Aionda Mail Security Whitepaper

Zero-Knowledge · Post-Quantum · Made in Germany

Version 1.0 — March 2026

Aionda GmbH
Stuttgart, Germany

contact-46epp9ba@contact.aionda.com
https://mail.aionda.com

PUBLIC DOCUMENT

Contents

1. Libro blanco de seguridad de Aionda Mail

Versión 1.0 — Marzo 2026 Aionda GmbH, Stuttgart, Alemania

1.1 Índice

1.2 1. Resumen ejecutivo

Aionda Mail es un servicio de correo electrónico Zero-Knowledge con cifrado post-cuántico, operado por Aionda GmbH en Stuttgart, Alemania. El servicio combina direcciones de correo desechables (DEAs) con un buzón de correo completamente cifrado — una combinación que ningún otro proveedor ofrece.

Propiedades de seguridad principales:

- **Arquitectura Zero-Knowledge:** Todo el cifrado y descifrado ocurre exclusivamente en el navegador del usuario. El servidor nunca tiene acceso al contenido de los correos en el buzón seguro, contraseñas ni claves de cifrado.
- **Seguridad post-cuántica:** El mecanismo de encapsulación de claves híbrido (X25519 + ML-KEM-1024) protege todos los datos contra ataques tanto de computadoras clásicas como cuánticas.
- **Autenticación Zero-Knowledge:** El protocolo OPAQUE (RFC 9807) asegura que las contraseñas nunca se transmiten ni se almacenan en el servidor — ni siquiera como hashes.

- **Shamir Secret Sharing (2-de-3):** La clave maestra del vault se divide en tres partes protegidas por contraseña, passkey y clave de recuperación. Dos cualesquiera de las partes reconstruyen la clave maestra.
- **Perfect Forward Secrecy:** Cada solicitud API utiliza un par de claves criptográficas único y de un solo uso. Comprometer una solicitud no afecta a ninguna otra.
- **Protección MITM (Guardian):** La extensión del navegador verifica de forma independiente todas las respuestas del servidor mediante firmas Ed25519 y detecta ataques de intermediario a través de la verificación de certificados TLS — incluso contra proxies corporativos y CDNs comprometidos.
- **Archivo de correo conforme a GoBD:** Las cuentas empresariales se benefician de una cadena de hashes a prueba de manipulación (blockchain SHA3-256) con contenido cifrado de extremo a extremo (Hybrid KEM), registro de auditoría completo, retención legal y retención configurable — conforme con la normativa alemana GoBD.
- **Sin recuperación de contraseña:** En caso de pérdida de la contraseña y todos los métodos de recuperación, los datos son irrecuperables. Esto es por diseño — demuestra que el servidor no puede acceder a los datos del usuario.

Jurisdicción: Legislación alemana (DSGVO/RGPD), sin intercambio de datos con agencias de inteligencia extranjeras.

1.3 2. Modelo de amenazas

1.3.1 2.1 Contra qué protege Aionda Mail

Amenaza	Protección
Compromiso del servidor (filtración de base de datos, acceso interno)	Todo el contenido del correo cifrado con claves que el servidor nunca posee
Espionaje de red (ISP, Wi-Fi, CDN)	Transporte API cifrado de extremo a extremo mediante Hybrid KEM
Inspección de CloudFlare	Las solicitudes API se cifran antes de salir del navegador; CloudFlare solo ve texto cifrado. La extensión Guardian detecta manipulación de respuestas mediante firmas Ed25519
Proxies MITM corporativos (ZScaler, Fortinet, etc.)	La extensión Guardian detecta certificados de proxy mediante lista de bloqueo de emisores (Firefox)
Ataques de computadoras cuánticas (“cosechar ahora, descifrar después”)	ML-KEM-1024 (NIST FIPS 203) proporciona resistencia post-cuántica
Robo de base de datos de contraseñas	OPAQUE almacena solo registros criptográficos, no hashes de contraseñas
Ataques de fuerza bruta offline contra contraseñas	OPAQUE previene ataques offline; limitación de velocidad del lado del servidor previene ataques online
Análisis de tamaño de correos	El Bucket Padding oculta los tamaños reales de los correos
Canales laterales de compresión (CRIME/BREACH)	Bucket Padding aplicado después de la compresión

Amenaza	Protección
Enumeración de usuarios	Respuestas falsas determinísticas para cuentas inexistentes

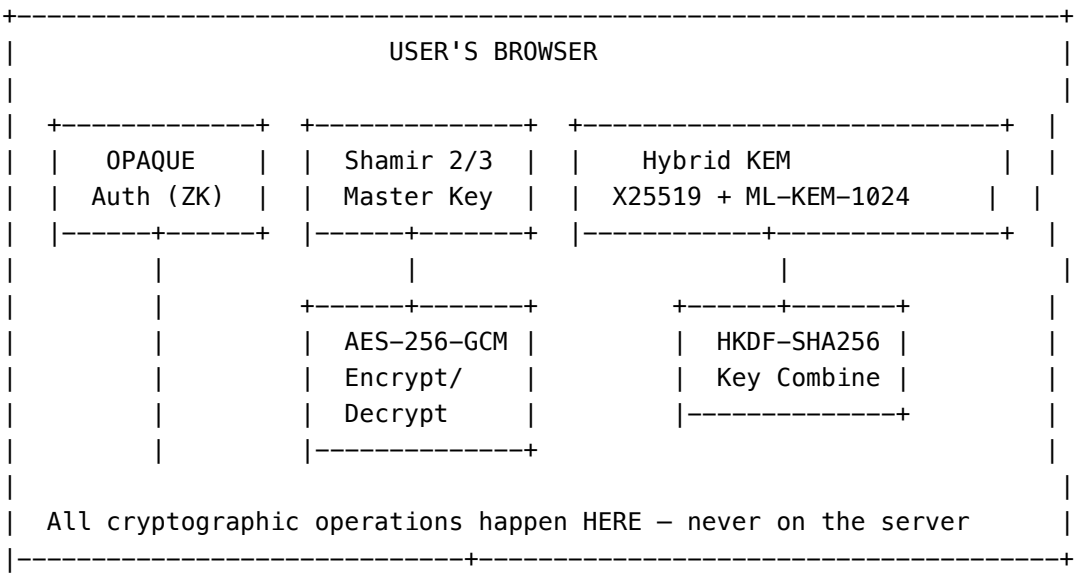
1.3.2 2.2 Contra qué NO protege Aionda Mail

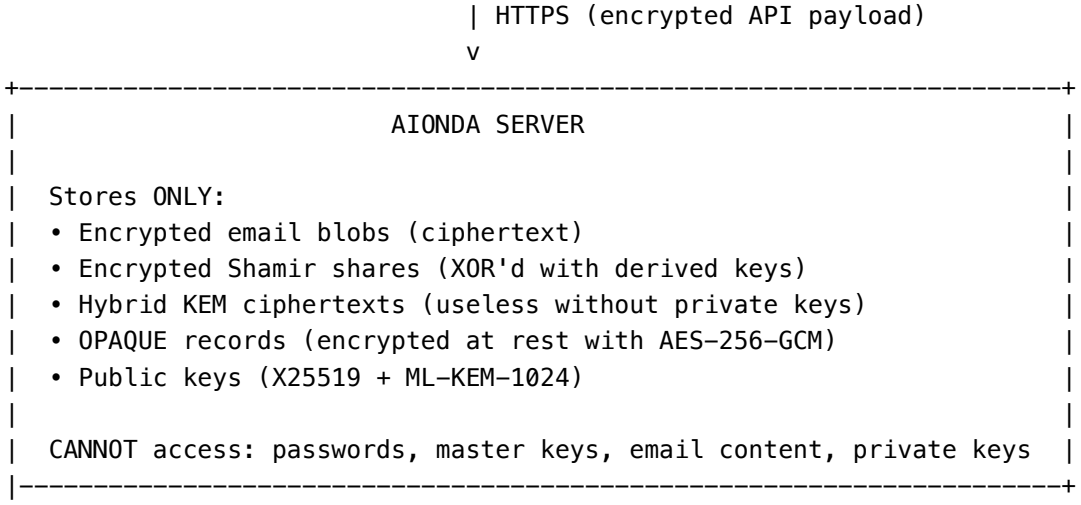
Limitación	Explicación
Dispositivo comprometido	Si un malware controla el navegador, puede leer el contenido descifrado
Metadatos hacia destinatarios externos	Los correos hacia Gmail/Outlook viajan sin cifrar tras abandonar nuestros servidores (a menos que se use PGP)
Metadatos de correo en nuestros servidores	Las marcas de tiempo, direcciones IP y tamaños de correos cifrados son visibles para el servidor
Confianza en la entrega web	El navegador descarga JavaScript de nuestros servidores en cada visita (véase la Sección 17 para mitigaciones)
Criptoanálisis mediante coacción física	Ningún sistema criptográfico protege contra la coerción física

1.3.3 2.3 Principio de diseño

Aionda Mail sigue el modelo de **"servidor honesto"**: el sistema está diseñado de manera que incluso un servidor completamente comprometido — o un operador malicioso — no pueda descifrar los datos del usuario. La seguridad no depende de confiar en nosotros. Depende de las matemáticas.

