TEMA 2
APLICACIONES DE LA CRIPTOGRAFÍA

Protegiendo tus datos de filtraciones y accesos no autorizados

José Manuel Redondo López Proyecto *"S-81 Isaac Peral"* v4.0

Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Logo: @creative_vanesa (*Instagram*)

ÍNDICE

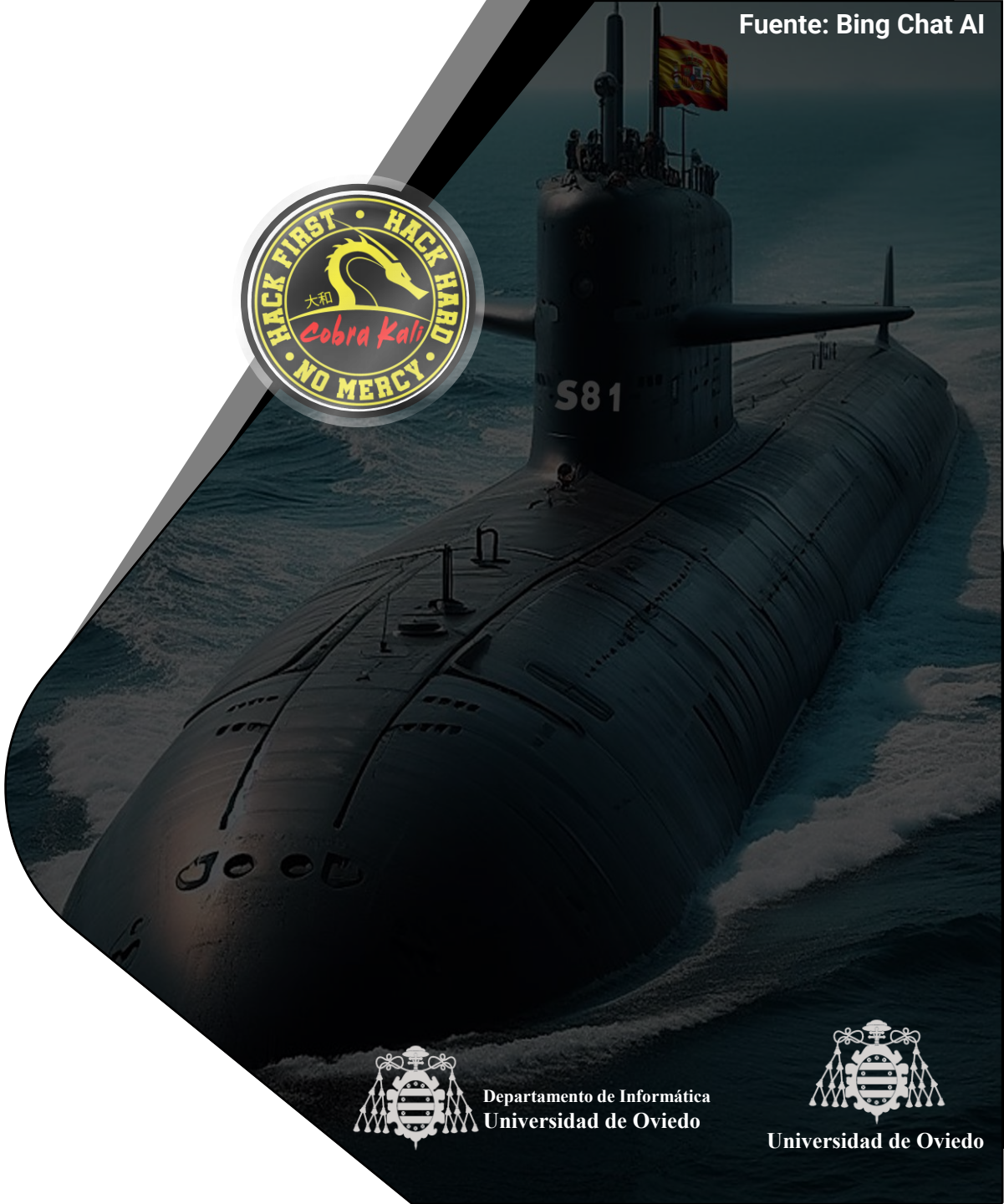
< Ir al Índice

INTRODUCCIÓN

¿De qué vamos a hablar?

*Iconos por Bing Chat AI

Fuente: Bing Chat AI



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿QUÉ TE VAS A ENCONTRAR EN ESTE BLOQUE?

