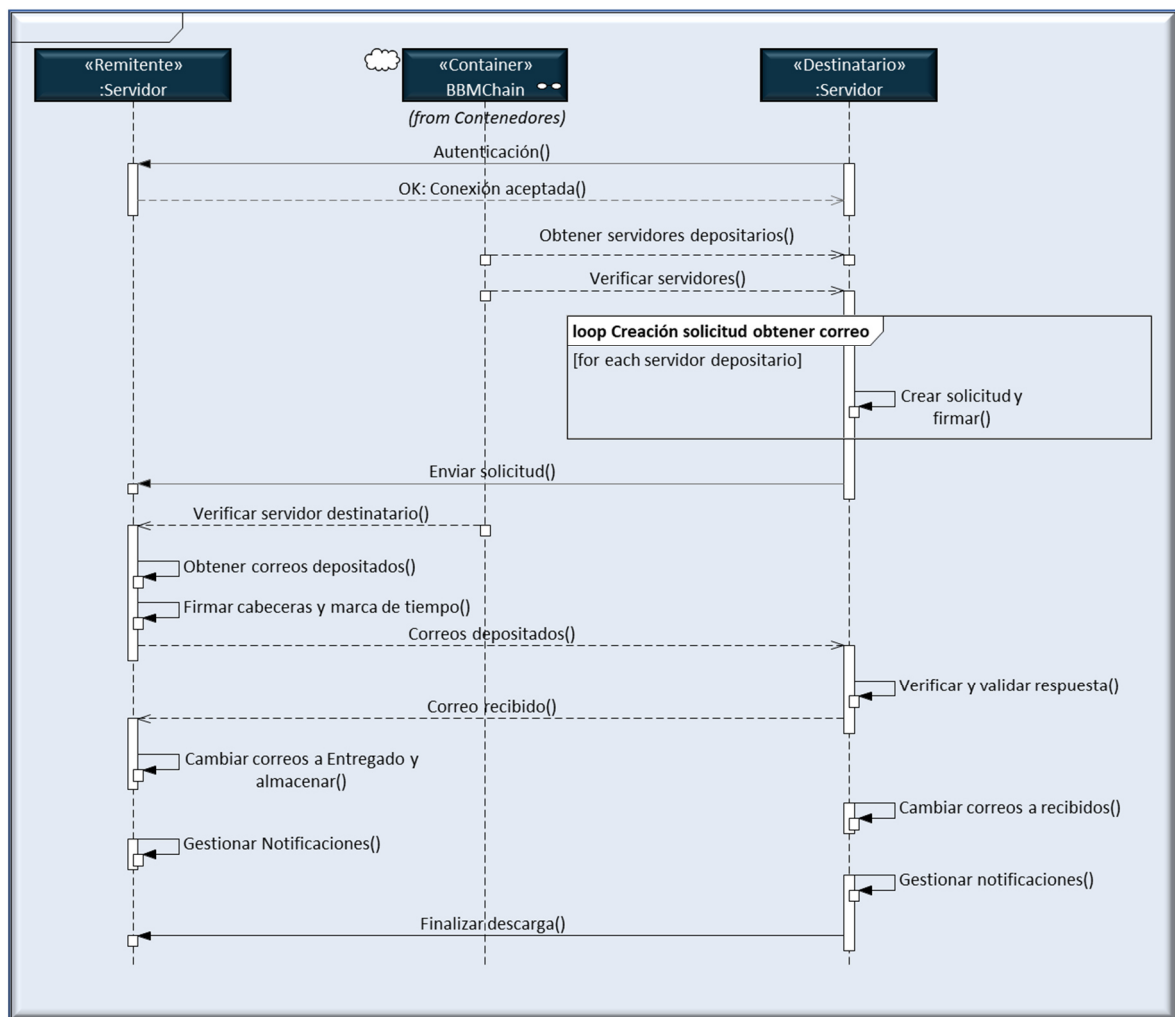
-  ***_requestId***: Número consecutivo que se genera para cada solicitud.
-  ***_fromServer***: Servidor origen de la petición. Contiene la clave pública del servidor que realiza la petición.
-  ***_type***: Tipo de solicitud que puede ser “notificación” o “recogida”. En el caso de este protocolo es “recogida”.
-  ***_date***: Fecha de la solicitud.
-  ***_requesSignature***: Firma de ***_fromServer*** de los datos de la solicitud.

Con estos datos el servidor del remitente podrá verificar si el servidor destinatario es un servidor seguro y conocido (protocolo PILS). También consultará todos los correos en estado “depositado” que han sido dirigidos a las cuentas registradas en el servidor destinatario, esto es, el valor del campo ***_ToServer*** de los correos depositados. La respuesta del servidor incluye como mínimo lo siguiente:

-  ***_headersSignature***: Firma del servidor remitente de las cabeceras de los correos y la marca de tiempo de la respuesta. La importancia de este valor radica en que los correos se enviarán en el formato en el que están almacenados y esto permite asegurar la integridad de los datos transmitidos.
-  ***_emailsList***: La lista de correos depositados.
-  ***_Responsetimestamp***: La marca de tiempo de la respuesta.

Una vez preparada la respuesta, el servidor la enviará al servidor del destinatario y pasará los correos almacenados a estado “entregado” si todo es correcto. El servidor del destinatario deberá leer en todos los correos sus respectivas cabeceras y descifrar el campo **RC+** para realizar las verificaciones y validaciones respectivas, comprobando los datos de contactos y firmas. Dependiendo de la configuración de las notificaciones, gestionará las notificaciones correspondientes. Todo este proceso se puede apreciar mejor en la Figura 83 donde se representa el Protocolo de Recepción de Correo en Servidor (PRECS) con un diagrama de secuencia.

Figura 83. Diagrama del protocolo PRECS.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PRECS son los siguientes:

- ✎ “Autenticación” indica el proceso de autenticación entre los servidores. Este puede ser realizado por medio de un establecimiento de sesión basado en las claves públicas que ambos conocen de cada uno. Este mensaje indica que antes de la ejecución del protocolo se parte de un proceso de autenticación,
- ✎ “Obtener los servidores depositarios” representa la consulta que realiza el servidor de las cuentas registradas con el objetivo de obtener la clave pública de los servidores, dato que es almacenado en el campo **_Server** de los contactos.
- ✎ “Verificar servidores” representa la comprobación del servidor para saber si con todos los servidores ha intercambiado su localización (protocolo PILS).
- ✎ “Crear solicitud y firma” indica la creación de la solicitud a cada servidor y la firma de la solicitud en **_requestSignature**.
- ✎ “Enviar solicitud” representa el envío de la solicitud firmada al servidor depositario.
- ✎ “Verificar servidor destinatario” indica el proceso que realiza el servidor depositario para identificar al servidor destinatario comprobando la firma en el registro en BBMChain.
- ✎ “Obtener correos depositados” indica la búsqueda que realiza el servidor depositario de los correos depositados, donde el destinatario en la relación de comunicación tenga como servidor en el campo **_ToServer** el valor de la clave pública del servidor solicitante.
- ✎ “Firmar cabeceras y marca de tiempo” indica el proceso de asignación de los valores **headersSignature**, **ResponseTimestamp** y **emailsList** de la respuesta del servidor.
- ✎ “Correos depositados” representa el envío de la respuesta al servidor solicitante.
- ✎ “Verificar y validar respuesta” representa la validación de la firma del campo **headersSignature** junto con las cabeceras de **emailsList** de la respuesta recibida del servidor depositario.
- ✎ “Correo recibido” representa la notificación que envía el servidor solicitante al servidor depositario para indicar que el correo ha sido recibido.
- ✎ “Cambiar correo a Entregado y almacenar” representa el cambio de estado de los correos entregados al estado “entregado” y el almacenamiento de estos.
- ✎ “Cambiar correos a recibidos” representa el proceso de modificación de los correos recibidos al estado “recibido” en el servidor solicitante.

✋ “Gestionar notificaciones” indica el procesamiento interno de los servidores para gestionar las notificaciones de los correos entregados y recibidos.

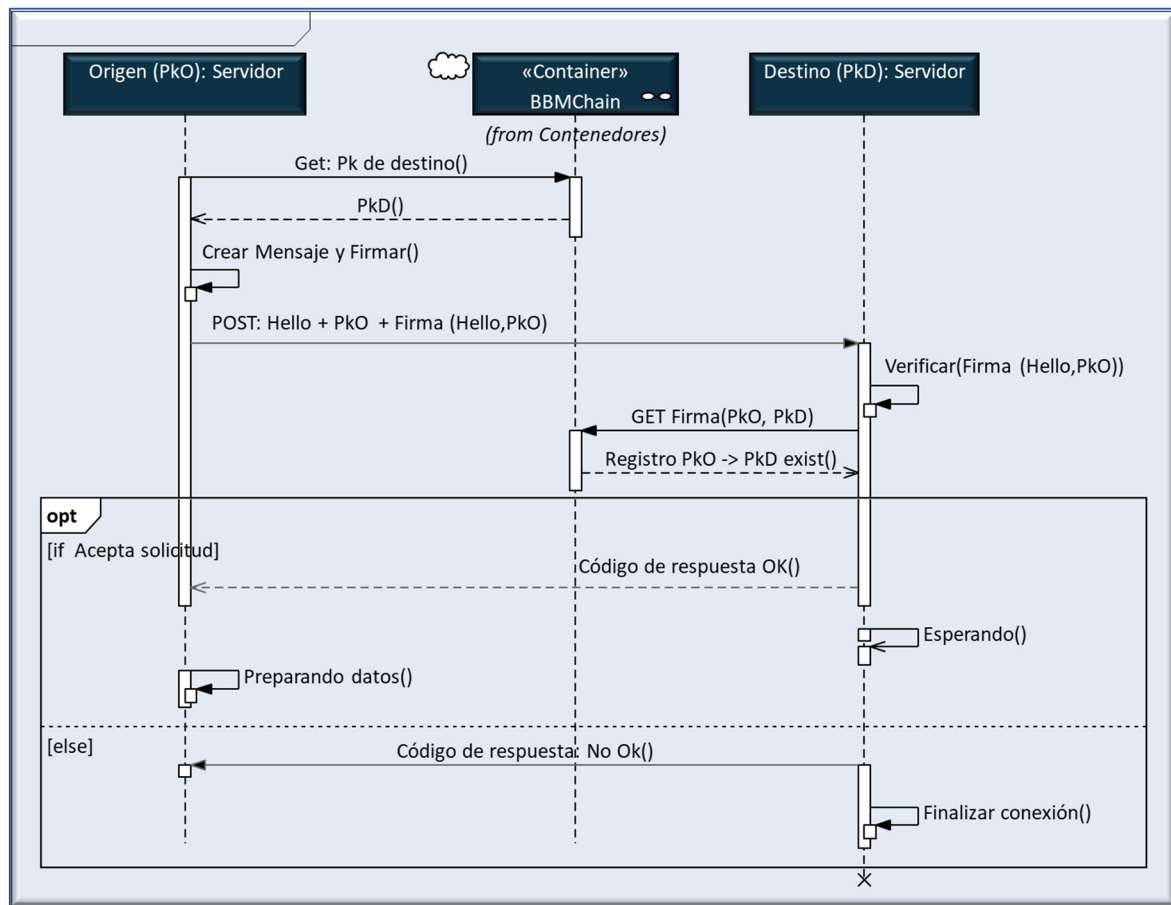
En lo relativo a los aspectos críticos de seguridad de PRECS, se comenta lo siguiente:

- Δ La solicitud solo puede ser realizada por un servidor registrado en BBM y no es posible suplantarla debido al uso de la firma en el mensaje de petición, el conocimiento previo de las claves públicas entre los servidores y el uso del protocolo PILS para generar confianza entre ellos.
- Δ En caso de que un tercero no autorizado intercepte los mensajes de solicitud y respuesta de este protocolo, no puede obtener información referente a las cuentas y correos, por lo tanto, se garantiza la confidencialidad del contenido de los correos.
- Δ No es posible modificar los contenidos de los correos ni los mensajes sin ser identificado. El uso de las firmas garantiza la integridad de los mensajes.
- Δ No se ponen en peligro las claves durante el intercambio de los mensajes debido a que no hay transporte de claves privadas. Las firmas son generadas desde los entornos seguros de las billeteras de cada servidor.
- Δ Las cabeceras de los correos pueden ser interpretadas por el servidor solicitante ya que fueron preparadas durante la ejecución del protocolo PEC.

4.16. Protocolo de Autenticación de Servidor (PAS)

La autenticación que se muestra en el diagrama del protocolo PRECS se basa en la misma filosofía utilizada en esquemas de autenticación conocidos, como TLS o similares, pero en BBM la negociación de las claves se realiza a nivel de *blockchain*, por lo tanto, la clave pública ya es conocida antes del establecimiento de la conexión, permitiendo que tempranamente pueda iniciarse una conexión segura. La Figura 84 muestra el esquema de autenticación utilizado en BBM que se utiliza durante el protocolo PRECS y que se denomina Protocolo de Autenticación de Servidor (PAS).

Figura 84. Diagrama del protocolo PAS.



Fuente: Elaboración propia.

Los aspectos para resaltar del protocolo PAS son los siguientes:

- ✎ “Get: Pk de destino” representa la obtención de la clave pública desde BBMChain y de la localización del servidor destino.
- ✎ “Crear mensaje y firmar” indica la creación del mensaje que será enviado al servidor destino. Incluirá un saludo “Hello” el envío de la clave pública del servidor origen PkO y la firma del mensaje Hello y PkO. En caso de requerir una conexión cifrada, se puede enviar una clave simétrica cifrada con la clave PkD.
- ✎ “Verificar (Firma (Hello, PkO))” representa en el servidor destino la recepción del mensaje y verificación de la firma.
- ✎ “GET Firma (PkO, PkD)” indica la consulta de los datos del servidor origen en BBMChain. Previamente los servidores han ejecutado el protocolo PILS para intercambiar datos.
- ✎ “Registro PkO -> PkD exist()” representa la respuesta de BBMChain indicando si existe dicho registro entre servidores.

- ✎ “Código de respuesta OK()” indica la respuesta positiva del servidor para el servidor origen aceptando la conexión.
- ✎ “Esperando” indica que el servidor destino se queda esperando datos provenientes desde el servidor origen.
- ✎ “Preparando datos” representa la preparación de los datos que serán enviados al servidor destino. La conexión se queda establecida hasta que se cierra la conexión.
- ✎ “Código de respuesta: No Ok()” indica una respuesta negativa en caso de que el servidor destino no acepte la solicitud.
- ✎ “Finaliza conexión” cierra la conexión entre los dos servidores.

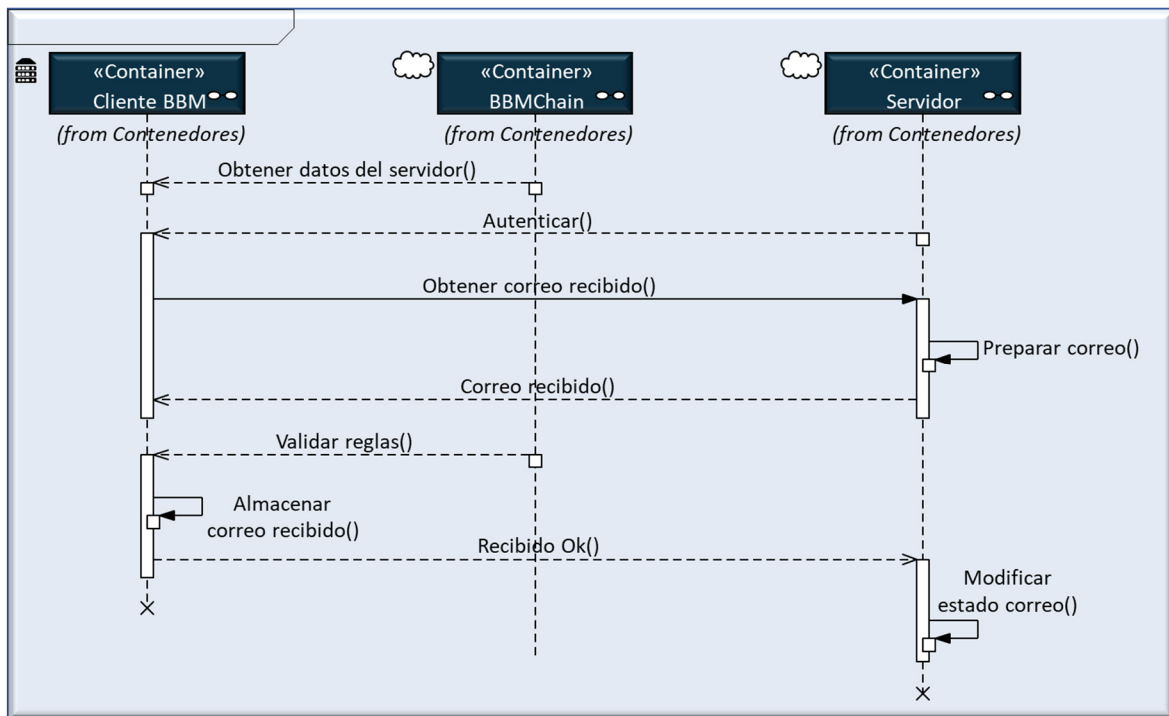
En cuanto a los aspectos críticos de seguridad de PAS, se comenta lo siguiente:

- ✎ El protocolo PAS es un protocolo de autenticación basado en clave pública.
- ✎ El conocimiento previo de las claves permite que desde el primer momento toda la comunicación pueda ir cifrada y sea muy difícil interceptar el mensaje y realizar modificaciones.
- ✎ La integridad de los mensajes está asegurada, ya que todos los mensajes van firmados.

4.17. Protocolo de Recepción de Correo (PRC)

La recepción de correo en BBM es un proceso que parte del cliente BBM hacia el servidor del destinatario. Durante el proceso, el cliente BBM solicita al servidor de la cuenta todos los correos recibidos. El servidor retorna todos los correos en estado “recibido” y los pasa a estado “leído”. El proceso de recepción de correo es simple debido a que el servidor tiene ya preparados todos los correos y validados. No obstante, como el cliente es un agente de gestión, también debe realizar las comprobaciones respectivas antes de que el usuario pueda leerlos en la bandeja de entrada del cliente BBM. La Figura 85 muestra un diagrama de secuencia con los mensajes mínimos que deben ser intercambiados por los participantes de la comunicación.

Figura 85. Diagrama del protocolo PRC.



Fuente: Elaboración propia.

Los aspectos para resaltar del Protocolo de Recepción de Correo (PRC) son los siguientes:

- ✎ No se representa en el diagrama la autenticación del usuario en el cliente BBM porque se ha explicado en otros protocolos.
- ✎ “Obtener datos del servidor” indica la consulta realizada por el cliente a BBMChain de los datos de localización del servidor. Este dato está almacenado en el campo ***_BRPrivateLocalization*** del registro del buzón de la cuenta a consultar.
- ✎ “Autenticar”, proceso de autenticación que utilizará un esquema de clave pública, ya que antes del establecimiento de la comunicación, los dos conocen sus respectivas claves públicas y eso facilita el procedimiento.
- ✎ “Obtener correo recibido” representa el llamado desde el cliente al servidor solicitando el correo recibido para una cuenta especificada como parámetro en el llamado.
- ✎ “Preparar correo” indica el proceso de consulta realizado por el servidor para obtener todos los correos dirigidos a la cuenta solicitada. Estos son todos los correos recibidos y verificados por el protocolo **PRECS**.
- ✎ “Validar reglas” indica que el cliente BBM debe confirmar que se cumplen los controles de seguridad, contactos, registro de seguridad etc. Aunque no aparece descrito en el diagrama,

el resultado negativo de la verificación indicaría que todos los agentes anteriores fallaron la validación y debería ser denunciado en BBM dicho evento. Esto es útil en el contexto de un sistema de clasificación o *ranking*.

- ✎ “Almacenar correo recibido” indica que el cliente BBM almacena de forma local en la Base de datos local los correos recibidos.
- ✎ “Recibido Ok” representa la confirmación de parte del cliente BBM de la recepción positiva de todos los correos.
- ✎ “Modificar estado de correo” representa el cambio que realiza el servidor de los correos entregados al cliente BBM, cambiando el estado de dichos correos de “recibido” a “leído”. El estado “leído” significa que el cliente BBM ha recibido el correo del servidor y no significa realmente que el usuario haya leído el correo, sino que lo tiene disponible para lectura. Esta característica es un requisito general de control de los usuarios y es de gran utilidad y uso en algunas aplicaciones específicas del correo electrónico.

En cuanto a los aspectos críticos de seguridad de PRC, se remarca lo siguiente:

- Δ Todos los mensajes intercambiados entre los agentes utilizan la firma para garantizar la integridad de la información que intercambian. Las solicitudes y respuestas en este protocolo siguen ese mismo principio.
- Δ El servidor solo entregará el correo al poseedor de la clave privada de la cuenta involucrada en la obtención del correo. La presentación de la firma en todos los mensajes enviados y recibidos garantiza que las dos partes confíen en la comunicación.
- Δ La utilización de BBMChain da solidez al protocolo, ya que garantiza la disponibilidad de la información necesaria para las validaciones y verificaciones de seguridad.
- Δ Los datos del campo ***_RC+*** del correo están cifrados con la clave pública de la cuenta del destinatario. Este cifrado fue realizado por el servidor antes de pasar el correo a estado “recibido”. El cliente BBM debe validar todos los contactos antes de pasar el correo al usuario. Como es necesario descifrar la cabecera, el cliente BBM lo descifra cuando el usuario abre el correo y, por lo tanto, no se realiza cuando el correo es recibido. La única validación que realiza el cliente BBM antes del descifrado es la validación de las firmas de los servidores.
- Δ Aunque no se muestra en el diagrama, todas las notificaciones deben ser gestionadas por todos los agentes de transporte del correo (Cliente BBM, Servidor, BBMChain). Los agentes

deben consultar si están activadas en los registros de seguridad las notificaciones y proceder con ellas. En la práctica las notificaciones son mensajes que se intercambian los servidores para anunciar cualquier cambio de estado de los correos.

5. Otros protocolos

En esta sección se han descrito los protocolos más destacados y la filosofía de seguridad propuesta con BBM. Hay una serie de protocolos en BBM que son complementarios o que realizan tareas muy específicas para proporcionar control o usabilidad a BBM. Estos protocolos siguen la misma filosofía de seguridad de los protocolos descritos hasta ahora, pero tienen objetivos distintos o complementan todos los anteriores protocolos. En este apartado solamente se describirán con el objetivo de mostrar su utilidad, pero con lo descrito hasta aquí son de fácil implementación y se considera que no es necesaria una descripción más detallada de los mismos ya que no aporta más al desarrollo de este trabajo.

5.1. Protocolo de Certificación de Identidad (PCI)

Cuando una cuenta solicita contacto a otra cuenta y la persona propietaria de la cuenta es desconocida, el usuario propietario de la cuenta solicitante podría requerir una certificación de identidad. Durante el proceso de creación del contacto, el campo ***_R/c*** puede ser modificado por el destinatario para indicar que el solicitante debería añadir en ***_AD*** datos adicionales que faciliten la identificación del usuario. En dicho campo es factible añadir una referencia IPFS a un fichero con los datos de un certificado de identidad. Igualmente, se podría crear un protocolo de certificación basado en las claves del usuario de BBM donde una CA registrada en BBM firme los certificados. Dicha CA podría ser una entidad local como una notaría o una entidad internacional que proporcione dicho servicio.

En general, el proceso se desarrollaría de la siguiente manera:

1. Una entidad se registra en BBM como CA y publica dicho registro de forma pública.
2. Los usuarios que confían en dicha entidad solicitan en BBM un certificado de identidad de persona física BBM, es aquí donde es útil el Protocolo de Certificación de Identidad (PCI).
3. Cuando una cuenta solicita contacto a otra, puede añadir el certificado de identidad BBM, que es la referencia a dicho registro en el CA.

El protocolo PCI debería ejecutarse con los siguientes pasos:

- 1.** El usuario realiza el registro en la CA utilizando BBM, sus datos personales y adjuntando la firma. Los datos personales deberían estar cifrados para que solo puedan ser interpretados por la CA.
- 2.** La CA de forma externa a BBM descifra y comprueba los datos del usuario y, cuando ha verificado la validez de estos, firma el registro dando conformidad a los mismos. La verificación de los datos la debería hacer la CA de forma externa a BBM.
- 3.** Dicho registro puede ser utilizado por el usuario como prueba de identidad dentro de BBM.

5.2. Protocolo de Recuperación y Bloqueo de Cuenta (PRBC)

Una de las grandes preocupaciones de los usuarios es la pérdida de su cuenta y, por consiguiente, de los datos. A nivel de cuentas personales, en BBM esto solo podría suceder cuando el usuario olvida su clave privada o la pierde. En cuentas corporativas, la recuperación de una cuenta o su bloqueo hace uso de las cuentas de gestión, por lo tanto, modificando la cuenta de gestión asociada y bloqueando dicha cuenta se puede recuperar fácilmente.

5.3. Protocolo de Acceso a Terceros (PAT)

Dar acceso a un tercero es una característica que entraña riesgos, sobre todo, si ese acceso no puede ser restringido de forma temporal. Hay dos formas de lograr este objetivo con BBM sin que se ponga en riesgo la seguridad de la cuenta.

Una opción es crear dentro del registro de seguridad la opción de compartir los datos entrantes o salientes, así todos los correos pueden ser redireccionados por el cliente BBM a dicha cuenta. Para cancelar el acceso del tercero a dichos correos, se modifica el registro de configuración. La aproximación anterior daría acceso a los correos a partir de una fecha y hora determinada, pero no podría acceder a los correos anteriores.

La otra aproximación a este protocolo es la creación de cuentas de *backup*, donde se puede crear un nuevo usuario y replicar todo el contenido a dicha cuenta. Esta solución podría ser utilizada para auditorías o seguimiento a cuentas específicas. Este tipo de protocolos solo podría ser utilizado con cuentas corporativas ya que es la única forma con BBM de lograr este tipo de control, gestión y auditoría.

5.4. Protocolo de Autorización Grupal (PAG)

Las listas de distribución son un requisito importante de los usuarios dentro de una solución de correo electrónico, ya que permite el envío de correo electrónico a un grupo de usuarios en lugar del envío individual de cada correo uno a uno. Dada la filosofía de BBM, las listas como son utilizadas usualmente presentan algunos problemas de seguridad ya que los usuarios deben percatarse de los aspectos de seguridad relacionados con sus contactos. En BBM, el enfoque utilizado para este tipo de requisito es el mismo que para los contactos, todo usuario debe estar plenamente consciente con quién se comunica y qué información comparte.

Una lista de distribución se presenta en BBM como un tipo de contacto que requiere autorización grupal, es un contacto creado por un usuario donde el resto se unen por medio de autorizaciones individuales. Un usuario crea un contacto en una cuenta de tipo grupal, el resto de los usuarios deben solicitar contacto para poder enviar correo a la lista; una vez autorizado el contacto, la cuenta grupal debe solicitar contacto para poder enviarle correo desde esa cuenta grupal. Con este proceso se tiene la autorización mutua y se puede enviar y recibir correo desde el contacto grupal. Los correos que se transmitan por ese medio nunca expondrán las cuentas individuales, ya que sería necesaria la autorización individual en ese caso.

A nivel de servidor, los contactos grupales se manejarían como cualquier otro contacto, recibiendo y enviando correo desde la cuenta propietaria del contacto a todos los contactos registrados, siendo la cuenta grupal el centro de dicha distribución de correos.

5.5. Protocolo de Supervisión y Autorización de Cuentas (PSATC)

Haciendo uso del esquema de jerarquía de cuentas y cuentas de gestión, se pueden controlar las cuentas a muchos niveles. Se ha visto que un usuario puede participar en una comunicación de correo bajo diferentes roles, pudiendo ser remitente, autor o autorizador en algunos casos. El aspecto de usabilidad es un requisito identificado indirectamente como requisito de seguridad en este trabajo, ya que como se ha descrito anteriormente, la seguridad empieza a perder su eficacia cuando la usabilidad de la solución es compleja o demasiado desconocida para el usuario.

Se ha descrito como, mediante la autoría de un correo, un usuario puede delegar el envío de un correo a otro, con la única intervención de su firma. Pero si el autor desea delegar el envío de los correos a una persona de su confianza sin necesidad de tener que firmar cada correo (esto es algo muy común en el mundo del correo electrónico), la solución de autoría en BBM presenta un problema directo de usabilidad. No obstante, BBM está planteado desde sus inicios para dar solución a esta casuística y se incluye como uno de los requisitos en el diseño de la arquitectura.

En BBM, se puede crear un nuevo tipo de contacto, denominado contacto de autorización. Esta solución permite que un usuario que aparece como autor previamente haya autorizado por medio de ese tipo de contacto especial la autoría de sus correos. Cuando en una comunicación de correo, un autor aparezca como contacto de autorización, la firma que debe solicitarse no es requerida ni validada por los servidores, solo se valida el tipo de contacto. Lo único que debería de hacerse es enviar una copia del correo, y no un correo en espera de firma como el que se envía al resto de autores. Es decir, dicho autor se comportaría como un destinatario más del correo, para que pueda realizar las labores de supervisión necesarias.

5.6. Protocolo de Borrado de Correo (PBC)

Aunque BBM es una solución basada en *blockchain*, el correo es almacenado en una red independiente. Dicha red puede ser implementada de muchas formas, pero uno de los objetivos es que el correo pueda eliminarse. En el Protocolo de Borrado de Correo (PBC) hay que tener en cuenta los registros de seguridad asociados a los usuarios cuyos campos *_ReqDelAuth*, *_DeleteAuth*, *_DeleteNotificacion* y *_LetDelete* están directamente relacionados con este protocolo.

El administrador de una cuenta puede activar los campos del registro de seguridad, lo que significa que este protocolo debe consultarlos activamente bajo las siguientes reglas:

- ➔ Si *_LetDelete* está activado, el correo de esa cuenta puede ser borrado. En caso contrario, el servidor no ejecutaría dicha acción y el correo lo almacenaría permanentemente hasta que se cumplan otras políticas fuera de BBM que permitan su borrado.
- ➔ Los campos *_ReqDelAuth* y *_DeleteAuth* están relacionados con la participación del administrador y el gestor respectivamente en el proceso de borrado. Si no se puede borrar el correo, los valores que tengan estos campos no aplican.
- ➔ El campo *_ReqDelAuth* indica al servidor como debe tratar la autorización del borrado. Si está activado, debe tenerse en cuenta que el administrador participa en la decisión del borrado.
- ➔ El campo *_DeleteAuth* indica la participación del gestor en el proceso de borrado. Si está activado, el gestor debe tomar parte de la decisión de forma pasiva o activa dependiendo del valor de *_ReqDelAuth*.
- ➔ Si el campo *_ReqDelAuth* no está activado y *_DeleteAuth* está activado, el gestor necesita autorizar el borrado. Un correo de autorización de borrado debería ser enviado desde el

servidor al gestor y es este quien decide el destino de dicho correo. Si el servidor recibe la autorización, procede al borrado.

- ➔ Si el campo ***_ReqDelAuth*** está activado y ***_DeleteAuth*** no está activado, el gestor no participa y es el administrador quien lo hace. Un correo de autorización de borrado debería ser enviado desde el servidor al administrador y es este quien decide el destino de dicho correo. Si el servidor recibe la autorización, procede al borrado.
- ➔ Si los dos campos ***_ReqDelAuth*** y ***_DeleteAuth*** están activados, el gestor y el administrador participan en la decisión del borrado. Un correo de autorización de borrado debería ser enviado desde el servidor al administrador y al gestor. El administrador y el gestor firman la autorización decidiendo el borrado de dicho correo. Si el servidor recibe las dos autorizaciones, procede al borrado.
- ➔ Por último, el campo ***_DeleteNotificacion*** indica la activación de notificaciones cuando un correo es borrado en el servidor.

5.7. Protocolo de Reenvío de Correo (PREC)

Reenviar correo implica el envío de un correo cuya autoría es de un tercero. Si el tercero lo autoriza, el correo se podría enviar sin problemas. El proceso se complica cuando ese tercero no quiere que su correo sea compartido. El autor, cuando firma un correo, el autoriza el contenido y también los contactos a los que va dirigido. Si cambia algo en el contenido del correo o en las RC originales, el autor puede no estar de acuerdo y debería ser requerida su autorización.

Dentro de BBM, los autores pueden autorizar el reenvío de un correo. Para poder saber si un correo puede ser reenviado, se hace uso del campo ***_LetReSend*** dentro del registro de seguridad de los autores. Si dicho campo está activado, un usuario destinatario podría realizar el reenvío sobre los correos que recibe de dichos autores sin requerir autorización alguna. El registro También contiene el campo ***_ReqReSendAuth***, que indicará si es necesario la autorización para reenviar un correo por parte de los autores. Esta autorización debe ser gestionada por el cliente BBM y por los servidores que actúan en dicha comunicación.

Una vez autorizado el reenvío, para hacer llegar el correo hasta un nuevo destinatario es necesario que se cumpla lo siguiente:

- 1.** Todos los autores deben tener activado el campo ***_LetReSend.***
- 2.** Los destinatarios deberá ser contactos autorizados de los autores.