1.4 3. Visión general de la arquitectura





1.4.1 3.1 Resumen de componentes

Componente	Propósito	Ubicación
OPAQUE (RFC 9807)	Autenticación de contraseña Zero-Knowledge	Cliente + Servidor
Shamir Secret Sharing	Protección de clave maestra del vault (umbral 2-de-3)	Solo cliente
Hybrid KEM (X25519 + ML-KEM-1024)	Encapsulación de claves post-cuántica para correos	Cliente + Servidor
AES-256-GCM	Cifrado simétrico autenticado	Cliente + Servidor
HKDF-SHA256	Derivación de claves a partir de secretos compartidos híbridos	Cliente + Servidor
BIP39	Codificación de clave de recuperación (mnemónico de 24 palabras)	Solo cliente
WebAuthn PRF	Desbloqueo del vault basado en passkey	Solo cliente
Bucket Padding	Protección contra canales laterales	Cliente + Servidor

---

1.5 4. Autenticación Zero-Knowledge (OPAQUE)

1.5.1 4.1 ¿Por qué no el hashing de contraseñas?

Los servicios tradicionales almacenan hashes de contraseñas (bcrypt, Argon2). Aunque es mejor que el texto plano, este enfoque tiene debilidades fundamentales:

- El servidor ve la contraseña durante el inicio de sesión (aunque solo brevemente en RAM)
- Los hashes de contraseñas pueden ser atacados por fuerza bruta offline si la base de datos es robada
- El servidor podría ser modificado para registrar contraseñas

OPAQUE elimina los tres problemas. La contraseña nunca sale del navegador — ni como texto plano, ni como hash, en ninguna forma.

1.5.2 4.2 Cómo funciona OPAQUE

OPAQUE (RFC 9807) es un protocolo de intercambio de claves autenticado por contraseña asimétrico (aPAKE). Utiliza un mecanismo criptográfico de desafío-respuesta donde el servidor puede verificar que el usuario conoce la contraseña correcta sin jamás conocer cuál es esa contraseña.

Registro (una sola vez):

```
Browser                                    Server
--------                                   ------
1. Generate OPAQUE registration     →
   request from password
                                           2. Create OPAQUE registration
                                              response (blind evaluation)
                                    ←
3. Finalize registration            →      4. Store encrypted OPAQUE
   (export_key available)                     record (AES-256-GCM at rest)
```

Inicio de sesión (cada sesión):

```
Browser                                    Server
--------                                   ------
1. Generate login request (KE1)     →      2. Look up encrypted OPAQUE record
   from password                              Decrypt with server key
                                              Generate response (KE2)
                                    ←
3. Verify server response
   Compute session key + KE3        →      4. Verify KE3 (cryptographic proof)
   Password verified locally!                 If valid: session authenticated
                                              If invalid: reject (max 3 attempts)
```

Propiedades clave:

- La contraseña se verifica **del lado del cliente** en el paso 3 — el servidor nunca la ve
- El servidor almacena un **registro OPAQUE**, que no es un hash de contraseña y no puede ser atacado por fuerza bruta offline
- Los registros OPAQUE están adicionalmente **cifrados en reposo** con AES-256-GCM usando una clave del servidor
- **Protección contra enumeración de usuarios**: Las cuentas inexistentes reciben respuestas falsas determinísticas con temporización idéntica
- **Limitación de velocidad**: Máximo 3 intentos de autenticación por sesión, tiempo de espera de sesión de 120 segundos

1.5.3 4.3 Implementación

- **Biblioteca:** `@serenity-kit/opaque` (basada en WASM, de grado producción)
- **Componente del servidor:** Microservicio dedicado para operaciones criptográficas OPAQUE
- **Formato Base64:** base64url (seguro para URLs, sin relleno) para compatibilidad del protocolo

- **Registro de auditoría:** Todos los eventos de autenticación registrados con marcas de tiempo y direcciones IP

1.5.4 4.4 Migración de SRP

Las cuentas heredadas que utilizan SRP-6a se migran automáticamente a OPAQUE en el siguiente inicio de sesión. Después de la migración, el verificador SRP se elimina permanentemente. La migración es unidireccional — las cuentas no pueden revertir a SRP.

1.6 5. Clave maestra del Vault y Shamir Secret Sharing

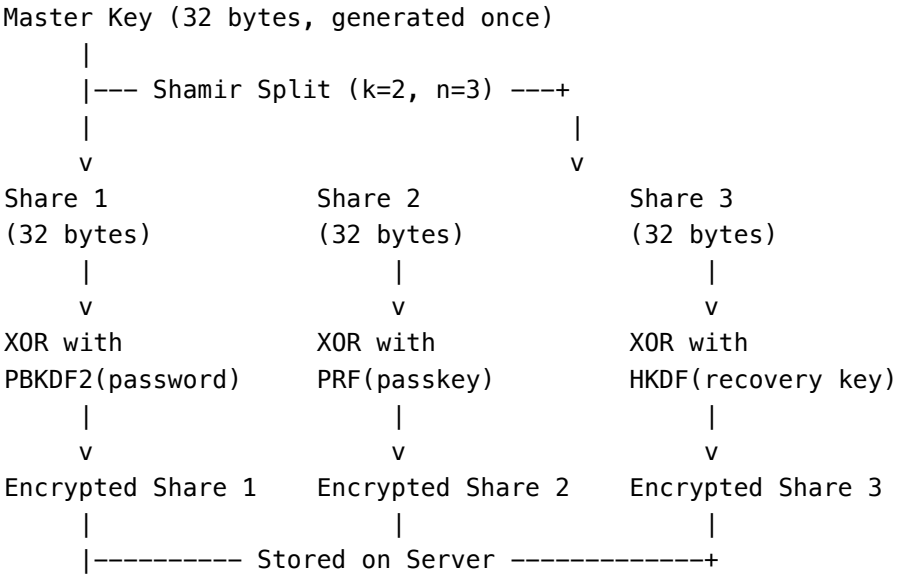
1.6.1 5.1 Generación de la clave maestra

Cuando un usuario activa el buzón cifrado, se genera una **clave maestra de 256 bits (32 bytes)** utilizando el generador de números aleatorios criptográficamente seguro del navegador (`crypto.getRandomValues`).

Esta clave maestra es la raíz de todo el cifrado. Nunca sale del navegador en texto plano. Nunca se almacena en ningún lugar — ni en el navegador, ni en el servidor, en ninguna forma.

1.6.2 5.2 Shamir Secret Sharing (2-de-3)

La clave maestra se divide en tres partes utilizando el esquema de **Shamir's Secret Sharing** sobre el campo de Galois GF(2^8) con el polinomio irreducible de AES (x^8 + x^4 + x^3 + x + 1).



Propiedad de umbral: Cualesquiera 2 de las 3 partes son suficientes para reconstruir la clave maestra mediante interpolación de Lagrange. El servidor almacena solo las partes cifradas — y no puede descifrar ninguna de ellas.

1.6.3 5.3 Protección de las partes

Cada parte se combina mediante XOR con una clave derivada de un factor de autenticación diferente:

Parte	Protegida por	Derivación de clave
Parte 1	Contraseña	PBKDF2-SHA256, 600.000 iteraciones, salt aleatorio de 32 bytes
Parte 2	Passkey (FIDO2)	Extensión WebAuthn PRF, vinculada al hardware
Parte 3	Clave de recuperación	HKDF-SHA3-256 con salt vinculado a la cuenta

Escenarios de reconstrucción:

- **Inicio de sesión normal:** Contraseña (Parte 1) + Passkey (Parte 2) Clave maestra
- **Passkey perdido:** Contraseña (Parte 1) + Clave de recuperación (Parte 3) Clave maestra
- **Cambio de contraseña:** Passkey (Parte 2) + Clave de recuperación (Parte 3) Clave maestra

1.6.4 5.4 Protección del almacenamiento de sesión

Incluso dentro de una sesión del navegador, la clave maestra nunca se almacena en texto plano:

1. Se genera una clave AES-256 efímera
2. La clave maestra se cifra con esta clave efímera
3. Solo el blob cifrado se coloca en `sessionStorage`
4. La clave efímera existe solo en la memoria de JavaScript (recolectada por el garbage collector al cerrar la pestaña)

1.7 6. KEM híbrido post-cuántico

1.7.1 6.1 ¿Por qué post-cuántico?