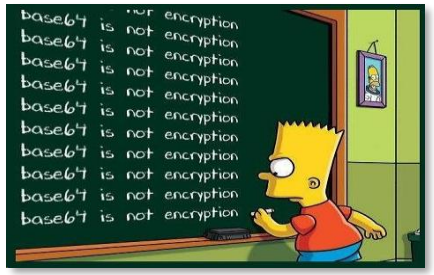
- **En este bloque aprenderás**
  - Qué son una serie de operaciones que manipulan (pero no cifran datos): **minificación**, **codificación** y **ofuscación**
    - Y las diferencias entre ellas
  - Qué es la **criptografía** y el **criptoanálisis**
  - La diferencia entre **criptografía** y **esteganografía**
  - Por qué la esteganografía **no garantiza la seguridad** por si sola
  - Qué es la "**CIA**" en lo relativo a la criptografía

OBJETIVOS GENERALES DEL TEMA

- **Conocer los fundamentos y aplicaciones de las diferentes técnicas de criptografía modernas**
  - Cómo y por qué se aplican en diferentes escenarios
  - Por qué unas técnicas son más adecuadas que otras para proporcionar ciertas características de seguridad
- **Eliminar falsas creencias**
  - **Minificación de código** y **ofuscación de archivos NO** son criptografía
    - Los programas siguen siendo ejecutables, sólo dificulta leerlos
  - La **codificación de archivos/datos TAMPOCO** es criptografía
    - Ej.: MIME (https://en.wikipedia.org/wiki/MIME), Base64 (https://en.wikipedia.org/wiki/Base64)...
    - Solo permite transferir contenidos con un juego de caracteres aceptable por el protocolo de transmisión
    - Es fácilmente reversible

JavaScript minificado. Fuente:
https://dev.to/aryaziai/minifying-code-shortcut-4d6c

```
{font-family:sans-serif;-ms-text-size-adjust:100%;-webkit-text-size-
body{margin:0}article,aside,details,figcaption,figure,footer,header,hgroup,mai
parent}a:active,a:hover{outline:0}b,strong{font-weight:bold}h1{font-size:2em;ma
,input,optgroup,select,textarea{color:inherit;font:inherit;margin:0}button{over
"button"],input[type="reset"],input[type="submit"]{-webkit-appearance:button;cu
r:0;padding:0}input{line-height:normal}input[type="checkbox"],input[type="radio
```




```
This is an MIME test email
--VGhpc0lzQW5FbWFpbEJvdW5kYXJ5Cg==
Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
Content-Transfer-Encoding: base64

Q29udGVudC1UeXBlOiBtdWx0aXBhcnQvbWI4ZWQ7CiAgICBib3VuZGFyeT0iMTI3LjAuMC4xLjUw
MS4xNjkyMy4xNTIyNDA4ODQ3Ljc1My4xIgpNaW1ILVZlcnNpb246IDEuMAoKaWYgeW91IGNhbiBy
ZWFkIHRoaXMsIHlvdXIgbWFpbGVyIGlzIG5vdCBNSU1FLWF3YXJlCgotLTEyNy4wLjAuMS41MDEu
MTY5MjMuMTUyMjQwODg0Ny43NTMuMQpDb250ZW50LVR5cGU6IHRleHQvcGxhaW4KQ29udGVudC1U
```

Fichero Excel enviado como adjunto de email, usando codificación Base64. Fuente:
https://stackoverflow.com/questions/49605212/failed-to-send-an-mime-email-body-with-mail-command

➢ Ejercicios **L3B1_ENCODE, L3B1_OBFUSCATE**

CRIPTOLOGÍA

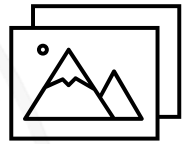
- **Ciencia que se ocupa de los problemas de intercambio seguro de información**
  - Siempre usa un **origen** y un **destino** de la información, junto con un **canal de comunicación**
  - Tiene como objetivo generar comunicaciones muy seguras e "irrompibles"
    - Romperlas significa poder leer sus contenidos sin la autorización adecuada
  - ¡Es muy antigua! (2000+ años)
- **Dos líneas principales de investigación**
  - **Criptografía**: sistemas capaces de **ocultar el significado de la información**
    - Generan información cifrada inaccesible para aquellos que no están destinados a conocerla
  - **Criptoanálisis**: sistemas capaces de **descifrar** información cifrada
    - Romper un sistema de criptografía

Fuente:
https://en.wikipedia.org/wiki/Enigma_machine

Fuente:
https://danbrown.fandom.com/wiki/Cryptex

¿Por qué la criptografía?