Con el cumplimiento de estas reglas se seguirán las reglas de BBM y el correo enviado seguirá las mismas reglas que un correo normal. Si el usuario destinatario quiere reenviar el contenido de un correo sin utilizar el protocolo de reenvío, puede ser realizarlo copiando el contenido y creando un correo normal, pero dicho correo no será un correo autorizado por los autores y se añadirán nuevos autores al correo, creando un problema de confidencialidad fuera del alcance de BBM.

5.8. Protocolo de Correo Certificado (PCCR)

Dentro del protocolo PRC se describió como el cliente BBM recibía el correo y esté pasaba a estado “leído”. Como se comentó, dicho estado significa que el correo está disponible para el usuario, pero no significa que lo haya leído.

Con BBM, lograr confirmar la recepción consciente del correo por parte del destinatario puede ser logrado con el Protocolo de Correo Certificado (PCCR). Esta propuesta de correo certificado se basa en añadir un campo dentro de la cabecera del correo que va dirigido al servidor del destinatario.

Una vez el servidor reciba un correo del que debe certificarse la recepción, el servidor envía la cabecera del correo sin cuerpo al cliente BBM y solicita la firma de recepción. Cuando el usuario abra el correo, firma la cabecera y la devuelve al servidor. Dicha respuesta llegará al servidor y éste, al verificar la firma, enviará el correo al cliente BBM marcándolo como “leído”. Todo el proceso de notificaciones se activará hasta que la confirmación de recepción del destinatario llegue al remitente.

5.9. Protocolo de Envío de Correo SMTP (PECSMTP)

La integración con el envío de correo SMTP es una necesidad de cualquier plataforma de correo que no haga uso de éste, como es el caso de BBM. BBM posee una filosofía de seguridad muy marcada que dificulta la integración de correo SMTP con las mismas garantías de seguridad.

BBM no es una capa que proporcione seguridad a la capa de SMTP, pero se puede lograr añadir cierta seguridad como se explica en la Figura 48 o haciendo uso de BBM utilizando clientes convencionales (Figura 49).

El objetivo de añadir una capa de seguridad BBM sobre SMTP es la divulgación de BBM como solución de correo seguro, pero en este trabajo se considera que es un error ir en esa dirección dados los problemas de seguridad del protocolo SMTP que se han encontrado en este trabajo y que han sido descritos en el capítulo 2 de esta tesis.

Se propone un protocolo para la integración de BBM y SMTP basado en los siguientes puntos:

- Un usuario en propiedad de una cuenta SMTP (alice@ejemplo.com), crea en BBM una cuenta y le asocia su cuenta SMTP. Esto puede ser realizado en el registro de la cuenta en el campo *_smtpData*, que debe ser añadido para implementar este protocolo.
- BBMI debe ser configurado con los datos del servidor SMTP y POP/IMAP.
- El cliente SMTP convencional (Outlook, ThunderBird, Mozilla, etc.) utilizará BBMI como servidor entrante y saliente.
- Todos los correos entrantes y salientes pasarán por BBMI, que hará de filtro para los correos SMTP y BBM. Los correos BBM deberán cumplir las reglas BBM para poder ser recibidos o enviados.
- Un correo BBM es identificado porque el remitente (correo entrante) o destinatario (correo saliente) es un contacto BBM y tiene asociada su cuenta SMTP en BBM.
- Un correo BBM tendrá el cuerpo con el mismo formato que un correo BBM convencional.
- La cabecera deberá tener el formato SMTP para que el correo sea transmitido por los servidores convencionales.

V. ASPECTOS CLAVE DE LOS PROTOCOLOS DE BBM

Tabla 15. Aspectos clave de los Protocolos de BBM.

1/5	
Protocolos	Aspectos clave
Protocolo de Generación de Claves (PGC)	- Genera una clave privada y una pública. - Creación de billetera BBM básica (UTC/JSON). - Uso de entorno de seguridad personal (PSE) para almacenar la clave privada.
Protocolo de Registro de Servidor (PRS)	- Ejecución de contrato inteligente para almacenar registro de servidor en BBMChain. - Uso de la clave pública como identificador. - Creación de la base de datos del servidor en BBMStorage. - Datos sensibles cifrados con ACD (AES +Scrypt +HMAC).
Protocolo de Consulta de datos de registro de Servidor (PCS)	- Ejecución de contrato inteligente para consultar datos de registro de servidor. - Descifrado de datos (AES +Scrypt +HMAC). - Descripción del algoritmo de cifrado ACDC (ECC +AES-128-CTR-HMAC) y de descifrado ADDC (ECC +AES-128-CTR-HMAC).
Protocolo de Modificación de datos de Servidor (PMS)	- Ejecución de contrato inteligente para modificar datos de servidor en BBMChain. - Contrato inteligente para máquina de estados de servidor. - Uso del protocolo PCS antes de la modificación del registro.
Protocolo de Registro de Usuario (PRU)	- Ejecución del contrato inteligente para el registro de usuario. - Claves jerárquicas basadas en BIB32 y BIB44 con la estructura BBM: NodoMaestro (m), Identificadores(m/i), Cuentas (m/i/z), Buzones (mI/z/k). - Algoritmo para la generación de claves (Figura 62) para los identificadores de usuario. - Tipos de usuario personal y corporativo (admin, delegado y supervisado). - Registro de configuración creado. - Propuesta de estructura corporativa. - Algoritmo de generación de claves de gestión AGCG.

Fuente: Elaboración propia.

Tabla 15. Aspectos clave de los Protocolos de BBM.

2/5	
Protocolos	Aspectos clave
Protocolo de Modificación de datos de Usuario (PMU)	⌂ Ejecución de contrato inteligente para modificar datos de usuario. ⌂ Contrato inteligente para máquina de estados de usuario. ⌂ Descripción del registro de seguridad. ⌂ Modificación de registro de seguridad de usuario por usuario gestor.
Protocolo de Creación de Cuentas (PCC)	⌂ Ejecución de contrato inteligente para creación de cuentas. ⌂ Generación de identificador de cuenta. ⌂ Propietario de la cuenta oculto por medio de cuentas de gestión.
Protocolo de Modificación de Cuentas (PMC)	⌂ Ejecución de contrato inteligente para modificación de cuentas. ⌂ Contrato inteligente para máquina de estados de cuentas.
Protocolo de Creación de Buzones (PCB)	⌂ Ejecución de contrato inteligente para creación de buzones. ⌂ Relación directa con la cuenta, no se utilizan cuentas de gestión. ⌂ Datos de localización del buzón cifrado por servidor.
Protocolo de Modificación de Buzón (PMB)	⌂ Ejecución de contrato inteligente para modificación de buzón y para máquina de estados.
Protocolo de Registro de Cuenta en el Servidor (PRCS)	⌂ Ejecución de contrato inteligente para registro de cuentas en servidores. ⌂ Uso de firma para autorización del servidor. ⌂ Modificación del buzón en la aprobación del registro.
Protocolo de Intercambio de Localización entre Servidores (PILS)	⌂ Ejecución de contrato inteligente para registro de autorización de intercambio de datos. ⌂ Uso de firma digital para confirmar intercambio. ⌂ Uso de ACDC y de ADDC para el intercambio de datos en el registro. ⌂ No necesario este protocolo en caso de red de almacenamiento basada en cadena de bloques privada.

Fuente: Elaboración propia.

Tabla 15. Aspectos clave de los Protocolos de BBM.

3/5	
Protocolos	Aspectos clave
Protocolo de Solicitud y Autorización de Contacto (PSAC)	⌂ Ejecución de contrato inteligente para creación de contacto. ⌂ Creación de solicitud en una sola dirección. ⌂ Uso de identificador basado en jerarquía de claves bajo el primer buzón (m/i/z/0/x). ⌂ Identificación pública del contacto destino de la solicitud solamente. ⌂ Control por registro de seguridad. ⌂ Uso de marca de tiempo para expiración. ⌂ Token criptográfico de autorización con uso de firma y algoritmos ACDC y ADDC. ⌂ Posibilidad de uso de certificado para identidad. ⌂ Servidor depositario público. ⌂ Ejecución de contrato inteligente para máquina de estados de contacto.
Protocolo de Rechazo de Contacto (PRZC)	⌂ Ejecución de contrato inteligente para rechazo de contrato. ⌂ Uso del campo <i>_Rjc</i> and <i>_AD</i> para información adicional. ⌂ Ejecución de contrato inteligente para máquina de estados de contacto.
Protocolo de Cambio de Estado de Contacto (PCEC)	⌂ Ejecución de contrato inteligente para modificación de contacto. ⌂ Se usa el mismo esquema que para los otros protocolos de modificación de datos en BBMChain.
Protocolo de Envío de Correo (PEC)	⌂ Ejecución de contratos inteligentes para verificar reglas y prerequisites que deben cumplir usuarios y servidores para depositar o rechazar correo. ⌂ Concepto de depósito de correo en lugar de envío. ⌂ Distinción de remitente, autorizador y autor. ⌂ Uso de ACDC para el cifrado del cuerpo del correo. ⌂ Cabecera cifrada con ACDC desde el remitente para el servidor. ⌂ Uso de multi correo para firmas de autorización y autoría. ⌂ Validación de tokens de los contactos para todas las relaciones de comunicación. ⌂ Máquina de estado controlada por el cliente BBM y los servidores BBM. ⌂ Almacenamiento de correos fuera de la cadena de bloques (BBMChain) o en una cadena de bloques privada de la red de proveedores de almacenamiento.

Fuente: Elaboración propia.

Tabla 15. Aspectos clave de los Protocolos de BBM.

4/5	
Protocolos	Aspectos clave
Protocolo de Recepción de Correo en Servidor (PRECS)	⌂ Ejecución de contratos inteligentes para verificar reglas y prerequisites que deben cumplir usuarios y servidores para depositar o rechazar correo. ⌂ Solicitud de correo por parte del destinatario basado en los usuarios registrados. ⌂ Solicitudes de mensajes firmados. ⌂ Uso de ADDC para el descifrado de cabeceras por parte del servidor del destinatario. ⌂ Uso del sub-protocolo de autenticación del servidor (PAS).
Protocolo de Recepción de Correo (PRC)	⌂ Ejecución de contratos inteligentes para verificar reglas y prerequisites que deben cumplir usuarios y servidores para recibir o rechazar el correo. ⌂ Validación de las relaciones de comunicación en BBMChain. ⌂ Uso de ADDC para el descifrado del cuerpo del correo.
Protocolo de Certificación de Identidad (PCI)	⌂ Ejecución de contrato inteligente para registro de CA en BBM. ⌂ Ejecución de contrato inteligente para solicitud de certificado al CA en BBM. ⌂ Ejecución de contrato inteligente para almacenamiento del certificado de identidad público en BBM. ⌂ Ejecución de contrato inteligente para la modificación de contacto cuando se solicite identificación.
Protocolo de Recuperación y Bloqueo de Cuenta (PRBC)	⌂ Regeneración de claves en la billetera BBM. ⌂ Ejecución de contrato inteligente para la modificación de cuentas de gestión y bloqueo de cuentas de gestión comprometidas.
Protocolo de Acceso a Terceros (PAT)	⌂ Ejecución de contrato inteligente para modificación del registro de seguridad para redireccionamiento de correo a otra cuenta. ⌂ Ejecución de contrato inteligente para añadir cuentas de auditoría paralelas donde se envía todo el correo de una cuenta específica.

Fuente: Elaboración propia.

Tabla 15. Aspectos clave de los Protocolos de BBM.

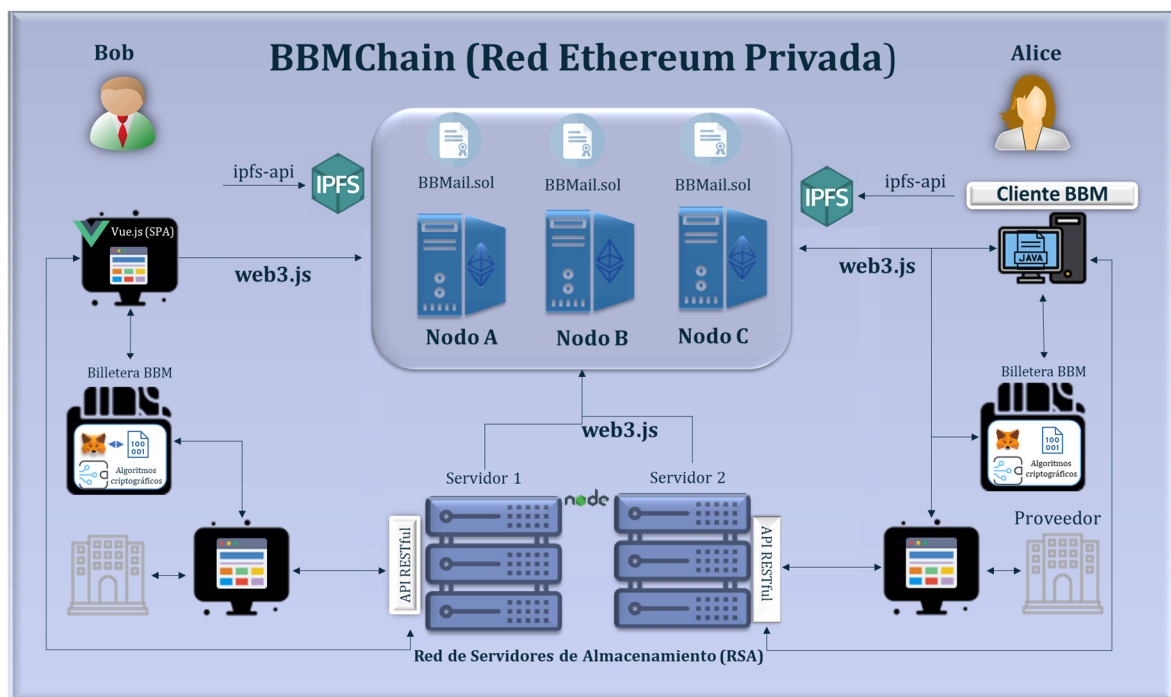
5/5	
Protocolos	Aspectos clave
Protocolo de Autorización Grupal (PAG)	⌚ Ejecución de contrato inteligente para la creación de contactos de grupo donde otras cuentas pueden solicitar contacto y viceversa.
Protocolo de Supervisión y Autorización De Cuentas (PSATC)	⌚ Ejecución de contrato inteligente para la creación de contactos de autorización. ⌚ Utilizado para la optimización del correo autorizando previamente el envío de los correos con el rol de autor a un originador remitente.
Protocolo de Borrado de Correo (PBC)	⌚ Control desde el gestor de la cuenta para la eliminación de correo por registro de seguridad almacenado en BBMChain.
Protocolo de Reenvío de Correo (PREC)	⌚ Control desde el gestor de la cuenta para el reenvío de correo por registro de seguridad almacenado en BBMChain. ⌚ Autorización de reenvío de correo con solicitud de firma por medio del registro de seguridad. ⌚ Verificación de contactos desde los nuevos remitentes.
Protocolo de Correo Certificado (PCCR)	⌚ Responsabilidad de los servidores de BBM con el control de entrega de correo previa solicitud de firma al destinatario. ⌚ Ejecución del protocolo PEC.
Protocolo de Envío de Correo SMTP (PECSMTP)	⌚ Integración de BBM con SMTP haciendo uso del Integrador BBM. ⌚ Ejecución de contratos inteligentes para el control de las relaciones de comunicación en las cabeceras de los correos. ⌚ Enrutamiento del correo a partir de las reglas del Integrador BBM.

Fuente: Elaboración propia.

VI. BBMAIL: PRUEBA DE CONCEPTO BASADA EN BBM

BBMail es una dApp basada en BBM que cumple con los objetivos de seguridad marcados en este trabajo y permite validar la factibilidad del uso de las tecnologías *blockchain* en el desarrollo e implementación de los protocolos de seguridad. En BBMail, se desarrollaron todos los componentes necesarios para la ejecución de los protocolos descritos en la sección anterior utilizando *blockchain* Ethereum como componente BBMChain, una aplicación SPA como cliente BBM *web* y, para la red de servidores de almacenamiento, se proporciona una API REST con tecnología *node.js* y MongoDB como repositorio de los correos. Se utilizan las librerías de *web3.js* para la ejecución de los contratos y diferentes librerías criptográficas *java* para proporcionar toda la capa de seguridad necesaria en los principales protocolos implementados. La Figura 86 muestra un esquema conceptual de la arquitectura BBMail y su correspondencia con la arquitectura BBM.

Figura 86. Arquitectura de BBMail.



Fuente: Elaboración propia.

Se decidió desarrollar el cliente BBM en dos versiones, una versión *web* y otra de escritorio, con el objetivo de comparar el desarrollo de este tipo de solución en entornos más o menos restrictivos. El caso de las aplicaciones *web* presentaba un reto por la poca seguridad y restricciones de librerías que de por sí presenta este tipo de entorno. En cambio, las aplicaciones de escritorio tienen más opciones de desarrollo y más librerías disponibles y hay

muchos lenguajes a la hora de elegir para su programación. Al final se creó el cliente de escritorio utilizando Java y el cliente *web* usando el Framework Vue.js.

Se desarrolló BBMChain bajo Ethereum desplegando una red de tres nodos privados para minimizar los costes asociados a las transacciones (*gas*) por el uso de esta red pública. Para la lógica de BBMChain, se crearon diferentes contratos inteligentes (BMail.sol) utilizando el lenguaje de programación Solidity. Dichos contratos exponían todos los métodos requeridos por los diferentes protocolos para su ejecución. Para la interacción con BBMChain, se utilizó la librería web3.js que permite la ejecución de los contratos inteligentes y la consulta de sus datos. Se creó una estructura de contratos utilizando herencia de clases y diferentes interfaces para solventar los problemas de almacenamiento debido a que los contratos en Ethereum tienen una limitación de 24kb y el código a desarrollar superaba ampliamente esa limitación.

Para realizar las funciones de billetera BBM, se utilizó *Metamask* para la gestión y generación de las claves, pero fue necesario añadir más algoritmos criptográficos y ampliar la billetera incorporando datos específicos de BMail. Se desarrolló todo el proceso de cifrado y descifrado fuera del entorno de Metamask porque éste no suministraba dichos algoritmos. Para proporcionar toda la capa criptográfica (cifrado ECIES, firma keccak256, cifrado simétrico AES, *hash* y funciones KDF para la generación de claves), se utilizaron diferentes librerías como son *eth-crypto*, *ciesjs*, módulo *crypto-node.js*, *smartbox-ecies-java*, entre otras. Se almacenaron todos los datos de la billetera utilizando el formato UTC/JSON y añadiendo datos adicionales específicos de algunos protocolos de BMail, como los datos de generación de las claves de las cuentas de gestión, entre otros.

Se desarrollaron los servidores de almacenamiento utilizando node.js para exponer las diferentes API por medio de una *API RESTful* que proporcionaba toda la funcionalidad para la interacción de los clientes BBM y de los proveedores, que son los que registran los servidores. Para almacenar toda la información de los servidores, se usó *MongoDb*, que proporciona opciones de alta disponibilidad mediante el uso de particiones (*sharding*) y replicación de la base de datos. Aunque los servidores proporcionan la funcionalidad de *backend*, se añadió la funcionalidad de *frontend* para no tener que desplegar otro servidor *web* y, para las pruebas, dicha capa fue utilizada por los dos roles de usuario y proveedor. Para la ejecución de las operaciones criptográficas (*hashing*, firma, cifrado, etc.), se crearon ficheros de configuración con las claves necesarias.

En cuanto a la funcionalidad, se creó el núcleo de BMail y los protocolos más importantes para realizar todas las pruebas. La Tabla 16 lista las funcionalidades implementadas en BMail y su correspondencia con los protocolos descritos en esta tesis.

Tabla 16. *Funcionalidad BBMail.*

Actor	Funcionalidad	Protocolo implementado
Usuario	Generación de claves	 Protocolo de Generación de Claves (PGC)
	Autenticación	 Protocolo de Generación de Claves (PGC)
	Gestión de usuario	 Protocolo de Registro de Usuario (PRU)
		 Protocolo de Modificación de Datos de Usuario (PMU)
	Gestión de cuentas	 Protocolo de Creación de Cuentas (PCC)
		 Protocolo de Modificación de Cuentas (PMC)
	Gestión de buzones	 Protocolo de Creación de Buzones (PCB)
		 Protocolo de Modificación de Buzón (PMB)
	Registro de cuenta en un servidor	 Protocolo de Registro de Cuenta en el Servidor (PRCS)
	Solicitud de contacto	 Protocolo de Solicitud y Autorización de Contacto (PSAC)
	Modificación de contacto	 Protocolo de Cambio de Estado de Contacto (PCEC)
	Creación de correo y envío	 Protocolo de Envío de Correo (PEC)
	Recepción de correo	 Protocolo de Recepción de Correo (PRC)
	Borrado de correo	 Protocolo de Borrado de Correo (PBC)
	Reenvío de correo	 Protocolo de Reenvío de Correo (PREC)
	Autorización de correo como Autor	 Protocolo de Envío de Correo (PEC)
Servidor	Generación de claves	 Protocolo de Generación de Claves (PGC)
	Gestión datos servidor	 Protocolo de Registro de Servidor (PRS)
		 Protocolo de Consulta de datos de registro de Servidor (PCS)
		 Protocolo de Modificación de datos de Servidor (PMS)
	Registro entre servidores	 Protocolo de Intercambio de Localización entre Servidores (PILS)
	Recepción de correo en servidor	 Protocolo de Recepción de Correo en Servidor (PRECS)

Fuente: Elaboración propia.

La Tabla 17 describe los aspectos de seguridad alcanzados con la implementación de los protocolos de BBM en BBMail. Aparecen listados los datos sensibles que se manejan en *BBMail* y cómo se ha logrado cada uno de los aspectos de seguridad objetivos de esta tesis.

Las claves constituyen la base de la seguridad de toda la solución, estando éstas en posesión, guardia y custodia del usuario. Se observa como la confidencialidad de los registros (usuario, cuentas, buzones, contactos, servidores, solicitudes) que se utilizan en la mayoría de los protocolos es lograda con el uso de algoritmos simétricos (*AES*) y algoritmos asimétricos híbridos (*ECIES*) y su almacenamiento en *blockchain* (*BBMChain*). Con *BBMail*, no solamente se logra la confidencialidad a nivel de los registros y correo, sino también a nivel de las relaciones de comunicación (contactos) cuando el correo es transportado. En caso de que un tercero intercepte los datos transportados con el objetivo de lograr un ataque de hombre en medio, la confidencialidad está asegurada, ya que los datos sensibles van cifrados, lo mismo que la integridad gracias al uso de las firmas digitales. El uso de las jerarquías de cuentas permite ocultar las relaciones de usuario, cuentas y contactos dentro de una *blockchain* pública (*BBMChain*).

La *blockchain* nos brinda la integridad por naturaleza propia, junto con esquemas de firma específicos en algunos protocolos. El no repudio de origen y destino está presentes en todos los protocolos, que están centrados en asegurarlo con las firmas y el almacenamiento descentralizado en *BBMChain*. Otra característica notable de *blockchain* es la disponibilidad de los datos, lo que asegura que la información esté disponible para todos los protocolos. En cuanto a la disponibilidad de los correos electrónicos, queda a cargo de la red de almacenamiento de servidores. El control como elemento crítico de seguridad presente en muchos de los requisitos está centralizado en *blockchain* (*BBMChain*) a través de los registros de seguridad y las cuentas de gestión.

Con *blockchain* (*BBMChain*), se proporciona toda la trazabilidad necesaria para auditar todos los cambios de los registros de los protocolos de seguridad. El uso de cuentas de gestión y los registros de seguridad facilita la labor de seguimiento y control, permitiendo que los usuarios corporativos puedan crear complejas estructuras en la organización, realizar auditorías y control a las acciones que ejecutan usuarios más subordinados (supervisados). En la Tabla 17, se puede observar que la usabilidad se logra alejando al usuario de las complejas operaciones de seguridad, ya que los protocolos han sido creados sobre dicha base. Solamente es importante que cuando el sistema solicite obtener su clave privada para realizar la firma o el cifrado, el usuario comprenda estos dos conceptos. Finalmente, en la última columna de la tabla se describe el aspecto integración, que es logrado añadiendo un puente con *BBMI*.

Tabla 17. Aspectos de seguridad de los datos sensibles manejados en BBMail.

Datos Sensibles	Confidencialidad	Integridad	No repudio	Disponibilidad	Control	Auditoría	Usabilidad	Integración
Claves	Cifrado AES de Sk Fichero JSON (keys.json)	Mac: keccak-256	Mac: keccak-256	Usuario	Usuario	Usuario	Generación Cliente BBM	Asociación entre cuentas BBMI
Registros de usuario	Algoritmo ACD Almacenamiento BBMChain Datos para cifrado/descifrado en fichero keys.json Jerarquía de claves basada en BIB32 y BIB44 Generación de cuenta de gestión con AGCG	Firma digital AGID BBMChain	Firma digital	BBMChain	Registros de seguridad Cuentas de gestión A cargo de BBMChain	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	BBMI
Registro de servidor	Algoritmo ACD Almacenamiento BBMChain Datos para cifrado/descifrado en fichero keys.json	Firma digital BBMChain	Firma digital	BBMChain	Proveedor	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	BBMI
Cuentas	Algoritmo ACD Almacenamiento BBMChain Datos para cifrado/descifrado en fichero keys.json Jerarquía de claves basada en BIB32 y BIB44 Generación de cuenta de gestión con AGCG	Firma digital BBMChain	Firma digital	BBMChain	Registros de seguridad Cuentas de gestión A cargo de BBMChain	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	BBMI
Buzones	Algoritmo ACD Almacenamiento BBMChain Datos para cifrado/descifrado en fichero keys.json Jerarquía de claves basada en BIB32 y BIB44	Firma digital BBMChain	Firma digital	BBMChain	Registros de seguridad Cuentas de gestión A cargo de BBMChain	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	BBMI
Contactos	Algoritmo ACD Almacenamiento BBMChain Datos para cifrado/descifrado en fichero keys.json Algoritmo ACDC Origen del contacto desconocido Token de autorización	Firma digital Cuenta origen y destino BBMChain	Firma digital Cuenta origen y destino	BBMChain	Registros de seguridad Cuentas de gestión A cargo de BBMChain	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	BBMI
Datos del correo	Algoritmo ACDC Almacenamiento "offchain" RC cifrado dirigido al servidor Cuerpo del correo cifrado para originador y destinatario Localización del servidor visible entre servidores	Firma digital Todos los participantes	Firma digital Todos los participantes	Servidor	Registros de seguridad Cuentas de gestión A cargo de BBMChain	Cliente BBM Servidores	Interfaz intuitiva Conceptos de firma y cifrado Conocer la filosofía BBM Adaptación	BBMI
Datos de servidores y sus cuentas	Públicos, no hay datos sensibles Propietario de la cuenta desconocido	Firma digital Cuenta y servidor BBMChain	Firma digital Cuenta servidor	BBMChain	Registros de seguridad Cuentas de gestión A cargo de BBMChain	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	BBMI
Registro de comunicación entre servidores	Algoritmo ACDR Almacenamiento en BBMChain Localización del servidor visible por el servidor destinatario	Firma digital	Firma digital de los servidores	BBMChain	Registros de seguridad Cuentas de gestión A cargo de BBMChain	BBMChain	Interfaz intuitiva Conceptos de firma y cifrado	NA
Datos en transporte	HTTP y Json-RPC Datos sensibles cifrados	Firma digital	Firma digital	NA	NA	Cliente BBM Servidores	BBMI	BBMI

Fuente: Elaboración propia.

Pruebas realizadas para la validación de los requisitos

Para las pruebas, se preparó BBMail con la siguiente configuración:

- ✎ Se registraron dos servidores: Servidor1 y Servidor2.
- ✎ Se creó: una estructura corporativa con usuario01 como usuario “admin”.
- ✎ Se crearon: 5 usuarios más a partir del usuario01: de usuario02 a usuario06.
- ✎ Se crearon: cuentas para cada usuario: de cuenta01 a cuenta06.
- ✎ Se crearon: buzones para cada cuenta: de buzón01 a buzón06.
- ✎ “Usuario autorizado” hace referencia a un usuario creado en BBMail.
- ✎ “Usuario no autorizado” hace referencia a un usuario no creado en BBMail.
- ✎ Cliente BBM modificado es un cliente que no valida las reglas en BBMChain. Es utilizado para realizar ataques.
- ✎ Servidor BBM modificado es un servidor que no valida las reglas en BBMChain. Es un servidor utilizado para realizar ataques.

La validación de **REQ001** se realiza con la validación del resto de requisitos.

Pruebas realizadas sobre los requisitos de los buzones

REQ002 y REQ023: Validados

Prueba 1:

- 📖 Se leyeron los datos de registro de los buzones en BBMChain con un usuario no autorizado.
- 📖 Fue imposible interpretar los datos de localización del servidor almacenados en el campo ***_BRPrivateLocalization*** de los buzones al estar cifrados.

Prueba 2:

- 📖 Se leyeron los datos de registro de los buzones desde el cliente BBM por los usuarios autorizados.
- 📖 La billetera BBM de cada usuario contenía la clave privada de la cuenta propietaria del buzón y el resto de los datos (DDC+salt+pke+MAC+IV+AESkC) se encontraban en el campo ***_BRPrivateLocalization*** del registro del buzón para descifrar la localización con el algoritmo ADDC.

 Se obtuvieron los datos del servidor donde está localizado el buzón.

Prueba 3:

 Se simuló un acceso no autorizado al Servidor1.

 Se copiaron los correos almacenados en el servidor para tratar de interpretarlos.

 No fue posible interpretar la cabecera y el cuerpo de ninguno de los correos sin tener las claves privadas de los participantes y del servidor donde estaban alojados los correos.

REQ003 y REQ005: Validados

Prueba 4:

 Para el envío de un correo en BBM solo se utilizan los identificadores de las cuentas, los buzones no son utilizados.

 Utilizando el protocolo PRCS se registró cuenta01 en Servidor1.

 Utilizando el protocolo PRCS se registró cuenta02 en Servidor1.

 Se creó el contacto entre cuenta01 y cuenta02 con el protocolo PSAC.

 Se envió un correo cumpliendo las reglas de seguridad de cuenta01 a cuenta02.

 Se cambió el registro de cuenta02 a Servidor 2.

 El protocolo PILS se ejecutó actualizando la localización de los servidores.

 Se envió de nuevo un correo desde cuenta01 a cuenta02 y se comprobó su recepción.

 Se pudo comprobar que la localización de los buzones fue modificada sin intervención de un tercero y que la modificación del servidor en cuenta02 se realizó de forma transparente para cuenta01.

REQ004: Validado

Prueba 5:

 Las cuentas en BBM que son utilizadas para enviar y recibir correo no tienen asociados los datos del buzón.

 Al modificar los datos del buzón en la prueba 4 se comprobó que no afectó a los datos de las cuentas relacionadas.

 En caso de publicar los datos de la cuenta, por ejemplo, en una red social o página web, no es necesario cambiar los datos publicados si se modifican los datos del buzón asociado.

REQ006: No Validado, dependerá de la implementación de RPA.

Prueba 6:

-  Se realizó una prueba de acceso simple a los correos debido a la limitación de la cantidad de servidores que se tenían disponibles.
-  El cliente BBM pudo acceder tanto a los correos almacenados localmente como a los almacenados en los dos servidores.
-  La disponibilidad, como se comentó en esta tesis, va a depender del sistema de alta disponibilidad utilizado.

REQ029: Validado

Prueba 7:

-  Se realizó desde el cliente BBM algunas modificaciones a los registros en BBMChain y un envío de un correo desde cuenta01 a cuenta02.
-  Se comprobó que las modificaciones de los registros se realizaron en el nodo local de BMMChain.
-  Quedó a la espera el depósito del correo en el servidor, pero con las validaciones de las reglas realizadas en BBMChain del nodo local.
-  El proceso se realizó sin conexión a internet con éxito.
-  Se activó la conexión a la red externa.
-  Se sincronizaron los nodos actualizándose BBMChain y se completó el proceso de envío del correo.

En la Tabla 18 se presenta un resumen de las pruebas sobre los requisitos de los buzones.

Tabla 18. *Pruebas requisitos buzones.*

Pruebas realizadas sobre los requisitos de los buzones			
Requisito	Prueba	Resultado	Nota
REQ002 y REQ023	1- 2 -3	Validados	
REQ003 y REQ005	4	Validados	
REQ004	5	Validado	
REQ006	6	No validado	Dependerá de la implementación de RPA.
REQ029	7	Validado	

Fuente: Elaboración propia.

Pruebas realizadas sobre los requisitos de los usuarios

REQ007: Validado

Prueba 8:

- 👤 Se activó el campo **OnlySenderauth** en la configuración del registro de seguridad para cuenta03.
- 👤 Se intentó crear un correo desde cuenta01 a cuenta03 con el protocolo PEC.
- 👤 El cliente BBM no permitió la creación del correo al validar la regla en BBMChain que indicaba que cuenta01 debía ser contacto autorizado para cuenta03 y no se había ejecutado el protocolo PSAC desde cuenta01 a cuenta03.
- 👤 Cuenta03 no pudo recibir correo de ningún remitente no autorizado.

