Resumen Teórica 7 : Criptografía (parte 4)

Tomás F. Melli

September 2025

Índice

1 Codificación

La clase comienza con este texto :

```
Tm9zLCBsb3MgcmVwcmVzZW50YW50ZXMgZGVsIHB1ZWJsbyBkZSBsYSBOYWNp824g
QXJnZW50aW5hLCByZXVuaWRvcyBlbiBDb25ncmVzbyBHZW5lcmFsIENvbnN0aXR1
eWVudGUgcG9yIHZvbHVudGFkIHkgZWxlY2Np824gZGUgbGFzIHByb3ZpbmNpYXMg
cXVlIGxhIGNvbXBvbmVuLCBlbiBjdW1wbGltaWVudG8gZGUgcGFjdG9zIHByZWV4
aXN0ZW50ZXMsIGNvbiBlbCBvYmpldG8gZGUgY29uc3RpdHVpciBsYSB1bmnzbiBu
YWNpb25hbCwgYWZpYW56YXIgbGEganVzdGljaWEsIGNvbnNvbGlkYXIgbGEgcGF6
IGludGVyaW9yLCBwcm92ZWVyIGEgbGEgZGVmZW5zYSBjb236biwgcHJvbW92ZXIg
ZWwgYmllbmVzdGFyIGdlbmVyYWwsIHkgYXNlZ3VyYXIgbG9zIGJlbmVmaWNpb3Mg
ZGUgbGEgbGliZXJ0YWQgcGFyYSBub3NvdHJvcywgcGFyYSBudWVzdHJhIHBvc3Rl
cmlkYWQgeSBwYXJhIHRvZG9zIGxvcyBob21icmVzIGRlbCBtdW5kbyBxdWUgcXVp
ZXJhbiBoYWJpdGFyIGVuIGVsIHN1ZWxvIGFyZ2VudGlubzsgaW52b2NhbmRvIGxh
IHByb3RlY2Np824gZGUgRGlvcywgZnVlbnRlIGRlIHRvZGEgcmF6824geSBqdXN0
aWNpYTogb3JkZW5hbW9zLCBkZWNyZXRhbW9zIHkgZXN0YWJsZWNlbW9zIGVzdGEg
Q29uc3RpdHVjafNuIHBhcmEgbGEgTmFjafNuIEFyZ2VudGluYS4gCg==
```

La idea es entender que si bien no se entiende, no significa que el texto esté cifrado. Está **codificado**, es decir, con un algoritmo de codificación se transforma información de un formato a otro. No se cifra ni se comprime la información.
Algunos ejemplos de estos algoritmos son :

- ASCII: convierte letras a números (ejemplo: "A" → 65).
- Base64 (*)
- URL encoding: transforma caracteres que no pueden ir en una URL (espacio → %20).
- UTF-8: codificación de caracteres que permite representar letras de cualquier idioma.

1.1 Base64

Base64 es un mecanismo de codificación que permite representar datos binarios (imágenes, archivos, claves, etc.) en texto ASCII. Se utiliza un alfabeto de 64 caracteres seguros y comunes, lo que facilita transmitir información por canales que solo aceptan texto (ejemplo: correo electrónico, JSON, XML, HTTP headers).

Funcionamiento

- **Agrupación de bytes:**
  - Toma la entrada binaria de a 3 bytes (24 bits).
  - Cada 24 bits se dividen en 4 grupos de 6 bits.
- **Mapeo a caracteres:**
  - Cada grupo de 6 bits se transforma en un número entre 0 y 63.
  - Ese número se representa con un carácter del alfabeto Base64:
    - `A--Z` → valores 0–25
    - `a--z` → valores 26–51
    - `0--9` → valores 52–61
    - `+` → 62
    - `/` → 63
- **Padding (`=`):**
  - Si la cantidad de bytes originales no es múltiplo de 3, se agregan ceros al final para completar.
  - Al decodificar, el carácter `=` indica que esos bits extra deben ignorarse.
- **Ejemplo:**
  - Si hay 1 byte → se codifica en 2 caracteres + `==`.
  - Si hay 2 bytes → se codifica en 3 caracteres + `=`.

2 MIME - Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extensions (MIME) es un estándar de Internet (definido en las RFC 2045 y siguientes) que extiende el formato tradicional de los correos electrónicos.
Originalmente, el email solo permitía transmitir texto en **US-ASCII** (7 bits), lo que lo hacía limitado. MIME amplía esas capacidades y permite:

- Soportar texto en diferentes conjuntos de caracteres (ej. UTF-8, ISO-8859-1).
- Incluir archivos binarios como adjuntos (imágenes, audio, documentos).
- Enviar mensajes que incluyan múltiples tipos de objetos en una misma estructura (texto + adjuntos + multimedia).