- El objetivo principal de un sistema seguro es **lograr una protección adecuada de los datos / información que gestiona**
- Para ello podemos utilizar dos técnicas
  - **Criptografía**: ocultar el **significado** de la información



→ **Z$5%tyc...zz**

    - La información se procesa y se transforma en una **versión no legible** (cifrada / encriptada)
    - Esto se consigue procesando la información a través de diferentes **algoritmos de encriptación**
    - El algoritmo elegido depende **de la finalidad de la información** (datos privados, contraseñas...)
    - **La ciberseguridad moderna depende en gran medida del uso de métodos de criptografía adecuados**
  - **Esteganografía**: ocultar la **existencia** de la información
    - La información está **incrustada** en otra información distinta (Ej.: una imagen)
      - Es inaccesible para aquellos que no saben que existe (oculta su presencia)
    - Sin embargo, **¡usar solo seguridad por ocultación de información no es aconsejable!**
      - En inglés se suele denominar "security by obscurity"
      - Normalmente, se combina con criptografía

Fuente: iconos *PowerPoint*

ESTEGANOGRAFÍA

Mensaje a ocultar

<secreto>
Esta imagen es menos inocente de lo que parece. De hecho, está ocultando este mensaje secreto. (sabes que es secreto porque está entre tags <secreto> </secreto> ☺)
</secreto>

Fichero "Contenedor"

Password

Proceso esteganográfico

Esteganograma: La imagen ahora contiene más información (el mensaje) ¡pero no se aprecia a simple vista!

Fuente: https://www.incibe-cert.es/blog/esteganografia-y-estegoanalisis-basicos

GENERACIÓN DE ESTEGANOGRAMAS

- Los esteganogramas se generan (y extraen) con herramientas adecuadas
- Algunas de las más populares son
  - Steghide
    - steghide.sourceforge.net/documentation/manpage_es.php
  - Steganos Security Suite
  - Outguess
  - Stegtools
  - SilentEye
  - Zsteg
    - https://github.com/zed-0xff/zsteg
  - **Para imágenes**
    - Exiftool: https://en.wikipedia.org/wiki/ExifTool
    - Stegsolve: http://www.caesum.com/handbook/Stegsolve.jar

Fuente: https://www.publicdomainpictures.net/es/view-image.php?image=215827&picture=mujer-de-yoga

Extraer un mensaje escondido con `steghide`

```
root@kali:~/Descargas# steghide extract -sf lady-in-yoga.jpg
Anotar salvoconducto:
anot? los datos extra?dos e/"secret.txt".
root@kali:~/Descargas#
```

```
root@kali:~/Descargas# cat secret.txt
Don't forget to knock: 3000,3001,3002
root@kali:~/Descargas#
```

La imagen aparenta ser normal y visualmente no hay diferencia, pero dentro contiene un mensaje

```
root@kali:~/Descargas# exiftool lady-in-yoga.jpg
ExifTool Version Number         : 11.65
File Name                       : lady-in-yoga.jpg
Directory                       :
File Size                       : 21 kB
File Modification Date/Time     : 2019:10:07 13:02:12+02:00
File Access Date/Time           : 2019:10:07 13:02:20+02:00
File Inode Change Date/Time     : 2019:10:07 13:02:12+02:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Image Width                     : 283
Image Height                    : 676
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 283x676
Megapixels                      : 0.191
root@kali:~/Descargas#
```

Esteganografía + Criptografía

- Simplemente ocultar un mensaje **no es una forma de implementar seguridad fiable**
  - **¡La "seguridad por oscuridad" (Security by obscurity) NUNCA es una buena idea!**
- Por eso **la esteganografía se suele combinar con la criptografía**
  - El mensaje oculto que se va a intercambiar debe cifrarse antes de colocarse en el objeto contenedor
  - Incluso se descubre el patrón esteganográfico, el mensaje sigue siendo desconocido
- Obtenemos una ventaja significativa
  - Si solo se usa cifrado, no podemos ver el significado, pero sabemos que algo se está transmitiendo
    - Por lo tanto, ataques de fuerza bruta (u otras variantes) se pueden usar contra el algoritmo de cifrado ("el algo" que se está transmitiendo)
  - Si se usan ambos, la probabilidad de transmitir información sin que sea detectada aumenta
    - ¡No puedes atacar algo que no conoces! ☺
- ¡Vamos a ver entonces las formas de cifrado que podemos usar para ello!