Las computadoras cuánticas ejecutando el algoritmo de Shor podrían romper la criptografía de clave pública clásica (RSA, ECDH, X25519) en tiempo polinómico. Aunque las computadoras cuánticas a gran escala aún no existen, la amenaza de los ataques **"cosechar ahora, descifrar después"** es real: los adversarios podrían almacenar datos cifrados hoy y descifrarlos cuando las computadoras cuánticas estén disponibles.

1.7.2 6.2 Enfoque híbrido

Aionda Mail utiliza un **mecanismo de encapsulación de claves híbrido** que combina:

- **X25519** (Curve25519 ECDH) — seguridad clásica probada, nivel de seguridad de 128 bits
- **ML-KEM-1024** (NIST FIPS 203, anteriormente Kyber-1024) — seguridad post-cuántica, NIST Security Level 5

El enfoque híbrido proporciona **defensa en profundidad**: la clave combinada es segura mientras **al menos uno** de los dos algoritmos permanezca sin ser roto.

1.7.3 6.3 Proceso de encapsulación

```
Sender (encrypting an email):

1. Generate ephemeral X25519 keypair
2. X25519 key agreement with recipient's public key
   → x25519SharedSecret (32 bytes)

3. ML-KEM-1024 encapsulation with recipient's public key
   → mlKemSharedSecret (32 bytes) + mlKemCiphertext (1568 bytes)

4. Combine secrets:
   combinedSecret = x25519SharedSecret || mlKemSharedSecret  (64 bytes)

5. Derive final key:
   sharedSecret = HKDF-SHA256(
       ikm = combinedSecret,
       salt = nil,
       info = "trashmail-hybrid-kem-v1",
       length = 32
   )

6. Use sharedSecret to wrap the email's ephemeral AES-256 key
```

1.7.4 6.4 Proceso de desencapsulación

```
Recipient (decrypting an email):

1. X25519 key agreement:
   x25519Shared = X25519(recipientPrivateKey, ephemeralPublicKey)

2. ML-KEM-1024 decapsulation:
   mlKemShared = ML-KEM-1024.Decapsulate(mlKemCiphertext, recipientPrivateKey)

3. Combine and derive (identical to sender):
   sharedSecret = HKDF-SHA256(x25519Shared || mlKemShared, "trashmail-hybrid-kem-v1")

4. Unwrap email's ephemeral AES-256 key using sharedSecret
5. Decrypt email content with ephemeral key
```

1.7.5 6.5 Tamaños de claves

Parámetro	Tamaño	Estándar
X25519 public key	32 bytes	RFC 7748
X25519 private key	32 bytes	RFC 7748
ML-KEM-1024 public key	1.568 bytes	NIST FIPS 203
ML-KEM-1024 private key	3.168 bytes	NIST FIPS 203
ML-KEM-1024 ciphertext	1.568 bytes	NIST FIPS 203
Combined shared secret	32 bytes	HKDF-SHA256 output

1.7.6 6.6 Biblioteca

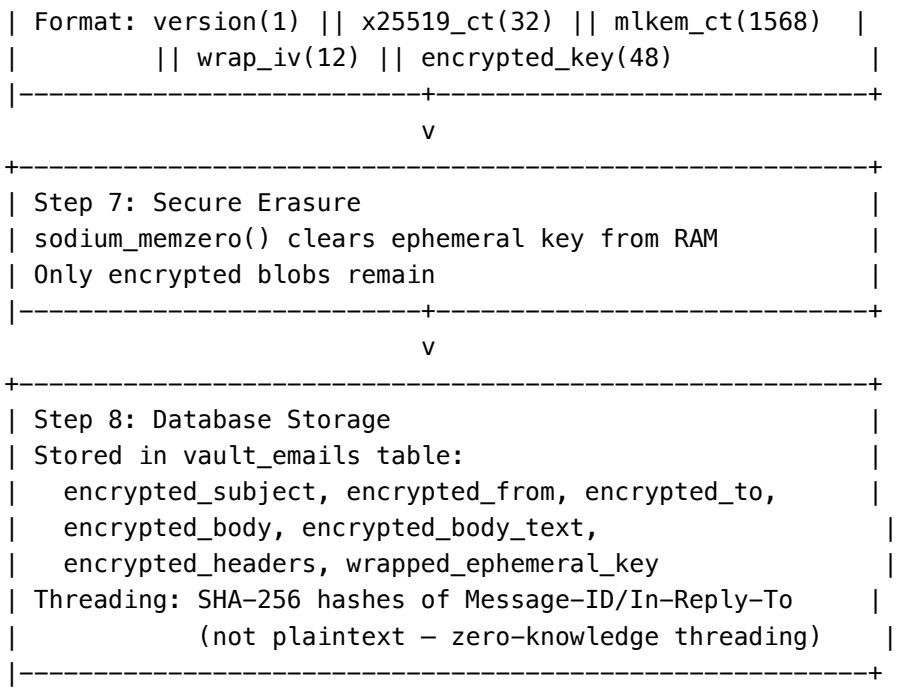
- **ML-KEM-1024:** `@noble/post-quantum` (auditada, implementación JavaScript pura)
- **X25519:** WebCrypto API (`crypto.subtle.deriveBits`)
- **HKDF:** `@noble/hashes` (conforme con RFC 5869)

1.8 7. Pipeline de cifrado de correos electrónicos

1.8.1 7.1 Correo entrante (SMTP Almacenamiento cifrado)

Cuando un correo llega al servidor SMTP de Aionda Mail:

```
+-----------------------------------------------------------+
| Step 1: SMTP Reception                                    |
| Mail server receives email → passes to SMTP handler       |
|---------------------------+-------------------------------+
                            v
+-----------------------------------------------------------+
| Step 2: Compression (optional)                            |
| gzip level 6 for text content                             |
| Skipped for: JPEG, PNG, MP4, ZIP, PDF, DOCX, etc.         |
|---------------------------+-------------------------------+
                            v
+-----------------------------------------------------------+
| Step 3: Bucket Padding                                    |
| Pad to fixed-size bucket (256B, 512B, 1KB, ..., 16MB)     |
| Format: 0xDEAD || length(4B) || data || random padding    |
|---------------------------+-------------------------------+
                            v
+-----------------------------------------------------------+
| Step 4: Generate Ephemeral Key                            |
| 32-byte random AES-256 key (unique per email)             |
| Generated via cryptographically secure RNG                |
|---------------------------+-------------------------------+
                            v
+-----------------------------------------------------------+
| Step 5: AES-256-GCM Encryption                            |
| Each field encrypted separately:                          |
|   • Subject     • From     • To                           |
|   • Body (HTML) • Body (plaintext) • Headers              |
| Nonce: 12 bytes random per field                          |
| Tag: 16 bytes authentication                              |
| Format: nonce(12) || tag(16) || ciphertext                |
|---------------------------+-------------------------------+
                            v
+-----------------------------------------------------------+
| Step 6: Hybrid KEM Key Wrapping                           |
| Ephemeral key wrapped with recipient's public keys:       |
|   X25519 + ML-KEM-1024 → shared secret                    |
|   AES-256-GCM(ephemeral_key, shared_secret)               |
```



1.8.2 7.2 Lectura de correo (descifrado en el navegador)

El proceso inverso ocurre completamente en el navegador:

1. Obtener el correo cifrado del servidor a través de la API cifrada
2. Analizar la clave efímera envuelta (extraer texto cifrado X25519 + texto cifrado ML-KEM)
3. **Desencapsulación Hybrid KEM** usando las claves privadas del vault secreto compartido
4. Desenvolver la clave AES-256 efímera
5. **Descifrado AES-256-GCM** de cada campo (asunto, remitente, destinatario, cuerpo, cabeceras)
6. Reordenar bytes: formato del servidor (nonce || tag || ct) formato WebCrypto (nonce || ct || tag)
7. Eliminar Bucket Padding (detectar bytes mágicos 0xDEAD)
8. Descomprimir si es gzip (detectar bytes mágicos 0x1F8B)
9. Decodificar UTF-8 a texto plano

1.8.3 7.3 Envío de correo

Al redactar y enviar un correo:

1. El cliente cifra el contenido del correo con un protocolo de desafío-respuesta
2. El servidor recibe la carga cifrada, descifra efímeramente (solo en memoria), envía vía SMTP
3. El servidor devuelve las cabeceras MIME generadas al cliente (cifradas)
4. El cliente cifra una copia con la clave maestra del vault y la almacena en la carpeta Enviados
5. El texto plano efímero del lado del servidor se descarta inmediatamente — nunca se escribe en disco

1.8.4 7.4 Adjuntos

Cada adjunto se cifra de forma independiente:

- Clave AES-256 efímera separada por adjunto
- Envoltura de clave Hybrid KEM separada por adjunto
- Nombre de archivo y tipo MIME cifrados por separado
- Sin compresión para formatos ya comprimidos (JPEG, ZIP, PDF, etc.)