Además, los tipos de contenido definidos por MIME son utilizados no solo en correo electrónico, sino también en otros protocolos como **HTTP** (por ejemplo, en la web para indicar el tipo de archivo).

Ejemplos de Content-Type

- **text**
  - `text/plain`
  - `text/richtext`
- **message**
  - `message/rfc822`
- **image**
  - `image/jpeg`
  - `image/gif`
- **video**
  - `video/mpeg`
- **application**
  - `application/postscript`
  - `application/octet-stream`
- **multipart**
  - `multipart/mixed`
  - `multipart/alternative`

2.1 S/MIME

Secure / Multipurpose Internet Mail Extensions (S/MIME) es un estándar para la seguridad del correo electrónico, basado en criptografía de clave pública.
Define el uso de un tipo especial de contenido, por ejemplo:

- `application/pkcs7-mime`
- `application/pkcs7-signature`

La funcionalidad de S/MIME está incorporada en la mayoría de los clientes de correo electrónico modernos.

2.1.1 Servicios provistos por S/MIME

- Autenticidad del emisor (autoría).
- Integridad del mensaje (protección contra modificaciones).
- No repudio (el emisor no puede negar haber enviado el mensaje).
- Confidencialidad de los datos (cifrado del contenido).

En resumen:

- **MIME**: extiende el formato del email para soportar múltiples tipos de contenido.
- **S/MIME**: añade seguridad al correo electrónico usando cifrado y firmas digitales.

3 Problemática actual

En el contexto de los documentos en formato digital, surgen diversas problemáticas relacionadas con la seguridad y la confianza:

- No es posible determinar con certeza el **autor** de un documento digital.
- Un documento digital es fácilmente **alterable**, y no suele existir evidencia confiable de dichas alteraciones.
- El autor puede **no reconocerlo**, y no existe un mecanismo fehaciente de verificación ante terceros.

Por estas razones, en muchos ámbitos aún se sostiene la idea de que:

Necesitamos...

Para resolver estos problemas, los sistemas de seguridad de la información deben proveer:

- **Autenticidad del autor:** atribuir el documento a su autor (persona o aplicación) de manera fehaciente, garantizando la identidad.
- **Integridad del contenido:** asegurar que el documento no haya sido modificado luego de ser firmado.
- **No repudio del documento:** garantizar que el emisor del mensaje no pueda negar su existencia ni su autoría, siendo verificable ante terceros.

3.1 Firma digital

La **firma digital** es un conjunto de datos expresados en formato digital que permite:

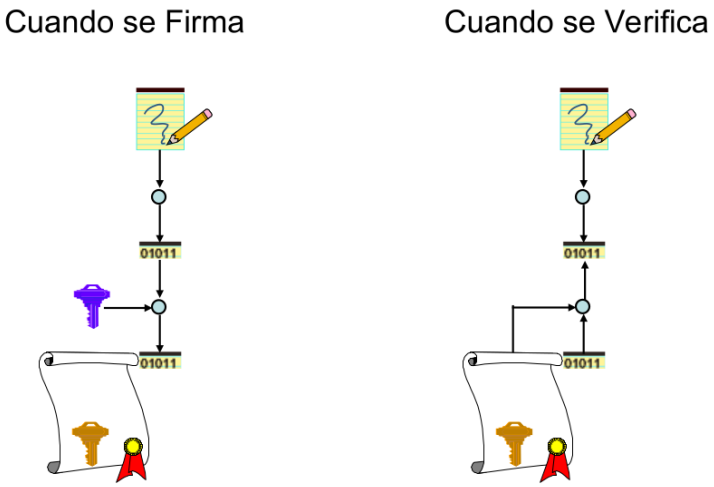
- **Identificar al firmante** de un documento.
- **Verificar la integridad** de su contenido.

Para que sea confiable, la firma digital debe cumplir con los siguientes requisitos:

- Pertenecer únicamente a su titular.
- Encontrarse bajo su absoluto y exclusivo control.
- Ser susceptible de verificación por terceros.
- Estar vinculada a los datos del documento digital, de modo tal que cualquier alteración sea evidente.

3.1.1 Funcionamiento

El proceso de firma digital se basa en criptografía de clave pública:



Cuando se firma

1. Se genera un **hash** del documento.
2. El hash se cifra con la **clave privada** del firmante.
3. El resultado es la **firma digital**, que se adjunta al documento.

Cuando se verifica

1. Se vuelve a calcular el hash del documento recibido.
2. Se descifra la firma digital con la **clave pública** del firmante.
3. Si ambos valores coinciden, se confirma la autenticidad y la integridad.