Objetivos de la criptografía

- **Las comunicaciones cifradas son básicas para lograr una seguridad real**
  - ¡Ocultemos o no el mensaje transferido!
- **La criptografía tiene como objetivo lograr esto (coloquialmente llamado la "CIA")**
  - **Confidencialidad**: Solo aquellos que conocen el método de descifrado pueden conocer el significado de la información
    - Esto se aplica a la información almacenada, así como al tráfico de red
    - Usa **algoritmos de cifrado**
  - **Integridad**: Garantía de que los datos transmitidos se recibieron igual que el remitente los envió
    - **Sin ninguna alteración / manipulación**: la información permanece inalterada de origen a destino
    - Usa **funciones hash**
  - **Autenticación**: Verificación de que la identidad de los remitentes y receptores es la que se declara
    - ¡No quieres enviar datos privados a / aceptarlos de entidades no confiables / falsas!
    - Esto se aplica a usuarios y a máquinas
    - **Para ello se usan firmas digitales y certificados** (**metadatos** y **marcas de agua** no son fiables)
- **Los algoritmos de cifrado autenticado (vistos luego) implementan los 3 objetivos**
  - https://en.wikipedia.org/wiki/Authenticated_encryption

¿Crees que lo has entendido todo? ¡Autoevalúate!

?

- **Eres capaz de hacer lo siguiente**
  - *Entender que minificar, codificar y ofuscar tienen sus usos, pero NO SON cifrado*
  - *Comprender que, si un código minificado ocupa menos, aparte de ser menos legible mejora el rendimiento de carga de las webs*
  - *Ser consciente de que la criptografía (y su "contraria", el criptoanálisis) son ciencias matemáticas muy antiguas*
  - *Tener claro que la esteganografía, independientemente del software o método usado para hacerla, es enviar información sin que nadie lo sepa y sin que, por tanto, sea evidente a simple vista*
  - *Entender que la esteganografía sin criptografía no sirve para garantizar la seguridad, porque ocultar la información solamente nunca es seguro*

< Ir al Índice

Fuente: Bing Chat AI

Simétricos >

Asimétricos >

CONFIDENCIALIDAD

Algoritmos de cifrado

*Iconos por Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

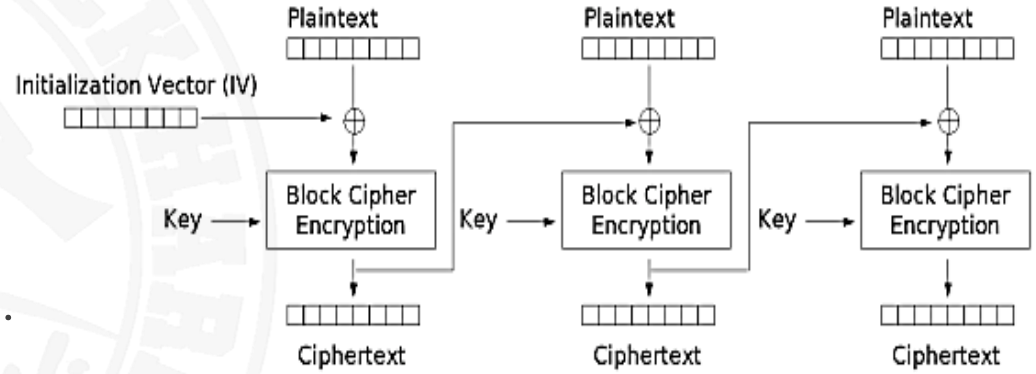
¿QUÉ TE VAS A ENCONTRAR EN ESTE BLOQUE?

En este bloque aprenderás

- Los dos tipos de algoritmos que **usan claves criptográficas** y otros tipos de cifrado
- Los principios básicos de las **claves criptográficas**
- Las bases teóricas de los algoritmos **de clave simétrica**, sus contrapartidas y algunas de sus aplicaciones más populares
- Las bases teóricas de los algoritmos **de clave asimétrica**, sus contrapartidas y algunas de sus aplicaciones más populares