1.8.5 7.5 Agrupación de hilos de correo (Zero-Knowledge)

La agrupación de hilos de correo (agrupar correos relacionados) utiliza únicamente **hashes SHA-256** de las cabeceras Message-ID e In-Reply-To. El servidor nunca ve las cadenas reales de Message-ID — puede agrupar correos por igualdad de hash sin conocer el contenido.

1.9 8. Capa de transporte API cifrada

1.9.1 8.1 Problema

Incluso con HTTPS, ciertos intermediarios pueden inspeccionar el tráfico:

- **CloudFlare** (CDN/protección DDoS) termina TLS y puede ver solicitudes en texto plano
- **Proxies corporativos** pueden realizar inspección TLS
- **Parámetros de API** (como `?cmd=read_email&id=123`) filtran metadatos

1.9.2 8.2 Solución: API cifrada de extremo a extremo

Toda la comunicación API está adicionalmente cifrada de extremo a extremo entre el navegador y el servidor de aplicación, dentro del túnel HTTPS:

```
Browser                                    Server
--------                                   ------

Phase 1: Key Exchange (once per session)
----------------------------------------
GET /get_encryption_keys          →        Return 20 pre-generated
                                           Hybrid KEM keypairs
                                  ←        {uuid, x25519_pub, mlkem_pub}

Phase 2: Encrypted Request (every API call)
----------------------------------------
1. Pick random keypair from cache
2. Hybrid KEM encapsulate → shared secret
3. gzip compress request payload
4. Bucket-pad compressed data
5. AES-256-GCM encrypt with shared secret
6. Generate ephemeral response keypair
7. POST /e {                      →        8.  Validate key ownership
     encrypted_payload,                    9.  Hybrid KEM decapsulate
     key_uuid,                             10. AES-256-GCM decrypt
     x25519_ciphertext,                    11. Decompress
     mlkem_ciphertext,                     12. Route to API controller
```

```
    response_x25519_pub,
    response_mlkem_pub
  }
                                    ←
16. Hybrid KEM decapsulate response
17. AES-256-GCM decrypt
18. Decompress → plaintext response
```

```
13. Execute business logic
14. Encrypt response with
    client's response keys
15. Return encrypted response
```

1.9.3 8.3 Propiedades clave

- **Uso único:** Cada par de claves API se utiliza exactamente una vez, luego se invalida permanentemente
- **Perfect Forward Secrecy:** Comprometer la clave de una solicitud no afecta a ninguna otra solicitud
- **Vinculada a la sesión:** Las claves son reclamadas por una sesión específica y no pueden ser reutilizadas por otra
- **Pool de claves:** El servidor mantiene aproximadamente 100.000 pares de claves pre-generados
- **Re-obtención automática:** El cliente solicita automáticamente nuevas claves cuando la caché baja de 10
- **TTL de claves:** Las claves reclamadas expiran después de 24 horas
- **Bidireccional:** Tanto la solicitud COMO la respuesta están cifradas — el servidor nunca devuelve texto plano

1.9.4 8.4 Lo que CloudFlare ve

Con esta arquitectura, CloudFlare (o cualquier proxy que termine TLS) solo ve:

- `POST /e` — un único endpoint opaco
- Un blob binario de datos cifrados
- Sin nombres de comandos API, sin parámetros, sin IDs de correo, sin datos de usuario

---

1.10 9. Criptografía de carpetas compartidas

1.10.1 9.1 Modelo de compartición

Los usuarios pueden compartir carpetas cifradas con otros usuarios de Aionda Mail. El mecanismo de compartición utiliza el Hybrid KEM para cifrar una clave específica de la carpeta para cada destinatario.

1.10.2 9.2 Flujo de compartición

```
Folder Owner                                        Recipient
------------                                        ---------

1. Derive folder key from master key:
   folderKey = HKDF-SHA256(masterKey, folderUuid)

2. Fetch recipient's public keys:
   recipient.x25519_pub (32 bytes)
   recipient.mlkem_pub (1568 bytes)
```

```
3. Hybrid KEM encapsulate:
   hybridEncapsulate(recipient.x25519_pub, recipient.mlkem_pub)
   → {x25519Ciphertext, mlKemCiphertext, sharedSecret}

4. Encrypt folder key:
   wrappedKey = AES-256-GCM(folderKey, sharedSecret, nonce)

5. Store on server:
   {x25519_ct, mlkem_ct, nonce, wrappedKey, permissions}
```

```
6. Fetch sharing record from server

7. Hybrid KEM decapsulate:
   hybridDecapsulate(x25519_ct, mlkem_ct,
     own_x25519_priv, own_mlkem_priv)
   → sharedSecret

8. Decrypt folder key:
   folderKey = AES-GCM-decrypt(
     wrappedKey, sharedSecret, nonce)

9. Decrypt emails in folder using folderKe
```

1.10.3 9.3 Modelo de permisos

Permiso	Capacidad
`readonly`	Leer correos de la carpeta (solo descifrar)
`einliefern`	Entregar nuevos correos en la carpeta
`bearbeiten`	Editar el contenido de la carpeta
`antworten`	Responder correos dentro de la carpeta
`vollzugriff`	Acceso completo incluyendo transferencia de propiedad

1.10.4 9.4 Copia entre vaults (Re-Wrapping)

Cuando un destinatario copia un correo de una carpeta compartida a su propio vault, la clave del correo debe ser **re-envuelta** para su propio par de claves Hybrid KEM. Esto se realiza completamente del lado del cliente:

1. Descifrar la clave de la carpeta compartida usando las claves privadas del destinatario
2. Descifrar la clave efímera del correo usando la clave de la carpeta
3. Re-encapsular la clave efímera con las claves públicas propias del destinatario
4. Almacenar la copia re-envuelta en el vault del destinatario

El servidor facilita la transferencia pero nunca ve ningún material de clave en texto plano.

1.11 10. Protecciones contra canales laterales

1.11.1 10.1 Bucket Padding

Problema: Los tamaños de los correos cifrados pueden filtrar información. Un atacante observando las longitudes del texto cifrado podría inferir contenido (p. ej., un correo de 50 bytes probablemente es “OK, gracias” mientras que un correo de 500 KB contiene adjuntos).

Solución: Todos los datos se rellenan a “buckets” de tamaño fijo antes del cifrado:

```
Bucket sizes: 256B, 512B, 1KB, 2KB, 4KB, 8KB, 16KB, 32KB,
              64KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB

Format: [0xDEAD magic][4-byte length][actual data][random padding to bucket boundary]
```

Ejemplo: Un correo de 523 bytes se rellena a 1.024 bytes. Un observador solo ve “correo de 1 KB” — no el tamaño real de 523 bytes.

1.11.2 10.2 Compresión antes del cifrado

Los datos se comprimen con gzip (nivel 6) **antes** del cifrado. Este es el único orden correcto:

- La compresión después del cifrado fallaría (los datos cifrados tienen entropía máxima)
- El Bucket Padding después de la compresión previene ataques tipo CRIME/BREACH que explotan las ratios de compresión

1.11.3 10.3 Privacidad en la agrupación de hilos

Los hilos de correo utilizan hashes SHA-256 de las cabeceras Message-ID en lugar de identificadores en texto plano. El servidor puede agrupar correos relacionados por igualdad de hash sin conocer los identificadores reales de los mensajes.