Prueba 9:

- 👤 Utilizando el cliente BBM modificado se simuló el depósito de un correo enviado a cuenta03 en el servidor de la cuenta01.
- 👤 Al recibir el mensaje el servidor1, este validó las reglas y rechazó el depósito del correo al comprobar el registro de configuración de la cuenta03 y verificar que no permitía la recepción de correos de contactos no autorizados.

Prueba 10:

- 👤 Utilizando el cliente y servidor BBM modificados para la cuenta01, se envió un correo al destinatario cuenta03.
- 👤 El servidor de cuenta03 nunca reclamó al servidor de la cuenta01 el correo, ya que no era contacto autorizado.
- 👤 El correo permaneció en el servidor1 modificado y cuenta03 nunca tuvo conocimiento de dicho correo.

Prueba 11:

- 👤 Haciendo uso del protocolo PSAC desde cuenta01 a cuenta03 se autorizó el contacto.
- 👤 Haciendo uso del protocolo PEC, se envió un correo desde cuenta01 a cuenta03.
- 👤 El cliente BBM y el servidor de cuenta01 validaron las reglas y permitieron el depósito del correo.
- 👤 El servidor de cuenta03 haciendo uso del protocolo PRECS reclamó el correo y lo depositó para cuenta03.
- 👤 Desde el cliente BBM de cuenta03 se pudo recibir el correo.

REQ008: Validado

Prueba 12:

- ✚ Se activó el campo **OnlyReceiverAuth** del registro de seguridad de la cuenta01.
- ✚ Se intentó crear un correo desde cuenta01 a cuenta04.
- ✚ El cliente BBM no aceptó la creación del correo al no ser cuenta04 un contacto autorizado.
- ✚ En el caso de las pruebas con el cliente BBM y el servidor modificado, los correos nunca llegaron al destino al no ser cuenta04 un contacto autorizado para cuenta01.

REQ009: Validado

Prueba 13:

- ✚ Con la realización de las pruebas 8 a la 12, se validó el requisito para contactos autorizados.

Prueba 14:

- ✚ En el registro de configuración de la cuenta01 y cuenta02 se desactivaron los campos **OnlyReceiverAuth** y **OnlySenderauth**.
- ✚ Haciendo uso del protocolo PEC, se envió un correo desde cuenta01 a cuenta02.
- ✚ Los clientes BBM y los servidores validaron las reglas en BBM y, al comprobar dichos campos de los registros de configuración, permitieron el envío y recepción del correo sin contacto autorizado.
- ✚ Quedó comprobada la configuración y uso de cuentas abiertas.

REQ010: No validado

- ✚ No se realizó ninguna prueba de este requisito. Es una opción de usabilidad que puede ser implementada a partir de los requisitos REQ007 y REQ008.

REQ014: Validado

Prueba 15:

- ✚ Se hace uso del protocolo PSAC para la solicitud de contacto desde cuenta01 a cuenta04.
- ✚ Se ejecutó el protocolo PSAC (rol destinatario) rechazando el contacto (campo **_Rjc**) y solicitando una verificación de identidad (campo **_AD**) por parte de cuenta04.
- ✚ Se comprobó que el contacto quedó en estado "rechazado", recibiendo cuenta01 la notificación.
- ✚ Aunque no se hizo uso de un certificado de identidad, se añadió un registro de datos firmado por cuenta01 con datos personales.

✚ Dicho registro se almacenó con IPFS y se proporcionó de forma cifrada para cuenta04 el enlace del fichero.

✚ Al ejecutar de nuevo el protocolo PSAC con el rol de destinatario, se comprobó que se tenía acceso al fichero y que se podía comprobar la identidad del usuario.

REQ015: Validado

Prueba 16:

✚ Desde cuenta03 se modificó el contacto con cuenta01 dejándolo en estado "bloqueado".

✚ Al ejecutar de nuevo la prueba 11 de envío de correo, se comprobó que al estar el contacto "revocado" no pudo realizarse el envío del correo desde el cliente BBM.

REQ028: Validado

Prueba 17:

✚ Se comprobó que todas las operaciones sobre BBMail requerían el uso de la firma como mecanismo de autenticación.

✚ Se realizaron varias simulaciones sin utilizar la firma enviando mensajes creados manualmente.

✚ Solo fue posible realizar las operaciones de consulta sobre BBMChain de los datos no cifrados, pero dicha información no revelaba información sensible.

REQ042: Validado

Prueba 18:

✚ El registro de contacto en el campo **_AD** es el único lugar donde se puede almacenar información personal.

✚ Al ejecutar la prueba 15 se pudo leer el registro de contacto directamente desde BBMChain, pero dicha información al estar cifrada no reveló ninguna información personal.

✚ En BBMail no se almacena información personal de ningún usuario y no es requerida para el registro ni para otro tipo de operación.

A continuación, se presenta una tabla resumen de las pruebas realizadas sobre los requisitos de los usuarios (Tabla 19).

Tabla 19. Pruebas requisitos usuarios.

Pruebas relacionadas sobre los requisitos de los usuarios			
Requisito	Prueba	Resultado	Nota
REQ007	8-9-10-11	Validado	
REQ008	12	Validado	
REQ009	13-14	Validado	
REQ010		No validado	Opción de usabilidad que puede ser implementada a partir de REQ007 y REQ008.
REQ014	15	Validado	
REQ015	16	Validado	
REQ028	17	Validado	
REQ042	18	Validado	

Fuente: Elaboración propia.

Pruebas realizadas sobre los requisitos de las cuentas

REQ033 y REQ036: Validados

Prueba 19:

-  El usuario02 se definió como usuario "delegado" bajo la supervisión del usuario01, que era el usuario "admin".
-  Los usuarios usuario03, usuario04 y usuario05 fueron creados como usuarios "supervisados".
-  El usuario02 se nombró como "delegado" del resto de usuarios modificando el campo **_authorizerUser** en el registro almacenado en BBMChain con la clave pública de la cuenta de gestión creada para esos usuarios.
-  Las billeteras de los usuarios usuario03, usuario04 y usuario05 pueden ser entregadas y controlados por terceros usuarios.

-  Se comprobó que el usuario "admin" tenía en su billetera los datos de las billeteras de todos los usuarios creados.
-  Se comprobó que también tenía las claves de gestión de las cuentas de los usuarios creados.
-  Usuario02 tenía la billetera de la cuenta de gestión asociada al resto de usuarios.
-  Se comprobó que todos los usuarios creados tenían almacenado en su registro en BBMChain en el campo **_mainUser** la clave privada de la cuenta de gestión que controlaba usuario01.

REQ034: Validado

Prueba 20:

-  Con el usuario01 se generó una nueva cuenta de gestión y se modificaron los datos del usuario "supervisado" usuario03 indicando en el campo **_authorizedUser** los datos de esa nueva cuenta.
-  Se intentó modificar el registro y estado de usuario03 con la cuenta de gestión anterior desde el usuario02 y se comprobó que no fue posible realizar el cambio.
-  Se intentó modificar el registro y estado de usuario03 con la nueva cuenta de gestión y se realizó exitosamente el cambio.
-  Quedó verificado que el cambio de la cuenta de gestión modificó los accesos y la supervisión desde el usuario "admin" quedó comprobada.

REQ035: Validado

Prueba 21 (Basada en la prueba 20):

-  Se comprobó que los registros de los usuarios solo podían ser modificados por las cuentas de gestión creadas y asignadas por el usuario01.
-  En cada modificación dentro de BBMChain los contratos comprobaron que solo el poseedor de la clave de gestión podía realizar las modificaciones de los registros asociados a los usuarios "delegados".
-  Se simuló con el usuario02, que era el antiguo delegado de esas cuentas, modificar los registros, pero BBMChain rechazó la modificación ya que la cuenta de gestión había sido modificada por el usuario "admin".

Prueba 22:

-  Se modificó el registro de seguridad del usuario03 en el campo **LetSend** desactivándolo, para simular un intento de envío de correo desde una de cuenta03.
-  Haciendo uso del protocolo PEC, se intentó crear un correo.
-  El cliente BBM en BBMChain comprobó que los datos del registro no permitían que la cuenta03 pudiera enviar correo.
-  Se invalidó la operación desde el cliente BBM.

Prueba 23:

-  Se simuló enviar un correo desde cuenta03 a cuenta04 con un cliente BBM modificado.
-  Se envió el correo al servidor de la cuenta03 y se comprobó que el correo fue rechazado debido al registro de seguridad que indicaba que no podía realizar envíos.

Prueba 24:

-  Haciendo uso del protocolo PSAC se creó el contacto de cuenta03 a cuenta04.
-  Se simuló con un cliente y servidor modificados el envío de un correo desde cuenta03 a cuenta04.
-  Se depositó el correo en el servidor modificado.
-  El servidor de cuenta04 reclamó el correo haciendo uso del protocolo PRECS y, al validar las reglas, identificó que el registro indicaba que cuenta03 no podía enviar correos.
-  Se invalidó la operación y se notificó en BBMChain que el servidor de la cuenta03 no era un servidor confiable.

REQ037: No validado

-  En la prueba de concepto no se implementó el protocolo PAT, pero se ha dejado especificado como debe ser implementado.

REQ038: No validado

-  En la prueba de concepto no se implementó el protocolo PRBC, pero se ha dejado especificado como debe ser implementado.

REQ041: Validado

Prueba 25:

-  Se simuló crear mensajes RPC donde no era utilizada una firma válida y con usuarios no registrados.
-  Todas las pruebas fallaron debido a que la comprobación por parte de los servidores BBMail y BBMChain no permitían ninguna operación sin ser autenticada por medio de la firma.

En la Tabla 20 se resumen las pruebas realizadas sobre los requisitos de las cuentas.

Tabla 20. *Pruebas requisitos cuentas.*

Pruebas realizadas sobre los requisitos de las cuentas			
Requisitos	Prueba	Resultado	Nota
REQ033 y REQ036	19	Validados	
REQ034	20	Validado	
REQ035	21-22-23-24	Validado	
REQ037		No validado	Protocolo PAT no implementado.
REQ038		No validado	Protocolo PRBC no implementado.
REQ041	25	Validado	

Fuente: Elaboración propia.

Pruebas realizadas sobre los requisitos de los correos

REQ011: Validado

Prueba 26:

-  Se modificó el registro del usuario03 indicando en el campo **DeliveryNotification** que se deseaba recibir notificaciones de entrega.
-  Cuenta04 era contacto autorizado de cuenta03.
-  Se envió un correo desde cuenta03 a cuenta04 haciendo uso del protocolo PEC.
-  Se ejecutó el protocolo PRECS sobre el servidor de cuenta04.
-  El servidor validó las reglas en BBMChain y se identificó que era necesario enviar notificación de entrega.
-  Se envió notificación de entrega desde el servidor de cuenta04 al servidor de cuenta03.
-  Se comprobó que la notificación de entrega fue recibida y que el cliente BBM del usuario03 recibió dicha notificación.

REQ012: Validado

Prueba 27:

-  Se activó el campo **ReadNotification** en el registro de seguridad de cuenta03.
-  Se ejecutó la prueba 26 de nuevo.
-  Se ejecutó el protocolo PRC en el servidor de cuenta04.
-  Se comprobó en el servidor de cuenta04 que el correo había cambiado a estado "leído".
-  Se comprobó que el servidor de cuenta04 había leído el registro de seguridad de cuenta03 y que envió notificación de lectura al servidor de cuenta03.
-  Se comprobó que el cliente BBM de cuenta03 recibió la notificación de lectura.

REQ013: Validado

Prueba 28:

-  Se comprobó que el usuario de la cuenta04 tenía en el registro de seguridad el campo **LetDelete** activado.
-  Se procedió al borrado del correo enviado por cuenta03 en la prueba 29.
-  Se hizo uso del protocolo PBC y se realizó el borrado del correo.
-  Tanto el cliente como el servidor comprobaron que el campo **LetDelete** estaba activado en BBMChain y borraron el correo de sus respectivas bases de datos.

Prueba 29:

-  Con el usuario02 como gestor de la cuenta04, se activó el campo **ReqDelAuth** del registro de seguridad de la cuenta04.
-  Se envió un correo desde la cuenta03 a la cuenta04 haciendo uso del protocolo PEC.
-  Con cuenta04 se intentó realizar el borrado del correo.
-  El Cliente BBM verificó en BBMChain que era necesaria la autorización de borrado.
-  Se envió una notificación de borrado al servidor1 indicando que el borrado debía ser autorizado.
-  El servidor1 al recibir la notificación añadió en la bandeja de notificaciones el correo que debía ser borrado y estaba dirigido a la dirección de gestión de la cuenta04.
-  La notificación fue leída por el poseedor de las claves de gestión asociada a la cuenta04, que era usuario02.
-  La notificación fue firmada como autorizada y enviada al servidor1 por parte de usuario02.
-  Se comprobó que el servidor1 recibió la autorización firmada con las claves de gestión y se procedió al borrado del correo.

REQ016: Validado

Prueba 30:

-  Se simuló el envío de varios correos desde cuenta03 a cuenta04 modificando los mensajes manualmente y utilizando firmas realizadas con usuarios diferentes a cuenta03.
-  Todos los correos fueron rechazados, ya que el cliente BMM y los servidores BBM verificaron para todos los envíos las reglas de seguridad asociadas a las firmas respectivas.
-  No se pudo enviar correos en nombre de cuenta03 desde otros usuarios.

REQ017: Validado

Prueba 31:

-  Se realizaron envíos de correo desde cuenta03 a cuenta04 con las firmas del correo realizadas por cuenta03.
-  Se pudo comprobar que los correos recibidos del remitente cuenta03 en cuenta04, solo podían ser realizados por el poseedor de las claves de cuenta03.

REQ018: No Validado

-  En la prueba de concepto no se implementó el protocolo PCCR, pero se ha dejado especificado como debe ser implementado.

REQ019: Validado

Prueba 32:

-  Se realizó un envío desde cuenta03 a cuenta04 añadiendo como autor a cuenta02.
-  El cliente BBM y el servidor1 comprobaron las reglas del protocolo PEC satisfactoriamente.
-  El servidor1 de cuenta03 envió el correo para firmar a cuenta02 como autor del correo.
-  Se comprobó que en el cliente BBM de cuenta02 recibió el correo para la firma como autor.
-  Se realizó la firma desde cuenta02 y se envió al servidor de cuenta03 la confirmación.
-  El Servidor1 realizó todas las comprobaciones en BBChain y se envió el correo a cuenta04.
-  El servidor de cuenta04 ejecutó el protocolo PRECS y comprobó que todas las reglas se cumplían.
-  Se ejecutó el protocolo PRC y el cliente BBM de la cuenta04 recibió el correo.
-  Se pudo comprobar que el remitente era cuenta03 y que el autor era cuenta02 y que estaban firmados por los dos.

REQ020: Validado

Prueba 33:

-  Se activó el campo **ReqSendAuth** en el registro de seguridad de cuenta03.
-  Se realizó la prueba 32 y en la creación del correo se añadió a cuenta01 también como autorizador en el correo. Cuenta01 es el gestor de cuenta03.
-  Se comprobó que el correo recibido tenía como remitente a cuenta03, que el autor era cuenta02 y que ha habido sido autorizado por cuenta01.

REQ021: Validado

-  La **prueba 32 y 33** validan este requisito.

REQ022: Validado

Prueba 34:

-  Haciendo uso del protocolo PSAC se crean los contactos desde cuenta03 a cuenta05 y cuenta06.
-  Se realizaron las pruebas 32 y 33 pero se añadieron como destinatarios a cuenta05 y cuenta06, que eran ya contactos de cuenta03.
-  Se comprobó que los servidores validaron las reglas en BBM y que las cuentas destinatarias recibieron los correos.

Prueba 35:

-  Se revocó la autorización de contacto de cuenta06 a cuenta03.
-  Se intentó realizar la prueba 34 con esta modificación.
-  El cliente BBM no permitió la creación del correo.
-  Con los clientes BBM y el servidor de cuenta03 modificados, se depositó un correo con cuenta06 como destinatario.
-  El correo nunca fue reclamado y nunca llegó al servidor de cuenta06, ya que no constaba que debía reclamarse correos para cuenta06 desde el servidor de cuenta03.

REQ024: Validado

Prueba 36:

Se modificó la configuración de seguridad de la cuenta03 activando el campo **LetReSend**:

-  Se envió un correo desde cuenta03 a cuenta04 y se comprobó que la recepción fue exitosa.
-  Haciendo uso del protocolo PRE, se simuló el reenvío del correo desde la cuenta04 a la cuenta05.
-  El cliente BBM verificó en BBMChain que la cuenta03 permitía el reenvío de sus correos cuando era remitente y/o autor.
-  Se realizó el reenvío del correo exitosamente al cumplirse las reglas de seguridad y se verificó que se podía tener control sobre el reenvío.

Prueba 37:

-  Se desactivó el campo **LetReSend** y se repitió la prueba 26.
-  Al intentar reenviar el correo desde la cuenta04 a cuenta05 el cliente BBM verificó que no se admitía el reenvío del correo y no se permitió la ejecución del protocolo.
-  Se verificó que se pudo reenviar el contenido del correo utilizando un correo nuevo como se especificó en el protocolo PREC.
-  El correo nuevo no era un correo autorizado y no contenía las firmas de los originadores iniciales.

REQ025: Validado

Prueba 38:

-  En BBM los ficheros adjuntos aparecen dentro del contenido del fichero que siempre va cifrado.
-  Se realizaron algunos envíos de correo adjuntando un fichero de texto de prueba.
-  Se comprobó que solamente los participantes del correo podían acceder a los ficheros y que no había forma de saber si los correos almacenados en los servidores llevaban un fichero adjunto.

REQ026: No Validado

-  En la prueba de concepto no se implementó el protocolo PECSMTP, pero se ha dejado especificado como debe ser implementado.

REQ030: Validado

Prueba 39:

-  En el cliente BBM en BBMail se añadió una opción en la interfaz que permitía activar la visualización de los correos cifrados.
-  Dentro del cliente BBM cuando la opción de visualización no era activada, no era posible visualizar los contenidos de los correos a menos que individualmente se abriera un correo y se proporcionara la clave para descifrarlo.
-  Si la opción era activada, se pedía la clave y se podían leer todos los correos.
-  Este es un requisito de usabilidad y seguridad que puede ser implementando con el objetivo de evitar la lectura de correos en caso de que el cliente BBM quede abierto y un usuario no autorizado acceda a él.

REQ031: No Validado

-  No se realizó la implementación de este requisito en BBMail.

REQ032: Validado

Prueba 40:

-  Se comprobó que a excepción del proceso de cifrado y descifrado de los correos, que exigen que el usuario tome algunas decisiones en la interfaz del cliente BBM, el resto de los procedimientos solo exigían que el usuario utilizara la firma y se ejecutaban de forma transparente.
-  El rendimiento de la prueba era aceptable, pero puede ser mejorado.
-  Se puede trabajar en esta dirección para dar más usabilidad a esta característica.

REQ039: No Validado

-  No se realizó la implementación en BBMail de este requisito y será implementado en versiones posteriores.

REQ040: Validado

-  Durante todas las pruebas realizadas se comprobó que las operaciones no autorizadas eran notificadas.
-  Todas las operaciones fueron canceladas en el momento de comprobar que no cumplían las reglas de seguridad de BBM.

REQ043 y REQ044: Validados

Prueba 41:

-  Se realizaron diferentes pruebas de envío y recepción de correo con las cuentas creadas para las pruebas descritas en esta memoria.
-  Se comprobó que el uso de BBMChain como registro de seguridad y la ejecución de los protocolos PEC, PRECS y PRC garantizaron el cumplimiento de los requisitos de seguridad mencionados en este trabajo y descritos durante las pruebas realizadas.
-  Las pruebas indicaron que hay margen de mejora a la hora de realizar una implementación completa de BBM y es necesario seguir trabajando en esta línea para abordar las dificultades encontradas durante la implementación y las pruebas.

En la Tabla 21 se resumen las pruebas realizadas sobre los requisitos de los correos.

Tabla 21. *Pruebas requisitos correos.*

Pruebas realizadas sobre los requisitos de los correos			
Requisitos	Prueba	Resultado	Nota
REQ011	26	Validado	
REQ012	27	Validado	
REQ013	28-29	Validado	
REQ016	30	Validado	
REQ017	31	Validado	
REQ018		No Validado	PCCR no implementado.
REQ019	32	Validado	
REQ020	33	Validado	
REQ021	32 - 33	Validado	
REQ022	34 - 35	Validado	
REQ024	36 - 37	Validado	
REQ025	38	Validado	
REQ026		No Validado	PECSMTP no implementado.
REQ030	39	Validado	
REQ031		No Validado	Necesario herramienta externa.
REQ032	40	Validado	
REQ039		NoValidado	Trabajos futuros.
REQ040	1 - 41	Validado	Se valida con todas las pruebas.
REQ043 y REQ044	41	Validado	

Fuente: Elaboración propia.

VII. DISCUSIONES ADICIONALES SOBRE LOS RESULTADOS

Durante la descripción de la contribución de este trabajo se han ido incluyendo discusiones alrededor de toda la propuesta con el objetivo de añadir claridad a los aspectos planteados en cada una de las secciones de este trabajo. No obstante, se considera necesario comentar alguna consideración adicional.

📖 Utilizar una *blockchain* para almacenar los correos electrónicos o el valor resumen de estos requiere demasiado almacenamiento en los nodos y al final genera problemas de rendimiento debido a la distribución de información hacia todos los nodos. El correo electrónico por naturaleza propia no es inmutable, sino que es un vehículo para el transporte de información y, por lo tanto, no tiene sentido utilizar *blockchain* para su almacenamiento. El uso de servidores responsables del almacenamiento se torna como la opción más adecuada debido a los costes y al rendimiento asociado en torno a este tipo de servicios. Gracias a la seguridad suministrada y contrastada de los protocolos de BBM, utilizar soluciones de alta disponibilidad para estos servidores proporciona la solución de disponibilidad requerida para los correos electrónicos.

📖 En BBM se considera el correo electrónico como dos bloques de información, la cabecera y el cuerpo, cuya seguridad requiere un tratamiento diferente. Se intenta que la cabecera que tiene la información de transporte estuviera oculta para terceros no autorizados, es decir, que solamente el remitente y destinatario pudieran interpretarla, e incluso el servidor del remitente no pudiera leerla. Debido a los costes de almacenamiento, el usuario debe registrarse frente al proveedor para pagar dichos costes, lo que lleva al servidor a tener que identificar al remitente e interpretarla la cabecera y, además, a realizar las validaciones de los protocolos. En esta investigación, la única solución que se encontró a este problema fue realizar el registro de los usuarios en el servidor de forma externa y, durante el transporte del correo, se presentaba un *token* que los servidores verificaban y toda la validación de los protocolos pasaba al cliente BBM; al final se descartó esta solución porque no era muy usable por los usuarios y proveedores.

📖 La *blockchain*, gracias a la inmutabilidad del libro mayor, brinda una oportunidad de proporcionar consistencia a los protocolos de seguridad, puesto que las reglas de validación están descentralizadas y fuera del alcance de entidades con fines mal intencionados. En la ejecución de los protocolos hay que tener en cuenta el tiempo de actualización de los nodos, siendo clave llevar o transmitir la versión del protocolo que se está utilizando.

-  En el caso del esquema de seguridad empleado para los contactos, el cliente BBM deberá facilitar esta tarea para que sea usable para los usuarios, ya que el esquema de jerarquía de clases se hace muy fácilmente utilizable cuando la frecuencia al utilizar las claves es baja, como el caso de los usuarios y cuentas. Al ser más frecuente para los usuarios el uso y gestión de contactos, es necesario proporcionar un mecanismo que facilite esta funcionalidad, pero que es necesaria para lograr una seguridad completa.
-  La integración de BBM con otros protocolos menos seguros como SMTP puede ser lograda mediante un Integrador, pero muchas de las características de seguridad esenciales de BBM pueden ser claramente perdidas. Aunque la integración pasa por la validación de las reglas BBM, este protocolo no alcanza todos los objetivos de seguridad que pueden ser logrados con una implementación completa de BBM. Por ejemplo, con este protocolo no pueden ser ocultadas las relaciones de comunicación del correo (de, para) ni restringir el envío de correo no deseado, solamente no permitir su entrada.

VIII. ASPECTOS DESTACABLES DE LA INVESTIGACIÓN

Las principales **fortalezas** de esta investigación que la diferencian de la mayoría de las soluciones publicadas por otros grupos de trabajo son:

- ❏ En BBM, el correo electrónico se almacena en el servidor del remitente y sólo podrá ser leído por los destinatarios que lo reclamen. Este esquema soluciona el problema del correo electrónico no deseado.
- ❏ Utilizar *blockchain* como registro inmutable para los datos gestionados por los protocolos de seguridad es una solución que puede ser incluso escalada a otros protocolos de seguridad. Al utilizar los protocolos de seguridad como contratos inteligentes, la actualización y distribución del protocolo queda asegurada gracias a los algoritmos de consenso utilizados.
- ❏ Con BBM, los MITM no producen el mismo efecto negativo que con los protocolos SMTP, POP e IMAP, ya que todos los protocolos garantizan la integridad y confidencialidad de los datos transportados.
- ❏ El libro mayor en *blockchain* muestra las transacciones y las direcciones involucradas en todos los llamados de los contratos realizados. Si una clave en *blockchain* que representa un usuario en BBM, es un usuario corporativo, todos los movimientos que realice en *blockchain* sobre otros usuarios (por ejemplo, cambiar su registro de seguridad) puede ser fácilmente trazado y de esta forma se puede deducir las relaciones entre las claves de las cuentas y por ende la estructura de claves corporativas.
- ❏ El presente enfoque da una solución a este problema con el uso de una estructura jerárquica de claves determinística y la utilización de claves de gestión para crear una relación de control entre dos claves de la jerarquía sin revelarlo en *blockchain*.
- ❏ No obstante, los datos personales del usuario nunca son almacenados en *blockchain* y por lo tanto solo dicha estructura pública se utilizaría por razones de trazabilidad, que es clave para la misma seguridad en la que se sustenta BBM.
- ❏ En esta tesis se han analizado los problemas de seguridad considerando un gran abanico de soluciones existentes y el estudio de las propiedades de seguridad ofrecidas. No se ha encontrado ninguna propuesta que abordara los problemas de seguridad del correo desde un punto de vista amplio, sino de aspectos de seguridad muy concretos. Al estar BBM diseñado a partir de los requisitos de seguridad de los usuarios, proporciona una solución muy completa que incluye muchos aspectos que otros autores no han considerado.

- ☐ Aunque los profesionales de ciberseguridad conocen muy bien que los objetivos de la seguridad de la información son la confidencialidad, integridad, no repudio y la disponibilidad, los otros aspectos tratados en esta tesis (control, auditoría, usabilidad e integración) no se incluyen directamente en ninguna de las propuestas analizadas, pero se resalta en el presente trabajo su importancia real y cómo afectan de forma directa o indirecta la seguridad del correo electrónico.

Por otra parte, se considera importante resaltar algunas de los problemas resueltos hallados durante el desarrollo de esta propuesta:

- ☐ La gestión de las claves en una solución basada en *blockchain* presenta un gran inconveniente a la hora de abordar la seguridad de la información en un libro mayor inmutable. La pérdida, robo u olvido de la clave privada o de la semilla para generarla pone en duda toda la seguridad que sobre ella se ha construido.
- ☐ A pesar de esta limitación, se ha realizado una propuesta basada en claves de gestión que permite la regeneración y bloqueo de cuentas debido a que, hasta ahora, no hay ninguna solución existente sin el uso de ellas. No obstante, el robo y el bloqueo de acceso a los correos de la clave privada comprometida queda por resolver desde BBM, aunque es posible fuera de BBM comunicarlo directamente al servidor con un bloqueo permanente.
- ☐ Utilizar billeteras existentes o comerciales de *blockchain* en BBM no fue posible completamente. Por esta razón, se diseñó una solución propietaria para el uso de la clave privada para el cifrado y descifrado de los datos.
- ☐ Actualmente, esta propuesta se encuentra aún en desarrollo, de modo que quedan varios aspectos por solventar y mejorar en futuras líneas de investigación.

IX. TRABAJOS FUTUROS

Este trabajo es una propuesta inicial del uso de los contratos inteligentes de *blockchain* en el diseño de protocolos de seguridad. El desarrollo este enfoque plantea una serie de retos y preguntas que estamos abordando en la actualidad o que deben ser tratados en futuros trabajos. Entre ellos, destacan los siguientes:

- Desarrollar propuestas más manejables para el uso de las claves privadas que permitan abordar el problema de la pérdida de la clave y la recuperación de los datos gestionados con dicha clave.
- Desarrollar propuestas de billeteras para gestionar, no solamente claves privadas, sino otro tipo de información propietaria, como la gestionada con BBM.
- Diseñar billeteras que incluyan diferentes algoritmos criptográficos, como, por ejemplo, el cifrado, el descifrado o la generación de claves, entre otros.
- Diseñar una *blockchain* propietaria para el contenedor BBMChain.
- Implementar una solución gratuita, disponible y de código abierto para el entorno académico con el objetivo de seguir desarrollando y mejorando los protocolos BBM.
- Proponer un diseño y desarrollar el Protocolo de Envío de Correo SMTP (PECSMTP) para la integración con los protocolos POP e IMAP.
- Desarrollar una propuesta para el protocolo PCCR que incluya toda la gestión de la CA dentro de BBM.
- Desarrollar diferentes propuestas para el Protocolo de Acceso a Terceros (PAT) que faciliten el acceso temporal a los correos de un tercero, con bloqueo de acceso posterior al mismo tercero.
- Diseñar y desarrollar la red privada de proveedores de almacenamiento sobre una *blockchain* privada con una propuesta sólida para el Protocolo de Borrado de Correo (PBC). Sobre este enfoque estamos desarrollando actualmente una propuesta.
- Implementar BBMChain con una estructura de información totalmente privada donde ninguno de los agentes pudiera relacionar la información.
- Desarrollar el modelo de gobernanza y reglas de suspensión para los usuarios y servidores dentro de BBM.
- Diseñar una solución para la interoperabilidad de diferentes implementaciones de BBM.

Capítulo 4

Conclusiones

CONCLUSIONES

En el estudio del estado del arte (capítulo 1), se identifica que el correo electrónico no solo es un mensaje con adjuntos que envía un usuario a otro, sino que hay otros elementos que necesitan ser asegurados. Debido a que no se plantearon desde el diseño inicial, tratar de abordarlos como una capa adicional a los protocolos existentes ha producido el fracaso de numerosas propuestas y ha dado lugar a soluciones implementadas hoy en día de baja eficacia según la evaluación de ellas realizada en esta investigación (Tabla 11).

En base a estos resultados y de acuerdo con los objetivos planteados, después de haber diseñado, desarrollado y validado la prueba de concepto de una solución segura para el correo electrónico, se extraen las siguientes conclusiones:

- Tras analizar los efectos económicos de los ataques informáticos cuyo vector de ataque era el correo, las diferentes amenazas a las que estaban expuestas las organizaciones y las vulnerabilidades de los protocolos utilizados por las diferentes herramientas de transporte de correo usadas, se pasa a identificar las necesidades de los usuarios y organizaciones para abordar el problema de la seguridad en el correo. Se definieron 44 requisitos (Tabla 13) orientados a resolver cuestiones de seguridad alrededor del correo electrónico que abarcara los siguientes nueve aspectos de seguridad: confidencialidad, integridad, no repudio, autenticación, disponibilidad, control, auditoría, usabilidad e integración.
- Se diseñó una arquitectura de correo electrónico basado en *blockchain* llamada BBM que permite la gestión de correo electrónico segura. Es una solución orientada a dar el control de los correos electrónicos al usuario y proporcionarles independencia de gestión de la plataforma.
- En BBM no es posible el correo no deseado. Dada la filosofía de depósito de correo en el servidor del remitente, un atacante que intente enviar correo basura lo único que conseguirá será almacenar en su propio servidor gran cantidad de correo que ningún usuario reclamará, recayendo sobre él mismo el coste de ese almacenamiento.
- Se diseñaron 17 protocolos (capítulo 3, sección IV.4) distribuidos sobre una *blockchain* que permiten el envío, la recepción, la gestión y el control del correo electrónico de forma segura.
- La arquitectura BBM junto con los 17 protocolos garantizan en gran medida todos los aspectos de seguridad requeridos, que se resumen en la Tabla 22 y se detallan a continuación de ésta.

Tabla 22. Resumen soluciones propuestas a los aspectos de seguridad.

Aspectos de seguridad	Solución propuesta en BBM
Confidencialidad	- Esquemas de cifrado y firma asegurando la cabecera y el cuerpo del correo. - Información solo accedida por los autorizados.
Integridad	Uso de <i>blockchain</i> y firma digital.
No repudio	Firma digital y <i>token</i> de autorización.
Autenticación	Claves privadas y claves de gestión.
Disponibilidad	- Disponibilidad de los protocolos utilizando <i>blockchain</i>. - Disponibilidad de los correos utilizando una solución <i>offchain</i> con alta disponibilidad.
Control	Utilización de un registro de seguridad almacenado en <i>blockchain</i> controlados por cuentas de gestión.
Auditoría	Trazabilidad en <i>blockchain</i> debido a su inmutabilidad y trazabilidad en la red de proveedores de almacenamiento.
Usabilidad	- Utilización de una billetera de gestión. - Poca participación del usuario. - Transparencia en los procedimientos criptográficos utilizados.
Integración	Utilización de un Integrador que hace de traductor entre los protocolos pero valida los protocolos en BBMChain.