3.2 Sistema de firma digital

En un sistema de firma digital intervienen cuatro actores principales:

- **El suscriptor:** la persona que firma el documento.
- **El tercero usuario:** quienes necesitan verificar la firma digital.
- **La autoridad certificante (CA):** quien testimonia y garantiza que una firma digital pertenece a una cierta persona.
- **El organismo de control:** encargado de regular y supervisar el sistema.

Más referencias

- **RFC 5751** – Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification.
- **RFC 5652** – Cryptographic Message Syntax (CMS).
- **RFC 5126** – CMS Advanced Electronic Signatures (CAdES).
- Comandos de **OpenSSL CA**.
- Blog de Cloudflare: https://blog.cloudflare.com/introducing-cfssl/.

Resumen: La firma digital aporta autenticidad, integridad, no repudio y confidencialidad, permitiendo trasladar al mundo digital las garantías que en el mundo físico provee el papel.

4 OpenSSL

OpenSSL es una implementación **Open Source** de diversos algoritmos y estándares criptográficos. Proporciona herramientas y librerías para realizar cifrado, firmas digitales, certificados y comunicación segura mediante protocolos como TLS/SSL.

- Sitio oficial: https://www.openssl.org
- Documentación de uso:
  - https://www.madboa.com/geek/openssl/
  - https://github.com/openssl/openssl/wiki

OpenSSL es ampliamente utilizado tanto en servidores web como en aplicaciones que requieren **seguridad criptográfica**, siendo una referencia clave en la implementación de protocolos seguros.

5 Side Channel (Canales Laterales)

Un **Side Channel** es un tipo de ataque que se basa en **información obtenida a partir de efectos secundarios** de la implementación de un algoritmo criptográfico, y **no en debilidades del algoritmo en sí**.

5.1 Tipos de Side Channels

- **Tiempo:** ataques basados en la cantidad de tiempo que tardan ciertos cálculos.
- **Consumo eléctrico:** diferencias de consumo del hardware dependiendo de la operación realizada.
- **Electromagnéticos:** fuga de información a través de radiación electromagnética.
- **Acústico:** análisis de sonidos emitidos durante el cómputo.
- **Otros:** cualquier información indirecta que permita inferir datos secretos.

Referencia: https://www.tau.ac.il/~tromer/acoustic/

6 Forward Secrecy (Secreto hacia adelante)

La **Forward Secrecy** protege las comunicaciones pasadas incluso si se compromete la clave privada del servidor.

- Sin Forward Secrecy: si un atacante obtiene la clave privada del servidor, podría descifrar todas las comunicaciones previas cifradas bajo esa clave.
- Con Forward Secrecy: las claves de sesión se generan de manera temporal y efímera, por lo que comprometer la clave privada no permite descifrar sesiones anteriores.

Referencia: https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices

7 Padding Oracle Attacks

Un **Padding Oracle Attack** explota la forma en que una aplicación maneja los errores de padding en cifradores de bloques (por ejemplo, modo CBC con padding PKCS#5).

Escenario típico

- Texto válido y correctamente cifrado: la aplicación responde normalmente.
- Texto inválido pero correctamente cifrado: la aplicación indica que el valor recibido no es válido.
- Texto con padding incorrecto: la aplicación indica un error de padding.

Consecuencia

- Un atacante puede **descifrar mensajes** y **cifrar mensajes arbitrarios** sin conocer la clave simétrica, aprovechando la información revelada en las respuestas de la aplicación.

Referencia: https://www.usenix.org/legacy/event/woot10/tech/full_papers/Rizzo.pdf

8 Actualidad en cifrado simétrico

Hasta ahora, los modos de cifrado para algoritmos simétricos por bloque se enfocaban únicamente en **cifrar**. Sin embargo, en la práctica moderna es necesario agregar **autenticación** para garantizar la integridad de los datos, sin depender únicamente de un MAC externo.

Modos y algoritmos modernos

- **Galois/Counter Mode (GCM):** combina cifrado y autenticación en un solo esquema eficiente y seguro.
- **AES-GCM para TLS:** ampliamente utilizado en protocolos TLS modernos. https://tools.ietf.org/html/rfc5288
- **Cifradores simétricos de flujo modernos:**
  - **Salsa20/ChaCha:** algoritmos rápidos y seguros, ampliamente implementados en TLS actual. https://cr.yp.to/chacha.html

Buenas prácticas

- Cuando se realizan operaciones de cifrado y autenticación por separado, suele ser más seguro **primero cifrar y luego autenticar**.