1.12 11. Gestión y ciclo de vida de claves

1.12.1 11.1 Jerarquía de claves

```
Vault Master Key (32 bytes, generated once per account)
   |
   |--- Vault Keypair (Hybrid KEM)
   |    |-- X25519 public key (32 bytes) — stored plaintext on server
   |    |-- X25519 private key (32 bytes) — encrypted with master key
   |    |-- ML-KEM-1024 public key (1568 bytes) — stored plaintext on server
   |    |-- ML-KEM-1024 private key (3168 bytes) — encrypted with master key
   |
   |--- Per-Email Ephemeral Keys (32 bytes each)
   |    |-- Wrapped with recipient's Hybrid KEM public keys
   |
   |--- Per-Attachment Ephemeral Keys (32 bytes each)
   |    |-- Wrapped independently per attachment
   |
   |--- Folder Keys (derived via HKDF per folder)
   |    |-- Shared via Hybrid KEM encapsulation per recipient
   |
   |--- Signature Encryption Key (derived from master key)
        |-- Encrypts email signature templates
```

1.12.2 11.2 Almacenamiento de claves

Clave	Ubicación de almacenamiento	Protección
Clave maestra	En ningún lugar (reconstruida bajo demanda a partir de las partes de Shamir)	Shamir 2-de-3
Claves privadas del vault	Servidor (cifradas)	AES-256-GCM con clave maestra
Claves públicas del vault	Servidor (texto plano)	No sensibles — públicas por definición
Claves efímeras de correo	Servidor (envueltas)	Encapsulación Hybrid KEM
Registros OPAQUE	Servidor (cifrados en reposo)	AES-256-GCM con clave del servidor
Partes de Shamir cifradas	Servidor	XOR con claves derivadas de contraseña/passkey/recuperación
Claves de transporte API	Servidor (pool pre-generado)	Uso único, TTL de 24h

1.12.3 11.3 Huellas digitales de claves

Cada par de claves del vault tiene una huella digital SHA-256 almacenada en el servidor. Esto permite:

- Registro de auditoría de rotaciones de claves
- Detección de cambios de claves no autorizados
- Verificación de integridad de claves del lado del cliente

1.13 12. Mecanismo de recuperación

1.13.1 12.1 Clave de recuperación (mnemónico BIP39)

Durante la configuración del vault, se presenta al usuario una **frase de recuperación de 24 palabras** generada a partir de 256 bits de entropía, codificada utilizando el estándar BIP39:

```
Example: apple river mountain sunset golden bridge falcon ocean
         crystal thunder meadow silver dolphin forest marble castle
         velvet compass harbor window ancient pepper rocket shield
```

1.13.2 12.2 Derivación de la clave de recuperación

```
1. Generate: 256 bits random entropy
2. Encode:   BIP39 mnemonic (24 words, 11 bits per word)
3. Derive:   verificationKey = HKDF-SHA3-256(
                 entropy,
                 salt = accountId,
```

```
                info = "trashmail-recovery-verify"
            )
4. Hash:    verificationHash = SHA3-256(verificationKey)
5. Store:   Server stores ONLY verificationHash (32 bytes)
```

1.13.3 12.3 Lo que el servidor almacena

El servidor almacena **solo el hash SHA3-256** de una clave de verificación derivada. No almacena:

- Las palabras de recuperación
- La entropía
- La clave de verificación en sí

1.13.4 12.4 Flujo de recuperación

1. El usuario introduce la frase de recuperación de 24 palabras
2. El cliente deriva entropía HKDF-SHA3-256 SHA3-256 hash de verificación
3. El cliente envía el hash de verificación al servidor (nunca la clave en texto plano)
4. El servidor compara con el hash almacenado
5. Si coincide: Todos los métodos 2FA se desactivan, el usuario configura autenticación nueva
6. La clave de recuperación se revoca después de un solo uso

1.13.5 12.5 Limitación de velocidad

- Máximo 3 intentos de verificación por hora
- Bloqueo de 60 minutos después de exceder el límite
- Uso único: la clave de recuperación se revoca permanentemente después del uso exitoso

1.13.6 12.6 Sin recuperación de contraseña

No hay restablecimiento de contraseña por correo electrónico. Si un usuario pierde su contraseña Y todos los demás factores de autenticación (passkey + clave de recuperación), sus datos son permanentemente inaccesibles. Esta es la prueba fundamental de que Zero-Knowledge funciona — si pudiéramos recuperar sus datos, también podríamos leerlos.

1.14 13. Autenticación sin contraseña (Passkeys)

1.14.1 13.1 Extensión WebAuthn PRF

Aionda Mail soporta passkeys FIDO2 (llaves de seguridad hardware, autenticadores biométricos) para inicio de sesión sin contraseña y desbloqueo del vault.

La **extensión WebAuthn PRF (Pseudo-Random Function)** proporciona una salida determinística de 32 bytes vinculada al passkey y credencial específicos. Esta salida se utiliza para proteger la Parte 2 de Shamir.

1.14.2 13.2 Cómo funciona

```
1. Registration:
   - User creates passkey via navigator.credentials.create()
```

```
   - PRF extension generates hardware-bound output
   - Output XOR'd with Shamir Share 2 → encrypted share stored on server

2. Authentication:
   - User authenticates with passkey (biometric/PIN)
   - PRF extension reproduces same 32-byte output
   - Output XOR'd with encrypted share → Shamir Share 2 recovered
   - Combined with Share 1 (password) → Master Key reconstructed

3. Vault Unlock (passwordless):
   - If both passkey (Share 2) and password (Share 1) available → immediate unlock
   - Password verified via OPAQUE (separate from passkey auth)
```

1.14.3 13.3 Múltiples Passkeys

Los usuarios pueden registrar múltiples passkeys (p. ej., MacBook Touch ID, iPhone Face ID, YubiKey). Cada passkey protege de forma independiente su propia copia de la Parte 2. Cualquier passkey individual combinado con la contraseña es suficiente para desbloquear el vault.

1.15 14. Guardian: Protección MITM y firma de respuestas

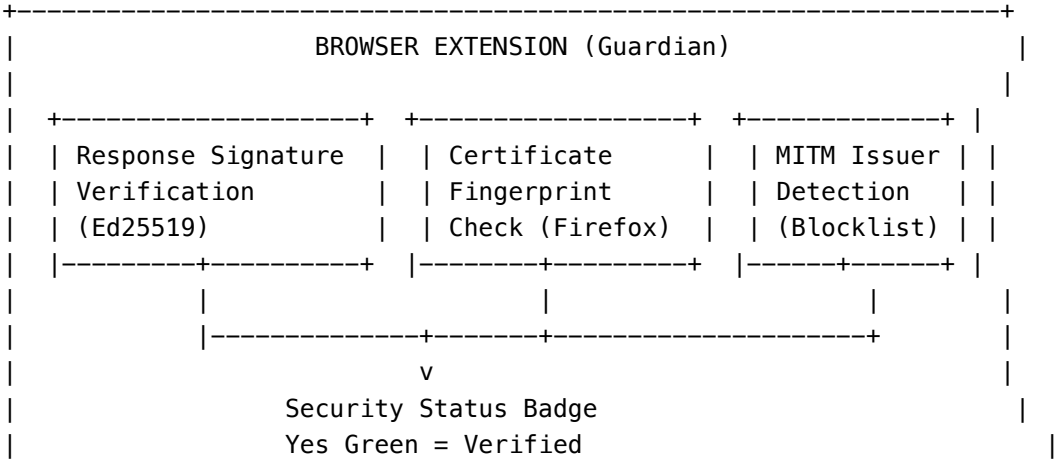
1.15.1 14.1 El problema con las aplicaciones web

Toda aplicación web tiene un problema inherente de confianza: el navegador descarga JavaScript del servidor en cada visita. Un atacante de intermediario (MITM) — ya sea un CDN comprometido, un proxy corporativo o un ISP malintencionado — podría teóricamente inyectar código modificado que exfiltre claves de cifrado.

Aionda Mail aborda esto con el **módulo Guardian**, un componente de extensión del navegador (disponible para Chrome y Firefox) que verifica de forma independiente la integridad del servidor.

1.15.2 14.2 Visión general de la arquitectura

El módulo Guardian opera dentro del Service Worker de la extensión del navegador — completamente independiente del JavaScript de la aplicación web. Realiza tres tipos de verificación:



```
|                  No Red = MITM Detected                              |
|                  ! Red = Missing Signatures                         |
|-----------------------------------------------------------------+
```

1.15.3 14.3 Verificación de firma de respuesta (Ed25519)

Cada respuesta API del servidor de Aionda Mail está firmada criptográficamente usando **Ed25519** (Edwards-curve Digital Signature Algorithm).