Fuente: Elaboración propia.

- Actualmente existen soluciones que logran en buena medida garantizar que el mensaje no sea leído por un tercero, pero en esta tesis, la confidencialidad en el contexto del correo es un concepto más amplio y se aborda el problema desde una perspectiva diferente. Se presenta el correo como dos componentes bien diferenciados pero relacionados criptográficamente: la cabecera y el cuerpo del correo. Se logra asegurar la confidencialidad tanto en el contenido del correo como en los datos de transporte utilizando diferentes esquemas que incluyen cifrado y firma digital. De esta forma, en BBM, un correo electrónico almacenado o transportado solo puede ser leído o interpretado por los participantes de la comunicación y eso depende del estado en el que este se encuentre.
- La integridad se alcanzó con el uso de la firma digital y *blockchain*. Todas las comunicaciones realizadas en BBM requieren la firma de los participantes y, además, en caso de registrar información que no puede ser alterada, se hace uso de *blockchain*.

- El no repudio también se logra haciendo uso de la firma digital. Se introdujo el uso del *token* de autorización, una figura criptográfica como medida de prueba de la aceptación de un contrato y no repudio al conceder una autorización a un tercero.
- La autenticación, que normalmente se presenta como un servicio, es garantizada gracias al uso de un par de claves pública y privada, que se enlazan al registro del usuario o de una cuenta y se almacenan en *blockchain* funcionando como un sistema de intercambio de claves. Un usuario se identifica en BBM cuando demuestra que es el propietario de la clave privada o secreta utilizada en el registro de alguna transacción. En BBM, todos los protocolos requieren el uso de la clave pública y una demostración de propiedad de la clave privada a través de una firma. Por ello, el servicio de autenticación está integrado en la propia solución. Además de la autenticación, la arquitectura BBM permite enlazar la identidad de un usuario a una clave pública dentro de BBM, lo que posibilita la creación de un tipo de certificado que puede ser utilizado para que un usuario pueda verificar la identidad de otro.
- Se proponen dos soluciones para el problema de la disponibilidad. Para los protocolos, se define que se ejecutaran en *blockchain*, y, por lo tanto, gozan de sus propiedades de descentralización y distribución. Por el contrario, se considera que la disponibilidad del correo electrónico como tal debe ser abordada como una solución fuera de *blockchain* (*offchain*) debido a sus características de no inmutabilidad. Para este último caso, pueden utilizarse las soluciones de alta disponibilidad existentes o futuras que puedan ir desarrollándose.
- El correo electrónico tradicional adolece de no proporcionar ninguna característica de control, problema que se solventa en esta tesis con el uso de registros de seguridad distribuidos en *blockchain* y controlados por cuentas de gestión que proporcionan todo el control necesario para delegación de responsabilidad, autorización de acciones, accesos y permisos, entre otros. El registro de seguridad es flexible y puede incluir más características si se requieren en un futuro.
- La trazabilidad se asegura desde *blockchain* proporcionando un registro inmutable de la ejecución de todas las modificaciones de los protocolos de BBM. Por otra parte, el diseño de los servidores BBM garantiza que la información pueda ser auditada por los usuarios autorizados.
- La usabilidad se asegura proporcionando transparencia en los procedimientos criptográficos utilizados que deben ser ejecutados dentro de la billetera BBM y que solo exigen la participación del usuario cuando debe firmar o generar una clave.

- En cuanto a la integración con el protocolo SMTP, puede ser lograda con el Integrador BBMI, pero después de estudiar muchas de las propuestas de seguridad en torno a ese protocolo, las mejores soluciones deben alejarse del protocolo en sí y partir de nuevos enfoques en los que la seguridad sea la base de las propuestas y se construyan todos los demás servicios sobre ellas.
- Se creó una dApp basada en BBM en la que se implementaron la mayoría de los protocolos y se comprobó que es factible el desarrollo de una solución de este tipo.

Capítulo 5

Referencias bibliográficas

REFERENCIAS BIBLIOGRÁFICAS

-  Aaron Davis. (2015). MetaMask. Retrieved May 12, 2020, from www.metamask.io
-  Abadi, M., & Blanchet, B. (2005). Computer-assisted verification of a protocol for certified email. *Science of Computer Programming*, 58(1–2), 3–27. <https://doi.org/10.1016/j.scico.2005.02.002>
-  Acronis. (2021). *Acronis Cyberthreats Report*.
-  ACSC. (2019). Threat Report. *Australian Cyber Security Center*, 1, 40.
-  Adida, B., Hohenberger, S., & Rivest, R. L. (2005). Lightweight Encryption for Email. *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, (July), 13. Retrieved from <http://dl.acm.org/citation.cfm?id=1251282.1251295>
-  Agbo, C., Mahmoud, Q., & Eklund, J. (2019). Blockchain Technology in Healthcare: A Systematic Review. *Healthcare*, 7(2), 56. <https://doi.org/10.3390/healthcare7020056>
-  Aggarwal, S., & Kumar, N. (2021a). Attacks on blockchain★Working model. *Advances in Computers*, 121. <https://doi.org/10.1016/bs.adcom.2020.08.020>
-  Aggarwal, S., & Kumar, N. (2021b). Blockchain 2.0: Smart contracts. *Advances in Computers*, 121. <https://doi.org/10.1016/bs.adcom.2020.08.015>
-  Aggarwal, S., & Kumar, N. (2021c). Core components of blockchain. *Advances in Computers*, 121. <https://doi.org/10.1016/bs.adcom.2020.08.010>
-  Aggarwal, S., & Kumar, N. (2021d). Cryptographic consensus mechanisms. In *Advances in Computers* (Vol. 121). <https://doi.org/10.1016/bs.adcom.2020.08.011>
-  Aggarwal, S., & Kumar, N. (2021e). History of blockchain-Blockchain 1.0: Currency. *Advances in Computers*, 121. <https://doi.org/10.1016/bs.adcom.2020.08.008>
-  Aggarwal, S., & Kumar, N. (2021f). Hyperledger. *Advances in Computers*, 121. <https://doi.org/10.1016/bs.adcom.2020.08.016>
-  Ahmed, S., Shah, M. A., & Wakil, K. (2020). Blockchain as a Trust Builder in the Smart City Domain: A Systematic Literature Review. *IEEE Access*, Vol. 8. <https://doi.org/10.1109/ACCESS.2020.2993724>
-  Ajina, A., & Kumar, U. (2019). Email spoofing & backlashes. *International Journal of Innovative Technology and Exploring Engineering*, 8(11). <https://doi.org/10.35940/ijitee.J9310.0981119>
-  Ako, M. A. (2017). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data. *Cryptography and Network Security*, (June).
-  Al-Fedaghi, S., & Alnasser, H. (2020). Modeling Network Security: Case Study of Email System. *ArXiv*, 11(3), 99–107.
-  Al-Shaibani, H., Lasla, N., & Abdallah, M. (2020). Consortium Blockchain-Based Decentralized Stock Exchange Platform. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3005663>
-  Alge, W. (2012). Email in the cloud: the challenges and benefits. *Computer Fraud & Security*, 2012(7), 10–12. [https://doi.org/10.1016/S1361-3723\(12\)70073-8](https://doi.org/10.1016/S1361-3723(12)70073-8)
-  Ali, O., Ally, M., Clutterbuck, & Dwivedi, Y. (2020). The state of play of blockchain technology in the financial services sector: A systematic literature review. *International Journal of Information Management*, Vol. 54. <https://doi.org/10.1016/j.ijinfomgt.2020.102199>
-  Alkhudary, R., Brusset, X., & Fenies, P. (2020). Blockchain in general management and economics: a systematic literature review. *European Business Review*, Vol. 32. <https://doi.org/10.1108/EBR-11-2019-0297>
-  Allman, E., & Hansen, T. (2004). SMTP Service Extension for Message Tracking. *RFC*, (3885). Retrieved from <http://www.ietf.org/rfc/rfc3885.txt>
-  Allocchio, C. (2001). *Minimal FAX address format in Internet Mail*. Retrieved from <http://www.ietf.org/rfc/rfc3192.txt>
-  Alwen, J., Chen, B., Pietrzak, K., Reyzin, L., & Tessaro, S. (2017). Scrypt is maximally memory-hard. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10212 LNCS. https://doi.org/10.1007/978-3-319-56617-7_2
-  Amin, R. M. (2002). Detecting targeted malicious email through supervised classification of persistent threat and recipient oriented features. *Computer and Telecommunications Engineering M.S. Telecommunications and Networking*.

- ☐ Ampel, B., Patton, M., & Chen, H. (2019). Performance modeling of hyperledger sawtooth blockchain. *2019 IEEE International Conference on Intelligence and Security Informatics, ISI 2019*. <https://doi.org/10.1109/ISI.2019.8823238>
- ☐ Anderson, D. P. (2020). BOINC: A Platform for Volunteer Computing. *Journal of Grid Computing*, 18(1). <https://doi.org/10.1007/s10723-019-09497-9>
- ☐ Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, Vol. 36. <https://doi.org/10.1145/1041680.1041681>
- ☐ Andy Yen, & Butler, B. (2021). ProtonMail. Retrieved from <https://protonmail.com/>
- ☐ Anti-Phishing Working Group. (2020). Phishing Activity Trends Report 3rd Quarter 2020. *Apwg*, (November), 1–12. Retrieved from https://docs.apwg.org/reports/apwg_trends_report_q3_2020.pdf
- ☐ Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005). *DNS Security Introduction and Requirements*. Retrieved from <http://www.ietf.org/rfc/rfc4033.txt>
- ☐ Armknecht, F., Karame, G. O., Mandal, A., Youssef, F., & Zenner, E. (2015). Ripple: Overview and outlook. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9229. https://doi.org/10.1007/978-3-319-22846-4_10
- ☐ Aslam, U., Manzoor, S., & Babar, M. I. (2019). Systematic Literature Review of Blockchain in Requisites of Big Data and its Applications. *Journal of Software Engineering & Intelligent Systems*, (August).
- ☐ Atzei, N., Bartoletti, M., Lande, S., & Zunino, R. (2018). A Formal Model of Bitcoin Transactions. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10957 LNCS. https://doi.org/10.1007/978-3-662-58387-6_29
- ☐ Australian Cyber Security Centre. (2020). *ACSC Annual Cyber Threat Report July 2019 to June 2020*. (June), 4–8. Retrieved from <https://www.cyber.gov.au/sites/default/files/2020-09/ACSC-Annual-Cyber-Threat-Report-2019-20.pdf>
- ☐ Babrahem, A. S., Alharbi, E. T., Alshiky, A. M., Alqurashi, S. S., & Kar, J. (2015). Study of the Security Enhancements in Various E-Mail Systems. *Journal of Information Security*, 06(01). <https://doi.org/10.4236/jis.2015.61001>
- ☐ Baldi, M., Chiaraluce, F., Kodra, M., & Spalazzi, L. (2019). Security analysis of a blockchain-based protocol for the certification of academic credentials. *ArXiv*.
- ☐ Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., & Danezis, G. (2019). Sok: Consensus in the age of blockchains. *AFT 2019 - Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 183–198. <https://doi.org/10.1145/3318041.3355458>
- ☐ Batta, A., Gandhi, M., Kar, A. K., Loganayagam, N., & Ilavarasan, V. (2020). Diffusion of blockchain in logistics and transportation industry: an analysis through the synthesis of academic and trade literature. *Journal of Science and Technology Policy Management*. <https://doi.org/10.1108/JSTPM-07-2020-0105>
- ☐ Baumgart, I., & Mies, S. (2007). IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3). *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, 2(Draft 3).
- ☐ Baza, M., Nabil, M., Mahmoud, M. M. E. A., Bewermeier, N., Fidan, K., Alasmay, W., & Abdallah, M. (2020). Detecting Sybil Attacks using Proofs of Work and Location in VANETs. *IEEE Transactions on Dependable and Secure Computing*. <https://doi.org/10.1109/TDSC.2020.2993769>
- ☐ Bellamy, A. J., & McDonald, M. (2002). 'The utility of human security': Which humans? What security? A reply to Thomas & Tow. *Security Dialogue*, 33(3), 373–377. <https://doi.org/10.1177/0967010602033003010>
- ☐ Benčić, F. M., & Žarko, I. P. (2018). Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. *Proceedings - International Conference on Distributed Computing Systems*, 2018-July. <https://doi.org/10.1109/ICDCS.2018.00171>
- ☐ Benet, J., & Greco, N. (2017). Filecoin: A Decentralized Storage Network. *Protocol Labs*.
- ☐ Bentov, I., Lee, C., Mizrahi, A., & Rosenfeld, M. (2014). Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. *Cryptology EPrint Archive*, 452(3).
- ☐ Bermeo-Almeida, O., Cardenas-Rodriguez, M., Samaniego-Cobo, T., Ferruzola-Gómez, E., Cabezas-Cabezas, R., & Bazán-Vera, W. (2018). Blockchain in agriculture: A systematic literature review. *Communications in Computer and Information Science*, 883. https://doi.org/10.1007/978-3-030-00940-3_4
- ☐ Bernal Bernabe, J., Canovas, J. L., Hernandez-Ramos, J. L., Torres Moreno, R., & Skarmeta, A. (2019). Privacy-Preserving Solutions for Blockchain: Review and Challenges. *IEEE Access*, Vol. 7. <https://doi.org/10.1109/ACCESS.2019.2950872>

- ☐ Bhagwan, R., Savage, S., & Voelker, G. M. (2003). Understanding availability. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2735. https://doi.org/10.1007/978-3-540-45172-3_24
- ☐ Bhaskar, P., Tiwari, C. K., & Joshi, A. (2020). Blockchain in education management: present and future applications. *Interactive Technology and Smart Education*. <https://doi.org/10.1108/ITSE-07-2020-0102>
- ☐ Bhatt, A. P., & Sharma, M. (2019). E-Mail security framework through various virus encryption techniques. *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*. <https://doi.org/10.1109/ICCS45141.2019.9065809>
- ☐ Bhattacharjee, K., Maity, K., & Das, S. (2018). A search for good pseudo-random number generators: Survey and empirical studies. *ArXiv*.
- ☐ Bhosale, S. T., Patil, T., & Patil, P. (2015). SQLite: Light Database System. *International Journal of Computer Science and Mobile Computing*, 44(4).
- ☐ Bhushan, A. K. (1971). File Transfer Protocol. *RFC114*. Retrieved from <http://www.ietf.org/rfc/rfc114.txt>
- ☐ Bhushan, A. K. (1973). FTP and Network Mail System. *RFC475*. Retrieved from <http://www.ietf.org/rfc/rfc475.txt>
- ☐ Bhushan, A. K., Pogran, K. T., Tomlinson, R. S., & White, J. E. (1973). Standardizing Network Mail Headers. *RFC561*. Retrieved from <http://www.ietf.org/rfc/rfc561.txt>
- ☐ Bhutta, M. N. M., Khwaja, A. A., Nadeem, A., Ahmad, H. F., Khan, M. K., Hanif, M. A., ... Cao, Y. (2021). A Survey on Blockchain Technology: Evolution, Architecture and Security. *IEEE Access*, Vol. 9. <https://doi.org/10.1109/ACCESS.2021.3072849>
- ☐ BigchainDB GmbH. (2018). *BigchainDB: The blockchain database [White Paper]*. (May). Retrieved from <https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf>
- ☐ Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1462. <https://doi.org/10.1007/BFb0055716>
- ☐ Bodkhe, U., Tanwar, S., Parekh, K., Khanpara, P., Tyagi, S., Kumar, N., & Alazab, M. (2020). Blockchain for Industry 4.0: A Comprehensive Review. *IEEE Access*, 8, 79764–79800. <https://doi.org/10.1109/ACCESS.2020.2988579>
- ☐ Boireau, O. (2018). Securing the blockchain against hackers. *Network Security*, 2018(1). [https://doi.org/10.1016/S1353-4858\(18\)30006-0](https://doi.org/10.1016/S1353-4858(18)30006-0)
- ☐ Boneh, D., & Franklin, M. (2003). Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3). <https://doi.org/10.1137/S0097539701398521>
- ☐ Borenstein, N., & Freed, N. (1993). *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. Retrieved from <http://www.ietf.org/rfc/rfc1521.txt>
- ☐ Bosselaers, A., Dobbertin, H., & Preneel, B. (1997). The RIPEMD-160 cryptographic hash function. *Dr. Dobb's Journal*, 22(1), 24–28.
- ☐ Braden, R. T. (1973). On mail service to CCN. *RFC498*. Retrieved from <http://www.ietf.org/rfc/rfc498.txt>
- ☐ Bray, T. (2014). *The JavaScript Object Notation (JSON) Data Interchange Format*. Retrieved from <http://www.ietf.org/rfc/rfc7159.txt>
- ☐ Bressler, R. D., & Thomas, R. (1973). Mail retrieval via FTP. *RFC458*. Retrieved from <http://www.ietf.org/rfc/rfc458.txt>
- ☐ Brown, D. R. L., & Research, C. (2010). SEC 2: Recommended Elliptic Curve Domain Parameters. *Standards for Efficient Cryptography*.
- ☐ Brownlee, J. (2011). Clever Algorithms - Nature-Inspired Programming Recipes. In *Search*.
- ☐ Burger, E., Candell, E., Eliot, C., & Klyne, G. (2003). *Message Context for Internet Mail*. Retrieved from <http://www.ietf.org/rfc/rfc3458.txt>
- ☐ Buterin, V., Reijnders, D., Leonardos, S., & Piliouras, G. (2020). Incentives in Ethereum's hybrid Casper protocol. *International Journal of Network Management*, 30(5). <https://doi.org/10.1002/nem.2098>
- ☐ Butler, K., Enck, W., Plasterr, J., Traynor, P., & Mcdaniel, P. (2006). Privacy Preserving Web-Based Email. *Information Systems Security. ICISS 2006*, 4332, 116–131. https://doi.org/10.1007/11961635_8
- ☐ Butler, M., Postel, J., Chase, D., Goldberger, J., & Reynolds, J. K. (1985). Post Office Protocol: Version 2. *RFC937*. Retrieved from <http://www.ietf.org/rfc/rfc937.txt>

- ❏ Cachin, C., & Vukolić, M. (2017). Blockchain consensus protocols in the wild. *Leibniz International Proceedings in Informatics, LIPIcs*, 91. <https://doi.org/10.4230/LIPIcs.DISC.2017.1>
- ❏ Cagigas, D., Clifton, J., Diaz-Fuentes, D., & Fernandez-Gutierrez, M. (2021). Blockchain for Public Services: A Systematic Literature Review. *IEEE Access*, Vol. 9. <https://doi.org/10.1109/ACCESS.2021.3052019>
- ❏ Cai, W., Wang, Z., Ernst, J. B., Hong, Z., Feng, C., & Leung, V. C. M. (2018). Decentralized Applications: The Blockchain-Empowered Software System. *IEEE Access*, 6. <https://doi.org/10.1109/ACCESS.2018.2870644>
- ❏ Caldarelli, G., & Ellul, J. (2021). Trusted academic transcripts on the blockchain: A systematic literature review. *Applied Sciences (Switzerland)*, Vol. 11. <https://doi.org/10.3390/app11041842>
- ❏ Callas, J., Donnerhacke, L., Finney, H., Shaw, D., & Thayer, R. (2007). OpenPGP Message Format. *RFC4880*. Retrieved from <http://www.ietf.org/rfc/rfc4880.txt>
- ❏ Calvaresi, D., Dubovitskaya, A., Calbimonte, J. P., Taveter, K., & Schumacher, M. (2018). Multi-agent systems and blockchain: Results from a systematic literature review. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10978 LNAI. https://doi.org/10.1007/978-3-319-94580-4_9
- ❏ Castro, M., & Liskov, B. (1999). Practical Byzantine Fault Tolerance. *Proceedings of the Symposium on Operating System Design and Implementation*, (February). <https://doi.org/10.1145/571637.571640>
- ❏ Castro, R. Q., & Au-Yong-Oliveira, M. (2021). Blockchain and Higher Education Diplomas. *European Journal of Investigation in Health, Psychology and Education*, 11(1). <https://doi.org/10.3390/ejihpe11010013>
- ❏ Chakravarty, M. M. T., Coretti, S., Fitzi, M., Gazi, P., Kant, P., Kiayias, A., & Russell, A. (2020). Hydra: Fast Isomorphic State Channels. *IACR Cryptol. EPrint Arch.*, 2020.
- ❏ Chamadoira-González, J., García-Díaz, V., Núñez-Valdez, E. R., Gómez, A. G., & Crespo, R. G. (2020). Replacing email protocols with blockchain-based smart contracts. *Cluster Computing*, 23(3), 1795–1801. <https://doi.org/10.1007/s10586-020-03128-9>
- ❏ Chan, W. K., Chin, J. J., & Goh, V. T. (2020). Evolution of Bitcoin Addresses from Security Perspectives. *2020 15th International Conference for Internet Technology and Secured Transactions, ICITST 2020*. <https://doi.org/10.23919/ICITST51030.2020.9351346>
- ❏ Chandhok, R., & Wenger, G. (2001). *List-Id: A Structured Field and Namespace for the Identification of Mailing Lists*. Retrieved from <http://www.ietf.org/rfc/rfc2919.txt>
- ❏ Chang, S. E., & Chen, Y. (2020). When blockchain meets supply chain: A systematic literature review on current development and potential applications. *IEEE Access*, Vol. 8. <https://doi.org/10.1109/ACCESS.2020.2983601>
- ❏ Chanti, S., & Chithralekha, T. (2020). Classification of Anti-phishing Solutions. *SN Computer Science*, 1(1). <https://doi.org/10.1007/s42979-019-0011-2>
- ❏ Chatterjee, R., & Chatterjee, R. (2018). An Overview of the Emerging Technology: Blockchain. *Proceedings - 2017 International Conference on Computational Intelligence and Networks, CINE 2017*. <https://doi.org/10.1109/CINE.2017.33>
- ❏ Chauhan, A. (2019). A Review on Various Aspects of MongoDB Databases. *AIP Conference Proceedings*, 08(05).
- ❏ Check Point. (2021). *Cyber security report 2021*.
- ❏ Chen, J., & Micali, S. (2019). Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777. <https://doi.org/10.1016/j.tcs.2019.02.001>
- ❏ Chen, X., Zhang, K., Liang, X., Qiu, W., Zhang, Z., & Tu, D. (2020). HyperBSA: A high-performance consortium blockchain storage architecture for massive data. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3027610>
- ❏ Chen, Y., Chen, S., Liang, J., Feagan, L. W., Han, W., Huang, S., & Wang, X. S. (2020). Decentralized data access control over consortium blockchains. *Information Systems*, 94. <https://doi.org/10.1016/j.is.2020.101590>
- ❏ Chohan, U. W. (2018). The Double Spending Problem and Cryptocurrencies. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3090174>
- ❏ Choukse, D., Singh, U. K., Laddhani, L., & Shahapurkar, R. (2012). Designing secure email infrastructure. *IFIP International Conference on Wireless and Optical Communications Networks, WOCN*. <https://doi.org/10.1109/WOCN.2012.6335534>
- ❏ Chowdhury, M. J. M., Ferdous, M. S., Biswas, K., Chowdhury, N., Kayes, A. S. M., Alazab, M., & Watters, P. (2019). A comparative analysis of distributed ledger technology platforms. *IEEE Access*, 7. <https://doi.org/10.1109/ACCESS.2019.2953729>

- ☐ Clark, D. D., Wroclawski, J., Sollins, K. R., & Braden, R. (2005). Tussle in cyberspace: Defining tomorrow's internet. *IEEE/ACM Transactions on Networking*, 13(3). <https://doi.org/10.1109/TNET.2005.850224>
- ☐ Clark, J., & Essex, A. (2012). CommitCoin: Carbon dating commitments with bitcoin. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7397 LNCS. https://doi.org/10.1007/978-3-642-32946-3_28
- ☐ Cocca, P. (2004). E-mail Security Threats. *SANS INSTITUTE*. Retrieved from <https://sansorg.egnyte.com/dl/ADSrVLiy5R>
- ☐ Cofense. (2020). *Phishing Review 2020*. Retrieved from <https://cofense.com/new-from-cofense-intelligence-q3-phishing-review-is-now-available/>
- ☐ Cohen, B., & Pietrzak, K. (2019). *The Chia Network Blockchain*. 1, 1–44. Retrieved from <https://www.chia.net/assets/ChiaGreenPaper.pdf>
- ☐ Crain, J., Opyrchal, L., & Prakash, A. (2010). Fighting phishing with trusted email. *ARES 2010 - 5th International Conference on Availability, Reliability, and Security*, 462–467. <https://doi.org/10.1109/ARES.2010.98>
- ☐ Crispin, M. (1994). *Distributed Electronic Mail Models in IMAP4*. Retrieved from <http://www.ietf.org/rfc/rfc1733.txt>
- ☐ Crispin, M. (2003). Internet Message Access Protocol - Version 4 rev1. *RFC3501*. Retrieved from <http://www.ietf.org/rfc/rfc3501.txt>
- ☐ Crispin, M. R. (1988). Interactive Mail Access Protocol: Version 2. *RFC1064*. Retrieved from <http://www.ietf.org/rfc/rfc1064.txt>
- ☐ Crocker, D. (1982). Standard For The Format of ARPA Internet Text Messages. *RFC822*, (822). Retrieved from <http://www.ietf.org/rfc/rfc822.txt>
- ☐ Crocker, D. (1995). *MIME Encapsulation of EDI Objects*. Retrieved from <http://www.ietf.org/rfc/rfc1767.txt>
- ☐ Crocker, D. (1997, May). *Mailbox Names for Common Services, Roles and Functions*. Retrieved from <http://www.ietf.org/rfc/rfc2142.txt>
- ☐ Crocker, D. (2009). *Internet Mail Architecture*. Retrieved from <http://www.ietf.org/rfc/rfc5598.txt>
- ☐ Crocker, D. (2011). RFC 6376 - DomainKeys Identified Mail (DKIM) Signatures. *Request for Comments*, 6376.
- ☐ Crocker, D., & Klyne, G. (2005). *Full-mode Fax Profile for Internet Mail (FFPIM)*. Retrieved from <http://www.ietf.org/rfc/rfc4142.txt>
- ☐ Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., ... Wattenhofer, R. (2016). On Scaling Decentralized Blockchains. In *Proc. 3rd Workshop on Bitcoin and Blockchain Research*.
- ☐ Crosby, M., Nachiappan, P., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain Technology - BEYOND BITCOIN. *Berkley Engineering*.
- ☐ CrowdStrike. (2021). Global Threat Report 2021. *CrowdStrike*, 75.
- ☐ Dabbagh, M., Choo, K. K. R., Beheshti, A., Tahir, M., & Safa, N. S. (2021). A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Computers and Security*, 100. <https://doi.org/10.1016/j.cose.2020.102078>
- ☐ Dan, K., Kitagawa, N., Sakuraba, S., & Yamai, N. (2019). Spam domain detection method using active DNS data and E-mail reception log. *Proceedings - International Computer Software and Applications Conference*, 1. <https://doi.org/10.1109/COMPSAC.2019.00133>
- ☐ DarkTrace. (2020). *Email Security Threat Report 4 key trends from spear phishing to credentials theft*. Retrieved from <https://www.darktrace.com/en/email-security-threat-report-2020/>
- ☐ Das, P., Erwig, A., Faust, S., Loss, J., & Riahi, S. (2021). The Exact Security of BIP32 Wallets. *Proceedings of the ACM Conference on Computer and Communications Security*. <https://doi.org/10.1145/3460120.3484807>
- ☐ de Haro-Olmo, F. J., Varela-Vaca, Á. J., & Álvarez-Bermejo, J. A. (2020). Blockchain from the perspective of privacy and anonymisation: A systematic literature review. *Sensors (Switzerland)*, 20(24). <https://doi.org/10.3390/s20247171>
- ☐ Decker, C., Seidel, J., & Wattenhofer, R. (2016). Bitcoin meets strong consistency. *ACM International Conference Proceeding Series*, 04-07-Janu. <https://doi.org/10.1145/2833312.2833321>
- ☐ Deloitte. (2020). *COVID-19 Global Cyber risks : Attack surfaces expand amid return to work efforts*. (7). Retrieved from <https://www2.deloitte.com/content/dam/Deloitte/global/Documents/About-Deloitte/COVID-19/gx-cyber-COVID-19-executive-briefing-issue-7-release-date-5.20.2020.pdf>
- ☐ Dennis, S. M. (2019). *Blockchain based Email Procedures (US Patent No. US010305833B1)*. United States of America.