9 Criptografía post-cuántica (PQC)

La **criptografía post-cuántica** está diseñada para ser segura frente a ataques de **computadoras cuánticas**.

- Los algoritmos actuales de criptografía asimétrica, como **RSA** y **ECC**, son vulnerables a algoritmos cuánticos como el de Shor.

9.1 Algoritmos PQC destacados

- **FIPS 203 – ML-KEM (Kyber):** reemplaza algoritmos de intercambio de claves como DH y ECDH. Ya se está usando en navegadores como Firefox, Chrome y Edge.
- **FIPS 204 – ML-DSA (Dilithium):** esquema de firma digital robusto.
- **FIPS 205 – SPHINCS+:** esquema de firma digital basado en funciones hash, resistente a ataques cuánticos y sin estructuras algebraicas ocultas.

Referencia: https://pq.cloudflareresearch.com/

10 Otras aplicaciones de criptografía

La criptografía moderna no solo protege comunicaciones, sino que también habilita aplicaciones avanzadas en distintos ámbitos:

- **Mental Poker:** juegos de cartas en línea sin confiar en un tercero.
- **Zero-Knowledge Proofs (ZKP):** permiten demostrar conocimiento de un secreto sin revelarlo. https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/
- **Smart Contracts:** contratos autoejecutables basados en blockchain.
- **Homomorphic Encryption y Secret Sharing:** permiten realizar operaciones sobre datos cifrados sin descifrarlos y compartir secretos de manera segura.
- **Blockchains:** registro distribuido y seguro de transacciones, impulsado por criptografía.

11 Apéndice

11.1 Algoritmos de cifrado simétrico modernos

11.1.1 AES-GCM (Galois/Counter Mode)

Modo: Galois/Counter Mode (GCM) combina cifrado y autenticación.
Cómo funciona:

- Utiliza AES en modo **contador (CTR)** para cifrar los bloques de datos, generando un flujo pseudoaleatorio que se combina con el mensaje mediante XOR.
- Calcula un **tag de autenticación** usando aritmética en el **campo de Galois**, garantizando la integridad del mensaje.
- Permite cifrado y autenticación en paralelo, lo que mejora la eficiencia.
- Ampliamente usado en **TLS 1.2 y 1.3**, VPNs y comunicaciones seguras.

Referencia: https://tools.ietf.org/html/rfc5288

11.1.2 Salsa20

Modo: Cifrador de flujo.
Cómo funciona:

- Genera un **stream pseudoaleatorio** basado en la clave y un nonce.
- Combina el stream con el mensaje mediante **XOR** para cifrarlo.
- Operaciones internas: suma modular, XOR y rotaciones de bits.
- No requiere padding y es muy eficiente en software.

Referencia: https://cr.yp.to/chacha.html

11.1.3 ChaCha

Modo: Cifrador de flujo, variante optimizada de Salsa20.
Cómo funciona:

- Similar a Salsa20, genera un stream pseudoaleatorio que se combina con el mensaje mediante XOR.
- Optimizado para **hardware y software moderno**, con mayor resistencia a ciertos ataques de análisis de claves.
- Utilizado en TLS modernos, VPNs y protocolos móviles.

Referencia: https://cr.yp.to/chacha.html

11.2 Algorítmos post-cuánticos

11.2.1 FIPS 203 – ML-KEM (Kyber)

Tipo: Algoritmo de intercambio de claves.
Cómo funciona:

- Diseñado para reemplazar algoritmos clásicos de intercambio de claves como **Diffie-Hellman (DH)** y **ECDH**.
- Basado en problemas de **lattice** resistentes a ataques cuánticos.
- Permite que dos partes generen una clave compartida segura incluso frente a un atacante con computadora cuántica.
- Ya implementado en navegadores modernos como **Firefox, Chrome y Edge**.

11.2.2 FIPS 204 – ML-DSA (Dilithium)

Tipo: Algoritmo de firma digital.
Cómo funciona:

- Permite generar y verificar firmas digitales de manera segura contra ataques cuánticos.
- Basado en problemas de **lattice** que no pueden ser resueltos eficientemente por computadoras cuánticas.
- Garantiza autenticidad, integridad y no repudio de los mensajes.

11.2.3 FIPS 205 – SPHINCS+

Tipo: Algoritmo de firma digital basado en funciones hash.
Cómo funciona:

- Utiliza únicamente **funciones hash** y estructuras de árbol para generar firmas digitales.
- No depende de estructuras algebraicas ocultas, lo que lo hace robusto frente a ataques cuánticos.
- Garantiza autenticidad, integridad y no repudio sin recurrir a problemas matemáticos vulnerables a computadoras cuánticas.