Proceso de firma (lado del servidor):

```
1. Server generates API response body (JSON)
2. Construct signing input: responseBody + "|" + unixTimestamp
3. Sign with Ed25519 private key → 64-byte signature
4. Attach HTTP headers:
   X-Aionda-Signature: <base64(signature)>
   X-Aionda-Timestamp: <unix_timestamp>
   X-Aionda-Key-Id: <key_identifier>
```

Proceso de verificación (extensión del navegador):

```
1. Extract signature, timestamp, and key ID from HTTP headers
2. Look up Ed25519 public key by key ID (bundled in extension)
3. Verify key has not expired (valid_from / valid_until)
4. Check timestamp freshness: |now − timestamp| ≤ 300 seconds
5. Reconstruct signing input: responseBody + "|" + timestamp
6. crypto.subtle.verify("Ed25519", publicKey, signature, data)
7. If invalid → MITM alert, red badge
```

Propiedades clave:

- **Protección contra repetición:** Ventana de tiempo de 5 minutos previene la reproducción de respuestas antiguas
- **Detección de manipulación:** Cualquier modificación al cuerpo de la respuesta invalida la firma
- **Aislamiento de claves:** Las claves públicas están empaquetadas dentro de la extensión (no se descargan del servidor)
- **Separación de entornos:** Las claves de desarrollo (`dev-2026-01`) no pueden usarse en URLs de producción y viceversa

1.15.4 14.4 Gestión de claves públicas Ed25519

Las claves públicas se distribuyen con la extensión del navegador en `public_key.json`:

```json
{
    "keys": {
        "prod-2026-01": {
            "algorithm": "Ed25519",
            "public_key": "<base64 SPKI DER>",
            "valid_from": "2026-01-13T00:00:00Z",
            "valid_until": "2027-01-13T00:00:00Z"
        }
    }
}
```

- **Formato de clave:** SPKI DER (Subject Public Key Info, Distinguished Encoding Rules)
- **Tamaño de clave:** 32 bytes (clave pública Ed25519 de 256 bits)
- **Tamaño de firma:** 64 bytes (fijo)
- **Rotación:** Se agregan nuevas claves antes de que expiren las anteriores; las actualizaciones de la extensión entregan las nuevas claves
- **Sin confianza en el servidor:** Las claves están incrustadas en el binario de la extensión, no se obtienen del servidor

1.15.5 14.5 Verificación de certificados TLS (Firefox)

En Firefox, el módulo Guardian realiza una verificación adicional de certificados TLS utilizando la API `browser.webRequest.getSecurityInfo()` (no disponible en Chrome debido a las limitaciones de Manifest V3).

Flujo de verificación:

```
1. Browser extension intercepts HTTPS response
2. Extract TLS certificate chain from browser's security info:
   - Leaf certificate fingerprint (SHA-256)
   - Issuer Distinguished Name (O=, CN=)
   - Subject (CN=)

3. Check against known MITM issuers (hardcoded blocklist):
   ZScaler, Netskope, Fortinet, Palo Alto, Blue Coat,
   Check Point, Barracuda, Sophos, WatchGuard, Cisco Umbrella
   → If match: MITM detected, show warning

4. Check against trusted issuers:
   Google Trust Services, Cloudflare, Let's Encrypt,
   DigiCert, Sectigo
   → If match AND subject matches expected domain: OK

5. If unknown issuer: Fetch server's own certificate fingerprint
   - Server connects to itself via external routing (prevents spoofing)
   - Response is Ed25519 signed (prevents MITM from lying about cert)
   - Compare issuer organization with browser's certificate issuer
   → If mismatch: MITM suspected, show warning
```

¿Por qué validación basada en emisor en lugar de pinning? CloudFlare (usado como CDN) rota los certificados hoja entre los servidores de borde. El pinning de certificados tradicional (que coincide con huellas digitales exactas) causaría falsos positivos. La validación basada en emisor es más robusta: la CA emisora es estable incluso cuando los certificados hoja cambian.

1.15.6 14.6 Obtención de certificado propio (Anti-Spoofing)

El endpoint de certificado del servidor utiliza una técnica ingeniosa anti-spoofing:

```
Server connects to cert.trashmail.com (or cert-subdomain.domain)
with SNI = mail.aionda.com

→ Forces external routing through CloudFlare
→ Receives the actual certificate that users see
```

```
→ Prevents localhost spoofing
→ Response signed with Ed25519 to prevent tampering
```

El servidor esencialmente pregunta “¿qué certificado ve el mundo exterior para mi dominio?” — y firma la respuesta para que la extensión pueda confiar en ella.

1.15.7 14.7 Indicadores de estado de seguridad

La extensión muestra una insignia en la barra de herramientas del navegador:

Insignia	Color	Significado
Yes	Verde	Todas las respuestas verificadas — firmas válidas
!	Naranja	Usando clave de firma obsoleta (rotación pendiente)
No	Rojo	MITM detectado — verificación de firma falló
!	Rojo	Firmas faltantes — respuestas no firmadas
Shield	Azul	Modo protegido — aún no se ha realizado verificación

1.15.8 14.8 Cobertura de amenazas

Ataque	Método de detección	Navegador
Proxy MITM corporativo (ZScaler, Fortinet)	Lista de bloqueo de emisores de certificados	Firefox
Respuestas API modificadas	Verificación de firma Ed25519	Chrome + Firefox
Ataques de repetición	Ventana de tiempo de 5 minutos	Chrome + Firefox
Compromiso de CDN (CloudFlare)	Discrepancia de firma de respuesta	Chrome + Firefox
Sustitución de certificado	Comparación de emisor + auto-verificación del servidor	Firefox
Confusión de clave dev/prod	IDs de clave vinculados al entorno	Chrome + Firefox

1.15.9 14.9 Limitaciones

- **Chrome Manifest V3:** No puede inspeccionar certificados TLS — solo está disponible la verificación de firma de respuesta
- **Extensión requerida:** Los usuarios sin la extensión no se benefician de las protecciones de Guardian
- **Ed25519 no es post-cuántico:** La verificación de firmas utiliza criptografía clásica. Una computadora cuántica suficientemente potente podría teóricamente falsificar firmas Ed25519.

1.16 15. Archivo empresarial de correo electrónico (Blockchain)

1.16.1 15.1 Visión general

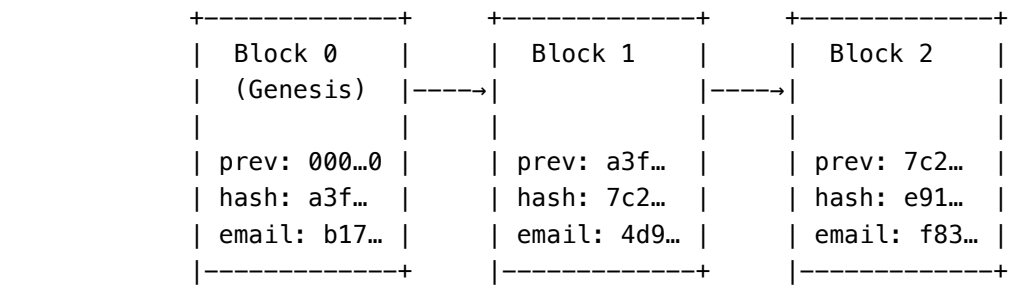
El plan Enterprise de Aionda Mail incluye un **archivo de correo electrónico conforme a GoBD** asegurado por una cadena de hashes criptográfica (blockchain). Cada correo archivado se convierte en un bloque inmutable en una cadena por empresa. Cualquier manipulación — modificación, eliminación o inserción de bloques — es criptográficamente detectable.

El archivo combina dos capas de seguridad independientes:

1. **Cadena de hashes (SHA3-256):** Garantiza integridad e inmutabilidad — demuestra que ningún correo fue alterado o eliminado después del archivado
2. **Cifrado Hybrid KEM (CAK):** Garantiza confidencialidad — el servidor no puede leer el contenido de los correos archivados

1.16.2 15.2 Arquitectura de la cadena de hashes

Cada correo archivado se convierte en un bloque en una cadena secuencial a prueba de manipulación:



Cálculo del hash de bloque:

```
block_hash = SHA3-256(
    prev_block_hash  || "|" ||
    timestamp        || "|" ||
    email_hash       || "|" ||
    direction        || "|" ||
    sender_domain    || "|" ||
    recipient_domain
)
```

Propiedades:

- **Algoritmo de hash:** SHA3-256 (NIST FIPS 202)
- **Bloque génesis:** `prev_block_hash` = 64 ceros, `block_number` = 0
- **Numeración secuencial:** Aplicada por la base de datos `UNIQUE KEY (company_uuid, block_number)`
- **Una cadena por empresa:** Aislamiento completo entre empresas
- **Hash de correo:** `SHA3-256(sender || recipient || timestamp || size)` — prueba de integridad de los datos del correo original

1.16.3 15.3 Detección de manipulación

El algoritmo de verificación de la cadena detecta cualquier forma de manipulación:

```
For each block (ordered by block_number ASC):
```

```
  1. Verify link:    block.prev_block_hash == expected_prev_hash
  2. Recalculate:    expected = SHA3-256(prev_hash | timestamp | email_hash | ...)
  3. Verify content: block.block_hash == expected
  4. Advance:        expected_prev_hash = block.block_hash

If ANY check fails → chain is broken at block N
```

Intento de manipulación	Detección
Modificar contenido del correo	`email_hash` cambia falla el recálculo de `block_hash`
Modificar metadatos (remitente, dominio, marca de tiempo)	Incluido en la entrada del hash discrepancia de `block_hash`
Eliminar un bloque	El `prev_block_hash` del siguiente bloque queda huérfano
Insertar un bloque	Rompe el `block_number` secuencial + cadena `prev_block_hash`
Reordenar bloques	La restricción `UNIQUE KEY` + verificación secuencial lo previene
Reemplazar toda la cadena	El hash del bloque génesis diferiría de cualquier copia de seguridad externa

El **resultado de verificación** reporta el número de bloque exacto donde se detectó la manipulación, con los valores de hash esperados vs. reales para análisis forense.