- ☐ Dhinakaran, C., Lee, J. K., Nagamalai, D., & Chae, C. J. (2007). An empirical study of spam and spam vulnerable email accounts. *Proceedings of Future Generation Communication and Networking, Main Conference Papers, Vol 1, 1*, 407–412. <https://doi.org/10.1109/FGCN.2007.61>
- ☐ Di Francesco Maesa, D., & Mori, P. (2020). Blockchain 3.0 applications survey. *Journal of Parallel and Distributed Computing*, 138. <https://doi.org/10.1016/j.jpdc.2019.12.019>
- ☐ Dib, O., Brousmiche, K., Durand, A., Thea, E., & Hamida, B. (2018). Consortium Blockchains: Overview, Applications and Challenges. *International Journal on Advances in Telecommunications*, 11(1&2).
- ☐ Doglio, F. (2018). REST API Development with Node.js. In *REST API Development with Node.js*. <https://doi.org/10.1007/978-1-4842-3715-1>
- ☐ Dominguez Perez, L. J., Ibarra, L., Alejandro, G. F., Rumayor, A., & Lara-Alvarez, C. (2020). A loyalty program based on Waves blockchain and mobile phone interactions. *Knowledge Engineering Review*. <https://doi.org/10.1017/S0269888920000181>
- ☐ Douceur, J. R. (2002). The sybil attack. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2429. https://doi.org/10.1007/3-540-45748-8_24
- ☐ Dukhovni, V., & Hardaker, W. (2015). *The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance*. Retrieved from <http://www.ietf.org/rfc/rfc7671.txt>
- ☐ Durand, A., & Dupont, F. (1995). *SMTP 521 Reply Code*. Retrieved from <http://www.ietf.org/rfc/rfc1846.txt>
- ☐ Dwork, C., & Naor, N. (1993). Pricing via Processing, Or, Combatting Junk Mail, *Advances in Cryptology. In CRYPTO'92: Lecture Notes in Computer Science No. 740*.
- ☐ Ekparinya, P., Gramoli, V., & Jourjon, G. (2020). *The Attack of the Clones Against Proof-of-Authority*. <https://doi.org/10.14722/ndss.2020.24082>
- ☐ Ekramifard, A., Amintoosi, H., & Seno, A. H. (2020). A Systematic Literature Review on Blockchain-Based Solutions for IoT Security. In *Lecture Notes on Data Engineering and Communications Technologies (Vol. 45)*. https://doi.org/10.1007/978-3-030-37309-2_25
- ☐ El-Ansary, S., Alima, L. O., Brand, P., & Haridi, S. (2003). Efficient broadcast in structured P2P networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2735. https://doi.org/10.1007/978-3-540-45172-3_28
- ☐ Elkins, M., Torto, D. Del, Leven, R., & Roessler, T. (2001). MIME Security with OpenPGP. *RFC3156*. Retrieved from <http://www.ietf.org/rfc/rfc3156.txt>
- ☐ Elrom, E. (2019). NEO Blockchain and Smart Contracts. In *The Blockchain Developer*. https://doi.org/10.1007/978-1-4842-4847-8_7
- ☐ Elz, R., & Bush, R. (1997). *Clarifications to the DNS Specification*. Retrieved from <http://www.ietf.org/rfc/rfc2181.txt>
- ☐ Eric Conrad. (2021). Detecting Spam with Genetic Regular Expressions. *SANS INTITUTE*. Retrieved from <https://www.sans.org/reading-room/whitepapers/vpns/cryptanalysis-rsa-survey-1006>
- ☐ European Union Agency for Network and Information Security. (2019). *ENISA Threat Landscape 2014*. (April).
- ☐ Falkon, S. (2017). The Story of the DAO — Its History and Consequences. *Medium*, (May 2016).
- ☐ Fan, C., Ghaemi, S., Khazaei, H., & Musilek, P. (2020). Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access*, Vol. 8. <https://doi.org/10.1109/ACCESS.2020.3006078>
- ☐ Fan, C. I., Tseng, Y. F., Su, H. P., Hsu, R. H., & Kikuchi, H. (2019). Secure Hierarchical Bitcoin Wallet Scheme Against Privilege Escalation Attacks. *DSC 2018 - 2018 IEEE Conference on Dependable and Secure Computing*. <https://doi.org/10.1109/DESEC.2018.8625151>
- ☐ Fang, W., Chen, W., Zhang, W., Pei, J., Gao, W., & Wang, G. (2020). Digital signature scheme for information non-repudiation in blockchain: a state of the art review. *Eurasip Journal on Wireless Communications and Networking*, Vol. 2020. <https://doi.org/10.1186/s13638-020-01665-w>
- ☐ Farrell, S. (2009). Why Don't We Encrypt Our Email? *IEEE Internet Computing*, 13(1), 82–85. <https://doi.org/10.1109/MIC.2009.25>
- ☐ FBI's Internet Crime Complaint Center. (2021). 2021 Internet Crime Report. *2021 Internet Crime Report*, 1–33. Retrieved from https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf
- ☐ Felician, A. (2013). Tracking the Digital Traces. *Oeconomics of Knowledge*, 5(3).
- ☐ Ferdous, M. S., Chowdhury, M. J. M., & Hoque, M. A. (2021). A survey of consensus algorithms in public blockchain systems for crypto-currencies. *Journal of Network and Computer Applications*, Vol. 182. <https://doi.org/10.1016/j.jnca.2021.103035>

1. Ferretti, S., & D'Angelo, G. (2020). On the Ethereum blockchain structure: A complex networks theory perspective. *Concurrency Computation*, 32(12). <https://doi.org/10.1002/cpe.5493>
2. Finlow-Bates, K. (2018). A Lightweight Blockchain Consensus Protocol. *National Institute of Standard and Technology*.
3. Francison, E. A., Nascimento, M. P., Granatyr, J., Weffort, M. R., Lessing, O. R., & Scalabrin, E. E. (2019). A systematic literature review of blockchain architectures applied to public services. *Proceedings of the 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design, CSCWD 2019*. <https://doi.org/10.1109/CSCWD.2019.8791888>
4. Freed, N. (1999). *Gateways and MIME Security Multiparts*. Retrieved from <http://www.ietf.org/rfc/rfc2480.txt>
5. Freed, N., & Borenstein, N. (1996a). *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*. Retrieved from <http://www.ietf.org/rfc/rfc2049.txt>
6. Freed, N., & Borenstein, N. (1996b). *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Retrieved from <http://www.ietf.org/rfc/rfc2045.txt>
7. Freed, N., & Borenstein, N. (1996c). *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Retrieved from <http://www.ietf.org/rfc/rfc2046.txt>
8. Freed, N., & Klensin, J. (2005a). *Media Type Specifications and Registration Procedures*. Retrieved from <http://www.ietf.org/rfc/rfc4288.txt>
9. Freed, N., & Klensin, J. (2005b). *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. Retrieved from <http://www.ietf.org/rfc/rfc4289.txt>
10. Freed, N., Newman, D., Belissent, J., & Hoy, M. (1998). *The Batch SMTP Media Type*. Retrieved from <http://www.ietf.org/rfc/rfc2442.txt>
11. Fu, Z., Dong, P., & Ju, Y. (2020). An intelligent electric vehicle charging system for new energy companies based on consortium blockchain. *Journal of Cleaner Production*, 261. <https://doi.org/10.1016/j.jclepro.2020.121219>
12. Gallagher, P. (2015). Secure Hash Standard (SHA1/2) FIPS PUB 180-4. *Processing, FIPS PUB 1*(October). <https://doi.org/10.6028/NIST.FIPS.180-4>
13. Galvin, J., Murphy, S., Crocker, S., & Freed, N. (1995). Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted. *RFC1847*. Retrieved from <http://www.ietf.org/rfc/rfc1847.txt>
14. Gamage, H. T. M., Weerasinghe, H. D., & Dias, N. G. J. (2020). A Survey on Blockchain Technology Concepts, Applications, and Issues. *SN Computer Science*, 1(2). <https://doi.org/10.1007/s42979-020-00123-0>
15. Garcia-Garcia, J. A., Sanchez-Gomez, N., Lizcano, D., Escalona, M. J., & Wojdyski, T. (2020). Using Blockchain to Improve Collaborative Business Process Management: Systematic Literature Review. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3013911>
16. Garfinkle, S. L. (2003). Email-based identification and authentication: an alternative to PKI? *IEEE Security & Privacy Magazine*, 1(6), 20–26. <https://doi.org/10.1109/MSECP.2003.1253564>
17. Gauld, S., Ancoina, F.V., & Stadler, R. (2017). The Burst Dymaxion An Arbitrary Scalable , Energy Efficient and Anonymous Transaction Network Based on Colored Tangles.
18. Gellens, R. (1999). *ON-DEMAND MAIL RELAY (ODMR) SMTP with Dynamic IP Addresses*. Retrieved from <http://www.ietf.org/rfc/rfc2645.txt>
19. Gellens, R. (2006). *Mapping Between the Multimedia Messaging Service (MMS) and Internet Mail*. Retrieved from <http://www.ietf.org/rfc/rfc4356.txt>
20. Gellens, R., & Klensin, J. (2006). *Message Submission for Mail*. Retrieved from <http://www.ietf.org/rfc/rfc4409.txt>
21. Georgiou, I., Nell, J. G., & Kokkinaki, A. I. (2020). Blockchain for Smart Cities: A Systematic Literature Review. *Lecture Notes in Business Information Processing*, 402. https://doi.org/10.1007/978-3-030-63396-7_12
22. Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., & Čapkun, S. (2016). On the security and performance of Proof of Work blockchains. *Proceedings of the ACM Conference on Computer and Communications Security, 24-28-October-2016*. <https://doi.org/10.1145/2976749.2978341>
23. Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2). <https://doi.org/10.1145/564585.564601>
24. Gol, D. A. (2019). An Analysis of Consensus Algorithms for the Blockchain Technology. *International Journal for Research in Applied Science and Engineering Technology*, 7(2). <https://doi.org/10.22214/ijraset.2019.2096>

- ☐ Golan Gueta, G., Abraham, I., Grossman, S., Malkhi, D., Pinkas, B., Reiter, M., ... Tomescu, A. (2019). SBFT: A Scalable and Decentralized Trust Infrastructure. *Proceedings - 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019*. <https://doi.org/10.1109/DSN.2019.00063>
- ☐ Gong, Y., van Engelenburg, S., & Janssen, M. (2021). A reference architecture for blockchain-based crowdsourcing platforms. *Journal of Theoretical and Applied Electronic Commerce Research*, 16(4). <https://doi.org/10.3390/jtaer16040053>
- ☐ Grassi, P. A., Garcia, M. E., & Fenton, J. L. (2020). NIST 800-63-3: Digital Identity Guidelines. *NIST Special Publications*.
- ☐ GridCoin. (2018). *GRIDCOIN [White Paper]*. Retrieved from <https://gridcoin.us/assets/docs/whitepaper.pdf>
- ☐ Grover, P., Kar, A. K., & Vigneswara Ilavarasan, P. (2018). Blockchain for businesses: A systematic literature review. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11195 LNCS. https://doi.org/10.1007/978-3-030-02131-3_29
- ☐ Guenther, P., & Showalter, T. (2008). *Sieve: An Email Filtering Language*. Retrieved from <http://www.ietf.org/rfc/rfc5228.txt>
- ☐ Gupta, B. B., Tewari, A., Jain, A. K., & Agrawal, D. P. (2017). Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications*, 28(12), 3629–3654. <https://doi.org/10.1007/s00521-016-2275-y>
- ☐ Guustaaf, E., Rahardja, U., Aini, Q., Maharani, H. W., & Santoso, N. A. (2021). Blockchain-based Education Project. *Aptisi Transactions on Management (ATM)*, 5(1). <https://doi.org/10.33050/atm.v5i1.1433>
- ☐ Haber, S., & Stornetta, W. S. (1991). How to time-stamp a digital document. *Journal of Cryptology*, 3(2). <https://doi.org/10.1007/BF00196791>
- ☐ Hahn, T., Kunz, T., Schneider, M., & Vowé, S. (2012). Vulnerabilities through usability pitfalls in cloud services: Security problems due to unverified email addresses. *Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012*, 850–856. <https://doi.org/10.1109/TrustCom.2012.297>
- ☐ Hamid, I. R. A., & Abawajy, J. (2011). Phishing email feature selection approach. *Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2011, 8th IEEE Int. Conf. on Embedded Software and Systems, ICESS 2011, 6th Int. Conf. on FCST 2011*, 916–921. <https://doi.org/10.1109/TrustCom.2011.126>
- ☐ Hansen, T., & Klensin, J. (2008). *A Registry for SMTP Enhanced Mail System Status Codes*. Retrieved from <http://www.ietf.org/rfc/rfc5248.txt>
- ☐ Hansen, T., & Vaudreuil, G. (2004, May). *Message Disposition Notification*. Retrieved from <http://www.ietf.org/rfc/rfc3798.txt>
- ☐ Hao, W., Zeng, J., Dai, X., Xiao, J., Hua, Q. S., Chen, H., ... Jin, H. (2020). Towards a Trust-Enhanced Blockchain P2P Topology for Enabling Fast and Reliable Broadcast. *IEEE Transactions on Network and Service Management*, 17(2). <https://doi.org/10.1109/TNSM.2020.2980303>
- ☐ Hasan, J. (2019). Overview and Applications of Zero Knowledge Proof (ZKP). In *IJCSN-International Journal of Computer Science and Network* (Vol. 8).
- ☐ Hassanzadeh-Nazarabadi, Y., Küpçü, A., & Özkasap, Ö. (2021). LightChain: Scalable DHT-Based Blockchain. *IEEE Transactions on Parallel and Distributed Systems*, 32(10). <https://doi.org/10.1109/TPDS.2021.3071176>
- ☐ Hatzivasilis, G. (2017). Password-hashing status. *Cryptography*, 1(2). <https://doi.org/10.3390/cryptography1020010>
- ☐ He, P., Tang, D., & Wang, J. (2020). Stake Centralization in Decentralized Proof-of-Stake Blockchain Network. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3609817>
- ☐ Herzberg, A. (2009). DNS-based email sender authentication mechanisms: A critical review. *Computers & Security*, 28(8), 731–742. <https://doi.org/10.1016/j.cose.2009.05.002>
- ☐ Hewa, T., Ylianttila, M., & Liyanage, M. (2021). Survey on blockchain based smart contracts: Applications, opportunities and challenges. *Journal of Network and Computer Applications*, Vol. 177. <https://doi.org/10.1016/j.jnca.2020.102857>
- ☐ Hinarejos, M. F., & Ferrer-Gomila, J. (2020). A Solution for Secure Multi-Party Certified Electronic Mail Using Blockchain. *IEEE Access*, 8, 102997–103006. <https://doi.org/10.1109/ACCESS.2020.2998679>
- ☐ Hoffman, P. (2002). SMTP Service Extension for Secure SMTP over Transport Layer Security. *RFC3207*. Retrieved from <http://www.ietf.org/rfc/rfc3207.txt>

- ☐ Holbrook, J. (2020). Understanding Enterprise Blockchain Consensus. In *Architecting Enterprise Blockchain Solutions*. <https://doi.org/10.1002/9781119557722.ch4>
- ☐ Houttuin, J. (1993). *A Tutorial on Gatewaying between X.400 and Internet Mail*. Retrieved from <http://www.ietf.org/rfc/rfc1506.txt>
- ☐ Huang, H., Lin, J., Zheng, B., Zheng, Z., & Bian, J. (2020). When Blockchain Meets Distributed File Systems: An Overview, Challenges, and Open Issues. *IEEE Access*, 8, 50574–50586. <https://doi.org/10.1109/ACCESS.2020.2979881>
- ☐ Huang, K., Zhang, X., Mu, Y., Wang, X., Yang, G., Du, X., ... Guizani, M. (2019). Building redactable consortium blockchain for industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, 15(6). <https://doi.org/10.1109/TII.2019.2901011>
- ☐ Hutzler, C., Crocker, D., Resnick, P., Allman, E., & Finch, T. (2007). Email Submission Operations: Access and Accountability Requirements. *RFC5068*. Retrieved from <http://www.ietf.org/rfc/rfc5068.txt>
- ☐ Idris, I., Selamat, A., & Omatu, S. (2014). Hybrid email spam detection model with negative selection algorithm and differential evolution. *Engineering Applications of Artificial Intelligence*, 28. <https://doi.org/10.1016/j.engappai.2013.12.001>
- ☐ Idris, I., Selamat, A., Thanh Nguyen, N., Omatu, S., Krejcar, O., Kuca, K., & Penhaker, M. (2015). A combined negative selection algorithm-particle swarm optimization for an email spam detection system. *Engineering Applications of Artificial Intelligence*, 39. <https://doi.org/10.1016/j.engappai.2014.11.001>
- ☐ Inc. Agari Data. (2021). *H1 2021 Email Fraud & Identity Deception Trends*. Retrieved from <https://www.agari.com/insights/ebooks/h1-2021-email-fraud-identity-deception-trends-report/>
- ☐ Irresberger, F., John, K., & Saleh, F. (2020). The Public Blockchain Ecosystem: An Empirical Analysis. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3592849>
- ☐ Ismail, L., & Materwala, H. (2019). A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions. *Symmetry*, 11(10). <https://doi.org/10.3390/sym11101198>
- ☐ J. Schaad, B. Ramsdell, & Turner, S. (2019). Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0. *IETF Request For Comments*.
- ☐ Jacomme, C., & Kremer, S. (2021). An Extensive Formal Analysis of Multi-factor Authentication Protocols. *ACM Transactions on Privacy and Security*, 24(2). <https://doi.org/10.1145/3440712>
- ☐ Jindal, N., & Liu, B. (2008). Opinion spam and analysis. *Proceedings of the International Conference on Web Search and Web Data Mining WSDM 08*, 219. <https://doi.org/10.1145/1341531.1341560>
- ☐ Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1), 36–63. Retrieved from <http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf>
- ☐ Jokić, S. (2019). Analysis and security of crypto currency wallets. *ZBORNIK RADOVA UNIVERZITETA SINERGIJA*, 19(4). <https://doi.org/10.7251/zrsng1801102j>
- ☐ Kambourakis, G., Gil, G. D., & Sanchez, I. (2020). What Email Servers Can Tell to Johnny: An Empirical Study of Provider-to-Provider Email Security. *IEEE Access*, 8, 130066–130081. <https://doi.org/10.1109/ACCESS.2020.3009122>
- ☐ Kannan, L., & Jebakumar, R. (2020). Public Sender Score System(S3) by ESPs for Email Spam Mitigation with Score Management in Mobile Application. *International Journal of Interactive Mobile Technologies*, 14(17). <https://doi.org/10.3991/ijim.v14i17.16609>
- ☐ Karamachoski, J., Marina, N., & Taskov, P. (2020). Blockchain-Based Application for Certification Management. *Tehnički Glasnik*, 14(4). <https://doi.org/10.31803/tg-20200811113729>
- ☐ Karantias, K., Kiayias, A., & Zindros, D. (2020). Proof-of-Burn. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12059 LNCS. https://doi.org/10.1007/978-3-030-51280-4_28
- ☐ Karim, A., Azam, S., Shanmugam, B., Kannoopatti, K., & Alazab, M. (2019). A comprehensive survey for intelligent spam email detection. *IEEE Access*, 7, 168261–168295. <https://doi.org/10.1109/ACCESS.2019.2954791>
- ☐ Kasten, J. (2020). Blockchain on the Farm: A Systematic Literature Review. *Journal of Strategic Innovation and Sustainability*, 15(2). <https://doi.org/10.33423/jsis.v15i2.2893>
- ☐ Katz, J., & Lindell, Y. (2014). Introduction to Modern Cryptography. In *Introduction to Modern Cryptography*. <https://doi.org/10.1201/b17668>

- ☐ Khacef, K., & Pujolle, G. (2019). Secure Peer-to-Peer Communication Based on Blockchain. *Advances in Intelligent Systems and Computing*, 927. https://doi.org/10.1007/978-3-030-15035-8_64
- ☐ Khettry, A. R., Patil, K. R., & Basavaraju, A. C. (2021). A Detailed Review on Blockchain and Its Applications. *SN Computer Science*, 2(1). <https://doi.org/10.1007/s42979-020-00366-x>
- ☐ Kiayias, A., & Panagiotakos, G. (2015). Speed-Security Tradeoffs in Blockchain Protocols. *Cryptology EPrint Archive*.
- ☐ King, S. (2013). Primecoin: Cryptocurrency with Prime Number Proof-of-Work. *King, Sunny*, 4(2).
- ☐ King, S., & Nadal, S. (2012). *PPCoin-Paper [WhitePaper]*. Retrieved from <https://www.peercoin.net/whitepapers/peercoin-paper.pdf>
- ☐ Klensin, J. (2001). Simple Mail Transfer Protocol. *RFC2821*. Retrieved from <http://www.ietf.org/rfc/rfc2821.txt>
- ☐ Klensin, J. (2008). *Simple Mail Transfer Protocol*. Retrieved from <http://www.ietf.org/rfc/rfc5321.txt>
- ☐ Klensin, J. (2015). *SMTP 521 and 556 Reply Codes*. Retrieved from <http://www.ietf.org/rfc/rfc7504.txt>
- ☐ Klyne, G., Iwazaki, R., & Crocker, D. (2002). *Content Negotiation for Messaging Services based on Email*. Retrieved from <http://www.ietf.org/rfc/rfc3297.txt>
- ☐ Kobeissi, N. (2018). *An Analysis of the ProtonMail Cryptographic Architecture*. 1–14. Retrieved from <https://protonmail.com>.
- ☐ Kogias, D. G., Leligou, H. C., Xevgenis, M., Polychronaki, M., Katsadouros, E., Loukas, G., ... Patrikakis, C. Z. (2019). Toward a Blockchain-Enabled Crowdsourcing Platform. *IT Professional*, 21(5). <https://doi.org/10.1109/MITP.2019.2929503>
- ☐ Kolb, J., AbdelBaky, M., Katz, R. H., & Culler, D. E. (2020). Core Concepts, Challenges, and Future Directions in Blockchain: A Centralized Tutorial. *ACM Comput. Surv.*, 53(1). <https://doi.org/10.1145/3366370>
- ☐ Kölvar, M., Poola, M., & Rull, A. (2016). Smart contracts. In *The Future of Law and eTechnologies* (pp. 133–147). https://doi.org/10.1007/978-3-319-26896-5_7
- ☐ Kondru, K. K., Saranya, R., & Chacko, A. (2021). A Review of Distributed Supercomputing Platforms Using Blockchain. In *Lecture Notes in Networks and Systems* (Vol. 127). https://doi.org/10.1007/978-981-15-4218-3_13
- ☐ König, T., Duran, E., Fessler, N., & Alton, R. (2018). *The Proof-of-Cooperation Blockchain FairCoin*. 2(July), 1–9. Retrieved from https://faircoin.org/sites/default/files/FairCoin2_whitepaper_V1.2.pdf
- ☐ Konno, K., Kitagawa, N., & Yamai, N. (2020). False Positive Detection in Sender Domain Authentication by DMARC Report Analysis. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3388176.3388217>
- ☐ Kovacs, A., Karakatsanis, I., & Svetinovic, D. (2014). Argumentation-based security requirements analysis: Bitmessage case study. *Proceedings - 2014 IEEE International Conference on Internet of Things, IThings 2014, 2014 IEEE International Conference on Green Computing and Communications, GreenCom 2014 and 2014 IEEE International Conference on Cyber-Physical-Social Computing, CPS 20, (iThings)*, 408–414. <https://doi.org/10.1109/iThings.2014.74>
- ☐ Kovalchuk, L., Kaidalov, D., Nastenkov, A., Shevtsov, O., Rodinko, M., & Oliynykov, R. (2018). Number of confirmation blocks for Bitcoin and GHOST consensus protocols on networks with delayed message delivery. *CRYBLOCK 2018 - Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Part of MobiSys 2018*. <https://doi.org/10.1145/3211933.3211941>
- ☐ Kraus, N., Kraus, K., & Manzhura, O. (2019). *Newest Digital Technology in Management of National Economic System*. <https://doi.org/10.2991/smtesm-19.2019.1>
- ☐ Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC: Keyed-Hashing for Message Authentication*. Retrieved from <http://www.ietf.org/rfc/rfc2104.txt>
- ☐ Kucherawy, M., & Zwicky, E. (2015). *Domain-based Message Authentication, Reporting, and Conformance (DMARC)*. Retrieved from <http://www.ietf.org/rfc/rfc7489.txt>
- ☐ Kudlick, M. D. (1973a). Meeting announcement to discuss a network mail system. *RFC453*. Retrieved from <http://www.ietf.org/rfc/rfc453.txt>
- ☐ Kudlick, M. D. (1973b). Network mail meeting summary. *RFC469*. Retrieved from <http://www.ietf.org/rfc/rfc469.txt>
- ☐ Kuhlman, C., Middleton, D., Bollen, B., & Davis, S. (2017). Hyperledger Burrow Proposal. *Hyperledger.Org*, (March 28).

1. Kumaraguru, P., Rhee, Y., Acquisti, A., Cranor, L. F., Hong, J., & Nunge, E. (2007). Protecting people from phishing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '07*, 905. <https://doi.org/10.1145/1240624.1240760>
2. Kummer, S., Herold, D. M., Dobrovnik, M., Mikl, J., & Schäfer, N. (2020). A systematic review of blockchain literature in logistics and supply chain management: Identifying research questions and future directions. *Future Internet*, 12(3). <https://doi.org/10.3390/fi12030060>
3. Kwak, K. H., Kong, J. T., Cho, S. I., Phuong, H. T., & Gim, G. Y. (2019). A study on the design of efficient private blockchain. *Studies in Computational Intelligence*, 787. https://doi.org/10.1007/978-3-319-96806-3_8
4. Laabs, M., & Dukanović, S. (2021). Blockchain in Industrie 4.0: Beyond cryptocurrency. *IT - Information Technology*, 60(3). <https://doi.org/10.1515/itit-2018-0011>
5. Labbi, M., Kannouf, N., Chahid, Y., Benabdellah, M., & Azizi, A. (2019). *Blockchain-Based PKI for Content-Centric Networking*. https://doi.org/10.1007/978-3-030-11196-0_54
6. Lafourcade, P., & Lombard-Platet, M. (2020). About blockchain interoperability. *Information Processing Letters*, 161. <https://doi.org/10.1016/j.ipl.2020.105976>
7. Lai, R., & Lee Kuo Chuen, D. (2018). Blockchain-From Public to Private. In *Handbook of Blockchain, Digital Finance, and Inclusion* (Vol. 2). <https://doi.org/10.1016/B978-0-12-812282-2.00007-3>
8. Lai, Y. (2018). NEM [White paper]. *Imid 2009*, (159679), 1069–1072.
9. Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3). <https://doi.org/10.1145/357172.357176>
10. Larimer, D. (2017). *EOS [White Paper v2]*. Retrieved from <https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md>
11. Lee, H., Girish, A., van Rijswijk-Deij, R., Kwon, T., & Chung, T. (2020). A longitudinal and comprehensive study of the DANE ecosystem in email. *Proceedings of the 29th USENIX Security Symposium*, 613–630.
12. Lee, M. S., & Kim, K. J. (2021). Survey on blockchain evolution and proof-of-stake consensus algorithm. *International Journal of Engineering Trends and Technology*, Vol. 69. <https://doi.org/10.14445/22315381/IJETT-V69I4P220>
13. Lee, W.-M. (2019). Using the web3.js APIs. In *Beginning Ethereum Smart Contracts Programming*. https://doi.org/10.1007/978-1-4842-5086-0_8
14. Leng, J., Zhou, M., Zhao, L. J., Huang, Y., & Bian, Y. (2020). Blockchain Security: A Survey of Techniques and Research Directions. *IEEE Transactions on Services Computing*. <https://doi.org/10.1109/TSC.2020.3038641>
15. Levison, L. (2014). Secrets, lies and Snowden's email: why I was forced to shut down Lavabit. *The Guardian*.
16. Levison, L., & Delgado, Ri. (2021). Lavabit. Retrieved from <https://lavabit.com>
17. Li, K., Li, H., Hou, H., Li, K., & Chen, Y. (2018). Proof of Vote: A High-Performance Consensus Protocol Based on Vote Mechanism & Consortium Blockchain. *Proceedings - 2017 IEEE 19th Intl Conference on High Performance Computing and Communications, HPCC 2017, 2017 IEEE 15th Intl Conference on Smart City, SmartCity 2017 and 2017 IEEE 3rd Intl Conference on Data Science and Systems, DSS 2017, 2018-Janua*. <https://doi.org/10.1109/HPCC-SmartCity-DSS.2017.61>
18. Li, S., Wang, N., & Du, X. (2020). Privacy protection mechanism of on-demand disclosure on blockchain. *Chinese Journal of Network and Information Security*, 6(3), 19–29. <https://doi.org/10.11959/j.issn.2096-109x.2020033>
19. Li, Xiaofan, & Whinston, A. B. (2020). Analyzing Cryptocurrencies. *Information Systems Frontiers*, 22(1). <https://doi.org/10.1007/s10796-019-09966-2>
20. Li, Xiaoqi, Jiang, P., Chen, T., Luo, X., & Wen, Q. (2020). A survey on the security of blockchain systems. *Future Generation Computer Systems*, 107. <https://doi.org/10.1016/j.future.2017.08.020>
21. Li, Y., Susilo, W., Yang, G., Yu, Y., Du, X., Liu, D., & Guizani, N. (2019). Toward privacy and regulation in blockchain-based cryptocurrencies. *IEEE Network*, 33(5). <https://doi.org/10.1109/MNET.2019.1800271>
22. Liang, W., Fan, Y., Li, K.-C., Zhang, D., & Gaudiot, J.-L. (2020). Secure Data Storage and Recovery in Industrial Blockchain Network Environments. *IEEE Transactions on Industrial Informatics*, 16(10), 6543–6552. <https://doi.org/10.1109/TII.2020.2966069>
23. Lina, Z., Jiang, M. H., & Jiang, M. H. (2012). The Security Email Based on Smart Card. *Physics Procedia*, 33, 1634–1639. <https://doi.org/10.1016/j.phpro.2012.05.263>
24. Lindberg, G. (1999). Anti-Spam Recommendations for SMTP MTAs. *RFC2505*. Retrieved from <http://www.ietf.org/rfc/rfc2505.txt>

- ☐ Liu, L., Han, M., Zhou, Y., Parizi, R. M., & Korayem, M. (2020). Blockchain-based certification for education, employment, and skill with incentive mechanism. In *Advances in Information Security* (Vol. 79). https://doi.org/10.1007/978-3-030-38181-3_14
- ☐ Lo, S. K., Liu, Y., Chia, S. Y., Xu, X., Lu, Q., Zhu, L., & Ning, H. (2019). Analysis of Blockchain Solutions for IoT: A Systematic Literature Review. *IEEE Access*, 7. <https://doi.org/10.1109/ACCESS.2019.2914675>
- ☐ Lone, A. H., & Naaz, R. (2021). Applicability of Blockchain smart contracts in securing Internet and IoT: A systematic literature review. *Computer Science Review*, Vol. 39. <https://doi.org/10.1016/j.cosrev.2020.100360>
- ☐ Long, S. (2019). A Comparative Analysis of the Application of Hashing Encryption Algorithms for MD5, SHA-1, and SHA-512. *Journal of Physics: Conference Series*, 1314(1). <https://doi.org/10.1088/17426596/1314/1/012210>
- ☐ Lozupone, V. (2018). Analyze encryption and public key infrastructure (PKI). *International Journal of Information Management*, 38(1). <https://doi.org/10.1016/j.ijinfomgt.2017.08.004>
- ☐ Ma, Y., Sun, Y., Lei, Y., Qin, N., & Lu, J. (2020). A survey of blockchain technology on security, privacy, and trust in crowdsourcing services. *World Wide Web*, 23(1). <https://doi.org/10.1007/s11280-019-00735-4>
- ☐ Madi, N. K. M., Salehian, S., Masoumian, F., & Abdullah, A. (2012). Implementation of secure email server in cloud environment. *2012 International Conference on Computer and Communication Engineering, ICCCE 2012*, 28-32. <https://doi.org/10.1109/ICCCE.2012.6271146>
- ☐ Maes, S., & Melnikov, A. (2006). *Internet Email to Support Diverse Service Environments (Lemonade) Profile*. Retrieved from <http://www.ietf.org/rfc/rfc4550.txt>
- ☐ Mahjabin, T., Xiao, Y., Sun, G., & Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, 13(12). <https://doi.org/10.1177/1550147717741463>
- ☐ Mandloi, J., & Bansal, P. (2020). An empirical review on blockchain smart contracts: Application and challenges in implementation. *International Journal of Computer Networks and Applications*, Vol. 7. <https://doi.org/10.22247/ijcna/2020/195718>
- ☐ Mardan, A. (2018). Boosting Node.js and MongoDB with Mongoose. In *Practical Node.js*. https://doi.org/10.1007/978-1-4842-3039-8_7
- ☐ Margolis, D., Brotman, A., Ramakrishnan, B., Jones, J., & Risher, M. (2018). SMTP TLS Reporting Abstract. *IETF Request For Comments*, 1-26.
- ☐ Margolis, D., Risher, M., Google, I., Ramakrishnan, B., Oath, I., Brotman, A., ... Microsoft, I. (2018). SMTP MTA Strict Transport Security (MTA-STS). *RFC8461*, 1-29.
- ☐ Masood, F., & Faridi, A. R. (2021). One-Stop-Shop For The Security Of IoT With Blockchain: A Systematic literature review. *International Journal of Sensors, Wireless Communications and Control*, 11. <https://doi.org/10.2174/2210327911666210308155600>
- ☐ Mayer, A. H., da Costa, C. A., & Righi, R. da R. (2020). Electronic health records in a Blockchain: A systematic review. *Health Informatics Journal*, 26(2). <https://doi.org/10.1177/1460458219866350>
- ☐ Mayer, H. (2016). ECDSA Security in Bitcoin and Ethereum: a Research Survey.
- ☐ Maymounkov, P., & Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the XOR metric. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2429. https://doi.org/10.1007/3-540-45748-8_5
- ☐ McAfee. (2020). McAfee Labs Threats Report. *McAfee Labs Threats Report November 2020*, pp. 1-27. Retrieved from <https://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2015.pdf>
- ☐ McDowell, M. (2009). Understanding Denial-of-Service Attacks. Retrieved from <https://www.cisa.gov/uscert/ncas/tips/ST04-015>
- ☐ Menegay, P., Salyers, J., & College, G. (2019). Secure Communications Using Blockchain Technology. *Proceedings - IEEE Military Communications Conference MILCOM, 2019-Octob.* <https://doi.org/10.1109/MILCOM.2018.8599771>
- ☐ Meneghetti, A., Sala, M., & Taufer, D. (2020). A survey on pow-based consensus. *Annals of Emerging Technologies in Computing*, Vol. 4. <https://doi.org/10.33166/AETiC.2020.01.002>
- ☐ Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. In *Handbook of Applied Cryptography*. <https://doi.org/10.5860/choice.34-4512>
- ☐ Merkle, R. C. (1982). Method of Providing Digital Signatures.