1.16.4 15.4 Company Archive Key (CAK) — Cifrado Zero-Knowledge

Los contenidos del archivo se cifran de extremo a extremo usando una **Company Archive Key** — un par de claves Hybrid KEM (X25519 + ML-KEM-1024) generado del lado del cliente por el propietario de la empresa.

```
Company Owner's Browser                          Server
-----------------------                          ------

1. Generate Hybrid KEM keypair (client-side):
   X25519 keypair (32 + 32 bytes)
   ML-KEM-1024 keypair (1568 + 3168 bytes)

2. Derive wrapping key from password:
   wrappingKey = HKDF-SHA256(
     password,
     salt = "trashmail-archive-{account_id}",
     info = "trashmail-archive-key-wrap",
     length = 32
   )
```

```
3. Wrap private keys:
   AES-256-GCM(x25519_priv || mlkem_priv, wrappingKey)

4. Send to server:                          →   Store:
   • Public keys (plaintext)                      archive_x25519_pub
   • Wrapped private keys (encrypted)             archive_mlkem_pub
                                                  wrapped_archive_key
```

Distribución de claves a otros empleados (Administrador, Oficial de Cumplimiento):

1. El propietario descifra las claves privadas del CAK con su contraseña
2. El propietario re-envuelve las claves privadas con la clave derivada de la contraseña del empleado objetivo
3. El servidor almacena la copia re-envuelta en el registro del empleado
4. Cada empleado autorizado tiene su propia copia envuelta de forma independiente

El servidor nunca ve las claves privadas del CAK en texto plano.

1.16.5 15.5 Qué se cifra

Cuando un correo se archiva, se aplican dos capas de cifrado:

Metadatos cifrados (AES-256-GCM con Hybrid KEM):

```json
{
  "d": "INBOUND",
  "s": "user@example.com",
  "r": "admin@company.de",
  "sd": "example.com",
  "rd": "company.de",
  "sz": 45000,
  "ts": "2026-02-27T10:30:00Z",
  "ha": true,
  "ac": 3,
  "en": "John Doe"
}
```

Contenido cifrado del correo (envoltura de clave Hybrid KEM separada):

```json
{
  "subject": "Meeting notes",
  "body": "<html>...</html>",
  "from": "sender@domain.com",
  "to": "recipient@company.de"
}
```

Aplicación Zero-Knowledge: Después del cifrado, los campos de metadatos en texto plano en la base de datos (sender_address, recipient_address, domains) se reemplazan con sus hashes SHA3-256. El servidor almacena solo hashes — los valores originales existen solo dentro de los blobs cifrados.

1.16.6 15.6 Registro de auditoría

Cada acción sobre el archivo se registra en una **cadena de auditoría independiente** (también encadenada por hashes con SHA3-256):

Acción	Cuándo se registra
`EMAIL_RECEIVED` / `EMAIL_SENT` / `DRAFT_ARCHIVED`	Correo archivado
`VIEW_EMAIL` / `VIEW_ATTACHMENT`	Empleado lee correo archivado
`SEARCH_ARCHIVE`	Búsqueda realizada
`EXPORT_EMAIL` / `EXPORT_REPORT`	Datos exportados
`VERIFY_CHAIN` / `CHAIN_VERIFIED_OK` / `CHAIN_VERIFIED_BROKEN`	Verificación de integridad
`LEGAL_HOLD_SET` / `LEGAL_HOLD_RELEASED`	Retención legal activada/desactivada
`ARCHIVE_DECRYPT`	CAK utilizado para descifrar contenido
`ADMIN_ACCESS`	Acción administrativa

Cada entrada de auditoría registra: actor (UUID + rol), dirección IP, ID de sesión, hash del correo objetivo, y si la cadena era válida en el momento del acceso.

1.16.7 15.7 Retención legal y período de retención

- **Período de retención:** Configurable por empresa (predeterminado: 10 años), calculado por correo como `archived_at + retention_years`
- **Retención legal:** Los correos individuales pueden ser puestos bajo retención legal, impidiendo su eliminación hasta que se libere. Incluye razón, actor y marca de tiempo.
- **Cumplimiento GoBD:** La combinación de cadena de hashes inmutable, registro de auditoría completo, retención configurable y retención legal satisface los requisitos de la normativa alemana GoBD (Grundsätze zur ordnungsmäßigen Führung und Aufbewahrung von Büchern, Aufzeichnungen und Unterlagen in elektronischer Form sowie zum Datenzugriff).

1.16.8 15.8 Exportación forense

Los usuarios autorizados (Propietario, Administrador) pueden exportar el estado completo de la cadena para verificación independiente:

- Datos completos de la cadena con todos los hashes de bloques
- Resultado de verificación (válida/rota, número de bloque roto si aplica)
- Valores de hash esperados vs. reales para análisis forense
- Últimas 50 entradas del registro de auditoría
- Formato JSON para re-verificación externa con cualquier implementación de SHA3-256

1.17 16. Lo que el servidor ve — y lo que no

Esta sección documenta explícitamente la frontera Zero-Knowledge.

1.17.1 16.1 El servidor PUEDE ver

Dato	Por qué es visible	Mitigación
Dirección IP	Requisito TCP/IP	Se recomienda el uso de VPN/Tor si se desea mayor privacidad
Marcas de tiempo	Hora de recepción del correo	Inherente al protocolo de correo
Blobs de correo cifrados	Almacenados para recuperación	Cifrados con AES-256-GCM, clave desconocida para el servidor
Tamaños de texto cifrado con relleno	Requisito de almacenamiento	El Bucket Padding oculta los tamaños reales
Dirección DEA del destinatario	Requisito de enrutamiento	La DEA es desechable, no la dirección real
Existencia de la cuenta	Flujo de autenticación	Protección contra enumeración de usuarios implementada
Claves públicas	Requeridas para el cifrado por el servidor	Públicas por definición, no sensibles
Partes de Shamir cifradas	Almacenamiento para el usuario	XOR con claves que el servidor no conoce
Registros OPAQUE	Protocolo de autenticación	No son hashes de contraseñas, cifrados en reposo

1.17.2 16.2 El servidor NO PUEDE ver

Dato	Por qué es invisible
Contenido del correo (asunto, cuerpo, cabeceras)	Cifrado con claves efímeras envueltas vía Hybrid KEM
Contraseña del usuario	OPAQUE — la contraseña nunca se transmite
Clave maestra	Reconstruida solo en el navegador a partir de las partes de Shamir
Claves privadas del vault	Cifradas con la clave maestra antes del almacenamiento
Claves efímeras de correo	Envueltas con Hybrid KEM, el servidor carece de claves privadas
Clave de recuperación / mnemónico	Solo se almacena el hash SHA3-256 de la clave derivada
Salidas PRF del passkey	Vinculadas al hardware, nunca salen del autenticador

Dato	Por qué es invisible
Nombres de carpetas	Cifrados con claves específicas de carpeta
Firmas de correo	Cifradas con la clave maestra
Contenido de solicitudes API	Cifrado vía capa de transporte /e
Contenido de respuestas API	Cifrado antes de la transmisión

1.17.3 16.3 Garantía criptográfica

Incluso con acceso completo a:

- La base de datos completa
- Todo el tráfico de red
- El código fuente y configuración del servidor
- Todos los registros OPAQUE y claves del servidor

…un atacante **no puede** descifrar un solo correo sin la contraseña del usuario (o passkey + clave de recuperación). Esto no es una política — es una imposibilidad matemática impuesta por el diseño criptográfico.

1.18 17. Referencia de algoritmos

1.18.1 17.1 Tabla completa de algoritmos

Componente	Algoritmo	Parámetros	Estándar
Autenticación de contraseña	OPAQUE	RFC 9807, aPAKE	RFC 9807
Derivación de clave de contraseña	PBKDF2-SHA256	600.000 iteraciones, salt 32B, salida 32B	NIST SP 800-132
Cifrado del vault	AES-256-GCM	Clave 256-bit, nonce 96-bit, tag 128-bit	NIST SP 800-38D
Intercambio de claves clásico	X25519	Curve25519, 256-bit	RFC 7748
KEM post-cuántico	ML-KEM-1024	Kyber-1024, NIST Level 5	NIST FIPS 203
Derivación de claves híbrida	HKDF-SHA256	64B IKM, info=“trashmail-hybrid-kem-v1”	RFC 5869
Compartición de secretos	Shamir SSS	k=2, n=3, GF(2^8)	Shamir (1979)
Codificación de clave de recuperación	BIP39	Entropía 256-bit, 24 palabras	BIP-0039
Derivación de clave de recuperación	HKDF-SHA3-256	Salt vinculado a la cuenta	NIST FIPS 202

Componente	Algoritmo	Parámetros	Estándar
Verificación de clave de recuperación	SHA3-256	Salida 32 bytes	NIST FIPS 202
Cifrado de registros OPAQUE	AES-256-GCM	Cifrado en reposo del lado del servidor	NIST SP 800-38D
Desbloqueo de vault por passkey	WebAuthn PRF	Basado en HMAC, vinculado al hardware	WebAuthn Level 2
Compresión	gzip	Nivel 6	RFC 1952
Bucket Padding	Personalizado	17 tamaños (256B–16MB), magic 0xDEAD	—
Firma de respuestas	Ed25519	Clave 256-bit, firma 512-bit	RFC 8032
Cadena de hashes del archivo	SHA3-256	Hash por bloque, encadenamiento secuencial	NIST FIPS 202
Envoltura de clave del archivo (CAK)	HKDF-SHA256 + AES-256-GCM	Clave de envoltura derivada de contraseña	RFC 5869 / NIST SP 800-38D
Verificación de certificados	SHA-256	Comparación de huella digital de certificado TLS	—
Agrupación de hilos de correo	SHA-256	Hash de Message-ID	NIST FIPS 180-4

1.18.2 17.2 Niveles de seguridad

Algoritmo	Seguridad clásica	Seguridad post-cuántica
X25519	128-bit	Roto por el algoritmo de Shor
ML-KEM-1024	Equivalente a 256-bit	NIST Level 5 (≈AES-256)
AES-256-GCM	256-bit	128-bit (algoritmo de Grover)
SHA-256	256-bit	128-bit (algoritmo de Grover)
SHA3-256	256-bit	128-bit (algoritmo de Grover)
Hybrid KEM (combinado)	128-bit (limitado por X25519)	Level 5 (limitado por ML-KEM)

1.19 18. Comparación con otros proveedores

Característica	Aionda Mail	Tuta Mail	Proton Mail
País	Alemania (Stuttgart)	Alemania (Hannover)	Suiza
Zero-Knowledge	Sí (OPAQUE + criptografía del lado del cliente)	Sí	Sí

Característica	Aionda Mail	Tuta Mail	Proton Mail
Post-cuántico	Sí (ML-KEM-1024 + X25519 Hybrid)	Sí (basado en Kyber)	En desarrollo
Protocolo de contraseña	OPAQUE (RFC 9807) — la contraseña nunca sale del navegador	bcrypt (contraseña enviada al servidor sobre TLS)	Basado en SRP
Asunto cifrado	Sí	Sí	No
Cabeceras cifradas	Sí	Parcial	No
Nombres de contactos cifrados	Sí (en el vault)	Sí	No
Direcciones de correo desechables	Sí (función principal, ilimitadas para Plus)	No	Sí (vía SimpleLogin)
Extensión del navegador	Sí (Chrome + Firefox)	No	Vía SimpleLogin
Compartición de carpetas	Sí (Hybrid KEM por destinatario)	Limitado	Sí
Código fuente abierto del cliente	No	Sí	Sí
Auditoría de seguridad	Planificada	Sí	Sí
Recuperación de contraseña	No (por diseño)	No (por diseño)	No (por diseño)
Soporte de passkey	Sí (FIDO2 + PRF)	Sí	Sí
Soporte PGP	Sí (entrante + saliente)	No (protocolo propio)	Sí (OpenPGP)
Archivo de correo conforme a GoBD	Sí (cadena de hashes SHA3-256 + Hybrid KEM)	No	No
Detección MITM (ext. navegador)	Sí (firmas Ed25519 + verificación TLS)	No	No
Perfect Forward Secrecy (API)	Sí (claves efímeras por solicitud)	Desconocido	Desconocido
Ofuscación de tamaño de correo	Sí (Bucket Padding)	Desconocido	No

---

1.20 19. Limitaciones y fronteras honestas

1.20.1 19.1 Modelo de confianza de aplicaciones web

Aionda Mail es una aplicación web. En cada carga de página, el navegador descarga JavaScript de nuestros servidores. Un atacante sofisticado que comprometa nuestros servidores podría teóricamente servir JavaScript modificado que exfiltre claves.

Mitigaciones actuales:

- Hashes de Subresource Integrity (SRI) en todas las etiquetas de script

- Cabeceras Content Security Policy (CSP) que restringen las fuentes de scripts
- Todo el código criptográfico crítico está incluido en el paquete principal de la aplicación
- **Extensión del navegador Guardian** (Sección 14): Verificación de firma Ed25519 en todas las respuestas API detecta manipulación del lado del servidor; verificación de certificados TLS (Firefox) detecta proxies MITM

Mitigaciones planificadas:

- Caché de Service Worker para operación offline (reduce la frecuencia de confianza en cada carga)

1.20.2 19.2 Visibilidad de metadatos

Aunque el contenido del correo está completamente cifrado, ciertos metadatos son visibles para el servidor:

- Cuándo se recibieron los correos (marcas de tiempo)
- Qué dirección DEA recibió el correo
- Tamaño aproximado del correo (dentro de los límites del bucket)
- Patrones de actividad de la cuenta

1.20.3 19.3 Registro de procesamiento de correos

Con fines de diagnóstico, Aionda Mail incluye un **registro de procesamiento de correos** opcional que puede almacenar temporalmente el contenido bruto de los correos entrantes. Esta función es configurable por dirección de correo desechable (DEA) y puede activarse o desactivarse en la configuración de la DEA (“Registrar contenido del correo”).

Cuando está activado (configurable por DEA):

- El mensaje SMTP bruto completo (encabezados + cuerpo) se almacena en texto plano en el servidor
- Eliminación automática tras un breve período de retención (menos de 7 días)
- Accesible únicamente para el propietario de la cuenta a través de la API autenticada
- Propósito: solución de problemas de entrega, verificación del reenvío, revisión de decisiones de filtrado de spam

Cuando está desactivado:

- No se almacena ningún contenido de correo en el registro de procesamiento
- Solo se registran metadatos (dirección del remitente, marca de tiempo, estado de entrega)
- El cifrado del vault sigue siendo el único mecanismo de almacenamiento

Importante: Este registro de procesamiento es independiente del vault cifrado. Los correos almacenados en el vault siempre están cifrados con Hybrid KEM, independientemente de la configuración del registro. El registro de procesamiento existe como una función heredada del sistema de reenvío de correos y proporciona transparencia operativa. Los usuarios que requieran almacenamiento estrictamente Zero-Knowledge para todos los correos deben desactivar esta opción.

1.20.4 19.4 Seguridad del correo externo

Los correos enviados a o recibidos de direcciones que no son de Aionda viajan a través de la infraestructura estándar de correo (SMTP). Aunque se almacenan cifrados en el vault, el contenido del correo fue visible durante el tránsito a menos que se haya utilizado cifrado PGP.

1.20.5 19.5 Sin custodia de claves

No existe clave maestra, puerta trasera ni mecanismo de recuperación disponible para Aionda GmbH. Si un usuario pierde su contraseña y todos los métodos de recuperación, sus datos se pierden permanentemente. Esta es una decisión de diseño intencional que demuestra la integridad del modelo Zero-Knowledge.

1.21 20. Hoja de ruta

Hito	Estado	Objetivo
Arquitectura Zero-Knowledge	Completado	—
Hybrid KEM post-cuántico (ML-KEM-1024)	Completado	—
Autenticación OPAQUE (RFC 9807)	Completado	—
Shamir Secret Sharing (2-de-3)	Completado	—
Capa de transporte API cifrada	Completado	—
Soporte Passkey/WebAuthn PRF	Completado	—
Calendario cifrado de extremo a extremo	Completado	—
Archivo de correo conforme a GoBD (Blockchain)	Completado	—
Protección MITM Guardian (Ed25519)	Completado	—
Verificación de certificados TLS (Firefox)	Completado	—

1.22 Historial del documento

Versión	Fecha	Cambios
1.0	Marzo 2026	Publicación inicial

1.23 Contacto

Aionda GmbH Stephan Ferraro Stuttgart, Alemania

Email: contact-46epp9ba@contact.aionda.com Web: https://mail.aionda.com

---

Este documento describe la arquitectura de seguridad de Aionda Mail a fecha de marzo de 2026. Los sistemas criptográficos evolucionan — este documento se actualizará conforme la arquitectura cambie.