- ☐ Merkle, R. C. (1988). A digital signature based on a conventional encryption function. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 293 LNCS. https://doi.org/10.1007/3-540-48184-2_32
- ☐ Microsoft. (2017). *The Coco Framework [WhitePaper]*. Retrieved from [https://github.com/EdisonDevS/Blockchain-1/blob/master/Coco Framework whitepaper.pdf](https://github.com/EdisonDevS/Blockchain-1/blob/master/Coco%20Framework%20whitepaper.pdf)
- ☐ Microsoft, I. (2008). About Simple Internet (RFC 822) Messages. Retrieved from [https://msdn.microsoft.com/en-us/library/ms527539\(v=exch.10\).aspx](https://msdn.microsoft.com/en-us/library/ms527539(v=exch.10).aspx)
- ☐ Miguel, P. A. (2005). Calidad en Investigación (2ªparte): Aproximación metodológica a la mejora de las actividades de investigación. *Revista Madri+ D*, (33), 3.
- ☐ Mirkin, M., Ji, Y., Pang, J., Klages-Mundt, A., Eyal, I., & Juels, A. (2020). BDoS: Blockchain Denial-of-Service. *Proceedings of the ACM Conference on Computer and Communications Security*. <https://doi.org/10.1145/3372297.3417247>
- ☐ Mockapetris, P. V. (1987a). *Domain names - concepts and facilities*. Retrieved from <http://www.ietf.org/rfc/rfc1034.txt>
- ☐ Mockapetris, P. V. (1987b). *Domain names - implementation and specification*. Retrieved from <http://www.ietf.org/rfc/rfc1035.txt>
- ☐ Moecke, C. T., & Volkamer, M. (2013). Usable secure email communications: criteria and evaluation of existing approaches. *Information Management & Computer Security*, 21(1), 41–52. <https://doi.org/10.1108/09685221311314419>
- ☐ Monika, Tomar, T., Kumar, V., & Kumar, Y. (2020). Implementation of elliptic - Curve cryptography. *International Journal of Electrical Engineering and Technology*, 11(2).
- ☐ Monrat, A. A., Schelén, O., & Andersson, K. (2019). A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, Vol. 7. <https://doi.org/10.1109/ACCESS.2019.2936094>
- ☐ Moore, K. (1996). *MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*. Retrieved from <http://www.ietf.org/rfc/rfc2047.txt>
- ☐ Moore, K. (2003). *Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)*. Retrieved from <http://www.ietf.org/rfc/rfc3461.txt>
- ☐ Moore, K. (2004). *Recommendations for Automatic Responses to Electronic Mail*. Retrieved from <http://www.ietf.org/rfc/rfc3834.txt>
- ☐ Moore, K., & Vaudreuil, G. (2003). *An Extensible Message Format for Delivery Status Notifications*. Retrieved from <http://www.ietf.org/rfc/rfc3464.txt>
- ☐ More, N., & Motwani, D. (2021). A blockchain-based decentralized framework for crowdsourcing. *Advances in Intelligent Systems and Computing*, 1200 AISC. https://doi.org/10.1007/978-3-030-51859-2_41
- ☐ Moriarty, K., Nystrom, M., Parkinson, S., Rusch, A., & Scott, M. (2014). *PKCS #12: Personal Information Exchange Syntax v1.1*. Retrieved from <http://www.ietf.org/rfc/rfc7292.txt>
- ☐ Möser, M. (2013). Anonymity of Bitcoin transactions. An analysis of mixing services. *Münster Bitcoin Conference, 17-18 July 2013*.
- ☐ Müller, J., Brinkmann, M., Poddebniak, D., Böck, H., Schinzel, S., Somorovsky, J., & Schwenk, J. (2019). "Johnny, you are fired!" - Spoofing OpenPGP and S/MIME signatures in emails. *Proceedings of the 28th USENIX Security Symposium*.
- ☐ Müller, J., Brinkmann, M., Poddebniak, D., Schinzel, S., & Schwenk, J. (2019). Re: What's Up Johnny?: Covert Content Attacks on Email End-to-End Encryption. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11464 LNCS. https://doi.org/10.1007/978-3-030-21568-2_2
- ☐ Müller, J., Brinkmann, M., Poddebniak, D., Schinzel, S., & Schwenk, J. (2020). Mailto: Me Your Secrets. on Bugs and Features in Email End-to-End Encryption. *2020 IEEE Conference on Communications and Network Security, CNS 2020*. <https://doi.org/10.1109/CNS48642.2020.9162218>
- ☐ Myers, J. (1996). *Local Mail Transfer Protocol*. Retrieved from <http://www.ietf.org/rfc/rfc2033.txt>
- ☐ Myers, J., & Rose, M. (1996, May). *Post Office Protocol - Version 3*. Retrieved from <http://www.ietf.org/rfc/rfc1939.txt>
- ☐ Nakamoto, S. (2008). Bitcoin: a peer-to-peer electronic cash system. Retrieved from <https://bitcoin.org/bitcoin.pdf>

- ☐ Nathan, J., & Jacobs, B. (2020). Blockchain consortium networks: Adding security and trust in financial services. *Journal of Corporate Accounting & Finance*, 31(2). <https://doi.org/10.1002/jcaf.22428>
- ☐ National Institute of Standards and Technology. (2009). Digital Signature Standard (DSS). *Fips Pub 186-4*, (July), 1–119. <https://doi.org/10.6028/NIST.FIPS.186-4>
- ☐ Neo. (2017). *Neo [Whitepaper]*. Retrieved from <https://docs.neo.org/docs/en-us/basic/whitepaper.html>
- ☐ Neufeld, G., & Baer, J. (1998). The Use of URLs as Meta-Syntax for Core Mail List Commands and their Transport through Message Header Fields. *RFC2369*. Retrieved from <http://www.ietf.org/rfc/rfc2369.txt>
- ☐ Neugebauer, R., Hippmann, S., Leis, M., & Landherr, M. (2016). Industrie 4.0 - From the Perspective of Applied Research. *Procedia CIRP*, 57. <https://doi.org/10.1016/j.procir.2016.11.002>
- ☐ Nijse, J., & Litchfield, A. (2020). A taxonomy of blockchain consensus methods. *Cryptography*, 4(4). <https://doi.org/10.3390/cryptography4040032>
- ☐ Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., & Koucheryavy, Y. (2018). Multi-factor authentication: A survey. *Cryptography*, 2(1). <https://doi.org/10.3390/cryptography2010001>
- ☐ Oswald, E. (2005). Introduction to elliptic curve cryptography. Institute for Applied Information Processing and Communication, Graz University Technology. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.2944&rep=rep1&type=pdf>
- ☐ P4Titan. (2014). Slimcoin: A Peer-to-Peer Crypto-Currency with Proof-of-Burn. *Whitepaper*.
- ☐ Pahlajani, S., Kshirsagar, A., & Pachghare, V. (2019). Survey on Private Blockchain Consensus Algorithms. *Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICICT 2019*. <https://doi.org/10.1109/ICICT1.2019.8741353>
- ☐ Pal, O., Alam, B., Thakur, V., & Singh, S. (2021). Key management for blockchain technology. *ICT Express*, 7(1). <https://doi.org/10.1016/j.icte.2019.08.002>
- ☐ Palatinus, R., & Marek, P. (2014). (bip-0044) Multi-Account Hierarchy for Deterministic Wallets.
- ☐ Panda, S. K., & Satapathy, S. C. (2021). *An Investigation into Smart Contract Deployment on Ethereum Platform Using Web3.js and Solidity Using Blockchain*. https://doi.org/10.1007/978-981-16-0171-2_52
- ☐ Park, A., & Li, H. (2021). The effect of blockchain technology on supply chain sustainability performances. *Sustainability (Switzerland)*, 13(4). <https://doi.org/10.3390/su13041726>
- ☐ Parmar, M., & Jit, H. (2021). Comparative Analysis of Secured Hash Algorithms for Blockchain Technology and Internet of Things. *International Journal of Advanced Computer Science and Applications*, 12(3). <https://doi.org/10.14569/ijacsa.2021.0120335>
- ☐ Payne, B. D., & Edwards, W. K. (2008). A brief introduction to usable Security. *IEEE Internet Computing*, 12(3), 13–20. <https://doi.org/10.1109/MIC.2008.50>
- ☐ Percival, C. (2009). Stronger key derivation via sequential memory-hard functions. *Self-Published*.
- ☐ Phil Zimmermann, Jon Callas, Mike Janke, and L. L. (2018). *Dark Internet Mail Environment* (Vol. 1).
- ☐ Piedrahita Castillo, D., Machío Regidor, F., Bermejo Higuera, J., Bermejo Higuera, J. R., & Sicilia Montalvo, J. A. (2020). A New Mail System for Secure Data Transmission in Cyber Physical Systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. <https://doi.org/10.1142/S0218488520400127>
- ☐ Pieter, W. (2012). (bip-0032) Hierarchical Deterministic Wallets. Retrieved from GitHub website: <https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>
- ☐ Pittalia, P. P. (2019). A Comparative Study of Hash Algorithms in Cryptography. *International Journal of Computer Science and Mobile Computing*, 8(6).
- ☐ Poddebniak, D., Dresen, C., Müller, J., Ising, F., Schinzel, S., Friedberger, S., ... Schwenk, J. (2018). Efail: Breaking S/MIME and OpenPGP email encryption using exfiltration channels. *Proceedings of the 27th USENIX Security Symposium*.
- ☐ Polge, J., Robert, J., & Le Traon, Y. (2020). Permissioned blockchain frameworks in the industry: A comparison. *ICT Express*. <https://doi.org/10.1016/j.icte.2020.09.002>
- ☐ Postel, J. (1979). Out-of-net host addresses for mail. *RFC754*. Retrieved from <http://www.ietf.org/rfc/rfc754.txt>
- ☐ Postel, J. (1981a). *Internet Protocol*. Retrieved from <http://www.ietf.org/rfc/rfc791.txt>
- ☐ Postel, J. (1981b). Simple Mail Transfer Protocol. *RFC788*. Retrieved from <http://www.ietf.org/rfc/rfc788.txt>
- ☐ Postel, J. (1982a). Computer mail meeting notes. *RFC805*, (805). Retrieved from <http://www.ietf.org/rfc/rfc805.txt>

- ☐ Postel, J. (1982b). Simple Mail Transfer Protocol. *RFC821*, (821). Retrieved from <http://www.ietf.org/rfc/rfc821.txt>
- ☐ Pournader, M., Shi, Y., Seuring, S., & Koh, S. C. L. (2020). Blockchain applications in supply chains, transport and logistics: a systematic review of the literature. *International Journal of Production Research*, 58(7). <https://doi.org/10.1080/00207543.2019.1650976>
- ☐ Praveen, G., Anand, M., Singh, P. K., & Ranjan, P. (2021). An Overview of Blockchain Consensus and Vulnerability. *Smart Innovation, Systems and Technologies*, 195. https://doi.org/10.1007/978-981-15-7078-0_44
- ☐ Proofpoint. (2018). *Q1-2018 Quarterly Threat Report*. 1–16. Retrieved from <https://www.proofpoint.com/sites/default/files/pfpt-us-tr-q118-quarterly-threat-report.pdf>
- ☐ Proofpoint. (2020). 2020 State of the Phish. *Proofpoint*, 1–48. Retrieved from <https://www.proofpoint.com/sites/default/files/gtd-pfpt-us-tr-state-of-the-phish-2020.pdf>
- ☐ Qasse, I. A., Talib, M. A., & Nasir, Q. (2019). Inter blockchain communication: A survey. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3333165.3333167>
- ☐ Qian, F., Pathak, A., Hu, Y. C., Mao, Z. M., & Xie, Y. (2010). A case for unsupervised-learning-based spam filtering. *Performance Evaluation Review*, 38(1 SPEC. ISSUE). <https://doi.org/10.1145/1811099.1811090>
- ☐ Raghuvanshi, K. K., Khurana, P., & Bindal, P. (2014). Study and Comparative Analysis of Different Hash Algorithm. *Journal of Engineering Computers & Applied Sciences*, 3(9).
- ☐ Ramsdell, B. (2004). *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification*. Retrieved from <http://www.ietf.org/rfc/rfc3851.txt>
- ☐ Ren, L. (2014). Proof of Stake Velocity: Building the Social Currency of the Digital Age. In *Self-published white paper*.
- ☐ Ren, Q., Mu, Y., & Susilo, W. (2007). SEFAP: An email system for anti-phishing. *Proceedings - 6th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2007; 1st IEEE/ACIS International Workshop on e-Activity, IWEA 2007*, 782–787. <https://doi.org/10.1109/ICIS.2007.162>
- ☐ Rescorla, E. (2017). TLS 1.3. *IETF Request For Comments*, 121(2).
- ☐ Resnick, P. (2008). *Internet Message Format*. Retrieved from <http://www.ietf.org/rfc/rfc5322.txt>
- ☐ Reuter, A., Boudaoud, K., Winckler, M., Abdelmaksoud, A., & Lemrazzeq, W. (2020). Secure Email - A Usability Study. In M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, & M. Sala (Eds.), *Financial Cryptography and Data Security* (pp. 36–46). Cham: Springer International Publishing.
- ☐ Reynolds, J. K. (1984). Post Office Protocol. *RFC918*. Retrieved from <http://www.ietf.org/rfc/rfc918.txt>
- ☐ Rogaway, P., & Shrimpton, T. (2004). Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3017. https://doi.org/10.1007/978-3-540-25937-4_24
- ☐ Ronen, E., Gillham, R., Genkin, D., Shamir, A., Wong, D., & Yarom, Y. (2019). The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations. *Proceedings - IEEE Symposium on Security and Privacy, 2019-May*. <https://doi.org/10.1109/SP.2019.00062>
- ☐ Rosenberg, J., & Jennings, C. (2008). The Session Initiation Protocol (SIP) and Spam. *RFC5039*. Retrieved from <http://www.ietf.org/rfc/rfc5039.txt>
- ☐ Ruano-Ordás, D., Fdez-Riverola, F., & R. Méndez, J. (2018). Using evolutionary computation for discovering spam patterns from e-mail samples. *Information Processing and Management*, 54(2). <https://doi.org/10.1016/j.ipm.2017.12.001>
- ☐ Rui, Z., & Yan, Z. (2019). A Survey on Biometric Authentication: Toward Secure and Privacy-Preserving Identification. *IEEE Access*, Vol. 7. <https://doi.org/10.1109/ACCESS.2018.2889996>
- ☐ S. Josefsson, & I. Liusvaara. (2017). Edwards-Curve Digital Signature Algorithm (EdDSA). *Request For Comments*, 8032.
- ☐ Saad, M., Spaulding, J., Njilla, L., Kamhoua, C., Shetty, S., Nyang, D. H., & Mohaisen, A. (2019). Exploring the attack surface of blockchain: A systematic overview. *ArXiv*.
- ☐ Saad, M., Spaulding, J., Njilla, L., Kamhoua, C., Shetty, S., Nyang, D. H., & Mohaisen, D. (2020). Exploring the Attack Surface of Blockchain: A Comprehensive Survey. *IEEE Communications Surveys and Tutorials*, 22(3). <https://doi.org/10.1109/COMST.2020.2975999>
- ☐ Saberhagen, N. Van. (2013). *CryptoNote v 2.0 [WhitePaper]*. 1–20. Retrieved from <https://cryptonote.org/whitepaper.pdf>

- ☐ Sajana M; Sethumadhavan, M, P. S. (2018). On Blockchain Applications: Hyperledger Fabric And Ethereum. *International Journal of Pure and Applied Mathematics*, 118(18).
- ☐ Saleh, F. (2021). Blockchain without Waste: Proof-of-Stake. *The Review of Financial Studies*, 34(3). <https://doi.org/10.1093/rfs/hhaa075>
- ☐ Samanta, S., Mohanta, B. K., Patnaik, D., & Patnaik, S. (2021). Introduction to Blockchain Evolution, Architecture and Application with Use Cases. In *Smart Innovation, Systems and Technologies* (Vol. 219). https://doi.org/10.1007/978-981-33-6470-7_1
- ☐ Samman, G. S. (2016). Chain: Simplified Byzantine Fault Tolerance (SBFT). *SAMMANTICS*. Retrieved from <http://sammantics.com/blog/2016/7/27/chain-1>
- ☐ Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2). <https://doi.org/10.14569/ijarai.2013.020206>
- ☐ Sayeed, S., & Marco-Gisbert, H. (2019). Assessing blockchain consensus and security mechanisms against the 51% attack. *Applied Sciences (Switzerland)*, 9(9). <https://doi.org/10.3390/app9091788>
- ☐ Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart Contract: Attacks and Protections. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.2970495>
- ☐ Schaub, A., & Rossi, D. (2015). Design and analysis of an improved bitmessage anti-spam mechanism. *2015 IEEE International Conference on Peer-to-Peer Computing, P2P 2015*. <https://doi.org/10.1109/P2P.2015.7328523>
- ☐ Schwenk, J., Brinkmann, M., Poddebniak, D., Müller, J., Somorovsky, J., & Schinzel, S. (2020). Mitigation of Attacks on Email End-to-End Encryption. *Proceedings of the ACM Conference on Computer and Communications Security*, 1647–1664. <https://doi.org/10.1145/3372297.3417878>
- ☐ Seigneur, J. M., & Jensen, C. D. (2003). Privacy recovery with disposable Email addresses. *IEEE Security and Privacy*, 1(6), 35–39. <https://doi.org/10.1109/MSECP.2003.1253566>
- ☐ Shahid, A., Almogren, A., Javaid, N., Al-Zahrani, F. A., Zuair, M., & Alam, M. (2020). Blockchain-Based Agri-Food Supply Chain: A Complete Solution. *IEEE Access*, 8, 69230–69243. <https://doi.org/10.1109/ACCESS.2020.2986257>
- ☐ Shahsavari, Y., Zhang, K., & Talhi, C. (2020). A Theoretical Model for Block Propagation Analysis in Bitcoin Network. *IEEE Transactions on Engineering Management*. <https://doi.org/10.1109/TEM.2020.2989170>
- ☐ Shang, J., Zhu, Y., & Liu, Q. (2020). Analysis of security extension protocol in e-mail system. *Chinese Journal of Network and Information Security*, 6(6). <https://doi.org/10.11959/j.issn.2096-109x.2020083>
- ☐ Sheikh, S. A., & Banday, M. T. (2021). A Cryptocurrency-based E-mail System for SPAM Control. *International Journal of Advanced Computer Science and Applications*, 12(1). <https://doi.org/10.14569/IJACSA.2021.0120139>
- ☐ Shen, C., & Pena-Mora, F. (2018). Blockchain for Cities - A Systematic Literature Review. *IEEE Access*, 6. <https://doi.org/10.1109/ACCESS.2018.2880744>
- ☐ Shen, K., Wang, C., Guo, M., Zheng, X., Lu, C., Liu, B., ... Yang, M. (2020). Weak links in authentication chains: A large-scale analysis of email sender spoofing attacks. *ArXiv*.
- ☐ Shi, L., Guo, Z., & Xu, M. (2021). Bitmessage Plus: A Blockchain-Based Communication Protocol with High Practicality. *IEEE Access*, 9. <https://doi.org/10.1109/ACCESS.2021.3056135>
- ☐ Shi, S., He, D., Li, L., Kumar, N., Khan, M. K., & Choo, K. K. R. (2020). Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey. *Computers and Security*, Vol. 97. <https://doi.org/10.1016/j.cose.2020.101966>
- ☐ Siemborski, R., & Melnikov, A. (2007). *SMTP Service Extension for Authentication*. Retrieved from <http://www.ietf.org/rfc/rfc4954.txt>
- ☐ Simonite, T. (2013). Bitcoin isn't the only Cryptocurrency in Town. Retrieved from <https://www.technologyreview.com/2013/04/15/16297/bitcoin-isnt-the-only-cryptocurrency-in-town/>
- ☐ Singh, A., Click, K., Parizi, R. M., Zhang, Q., Dehghantanha, A., & Choo, K. K. R. (2020). Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, Vol. 149. <https://doi.org/10.1016/j.jnca.2019.102471>
- ☐ Siris, V. A., Nikander, P., Voulgaris, S., Fotiou, N., Lagutin, D., & Polyzos, G. C. (2019). Interledger Approaches. *IEEE Access*, 7. <https://doi.org/10.1109/ACCESS.2019.2926880>
- ☐ Skh Saad, S. M., & Raja Mohd Radzi, R. Z. (2020). Comparative Review of the Blockchain Consensus Algorithm Between Proof of Stake (POS) and Delegated Proof of Stake (DPOS). *International Journal of Innovative*

- Computing*, 10(2). <https://doi.org/10.11113/ijic.v10n2.272>
- Sluizer, S., & Postel, J. (1980). Mail Transfer Protocol. *RFC772*, (772). Retrieved from <http://www.ietf.org/rfc/rfc772.txt>
- Sluizer, S., & Postel, J. (1981). Mail Transfer Protocol. *RFC780*. Retrieved from <http://www.ietf.org/rfc/rfc780.txt>
- Sompolinsky, Y., & Zohar, A. (2013). Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR Cryptology EPrint Archive*, 881.
- SophosLabs. (2021). *Sophos 2021 Threat Report*. 36. Retrieved from <https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophos-2021-threat-report.pdf>
- Srivastava, A., Bhattacharya, P., Singh, A., & Mathur, A. (2018). A Systematic Review on Evolution of Blockchain Generations. *International Journal of Information Technology and Electrical Engineering*, 7(6).
- Supreet, Y., Vasudev, P., Pavitra, H., Naravani, M., & Narayan, D. G. (2020). Performance Evaluation of Consensus Algorithms in Private Blockchain Networks. *Proceedings - 2020 International Conference on Advances in Computing, Communication and Materials, ICACCM 2020*. <https://doi.org/10.1109/ICACCM50413.2020.9213019>
- Surjandy, Meyliana, Hidayanto, A. N., & Prabowo, H. (2019). The latest adoption blockchain technology in supply chain management: A systematic literature review. *ICIC Express Letters*, 13(10). <https://doi.org/10.24507/icicel.13.10.913>
- Sussman, J. (1974). FTP Error Code Usage for More Reliable Mail Service. *RFC630*. Retrieved from <http://www.ietf.org/rfc/rfc630.txt>
- Tandon, A., Dhir, A., Islam, N., & Mäntymäki, M. (2020). Blockchain in healthcare: A systematic literature review, synthesizing framework and future research agenda. *Computers in Industry*, 122. <https://doi.org/10.1016/j.compind.2020.103290>
- Taylor, P. J., Dargahi, T., Dehghantanha, A., Parizi, R. M., & Choo, K. K. R. (2020). A systematic literature review of blockchain cyber security. *Digital Communications and Networks*, Vol. 6. <https://doi.org/10.1016/j.dcan.2019.01.005>
- The Radicati Group, I. (2020). *Email Statistics Report , 2020 - 2024* (Vol. 0). Retrieved from <https://www.radicati.com/?p=16510>
- Thirupalu, U., Scholar, R., & Reddy, E. K. (2020). Performance Analysis of Cryptographic Algorithms in the Information Security; Performance Analysis of Cryptographic Algorithms in the Information Security. *International Journal of Engineering Research and Technology (IJERT)*, 8(2).
- Toyoda, K., & Crocker, D. (2005). *Facsimile Using Internet Mail (IFAX) Service of ENUM*. Retrieved from <http://www.ietf.org/rfc/rfc4143.txt>
- Tran, N. K., Ali Babar, M., & Boan, J. (2021). Integrating blockchain and Internet of Things systems: A systematic review on objectives and designs. *Journal of Network and Computer Applications*, Vol. 173. <https://doi.org/10.1016/j.jnca.2020.102844>
- Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys and Tutorials*, 18(3). <https://doi.org/10.1109/COMST.2016.2535718>
- Tseng, L. (2017). Bitcoin's consistency property. *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing, PRDC*. <https://doi.org/10.1109/PRDC.2017.39>
- Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., ... Strackx, R. (2019). Breaking Virtual Memory Protection and the SGX Ecosystem with Foreshadow. *IEEE Micro*, 39(3). <https://doi.org/10.1109/MM.2019.2910104>
- Varghese, A. (2019). *Email Verification Service Using Blockchain*. Technical Disclosure Commons. Retrieved from https://www.tdcommons.org/cgi/viewcontent.cgi?article=3542&context=dpubs_series
- Vassilakis, C. (2019). Blockchain technologies for leveraging security and privacy. *Homo Virtualis*, 2(1). <https://doi.org/10.12681/homvir.20188>
- Vaudreuil, G., & Parsons, G. (2004). *Voice Profile for Internet Mail - version 2 (VPIMv2)*. Retrieved from <http://www.ietf.org/rfc/rfc3801.txt>
- Velásquez, I., Caro, A., & Rodríguez, A. (2018). Authentication schemes and methods: A systematic literature review. *Information and Software Technology*, Vol. 94. <https://doi.org/10.1016/j.infsof.2017.09.012>

- ☐ Verizon. (2020). *Mobile Security Index*. 3–6. Retrieved from <https://enterprise.verizon.com/resources/reports/mobile-security-index/>
- ☐ Virgile, K., & Yu, H. (2015). Securing cloud emails using two factor authentication based on password/apps in cloud computing. *International Journal of Security and Its Applications*, 9(3). <https://doi.org/10.14257/ijisia.2015.9.3.12>
- ☐ Vleck, T. Van. (2001). The History of Electronic Mail. Retrieved from <http://www.multicians.org/thvv/mail-history.html>
- ☐ Vogels, W. (2009). Eventually consistent. *Communications of the ACM*, 52(1). <https://doi.org/10.1145/1435417.1435432>
- ☐ Wang, D., Zhao, J., & Wang, Y. (2020). A Survey on Privacy Protection of Blockchain: The Technology and Application. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.2994294>
- ☐ Wang, Di, & Zhang, X. (2020). Secure data sharing and customized services for intelligent transportation based on a consortium blockchain. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.2981945>
- ☐ Wang, L., Shen, X., Li, J., Shao, J., & Yang, Y. (2019). Cryptographic primitives in blockchains. *Journal of Network and Computer Applications*, 127. <https://doi.org/10.1016/j.jnca.2018.11.003>
- ☐ Wang, Q., Yu, J., Peng, Z., Bui, V. C., Chen, S., Ding, Y., & Xiang, Y. (2020). Security Analysis on dBFT Protocol of NEO. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12059 LNCS. https://doi.org/10.1007/978-3-030-51280-4_2
- ☐ Wang, Y., Han, J. H., & Beynon-Davies, P. (2019). Understanding blockchain technology for future supply chains: a systematic literature review and research agenda. *Supply Chain Management*, Vol. 24. <https://doi.org/10.1108/SCM-03-2018-0148>
- ☐ Warren, J. (2012). Bitmessage: A Peer-to-Peer Message Authentication and Delivery System. *White Paper (27 November 2012)*, <https://bitmessage.org/Bitmessage.Pdf>.
- ☐ Warren, J. (2021). Bitmessage. Retrieved from https://bitmessage.org/wiki/Main_Page
- ☐ Watson, R. W. (1971a). Mail Box Protocol: Version 2. *RFC221*. Retrieved from <http://www.ietf.org/rfc/rfc221.txt>
- ☐ Watson, R. W. (1971b). Mail Box Protocol. *RFC196*. Retrieved from <http://www.ietf.org/rfc/rfc196.txt>
- ☐ Wei, J., Chen, X., Wang, J., Hu, X., & Ma, J. (2019). Forward-Secure Puncturable Identity-Based Encryption for Securing Cloud Emails. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11736 LNCS. https://doi.org/10.1007/978-3-030-29962-0_7
- ☐ Wei, J., Chen, X., Wang, J., Hu, X., & Ma, J. (2021). Enabling (End-to-End) Encrypted Cloud Emails With Practical Forward Secrecy. *IEEE Transactions on Dependable and Secure Computing*. <https://doi.org/10.1109/TDSC.2021.3055495>
- ☐ White, J. E. (1973a). Proposed Mail Protocol. *RFC524*. Retrieved from <http://www.ietf.org/rfc/rfc524.txt>
- ☐ White, J. E. (1973b). Request for network mailbox addresses. *RFC510*. Retrieved from <http://www.ietf.org/rfc/rfc510.txt>
- ☐ Wilkinson, S., Boshevski, T., Brandoff, J., Prestwich, J., Hall, G., Gerbes, P., ... Pollard, C. (2016). Storj A Peer-to-Peer Cloud Storage Network. *Technical Report*. Retrieved from <https://www.storj.io/storjv2.pdf>
- ☐ Winter, J. De. (1996). *SMTP Service Extension for Remote Message Queue Starting*. Retrieved from <http://www.ietf.org/rfc/rfc1985.txt>
- ☐ Wong, M., & Schlitt, W. (2006). Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. *RFC4408*. Retrieved from <http://www.ietf.org/rfc/rfc4408.txt>
- ☐ Wu, Yu, Yang, Z., Qu, Z., Xiao, Z., & Dai, Y. F. (2011). User-experience-based availability analysis model and its application in P2P storage systems. *Science China Information Sciences*, 54(8). <https://doi.org/10.1007/s11432-011-4313-9>
- ☐ Wu, Yue, Lu, Z., Yu, F., & Luo, X. (2020). Rapid consortium blockchain for digital right management. *Advances in Intelligent Systems and Computing*, 1107 AISC. https://doi.org/10.1007/978-981-15-3308-2_48
- ☐ Wu, Z., Tian, J., Member, S., Li, L., Jiang, C., & Yangi, X. (2007). A Secure Email System Based on Fingerprint Authentication Scheme. *Intelligence and Security Informatics, 2007 IEEE*, 250–253. <https://doi.org/10.1109/ISI.2007.379480>
- ☐ Xie, J., Tang, H., Huang, T., Yu, F. R., Xie, R., Liu, J., & Liu, Y. (2019). A Survey of Blockchain Technology Applied to Smart Cities: Research Issues and Challenges. *IEEE Communications Surveys and Tutorials*, 21(3), 2794–2830. <https://doi.org/10.1109/COMST.2019.2899617>

- Xiong, W., & Xiong, L. (2021). Data Trading Certification Based on Consortium Blockchain and Smart Contracts. *IEEE Access*, 9. <https://doi.org/10.1109/ACCESS.2020.3047398>
- Xu, P., Jiao, T., Wu, Q., Wang, W., & Jin, H. (2016). Conditional Identity-Based Broadcast Proxy Re-Encryption and Its Application to Cloud Email. *IEEE Transactions on Computers*, 65(1). <https://doi.org/10.1109/TC.2015.2417544>
- Xu, X., Weber, I., & Staples, M. (2019). Architecture for Blockchain Applications. In *Architecture for Blockchain Applications*. <https://doi.org/10.1007/978-3-030-03035-3>
- Yadav, J., & Shevkar, R. (2021). Performance-Based Analysis of Blockchain Scalability Metric. *Tehnički Glasnik*, 15(1). <https://doi.org/10.31803/tg-20210205103310>
- Yaqoob, S., Khan, M. M., Talib, R., Butt, A. D., Saleem, S., Arif, F., & Nadeem, A. (2019). Use of blockchain in healthcare: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, 10(5). <https://doi.org/10.14569/ijacsa.2019.0100581>
- Ye, C., Li, G., Cai, H., Gu, Y., & Fukuda, A. (2018). Analysis of security in blockchain: Case study in 51%-attack detecting. *Proceedings - 2018 5th International Conference on Dependable Systems and Their Applications, DSA 2018*. <https://doi.org/10.1109/DSA.2018.00015>
- Yumna, H., Khan, M. M., Ikram, M., & Ilyas, S. (2019). Use of Blockchain in Education: A Systematic Literature Review. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11432 LNAI. https://doi.org/10.1007/978-3-030-14802-7_17
- Zahed Benisi, N., Aminian, M., & Javadi, B. (2020). Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 162, 102656. <https://doi.org/https://doi.org/10.1016/j.jnca.2020.102656>
- Zavolokina, L., Ziolkowski, R., Bauer, I., & Schwabe, G. (2020). Management, governance, and value creation in a blockchain consortium. *MIS Quarterly Executive*, 19(1). <https://doi.org/10.17705/2msqe.00022>
- Zhang, J., Cui, W., Ma, J., & Yang, C. (2019). Blockchain-based secure and fair crowdsourcing scheme. *International Journal of Distributed Sensor Networks*, 15(7). <https://doi.org/10.1177/1550147719864890>
- Zhang, R., Xue, R., & Liu, L. (2019). Security and Privacy on Blockchain. *ACM Comput. Surv.*, 52(3). <https://doi.org/10.1145/3316481>
- Zhang, Shaomin, Pu, M., Wang, B., & Dong, B. (2019). A Privacy Protection Scheme of Microgrid Direct Electricity Transaction Based on Consortium Blockchain and Continuous Double Auction. *IEEE Access*, 7. <https://doi.org/10.1109/ACCESS.2019.2946794>
- Zhang, Shijie, & Lee, J. H. (2020). Analysis of the main consensus protocols of blockchain. *ICT Express*, 6(2). <https://doi.org/10.1016/j.ict.2019.08.001>
- Zhao, H., Wen, J., Huang, Q., Zhou, W., & Yang, Z. (2020). The application of consortium blockchain in health inspection system for special equipments on service security and privacy protection. *Chongqing Daxue Xuebao/Journal of Chongqing University*, 43(7). <https://doi.org/10.11835/j.issn.1000-582X.2020.245>
- Zhou, J., Onieva, J., & Lopez, J. (2005). Optimized multi-party certified email protocols. *Information Management & Computer Security*, 13, 350–366. <https://doi.org/10.1108/09685220510627250>
- Zhou, Q., Huang, H., Zheng, Z., & Bian, J. (2020). Solutions to Scalability of Blockchain: A Survey. *IEEE Access*, 8, 16440–16455. <https://doi.org/10.1109/ACCESS.2020.2967218>
- Zhu, X., & Wang, D. (2019). Application of Blockchain in Document Certification, Asset Trading and Payment Reconciliation. *Journal of Physics: Conference Series*, 1187(5). <https://doi.org/10.1088/1742-6596/1187/5/052080>

Capítulo 6

Anexo

ANEXO. CONTRIBUCIONES CIENTÍFICAS

Publicaciones del doctorando en revistas científicas de impacto y congresos directamente relacionadas con la tesis doctoral:

- ☐ Piedrahita Castillo, D., Machío Regidor, F., Bermejo Higuera, J., Bermejo Higuera, J. R., & Sicilia Montalvo, J. A. (2020). A New Mail System for Secure Data Transmission in Cyber Physical Systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. <https://doi.org/10.1142/S0218488520400127>
- ☐ Piedrahita, D., Bermejo, J., Machío, F. (2022). A Secure Email Solution Based on Blockchain. In: Prieto, J., Partida, A., Leitão, P., Pinto, A. (eds) *Blockchain and Applications. BLOCKCHAIN 2021. Lecture Notes in Networks and Systems*, vol 320. Springer, Cham. https://doi.org/10.1007/978-3-030-86162-9_36

A New Mail System for Secure Data Transmission in Cyber Physical Systems

Diego Piedrahita Castillo^{*}, Francisco Machío Regidor[†], Javier Bermejo Higuera[‡],
Juan Ramón Bermejo Higuera[§] and Juan Antonio Sicilia Montalvo[¶]

*Universidad Internacional de la Rioja,
ESIT-Escuela Superior de Ingeniería y Tecnología,
Av. de la Paz, 137, 26006 Logroño, La Rioja, Spain*

^{}diego.piedrahita@comunidadunir.net*

[†]francisco.machio@unir.net

[‡]javier.bermejo@unir.net

[§]juanramon.bermejo@unir.net

[¶]juanantonio.sicilia@unir.net

Received 25 May 2020

Revised 19 June 2020

This paper provides a complete study on email requirements, with special emphasis on its security aspects and architecture. It explores how current protocols have evolved, the environment in which they have been developed and the evolution of security requirements. This paper also analyzes email vulnerabilities and the reasons that have motivated the exploitation of them. The threats and solutions of the most used email protocols today are detailed, such as Simple Mail Transfer Protocol, Post office Protocol, Internet Message Access Protocol protocols, among others. An analysis of the main security solutions proposed in recent years is carried out and how these threats are solved, as well as a comparison of each of them. The result of this work leads us to conclude that it is necessary to make an integral change in the protocols used in the electronic mail in order to have a secure message exchange system that meets all the security requirements demanded today. We are working on a proposal based on blockchain that solves the security problems identified in this work.

Keywords: Email; security; SMTP; POP; IMAP; Blockchain.

1. Introduction

Email is an asynchronous electronic messaging system widely used by people and organizations, it is the formal messaging solution par excellence, and thanks to its simplicity of use, it is available to everyone. It has been replacing postal messaging little by little, facilitating, for example, document management between sender and receiver. Email has evolved along with technology. The operating systems allowed

[¶]Corresponding author.

the shared use of information and facilitated communication among users.¹ This work shows how these needs emerged and how they were given a solution with different proposals, either improving the protocols or with external solutions to them.

The RFC780² and RFC788³ proposed a protocol called Simple Mail Transfer Protocol. This protocol continues to this day with some modifications and has become a standard. As of SMTP, new features were added to the protocol and other protocols appeared, but the basic structure of SMTP remained the same and the security elements confidentiality, integrity and availability were left aside. The solutions to preserve these security elements have been incorporated outside the protocol, which has made it easier for attackers to take advantage of vulnerabilities when a mail server does not use any of the security solutions available.

Table 1. Email Spam Rate by Industry.⁴

EMAIL SPAM RATE BY INDUSTRY (YEAR)	
INDUSTRY	EMAIL SPAM RATE (%)
Mining	58.3
Finance, Insurance, & Real Estate	56.7
Manufacturing	55.1
Public Administration	54.9
Agriculture, Forestry, & Fishing	54.6
Transportation & Public Utilities	54.6
NonClassifiable Establishments	54.2
Services	54.1
Retail Trade	53.7
Construction	53.6
Wholesale Trade	52.6

Email continues to be the effective path cybercriminals use to achieve their goals and that trend continues to grow.⁴ More than half of email traffic was spam in 2019 as can be seen in Table 1. Although other types of email threats, such as phishing (with its variants) and malware, worry the community. Almost 190 billion emails circulated every day in 2019 and it is estimated that by the end of 2020 it will grow more than four percent. The users of the companies send an average of 42 emails and the number of individuals who read their mail on mobile devices grows.⁴ It is enough reason to worry and raise the security of email as a relevant problem to study and propose solutions.

2. State of the Art, Threats, Vulnerabilities and Security

The line that separates the historical documents and the state of the art, considered as current works, is very tenuous, since many contributions of yesteryear are still used today, such as the SMTP, POP3 and IMAP protocols or in security issues the certificates and cryptography.

Figure 1 shows schematically the possible attacks that can be perpetuated using the current email system. The main objective of a comprehensive secure email proposal is to minimize or resolve this scenario. The perspective that is used in this study is to analyze the works that go in that way. Since it is a well-known and studied problem today, only the originals works have been included; that is why derivative works have been left aside in the study.

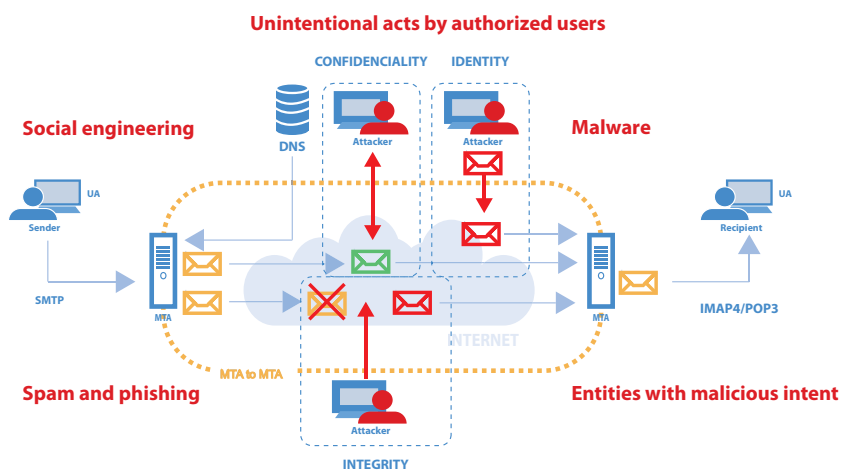


Fig. 1. Attacks on Mail Server.⁵

All this problematic will be treated in the following paragraphs and in this section focuses more on detailing the solutions that are considered interesting and that contribute some aspect to consider in a security proposal on email. It has been organized in two sections. First the most outstanding articles about proposals related to e-mail have been grouped by subject. In the second, the most robust technological proposals that are currently being used or for various reasons have not yet been implemented.

Attacks and exploitation of vulnerabilities related to the use of electronic mail have increased in recent years. Internet is available to everyone; good and bad people share an information highway, and much has been developed and written about it. When searching for information with the terms “Email security + email threats + email vulnerabilities” in different bibliographic databases, the results include more than 10,000 available publications. The most relevant publications and those best related to this work have been selected. The selected articles have been classified into four sections. Usability, Phishing, Spam and Security Solutions.

2.1. Usability

Payne and Edwards in 2008 conducted a study of the security systems from the point of view of usability, although they focused on two problems, authentication and encryption.⁶ The authors travel in time illustrating how authentication methods have changed, from the simple 8-character password to the complicated passwords generated by pixel selections in images called Passpoints. The study showed the usability problems that the encryption solutions for email have had, such as PEM in 1985, due to the problems of organizational usability with CAs. PGP in 1991 had scalability problems, therefore, it could only be used by small communities. S/MIME on the other hand, is an intermediate solution between PEM and PGP, which requires a nonspecific CA. Although mail encryption is available, many users do not use it because of the usability problems.

In the article commented on the work of Kai-Ping Yee, one of the most cited in terms of design guides on the usability of security, a study is made of some valued tools related to this feature. Following the same line, Stephen Farrell in 2009 makes a review of the security features present in the MUA (Thunderbird, Outlook, etc.) with the aim of obtaining the security features that users need for the treatment of electronic mail.⁷

Putting a little more focus on this article, you can get very valuable conclusions to be mentioned in this work. In the first place, the author comments on the reasons why users do not use security when they manage their e-mail and coincide with the other works described above, ease of use and ignorance are the main problem. The way in which the user perceives security and the way in which the experts propose security has great differences. According to the author, for many years, the user has tried to approach the most sophisticated techniques, but the way to approach it has failed. For a user it is too complicated to have to decide if a destination mailbox has a public key or even worse, the use of certificates is too cumbersome for most. As a conclusion of the article, the use of security techniques is not intuitive.

Another article selected for this work is the one written by Cristian Thiago Moecke and Melanie Volkamer, entitled “Usable secure email communications: criteria and evaluation of existing approaches”.⁸ In this work the desirable security, interoperability and security criteria are evaluated in any email system. Other aspects such as spam, traceability, authenticity and confidentiality, legal aspects and anonymous communications such as alias, are left for future work.

The criteria for the evaluation described in the article are the following:

- (A.1) Authenticity of the address of the issuer.
- (A.2) Strong authentication.
- (A.3) Authenticity of the person.
 - (I) Integrity.
 - (C) Confidentiality.
- (IN.1) Interoperability with other suppliers.
- (IN.2) Secure interoperability with other providers.

- (IN.3) Open standards. Possibility of a company to operate with its own services.
 (IN.4) Independence of the provider. The service can be offered by other providers.
 (IN.5) Security extension. Possibility of using other security techniques that are not implemented by default.
 (U.1) Starting cost. It should be easy to configure and start using the service.
 (U.2) Appropriate metaphors. The security terminology should be used with metaphors.
 (U.3) Easy trust decisions.
 (U.4) PKI is hidden.
 (U.5) Checks should not be made outside the system, such as comparing hash values.

Table 2. Results Authentication, Integrity and Confidentiality.⁸

	A.1	A.2	A.3	I	C
Gmail	Partially (see U.3 Tr: Gmail/DNS). Detect forged Gmail(Tr: Gmail)	2 factor (not visible to receiver)	No	Yes (Tr: Gmail and DNS)	End to Server and Server to Server (if available)
DE-Mail	Yes (Tr: Provider)	UnKnown	Yes (Tr: Provider)	Yes (Tr: Provider)	Yes (Tr: Provider)
EPostBrief	Yes (Tr: Provider)	SMS TAN (not visib. to receiver)	Yes (Tr: Provider)	Yes (Tr: Provider)	Yes (Tr: Provider)
Hushmail	Yes (Tr: Hushmail)	Password only	No	Yes (Tr: Hush-mail/Softw.)	Yes (Tr: Hush-mail/Softw.)
Gmail & PGP	Restricted contexts (no U.5)	No change	Restricted contexts (no U.5)	Yes	Yes
Gmail & S/MIME	If CA verifies it (Tr: CA)	No change	If CA verifies it (Tr: CA)	Yes	Yes
Gmail & Opp	If had previous secure comm.	No change	No	No	If receiver key is known

Tables 2, 3 and 4 show the results obtained when these criteria were used to evaluate the existing secure email providers.

Along the same line, the work of Routi shows some very interesting conclusions to be considered in any proposal of secure mail. The way in which the security elements are presented to the user and the ease of use thereof, will condition the diffusion and use of any security tool for the email. For other side, they found further evidence that hiding the security details can lead to a lack of trust in the secure email system. As a final conclusion in the study, they found that there is a gap between participants' interest in using secure email and how and when they should use it.⁹

Table 3. Results Usability.⁸

	U.1	U.2	U.3	U.4	U.5
Gmail	Easy	No methaphors	Decide if domain is trustworthy	No PKI	Domain trust worthiness
DE-Mail	Identification on provider	Unknown	Easy: Trust provider	PKI hidden	No out of bounds data
EPostBrief	Identification on Post office	Encryption and signature	Easy: Trust provider	PKI hidden	No out of bounds data
Hushmail	Easy	Encryption and signature	Easy: Trust provider	PKI hidden	No out of bounds data
Gmail & PGP	Hard: create /publish key, get recipient key	Depends on implementation	Decide trusted keys based on WoT	Understand PGP	Obtain information to trus keys
Gmail & S/MIME	Hard: obtain own/recipient certificate	Depends on implementation	If known CA easy. If not, hard	If verification fails or CA is untrusted	If verification fails or CA is untrusted
Gmail & Opp	Easy	Not necce-sary	Only in key conflict	PKI hidden	Only in key conflict

Table 4. Results Interoperability.⁸

	IN.1	IN.2	IN.3	IN.4	IN.5
Gmail	Yes	Yes	Open standards	Open standards	PGP, S/MIME
DE-Mail	Closed environ.	Closed Environ.	High cost	High cost	PGP, S/MIME
EPostBrief	Closed environ.	Closed environ.	Closed environ.	No	PGP, S/MIME
Hushmail	Yes	Yes	Open standars	Open Standards	DKIM/SPF/ S/MIME

2.2. Phishing

One of the most serious attacks for Internet users is Phishing, that is, identity theft. Despite the variety of existing anti-phishing systems, its effectiveness and usability has been compromised, as seen in the previous section. The problem with this type of threat is that the victim provides sensitive information to the attacker through deception perpetrated by social engineering techniques or forgery. One of its most lethal variants is the Spear Phishing, where the attack is directed using sophisticated social engineering techniques and the target of the attacker has been studied deeply, that is, it is clear to the attacker who his victim is and it has studied very well previously.

In 2007, Ren and Susilo made an anti-phishing proposal through a system called SEFAP.¹⁰ SEFAP which means Signed Email for Anti-Phishing is a system as its name indicates, based on several signature schemes, and can adopt any scheme, if two properties are met: this is based on identity and has the capacity to repudiate. Based on identity means it leads to eliminate the infrastructure of PKI and having

the repudiation property means it protect the privacy of the issuer. SEFAP has the great advantage that it can be integrated into any standard SMTP server. The only problem is that both sender and receiver must use SEFAP, which adds a layer of complexity and any user cannot use it.

In the same year, other types of less technical solutions and more focused on users were also proposed. Of all of them, one of the most interesting was the work of Kumaraguru in which it was proposed to instruct the user through training emails. The objective of the technique was to teach about the potential dangers of phishing, sending them to the false pages and teaching them to observe the differences between the page that appears in the mail link and the one they have visited.¹¹ Although it is very interesting, a careless user throws away all the training. It would be best to combine it with an automated phishing solution.

Trying to distance the user from the complexity of the certificates, some authors from the University of Miami devised a solution called "Trusted Email",¹² which used a proxy installed on the user's machine or the institution to which it belongs and another in the web server of the company, bank or institution the user wishes to communicate with. Proxies are responsible for secure communication, distancing the user a bit from that complexity. The entire process is done through a plugin that is installed in the user's browser. It is a solution based on user authentication.

The attackers change the way to carry out the phishing constantly, that makes the detection techniques become obsolete in a short time. Keep in mind that the problem is because SMTP is a protocol without authentication and the best anti-phishing techniques are those that provide authentication to the protocol in some way. Within detection techniques, in which phishing has already occurred but has not reached the user, there are two approaches: Based on behavior and based on Content. Hamid and Abawajy in 2011 conducted a study of different techniques that used some of these approaches and proposed a hybrid solution.¹³

The problem with the techniques studied on the solutions to phishing attacks, especially the dangerous spear phishing (specialized form of social engineering perpetuated on a specific objective) is that it focuses on identifying the attack once the attacker's email is accessed. That is, they are reactive techniques (the message has arrived) and not proactive techniques (the message has a low probability or zero of being received). For example, the work of Meyers,¹⁴ Petelka¹⁵ and Lawson¹⁶ among others, they are very good works but they are focused in the reactive direction. Being reactive techniques does not mean that they are considered bad, but in this work, we want to give a different approach and is to identify the problem of origin and analyze the studies conducted in that direction.

Other works that are wanted to comment here, go in the line of the use of very sophisticated techniques of artificial intelligence and that are in peak in these times. Reference 17 conducted a systematic study on phishing attacks using Machine Learning techniques. In this study, the different contributions are analyzed, and it is concluded that the URL-based approach is the most used.

2.3. Spam

Spam is the main problem today within the threats to email, not only because of the disorder it generates but also because the content is the most varied, from advertising, dangerous viruses, malware and phishing seen above. The study of antispam methods is outside the scope of this work, and there is a lot of literature about that, but reference is made to the relevant academic papers on this subject.

Studies conducted in 2002 revealed the existence of TME (malicious targeted mails) as a form of attack very different from Spam, Phishing and other non-legitimate emails.¹⁸ The author makes an extensive study on the techniques of TME detention and the procedures of analysis and filtering of emails.

The publications of RFC5068¹⁹ and RFC5039²⁰ dealing directly with the subject could not be left out of this study. The RFC5068 first makes a series of recommendations when sending an email, including as best practices the use of authentication, authorization and temporary traceability. The RFC5039, on the other hand, analyzes the use of the session initiation protocol (SIP) which also has the threat of spam. The proposed solution space includes: Content filtering, blacklists, whitelists, content-based communication, reputation systems, address obfuscation, limited-use addresses, Turin tests, computer puzzles, legal actions, etc. All these solutions can be perfectly applied to email.²⁰

All the techniques normally used against spam are based on behavioral studies or mail content. Empirical studies have been carried out, for example, to characterize how spam behaves in order to obtain vulnerable email accounts and to know what their content is. To do this, millions of emails with spam have been analyzed and collections of them have been gathered for their study.²¹ In the same line of this study Jindal and Liu analyzed the Spam of opinion and proposed a technique to detect this type of Spam.²²

The techniques used to solve the problem of spam all start with their detection. The problem has changed in such a way that there is legislation about the topic.²³ The scientific community is working hard, and the most sophisticated techniques are being used to solve the problem. The following works are clear examples of all this: Gbenga,²⁴ Alghoul,²⁵ Barber²⁶ and Habib.²⁷ A more recent work is the patent published by Dennis: Blockchain based email procedures. In this work, the author proposes a solution to the emails of unknown issuers using an authorization replication mechanism based on blockchain. The interesting thing about the technique is that the receiver does not receive the message until the system authorizes the sender.²⁸

The problem with spam is that it is a path that attackers use for other types of threats: Ransomware, Spear phishing, Spoofing, man-in-the-Middle Attacks, Whaling/Business Email Compromise, Key Loggers, Zero-Day Exploits, Social Engineering, etc.

2.4. Comprehensive security solutions

Bellamy and McDonald dealt with a problem that we want to raise in this work, human security based on the use of electronic mail. The authors commented on a debate about what security really is and for whom it should be provided.²⁹ This topic is interesting to consider when defining security aspects in the requirements of a security proposal, since the security of the mail is considered from the point of view of citizen security, a very controversial issue.

On the other hand, proposals for asymmetric encryption, such as that proposed by Garfinkel, demonstrate the great need to offer more manageable solutions with a high degree of usability.³⁰ The author criticizes solutions such as PGP and S/MIME for the complexity they add when implemented and enhances the characteristics of Stream, an email encryption system whose most representative vulnerability is the attack of man-in-the-middle.

Garfinkle proposed the use of EBIA instead of PKI.³⁰ The thesis on which the article is based argued that EBIA was a reasonable approximation for many commercial and governmental applications, since it provided better usability, privacy, autonomy, resilience and coupling to real-world business requirements than PKI technology. Among the PKI strengths that are exposed, the fact of replacing traditional identifiers such as the SSN and the use of smart cards or devices to store the keys is a clear advantage over other solutions. PKI provides non-repudiation, a feature much desired by governments. Among the problems that have prevented the expansion of PKI, the authors relate the following:

- Usability. PKI clients are very difficult to use.
- Cost. Users must buy certificates in many cases.
- Delegation. The delegation of authority, for example, for a director to delegate to his secretary the management of his mail is very difficult.

EBIA was simpler to use: The use of an email address as an identification mechanism and the reception of an email as an authentication mechanism. The authors advocate the use of additional security mechanisms to protect the vulnerabilities present when sending a confirmation email on an insecure network.

As an email address is easier to use, EBIA is quick to adopt. The great advantage of EBIA over PKI is that PKI requires specialized software and a relationship of mutual trust with a CA, while EBIA works with any Email client.

At the level of business requirements, EBIA outperforms PKI in:

- PKI must establish a personal identifier. Email addresses are easy to create and destroy. Users want to be able to identify themselves differently on many websites.
- Users do not want to be personally identified on some websites, EBIA allows it, PKI does not.
- It is very difficult for organizations to react to loss of keys or certificate errors with PKI, with EBIA no.
- If a key in PKI is stolen, the thief can use it again and again without the owner's knowledge.

- Users tend to trust more in the organizations where they get their email addresses than in the unknown and non-visible CAs that PKI uses.
- In organizations, the use of roles, as Financial Director, or groups, as field technicians, is very similar to EBIA, since they can use mailing lists. With PKI it is not possible, because the certificates are unipersonal.

In his work, Garfinkle referred to other works related to EBIA, such as IBE (identification based on Email), PGP and RSA Security.

The use of DEA (Disposable Email address) has also been proposed as a security mechanism against spam. The use of DEA together with the REAP protocol (Rolling Email Addresses Protocols) basically allows a user who wishes to register with a provider, use a REAP server. The server must provide a mail account that can be deleted if an email is received with deletion instructions or changed by the same REAP server if the user wishes. The server would be responsible for making changes to the email account with the suppliers.³¹

The protection of emails to prevent the spread of viruses in organizations is another aspect that has been considered when talking about security, as proposed by Shih and Chiang.³² The authors explain how organizations can protect emails and how to develop a security plan. This paper is an exhaustive study on the types of viruses, the causes and the effects of their spread in organizations.

Protocols for certified email allow an issuer to send a message to a receiver, so that the recipient gets the message if and only if the sender receives a corresponding return notification. The problem with this type of protocol is that anyone can simulate sending the message. Abadi and Blanchet propose in 2005 in a paper, by means of a tool, a way to verify this type of protocols. The general idea of this protocol is that a third party (TTP) intervenes in the communication, in this way, the issuer encrypts a temporary key with the public key of TTP. That temporary key is used to encrypt the message. In this way, the receiver can only decrypt the message if it has the key. To decrypt it, it sends the key to the TTP to decipher it, and it is at that moment, when TTP notifies the sender that the recipient has received the message.³³

There are proposals based on PKI. But with light encryption for email authentication.³⁴ The objective of these proposals was to preserve the natural use of electronic mail, without entering conflicts with the expectations when the mail is used. In this type of solution, each email domain becomes a master authority for the identity-based scheme. The public and private key pair is generated, and the public key is distributed via the DNS as an associated text record, and the private key is obtained by EBI.

Not only the problem of guaranteeing the reception through certified emails commits an issuer and receiver, there are cases in which it is necessary to guarantee the reception of many recipients of a certified email. Zhou, along with other authors, present two multi-device certified email protocols.³⁵

The proliferation of online mail services has originated the creation and use of tools that promulgate protect the integrity and confidentiality of the user. In 2006, a paper refers to Aquinas, a system that allows to encrypt the content using a message authentication code (MAC) for encryption and stenography to hide the keys and the encrypted text.³⁶

Biometric authentication techniques abound today, and it was expected that one of them would be used as an authentication scheme. The fingerprint was designed to be used within secure email systems, with a technique that used identity-based encryption.³⁷ Authentication mechanisms such as Sender Policy Framework (SPF), Sender ID Framework (SIDF) and DomainKeys Identified Mail (DKIM) were analyzed within DNS-based authentication mechanisms, a quite vulnerable protocol.³⁸ The author's recommendation is the use of DKIM and SPF as effective mechanisms to authenticate emails sent by a domain, although it is advisable to use S-DNS, but recommend discarding the use of SIDF because it does not really comply with what it promises. The three protocols will be analyzed in detail later.

In 2012, Choukse et al defined the basic principles when designing a secure email infrastructure and proposed a design according to the needs of the organizations, according to the resources and based on the principles of security by design and availability. In short, the solution is to develop a mix of the best solutions proposed so far based on an analysis of threats and weaknesses in email infrastructure.³⁹ Although it is criticized for its complexity, PKI is a robust solution and many authors still pointed in that direction. In 2012, Lina and Jiang advocated the use of smart cards within an authentication scheme and encryption for mail electronic.⁴⁰

Alge commented on how alarming it was, that mail servers were being hosted in the cloud, considering the little innate security that this service had and the existing security problems inherent in cloud solutions.⁴¹ One of the reasons for the loss of security is that many providers omitted verification for usability reasons, allowing attacks such as incrimination, malware distribution to be perpetuated.⁴² The solution to the problem of the cloud is to implement secure email servers in the cloud, such as the one proposed by Madi et al in 2012. The solution is based on using technologies such as TLS, SMTP-AUTH and SASL.⁴³

With the appearance of all kinds of solutions using blockchain, it was expected that some security solutions were proposed. Lukasz Dykcik proposed BlockPKI, which is said to be a blockchain-based public-key infrastructure that enables automated, resilient, and transparent issuance of digital certificates.⁴⁴ In the same year, in his work about public key infrastructure using blockchains, Smith details different techniques to use PKI on blockchain.⁴⁵

3. Data and Statistics

In this section, we will present some data that support the development of this work. The attacks, threats and costs associated with the abuse of email will be exposed in this section as a sample of the relevance and impact of the email security problem in the whole community in general.

The disponibility, confidentiality and integrity of information resources are the main objective of information security. All these aspects of information security can also be applied in the field of email security. In 2004, SANS Institute carried out a study by Pam Cooca in which e-mail security threats were defined at that time. According to the study, viruses, SPAM and Phishing were the main problems identified. As Table 5 shows, most of the viruses were introduced as an attachment, closely following the Internet downloads.⁴⁶

Table 5. Main sources of virus infection.⁴⁶

Virus Source	1996	1997	1998	1999	2000	2001	2002	2003
Email Attachment	9	26	32	56	87	83	86	88
Internet Downloads	10	16	9	11	1	13	11	16
Web Browsing	0	5	2	3	0	7	4	4
Don't Know	15	7	5	9	2	1	1	3
Other Vector	0	5	1	1	1	2	3	11
Software Distribution	0	3	3	0	1	2	0	0
Diskette	71	84	64	27	7	1	0	0

Other data obtained from market studies showed the expected growth in the number of users over the next four years (2019 to 2023).⁴⁷ According to the study, the expected growth is almost 400 thousand email users between 2019 and 2023, as shown in the Table 6.

Table 6. Worldwide Email Market Forecast.⁴⁷

Daily Email Traffic	2019	2020	2021	2022	2023
Total Worldwide Emails Per Day (B)	293.6	306.4	319.6	333.2	347.3
% Change	4%	4%	4%	4%	4%

In terms of traffic, at the end of 2019 a total of 293.6 billion emails sent and received per day are expected, and more than half of those emails correspond to business mail as shown in the Table 7.

Table 8 shows the amount of spam received per year by users of different sized organizations.⁴ The problem is that once the filters are done, those emails reach the users who must decide whether to open the message. The probability that the message is malignant is very high, given the data observed in previous tables.

Another piece of information provided by the American Institute Radicati Group is the cost of spam, which amounts to 1.3 million a year for companies. According to the study, a company with about 1000 users can spend 1.3 million a year for the management and control of spam. It is more than a good reason to address solutions to this problem.

Table 7. Daily email traffic in the world.⁴⁷

Worldwide Email Market Forecast	2019	2020	2021	2022	2023
Worldwide Email Users (M)	3,930	4,037	4,147	4,258	4,371
% Change	3%	3%	3%	3%	3%
Worldwide Email Market Revenues (\$M)	\$39,633	\$46,821	\$54,833	\$63,601	\$73,317
% Change	20%	18%	17%	16%	15%

Table 8. Email spam per user by organization size.⁴

EMAIL SPAM PER USER BY ORGANIZATION SIZE (YEAR)	
ORGANIZATION SIZE	SPAM PER USER
1-250	55
251-500	57
501-1000	109
101-1500	125
1501-2500	107
2501+	55

One of the most complete reports is that one developed by Symantec on Internet security threats.⁴ The report shows a decline in some ratios. For example, the number of phishing since 2017 has had a considerable decline, it has gone from 1 in 2995 emails in 2017 to 1 in 3500 emails in 2018.

Its descent comes hand in hand with the sophistication and new phishing techniques that are appearing. What’s more, Symantec has reported the number and sophistication of phishing attempts in recent years. Obviously, these new methods are not reflected in the Figure 2. If you look at the results of the study by company size, you can conclude that small companies are the most vulnerable, especially because of the cost of introducing additional security mechanisms. Table 9 shows the results of the analysis.

The results of the report also show the malware ratios that are attached in the reported emails. Among the documents that come as an attachment, the Word and Excel files are the most infected and, of course, companies are usually the most attacked. Table 10 shows the data.

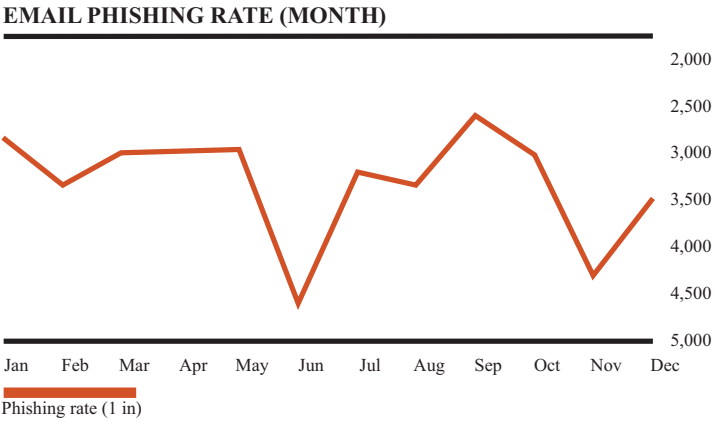


Fig. 2. Email Phishing rate in 2018.⁴

Table 9. Email phishing rate per user by organization.⁴

EMAIL PHISHING RATE PER USER BY ORGANIZATION SIZE (YEAR)	
ORGANIZATION SIZE	USERS TARGETED (1 IN)
1-250	52
251-500	57
501-1000	30
101-1500	56
1501-2500	36
2501+	82

Table 10. Top malicious email attachment types in 2018.⁴

TOP MALICIOUS EMAIL ATTACHMENT TYPES (YEAR)	
FILE TYPE	PERCENT
.doc, .dot	37.0
.exe	19.5
.rtf	14.0
.xls, .xlt, .xla	7.2
.jar	5.6
.html, htm	5.5
.docx	2.3
.vbs	1.8
.xlsx	1.5
.pdf	0.8

Table 11. Malicious email rate by organization size in 2018.⁴

MALICIOUS EMAIL RATE BY ORGANIZATION SIZE (YEAR)	
ORGANIZATION SIZE	MALICIOUS EMAIL RATE (1 IN)
1-250	323
251-500	356
501-1000	391
101-1500	823
1501-2500	440
2501+	556

As the latest data, the following statistics are very worrisome, considering the number of emails circulating in the world. According to the study, one in 323 emails of a user in a small organization is malicious. Table 11 shows the summary. As a conclusion to these reports, the companies are exposed to a problem that generates a high cost, reduces the overall performance of employees, without counting the associated cost problems due to breaches of confidentiality, integrity and availability of information.

4. Tools and Integral Solutions

4.1. STARTTLS

STARTTLS is an extension of SMTP that allows the use of TLS to provide communication on private and authenticated internet. TLS is widely used by the HTTP protocol, but it is being used to add security to other protocols such as in this case to SMTP.⁴⁸ The communication between client and server is done, as shown in Figure 3.

Just after the STARTTLS command is sent, the typical TLS negotiation starts and once the security levels have been agreed, the algorithms used, the key exchange, i.e. the “handshake”, the encrypted mail is sent.

The problem with this scheme is that there is communication in plain text that can alter the negotiation process, hence an attacker could modify it and the communication would be compromised, that is, there would be a man-in-the-middle attack (MITM) that would cause a degradation of the encryption.

4.2. SMTP STS

SMTP STS is an effort to make email more secure. STS is the acronym for Strict Transport Security and it is postulated as a strong security standard for email. The draft protocol was written in 2016.⁴⁹ The main idea is to improve the aspects

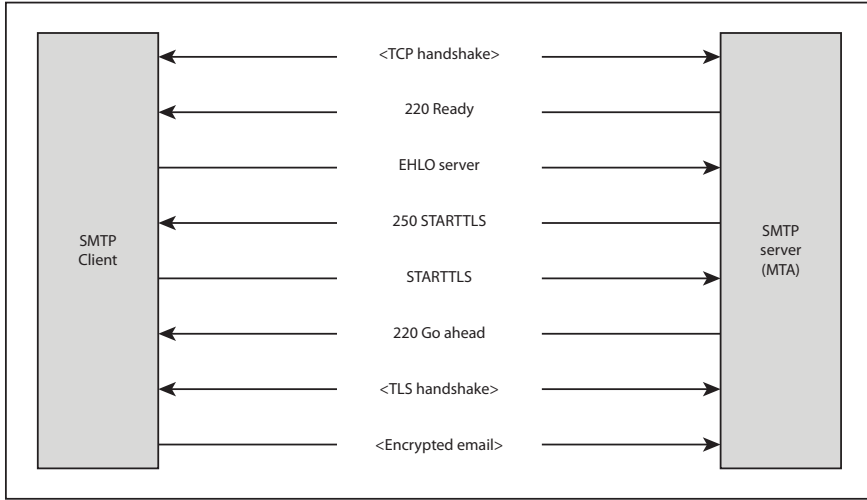


Fig. 3. SMTP STARTTLS session between client and server.⁴⁸

where SMTP TLS fails, such as the use of validated digital certificates when an SMTP connection is made, in such a way that the connection between the servers is encrypted and the famous attack called man- in-the-middle is avoided.

4.3. Protocol DANE

RFC6698 presented the DANE protocol (DNS-Based Authentication of Named Entities).⁵⁰ The DANE protocol is based on DNSSEC and aims to improve the TLS scenario and the unwanted CAs. The use of certificates has the problem that the signature of the certificate only makes sense if you trust in the key that signed it, which means that you must have a list of CA at hand.

DNSSEC works like CAs, following a chain of trust, but differs from CAs in the fact that a domain will only be signed by its parent domain. So, DANE uses the DNSEC infrastructure as storage of the signatures of the keys and certificates of any TLS service.

It can be said that DANE works as a normal TLS service but when the client obtains the certificate to validate the service, it makes a DNS request to see if it matches the stored certificate. If they agree is that everything is perfect.

What makes DANE so special is its main problem, DNSSEC is not very widespread, browsers do not support it and there are few sites that have it implemented.

4.4. DARK MAIL

Dark Mail is a new approach for email that aims to completely replace SMTP and the other access protocols such as POP and IMAP. The project can be read under

the name DIME (Dark Internet Mail Environment) and proposes the DMTP and DMAP protocols.⁵¹

Basically, the header of the message is deleted replacing it with routing information. Only the IP address is exposed arguing that Tor and VPNS need them too and it is impossible to eliminate them. It provides point-to-point security over routing and the message. All the keys are generated at the end point and the servers do not know them. It uses AES or equivalent to encrypt the content and continues with the use of DNS, but the idea is to use another system. For now, it is experimental and is expected to grow in the coming years.

4.5. *PROTONMAIL*

Protonmail is a solution that was born with the aim of protecting all people from the surveillance of governments and is a strong candidate to be one of the future safe mail alternatives in the world.⁵²

The solution allows end-to-end encryption, so that users are the only ones who know the password to decrypt the mailbox, that is, the servers only store the data in an encrypted form. It uses cryptosystems of public key, in such a way that when a user creates an account, the browser generates a pair of RSA keys, one public and one private. The public key is used to encrypt mailbox messages using AES-256. The public and private keys are stored in the Protonmail servers in an encrypted way. TLS is used to protect all traffic between clients and servers.

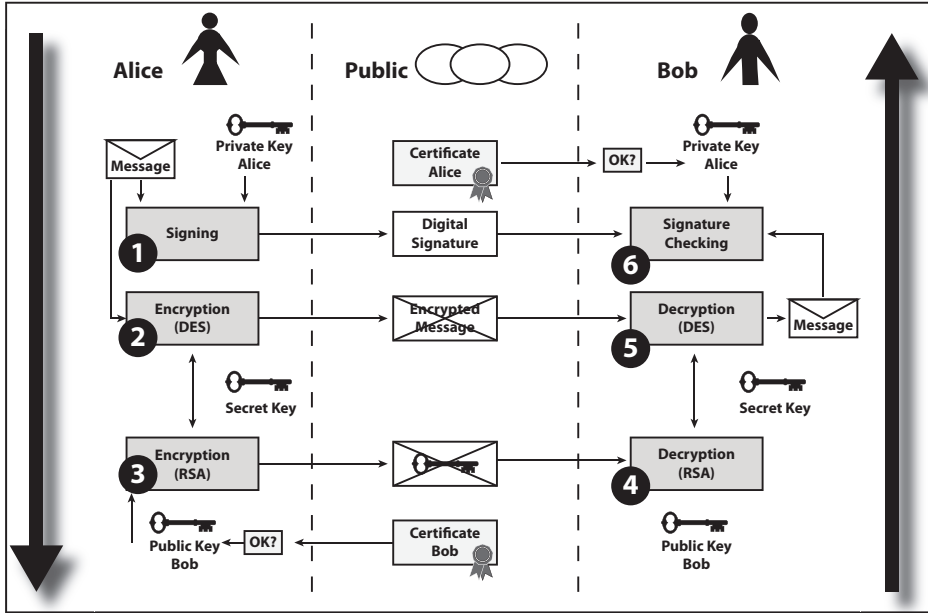
4.6. *BITMESSAGE*

It uses a totally decentralized system under a point-to-point communication protocol based on the famous bitcoins. The message is sent to everyone on the network, but only the receiver can decode it with their private key. Users can subscribe to messages from an address. It has the problem that the addresses are very long and unfriendly. It is not scalable and has usability problems.⁵³

4.7. *S/MIME*

RFC1847 defines a framework in which security services can be applied to the part of the mail body called MIME.⁵⁴

This standard is based on public key cryptosystems; in S/MIME what is encrypted is the message and not the communication. Each section of the message is encrypted using a session key. The session key is inserted in each heading of the sections and encrypted with the public key of the recipient, so that only the recipient can read it, deciphering it with its private key. Likewise, the signature of the message is encrypted with the private key of the sender, and the integrity of the message can be verified. Figure 4 illustrates a typical scenario of S/MIME.

Fig. 4. Typical S/MIME scenario.⁵⁵

4.8. OpenPGP

OpenPGP is defined in RFC4880 (Callas et al., 2007) in 2007, although its first specification appears in RFC2440 (Callas et al., 1998) in the year 1998. It is not really an email-oriented solution but has been used in many mail security solutions such as RFC3156 “MIME Security with OpenPGP”⁵⁶ and is widely used by many user communities on the Internet to encrypt.

Many email clients offer security with OpenPGP through signing and encryption of messages. It is a solution that is not very scalable, but we wanted to refer it in this work because it is widely accepted.

4.9. DKIM — DomainKeys Identified Mail

As the 2007 RFC4870 says, DomainKeys provides a framework for domain-level authentication for email using public key technology and DNS to prove the origin and content of an email.⁵⁷

The DKIM specification has been derived from DomainKeys and was defined in RFC4871.⁵⁸ This document states that one of DKIM’s ultimate goals is to protect the identity of the signer and the integrity of the messages while retaining the mail functionality as it is known today.

Among the advantages of DKIM over other solutions:

- It is compatible with existing mail infrastructures and it is transparent.
- It requires very little to implement it.

- It can be implemented independently of the clients.
- It can be expanded incrementally.
- It allows the delegation of signatures to third parties.

Figure 5 illustrates the process of sending and receiving mail with DKIM.

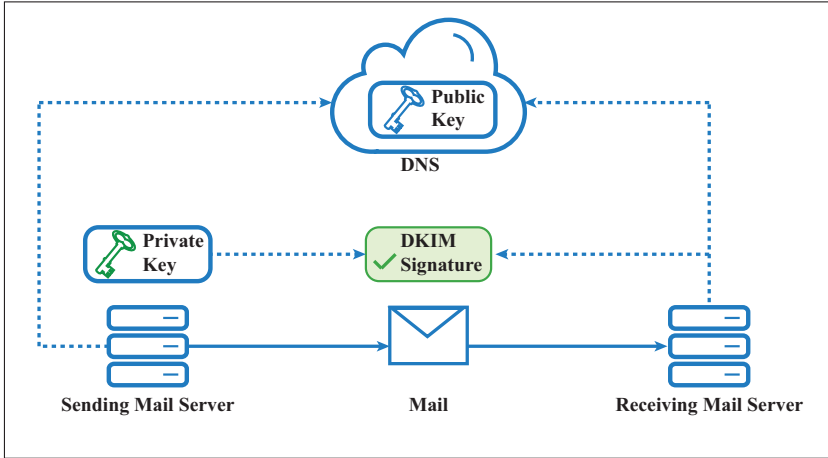


Fig. 5. Sending and receiving with DKIM.⁵⁹

4.10. SPF

Sender Policy Framework (SPF) is a system whose main objective is to protect against email address spoofing. It uses DNS to identify the domain names of SMTP servers that have been authorized for mail transport.⁶⁰

Although it is a very simple idea, to have a white list of machines that can send email, SPF is quite widespread, since it facilitates the identification of origin by the SMTP servers of destination.

4.11. SIDF

Sender-ID Framework (SIDF) is based on the same SPF concept. But SPF authenticates the domain in the “return-path” field found in the email headers, while SIDF also authenticates the from field visible to the user, which is called PRA (Purported Responsible Address). This validation allows one more layer of protection against Spammers and Phishers⁶¹

4.12. Other solutions

Other different services have been emerging, but all of them are based on same protocols SMTP, POP3 and IMAP. Some services for example are CryptoMail, LavaBit, CryptoMail, etc. All these applications use the services or solutions discussed above, which is why they are only mentioned.

5. Conclusions and Future Work

This work begins with the study of the evolution of electronic mail from its inception, to what is considered a state of the art. The objective was to try to understand why today an asynchronous messaging solution full of vulnerabilities and weaknesses is available to users. Although it seems contradictory, these vulnerabilities gave a boost to the evolution of technology itself. Once the study is finished, a compilation of the security requirements that should be included in a new proposal on e-mail has been made.

In summary, the contribution of this work is the specification of the requirements for a solution to the security of email that meets the security requirements and objectives needed for an asynchronous messaging system such as Confidentiality, Integrity, Availability, Traceability, Authentication and access control or Authorization.

We must remember the objectives pursued in this work were:

- Understand the historical security framework and the evolution of electronic mail since its inception. This objective has been achieved with the study of the history of use of electronic mail, as well as the considerations that have been contemplated for its evolution.
- Analyze in detail the reasons why today there is no concrete and formal solution to the problems of email security. In addition to the background, we have studied the most significant proposals that have been appearing with the specification of the current mail. It can be observed that only small cures have been added to the problems that have been appearing.
- Determine the requirements on which the current and old email systems have been developed. These requirements have always revolved around the need to share information but have changed radically at the time that the model has been required to be safe.
- Understand the current threats and vulnerabilities that determine the security requirements of the solutions currently proposed. The state of the art of the most important or notable threats and vulnerabilities has been studied, and from them security requirements have been defined that should be incorporated into a new model.
- Understand from the point of view of the developers how security needs have been added and on which they have been based. Until now, these security needs have been based on the attacks received and not on an in-depth study of the problems that the initial model was dragging and in which the security requirements had not been considered.
- Identify the needs of the users and how they have been supplying with the proposed electronic mail solutions. Users are the most affected, due to their ignorance on security issues and have been the main victims of security deficiencies in the current email. Until now they have had to be trained to the point that it is well accepted that a security problem is the low training of users.

The proposed solution should consider the following requirements and propose an email solution that is usable and meets the minimum-security requirements, such as:

- (1) The security of the mailboxes should oversee the security where the messages are hosted.
- (2) It is necessary that the solution provide in an integral and proprietary way the means to control spam.
- (3) The location of mailboxes should be controlled by the user, but the proposal should provide security for them.
- (4) The new email system should allow traceability at the level of the dialogue.
- (5) The user should be able to determine who can send an email.
- (6) Mailbox should be differentiated from Mailbox addresses and this must be transparent to the user.
- (7) A distinction should be made between author and sender
- (8) The author should always be verified to avoid falsification.
- (9) The distribution lists should be signed.
- (10) Access to stored mail should be controlled by means of an access list or similar. That is, more than one user should be able to access the mail, not just the owner.
- (11) The user should always be validated.
- (12) The signature should always be validated.
- (13) The user should be able to use several mailboxes.
- (14) It is necessary to provide security mechanisms for the forwarding and validation and verification of the sender and recipient when passing through a server.
- (15) A solution must be provided for the designation, location and mobility of the mailboxes. A mailbox can be hosted anywhere the user designates and that change must be controlled by the new model.
- (16) It is necessary that the message can contain any multimedia content.
- (17) It is necessary that the message can contain any multimedia content and the model must support it.
- (18) The mailbox must be uniquely named and there should be no confusion between the user and the mailbox, as they should be independent names.
- (19) It is imperative to trace the mail, but the content of the message should only be read by its recipient, not even by computers or intermediate processes. All transport information should be separated from the content.
- (20) The new model should allow the use of different authentication methods in order to be more flexible.
- (21) The mail should always be available, and the user could work "offline".
- (22) The user should be able to access all the content or a summary of the message with the aim of not charging customers and optimizing communications.

- (23) The new model should ensure the information of the mail in its three security dimensions: Confidentiality, Availability and Integrity.
- (24) The new model should be able to easily integrate other proposals or already consolidated tools such as PGP or PKI, but it should be clear that such integration should be transparent to the user.
- (25) The message management system should be considered in the new proposal or model, with the aim of supporting activities of management and control of tasks, as well as management threads of meetings or professional classifications.
- (26) The security mechanisms used should not affect the speed of use of the mail.
- (27) Communication should not be planned; communication is desirable to be asynchronous.
- (28) Security must be free.
- (29) Under certain circumstances, it should be possible to control, locate and trace the email.
- (30) Sending emails to services is indispensable.
- (31) Spoofing and the uncontrolled spread of viruses should be avoided in the proposal.
- (32) An email should have an associated author and is indivisible from its content.
- (33) It should be possible to distinguish organizational mailboxes from personal mailboxes; this feature must be included in the proposal.

These 33 requirements must be implemented by any tool that wants to provide a security solution for email and that meets the requirements of today's users. We are working in an integral solution that meets all requirements shown here in this paper.

Our proposal focuses on the creation of a Blockchain-based email protocol. A blockchain is public ledger of all transactions that a group of participants have executed. The interesting thing about Blockchain is that all the information stored on it have been verified by consensus of most of the participants in the system. The information can never be modified or erased. The blockchain contains information of the security of every single transaction saved.

The proposed protocol will allow the decentralization of email both at the storage and security level, since the content of the email will be distributed and the security will be organized in layers, with the first layer of security being the one that provides a protocol based on Blockchain by default.

The solution we are working on is basically focused on solving the 33 requirements presented in this study. For the decentralization of email we are working on two key aspects: Addressing and storage. The addressing part basically focuses on using a blockchain that allows this distribution and uniqueness of addresses, without using external servers to validate addresses. Regarding the distribution of the storage, dynamic storage is proposed based on a smart contract associated with the email address. To guarantee the confidentiality, integrity and availability of the

service, the blockchain is ideal due to its distributed and secure nature. Due to the distributed nature of the solution itself, the system appears as an open system with a large amount of data available. In this type of solutions, many security aspects must be considered. Some authors^{62–64} make a deep review of those aspects that we have considered vital in our work.

References

1. T. V. Vleck, *The History of Electronic Mail* (2001).
2. S. Sluizer and J. B. Postel, *Mail Transfer Protocol*, RFC 780 (Information Sciences Institute, University of Southern California, 1981).
3. J. B. Postel, *Simple Mail Transfer Protocol*, RFC788 (Information Sciences Institute, University of Southern California, 1981).
4. Symantec Corporation, *ISTR Internet Security Threat Report Volume 24*, February 2019, Technical Report February, Symantec Corporation, 2019.
5. I. Sanchez and G. Draper-Gil, *My Email Communications Security Assessment (MECSA): 2018 Results*, Technical report, Joint Research Centre (JRC), 2019.
6. B. D. Payne and W. K. Edwards, A brief introduction to usable security, *IEEE Internet Computing* **12** (2008) 13–20.
7. S. Farrell, Why Don't We Encrypt Our Email?, *IEEE Internet Computing* **13** (2009) 82–85.
8. C. T. Moecke and M. Volkamer, Usable secure email communications: criteria and evaluation of existing approaches, *Information Management & Computer Security* **21** (2013) 41–52.
9. S. Ruoti, J. Andersen, S. Heidbrink, M. O'Neill, E. Vaziripour, J. Wu, D. Zappala and K. Seamons, "We're on the Same Page": A usability study of secure email using pairs of novice users, in *Proc. 2016 CHI Conference on Human Factors in Computing Systems – CHI '16* (2016), pp. 4298–4308.
10. Q. Ren, Y. Mu and W. Susilo, SEFAP: An email system for anti-phishing, in *Proc. 6th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2007; 1st IEEE/ACIS International Workshop on e-Activity, IWEA 2007* (2007), pp. 782–787.
11. P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong and E. Nunge, Protecting people from phishing, in *Proc. SIGCHI Conference on Human Factors in Computing Systems – CHI '07* (ACM Press, New York, New York, USA, 2007), p. 905.
12. J. Crain, L. Opyrchal and A. Prakash, Fighting phishing with trusted email, in *ARES 2010 – 5th International Conference on Availability, Reliability, and Security* (2010), pp. 462–467.
13. I. R. A. Hamid and J. Abawajy, Phishing email feature selection approach, in *Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2011, 8th IEEE Int. Conf. on Embedded Software and Systems, ICESS 2011, 6th Int. Conf. on FCST 2011* (2011), pp. 916–921.
14. J. J. Meyers, D. L. Hansen, J. S. Giboney and D. C. Rowe, Training future cybersecurity professionals in spear phishing using sieve, in *Proc 19th Annual SIG Conference on Information Technology Education – SIGITE '18* (Association for Computing Machinery, New York, NY, USA, 2018), pp. 135–140.
15. J. Petelka, Y. Zou and F. Schaub, Put your warning where your link is: Improving and evaluating email phishing warnings, in *Proc. 2019 CHI Conference on Human Factors in Computing Systems – CHI '19* (Association for Computing Machinery, New York, NY, USA, 2019), pp. 1–15.

16. P. A. Lawson, A. D. Crowson and C. B. Mayhorn, Baiting the hook: Exploring the interaction of personality and persuasion tactics in email phishing attacks, in *Proc. 20th Congress of the International Ergonomics Association (IEA 2018)*, eds. S. Bag-nara, R. Tartaglia, S. Albolino, T. Alexander and Y. Fujita (Springer International Publishing, Cham, 2019), pp. 401–406.
17. E. Benavides, W. Fuertes, S. Sanchez and M. Sanchez, Classification of Phishing Attack Solutions by Employing Deep Learning Techniques: A Systematic Literature Review, in *Developments and Advances in Defense and Security. Smart Innovation, Systems and Technologies*, Vol. 152, eds. Á. Rocha and R. P. Pereira (Springer, Singapore, 2019), pp. 51–64.
18. R. M. Amin, Detecting targeted malicious email through supervised classification of persistent threat and recipient oriented features, *Computer and Telecommunications Engineering M.S. Telecommunications and Networking* (2011).
19. C. Hutzler, D. Crocker, P. Resnick, E. Allman and T. Finch, Email Submission Operations: Access and Accountability Requirements, *RFC5068*, Network Working Group (2007).
20. J. Rosenberg and C. Jennings, The Session Initiation Protocol (SIP) and Spam, *RFC5039*, Network Working Group (2008).
21. C. Dhinakaran, J. K. Lee, D. Nagamalai and C. J. Chae, An empirical study of spam and spam vulnerable email accounts, *Proc. Future Generation Communication and Networking, Main Conference Papers, Vol. 1*, **1** (2007) 407–412.
22. N. Jindal and B. Liu, Opinion spam and analysis, *Proc. International Conference on Web Search and Web Data Mining (WSDM 08)* (2008), pp. 219–220.
23. A. C. Kigerl, Email spam origins: does the CAN SPAM act shift spam beyond United States jurisdiction?, *Trends in Organized Crime* **21** (2018) 62–78.
24. E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi and O. E. Ajibuwa, Machine learning for email spam filtering: review, approaches and open research problems, *Heliyon* **5** (2019) e01802.
25. A. Alghoul, S. Al Ajrami, G. Al Jarousha, G. Harb and S. S. Abu-Naser, Email classification using artificial neural network, *Int. Journal of Academic Engineering Research* **2** (2018) 8–14.
26. M. H. Barber, C. Hagemann, C. J. Hockings and B. Waters, *Similar Email Spam Detection*, Patent (International Business Machines Corporation, Armonk, NY, 2018).
27. M. Hassonah, A. Sheta, J. Alqatawna, H. Faris, A. M. Al-Zoubi and M. Habib, Automatic email spam detection using genetic programming with smote, in *The Fifth HCT Information Technology Trends (ITT 2018)* (2018), pp. 185–190.
28. S. M. Dennis, Blockchain Based Email Procedures (2019).
29. A. J. Bellamy and M. McDonald, “The utility of human security”: Which humans? What security? A reply to Thomas & Tow, *Security Dialogue* **33** (2002) 373–377.
30. S. L. Garfinkel, Enabling email confidentiality through the use of opportunistic encryption, in *Proc. 2003 Annual National Conference on Digital Government Research* (2003), pp. 1–4.
31. J. M. Seigneur and C. D. Jensen, Privacy recovery with disposable Email addresses, *IEEE Security and Privacy* **1** (2003) 35–39.
32. D.-H. Shih and H.-S. Chiang, E-mail viruses: how organizations can protect their e-mails, *Online Information Review* **28** (2004) 356–366.
33. M. Abadi and B. Blanchet, Computer-assisted verification of a protocol for certified email, *Sci. Comput. Program.* **58** (2005) 3–27.
34. B. Adida, S. Hohenberger and R. L. Rivest, Lightweight encryption for email, *Proc. Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted*

- Traffic on the Internet Workshop* (USENIX Association, 2005), p. 13.
35. J. Zhou, J. Onieva and J. Lopez, Optimized multi-party certified email protocols, *Information Management & Computer Security* **13** (2005) 350–366.
36. K. Butler, W. Enck, J. Plasterr, P. Traynor and P. McDaniel, Privacy Preserving Web-Based Email, *Information Systems Security (ICISS 2006)* **4332** (2006) 116–131.
37. Z. Wu, J. Tian, L. Li, C.-P. Jiang and X. Yang, A secure email system based on fingerprint authentication scheme, in *Intelligence and Security Informatics, 2007 IEEE*, New Brunswick, NJ, USA (IEEE, 2007), pp. 250–253.
38. A. Herzberg, DNS-based email sender authentication mechanisms: A critical review, *Computers & Security* **28** (2009) 731–742.
39. D. Choukse, U. Singh, L. Laddhani and R. Shahapurkar, Designing secure email infrastructure, in *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)* (2012), pp. 1–9.
40. Z. Lina and M. H. Jiang, The Security Email Based on Smart Card, *Physics Procedia* **33** (2012) 1634–1639.
41. W. Alge, Email in the cloud: the challenges and benefits, *Computer Fraud & Security* **2012** (2012) 10–12.
42. T. Hahn, T. Kunz, M. Schneider and S. Vowé, Vulnerabilities through usability pitfalls in cloud services: Security problems due to unverified email addresses, *Proc. 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 – 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012* (2012), pp. 850–856.
43. N. K. M. Madi, S. Salehian, F. Masoumiyan and A. Abdullah, Implementation of secure email server in cloud environment, in *2012 International Conference on Computer and Communication Engineering, ICCCE 2012* (2012), pp. 28–32.
44. L. Dykcik, L. Chuat, P. Szakachowski and A. Perrig, BlockPKI: An automated, resilient, and transparent public-key infrastructure, in *IEEE International Conference on Data Mining Workshops (ICDMW 2018)* (IEEE, Singapore, 2018), pp. 105–114.
45. N. M. Smith and D. W. Grawrock, *Public Key Infrastructure Using BlockChains* (2018).
46. P. Cocca, E-mail Security Threats, *SANS Institute* (2004).
47. The Radicati Group, Email Market, 2019–2023, Technical Report 0 (The Radicati Group, 2019).
48. P. Hoffman, SMTP Service Extension for Secure SMTP over Transport Layer Security, *RFC3207* (Network Working Group, 2002).
49. D. Margolis, M. Risher, B. Ramakrishnan, A. Brotman and J. Jones, SMTP MTA Strict Transport Security (MTA-STS), *RFC8461* (2018), pp. 1–29.
50. P. Hoffman and J. Schlyter, The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA, *RFC6698* (Internet Engineering Task Force (IETF), 2012).
51. P. Zimmermann, J. Callas, M. Janke and L. Levison, Dark Internet Mail Environment, Technical report, 2015.
52. ProtonMail Website (2016).
53. BITMESSAGE Website (2016).
54. J. Galvin, S. Murphy, S. Crocker and N. Freed, Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted, *RFC1847* (Network Working Group, 1995).
55. Hakawati, Hakawati Security Lab (2016).
56. M. Elkins, D. D. Torto, R. Levien and T. Roessler, MIME Security with OpenPGP, *RFC3156* (Network Working Group, 2001).

57. M. Delany, Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys), *RFC4870* (Network Working Group, 2007).
58. E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton and M. Thomas, Domain Keys Identified Mail (DKIM) Signatures, *RFC4871* (Network Working Group, 2007).
59. Barracuda Sentinel, Configuring SPF, DKIM and DMARC (2019).
60. M. Wong and W. Schlitt, Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1, *RFC4408* (Network Working Group, 2006).
61. J. Lyon and M. Wong, Sender ID: Authenticating E-Mail, *RFC4406* (Network Working Group, 2006).
62. J. F. Herrera-Cubides, P. A. Gaona-García, C. E. Montenegro-Marín D. M. Cataño, and R. Gonzáles-Crespo, *Security Aspects in Web of Data Based on Trust Principles. A Brief of Literature Review* **11** (2019) 365–379.
63. C. Moreno, R. A. Carrasco and E. Herrera-Viedma, Data and Artificial Intelligence Strategy: A conceptual enterprise big data cloud architecture to enable market-oriented organisations, *Int. Journal of Interactive Multimedia and Artificial Intelligence* **5** (2019) 7.
64. J. Rodríguez, B. Corredor and C. Suárez, Genetic operators applied to symmetric cryptography, *Int. Journal of Interactive Multimedia and Artificial Intelligence* **5** (2019) 39.



A Secure Email Solution Based on Blockchain

Diego Piedrahita^(✉) , Javier Bermejo , and Francisco Machío 

Universidad Internacional de la Rioja, Logroño, Spain
diego.piedrahita@comunidadunir.net, {javier.bermejo,
francisco.machio}@unir.net

Abstract. Email is one of the most important online communication services between individuals and businesses. The large amount of information that passes through this medium is the object of desire for many attackers who use a whole series of malicious maneuvers to get hold of it. The number of emails circulating in the world per day is growing, going from 319.6 billion in 2021 to more than 361 billion by the end of 2024. Attackers use all kinds of attacks to participate in these communications, perpetuating attacks that range from spam to very sophisticated attacks such as phishing, scams, among others. There are many solutions available in the market and the email security problem continues to grow and there seems to be no solution in sight. We are working on a different approach. First, we have deeply analyzed the state of the art of email in terms of its design, the security problems identified and the proposed solutions. Subsequently, we have identified the security requirements necessary to address the problem and based on them we have designed an architecture based on a Blockchain platform, whose components interact through different protocols to achieve the objectives proposed in our research.

Keywords: Email · Blockchain · Security

1 Introduction and Problem Statement

The integrity, confidentiality and availability of information resources are the primary objectives of information security. Electronic mail (email), defined as a digital message that is transmitted between a sender and a receiver, contains information and any vulnerability in these three key areas will expose the system to any exploitation [1]. Email is still the most widely used electronic communication method on the internet by many users [2]. In 2004 SANS Institute conducted a study that reflected what were the email security threats at that time. Viruses, unwanted mail (spam) and phishing were the main problems and most of the viruses were hidden in the attachments of the mail messages [1]. After many years of the publication of that report, the same problems persist [3]. These attacks generate large economic losses to organizations and individuals [4].

2 Related Work

We have studied the countless applications of Blockchain in different domains [5], and we have identified that many companies do not take risks when approaching a project with

a technology where security aspects are under study. We see in the scientific literature different approaches to address this problem using Blockchain [6–8].

There are several commercial proposals available in the market offering secure mail on Blockchain, but there is little technical documentation on their construction, such as Mailchain [9]. Blockchain has been used as an extension to email to perform activities such as validation of unknown senders [10], validation of emails to verify integrity and to know if the email has not been forged [11], the use of payment with cryptocurrencies to be able to send emails [12], the use of decentralized reputation systems on Blockchain to avoid spam [13] or the use of Blockchain for sending certified mail without a trusted third party [14]. This type of work focuses on the use of Blockchain as an additional module to the traditional email system that provides a service to mitigate some of the problems that we have previously discussed.

Another type of proposal focuses on providing a comprehensive solution using Blockchain as the central email service. One of the papers proposes the use of the same identifiers used in Blockchain transactions as user identifiers [15]. Another proposal makes use of Blockchain to store messages using a proprietary server [16]. Finally, one of the analyzed works is the one presented by González in which he proposes to replace email protocols with smart contracts [17]. We must consider that some proposals are in development and apparently the architecture has not been designed in its entirety, but aspects are defined as progress is made.

3 Hypothesis

The research work hypothesis is stated as follows: Email today presents innumerable security vulnerabilities that can be resolved using a distributed and secure digital alternative mail system based on a new protocol, according to the technological advances of recent times in computer security and that complies with the basic principles of information security and the requirements of the different participants who make use of the solution.

4 Proposal

Our goal is to design a secure email solution based on a flexible and integrable architecture that is usable and integrable by users and that solves the problem of email security by providing services of confidentiality, integrity, availability, non-repudiation, authentication, control, and audit. The architecture will use Blockchain as one of its core components and the interaction of its components will be regulated by a series of distributed protocols that will be responsible for providing security to all processes (Fig. 1).

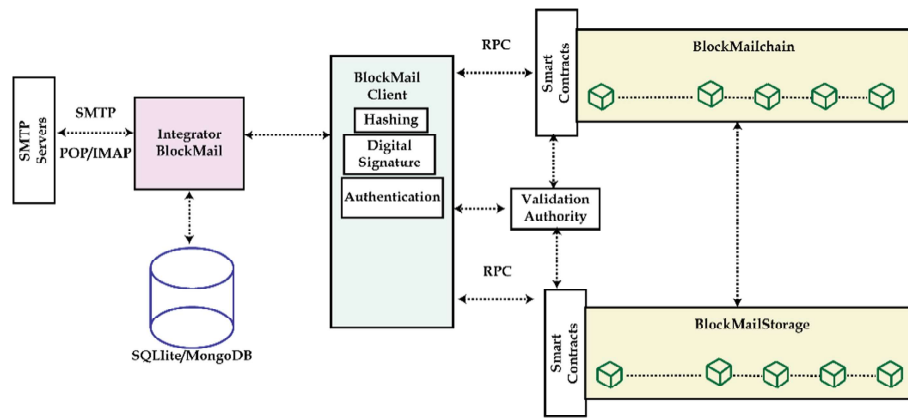


Fig. 1. Proposed architecture

5 Preliminary Results

We have developed the state of the art of email and the different proposals developed in the last 5 years. We are developing the proof of concept with different use cases that we have been defining. In our study we have identified eight aspects that must be considered when addressing security in the context of email and they are confidentiality, integrity, non-repudiation, authentication, availability, control, auditing, usability, and integration. Based on these eight aspects and the literature reviewed on email security, we have identified 43 security requirements that an email solution must meet to be considered secure. We have designed an architecture using a public blockchain that controls security aspects and another private blockchain that manages email information. The protocols are smart contracts that provide solutions to the identified security requirements. The architecture also provides integration with SMTP servers through a gateway called the integrator, but it is a partial solution while the new model is adopted.

6 Reflections

Blockchain technology can be used to provide security through smart contracts. The use of smart contracts seen as a protocol facilitates the standardization of many processes. However, Blockchain is not ideal for intensive user-centric systems or for data models whose rules do not require strict and detailed historical state management, as in the case of email. The decentralization feature offered by this technology is ideal for minimizing availability and integrity issues, but not for solving confidentiality issues or for storing large volumes of data such as those handled in email. To solve this problem, we had to develop a hybrid solution between private and public Blockchains, but that meets the identified requirements. We hope that our work will be a starting point for many other works and together we can offer an ideal solution to this problem that affects us all.

References

1. Cocca, P.: E-mail security threats. Sans Institute (2004). <https://doi.org/10.9780/22307850>
2. The Radicati Group, I. Email statistics report, 2020–2024 (2020)
3. Proofpoint: Q1–2018 quarterly threat report 1–16 (2018)
4. FBI's Internet Crime Complaint Center: 2019 Internet Crime Report. 2019 Internet Crime Report 1–28 (2019)
5. Pournader, M., Shi, Y., Seuring, S., Koh, S.C.L.: Blockchain applications in supply chains, transport and logistics: a systematic review of the literature. *Int. J. Prod. Res.* **58** (2020). <https://doi.org/10.1080/00207543.2019.1650976>
6. Taylor, P.J., Dargahi, T., Dehghantanha, A., Parizi, R.M., Choo, K.K.R.: A systematic literature review of blockchain cyber security. *Digital Commun. Netw.* **6**(2), 147–156 (2020)
7. Bernal Bernabe, J., Canovas, J.L., Hernandez-Ramos, J.L., Torres Moreno, R., Skarmeta, A.: Privacy-preserving solutions for blockchain: review and challenges. *IEEE Access* **7**, 164908–164940 (2019)
8. Vassilakis, C.: Blockchain technologies for leveraging security and privacy. *Homo Virtualis*. **2** (2019). <https://doi.org/10.12681/homvir.20188>
9. Mailchain <https://mailchain.xyz/>
10. Dennis, S.M.: Blockchain based email procedures (US Patent No. US010305833B1) (2019)
11. Varghese, A.: Email verification service using blockchain 0–7 (2019)
12. Sheikh, S.A., Banday, M.T.: A cryptocurrency-based E-mail system for SPAM control. *Int. J. Adv. Comput. Sci. Appl.* **12** (2021). <https://doi.org/10.14569/IJACSA.2021.0120139>
13. Kannan, L., Jebakumar, R.: Public Sender Score System (S3) by ESPs for Email spam mitigation with score management in mobile application. *Int. J. Interact. Mob. Technol.* **14** (2020). <https://doi.org/10.3991/ijim.v14i17.16609>
14. Hinarejos, M.F., Ferrer-Gomila, J.: A solution for secure multi-party certified electronic mail using blockchain. *IEEE Access* **8**, 102997–103006 (2020). <https://doi.org/10.1109/ACCESS.2020.2998679>
15. Khacef, K., Pujolle, G.: Secure peer-to-peer communication based on blockchain. In: Barolli, L., Takizawa, M., Xhafa, F., Enokido, T. (eds.) *WAINA 2019. AISC*, vol. 927, pp. 662–672. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15035-8_64
16. Menegay, P., Salyers, J., College, G.: Secure communications using blockchain technology. In: *Proceedings of the IEEE Military Communications Conference MILCOM* (2019)
17. González, J.C., García-Díaz, V., Núñez-Valdez, E.R., Gómez, A.G., Crespo, R.G.: Replacing email protocols with blockchain-based smart contracts. *Clust. Comput.* **23**(3), 1795–1801 (2020). <https://doi.org/10.1007/s10586-020-03128-9>