Clasificación de algoritmos de cifrado: Por cómo procesan la información

- **Los algoritmos de (des)cifrado son funciones que transforman la información**
  - El **cifrado** oculta el significado de la información
  - **Descifrar** lo recupera
    - Si se aplica sobre **un algoritmo de cifrado compatible**
- **Los algoritmos de cifrado se pueden clasificar en**
  - **Cifrado por bloques**: IDEA, AES, RSA...
    - Cifran la información por fragmentos (bloques)
    - El tamaño del bloque (en bits) varía con el algoritmo: 64, 128...
    - Usar un **vector de inicialización** (IV) aleatorio
      - Datos para inicializar el cifrado
      - La forma en que usan este IV determina su **modo de operación**
        - BCE, CBC, PCBC, OFB...
      - El modo podría ser parte del nombre del algoritmo. Ej. : AES-CBC
      - No vamos a ver los modos de estos algoritmos
  - **Cifrados de flujo** (stream ciphers): A5, RC4, SEAL ...
    - Cifran la información bit a bit



Cipher Block Chaining (CBC) mode encryption

Fuente:
https://es.wikipedia.org/wiki/Modos_de_operaci%C3%B3n_de_una_unidad_de_cifrado_por_bloques

Clasificación de algoritmos de cifrado: Por cómo usan las claves

- **Cifrados sin clave** (o cifrados de transposición)
  - **Permutan** posiciones de unidades de texto con un sistema regular
    - Las unidades de texto son caracteres o grupos de caracteres
    - El texto cifrado es una permutación del texto (es decir, **se reordena**)
  - Para cifrar se usa una función matemáticamente biyectiva sobre las posiciones de los caracteres (y una función inversa para descifrar)
  - **Ejemplos**: ROT13, ROT47, ... (pruébalos en https://cryptii.com/)
- **Cifrados con clave criptográfica** (los que estudiaremos)
  - Usan **claves**: los que las tengan pueden manipular la información
  - **Algoritmos de clave simétrica**
    - El emisor y el receptor **comparten** la misma clave privada
  - **Algoritmos asimétricos**
    - **El emisor y el receptor tienen dos claves cada uno**
    - **Una es Pública** (para cifrar)
    - **La otra Privada** (para descifrar / autenticar)
    - **¡Es la base del comercio electrónico moderno!**

	SIMÉTRICA	ASIMÉTRICA
Tipo de clave	Clave secreta compartida	Par de claves publica y privada único
Fortalezas	• **Mejor rendimiento** • Fácil de entender (tener una sola clave implica poder acceder a la información)	• **Las claves privadas nunca se exponen** • Permite autenticar al origen de contenidos
Debilidades	• Secreto compartido (clave expuesta) • No autentica el origen (si la clave queda expuesta, la información puede falsificarse)	• El manejo de claves públicas necesita infraestructura adicional (*CAs*, *Trusted Rings*) • **Computacionalmente intensiva** (comparada con la simétrica)
Principales diferencias	Las aplicaciones deben guardar las claves, pero si se encuentran, **pueden falsificarse mensajes**	**Es resistente a la falsificación** si las claves privadas no son comprometidas

CIFRADO CON CLAVE CRIPTOGRÁFICA

- Los **principios de Kerckhoffs** son las principales pautas para crear un algoritmo de cifrado basado en claves irrompible
  - ¡Enunciado en 1883!
  - **La efectividad no debe depender de que el diseño del algoritmo sea secreto**
    - La **clave debe ser el elemento más importante** para descifrar con éxito un mensaje cifrado
    - Incluso si el algoritmo es público, la información no se puede descifrar sin una clave adecuada.
  - **Las claves deben ser fáciles de recordar**
    - Para evitar escribirlas en medios de almacenamiento no seguros
  - La información cifrada resultante debe ser **alfanumérica**
  - Los sistemas de cifrado deben ser **manejables por una sola persona**
  - Los sistemas de cifrado deben ser **fáciles de usar**
- La criptografía moderna está influenciada por la **teoría de la información de Claude Shannon**
  - https://en.wikipedia.org/wiki/Information_theory

CLAVE CRIPTOGRÁFICA

- **Patrón** de información que utilizan los algoritmos criptográficos para **cifrar y descifrar** la información
- Obtendremos una salida totalmente diferente si cambiamos la clave
  - ¡Incluso si usa el mismo algoritmo y datos iniciales!
- La **fortaleza de la clave** (**key strength**) se mide en **bits** o **número de dígitos**
- Para un algoritmo dado, las claves más largas significan claves más fuertes
  - Sin embargo, las claves más largas necesitan más tiempo para procesar la información
  - Fuerte significa **difícil de romper**
    - Es decir, descifrar la información sin tener la clave

```
Información_Original = Algoritmo_De_Descifrado (Clave_de_cifrado,
        Algoritmo_de_cifrado (Información_Original, Clave_de_cifrado) )
```

Algoritmos de clave simétrica

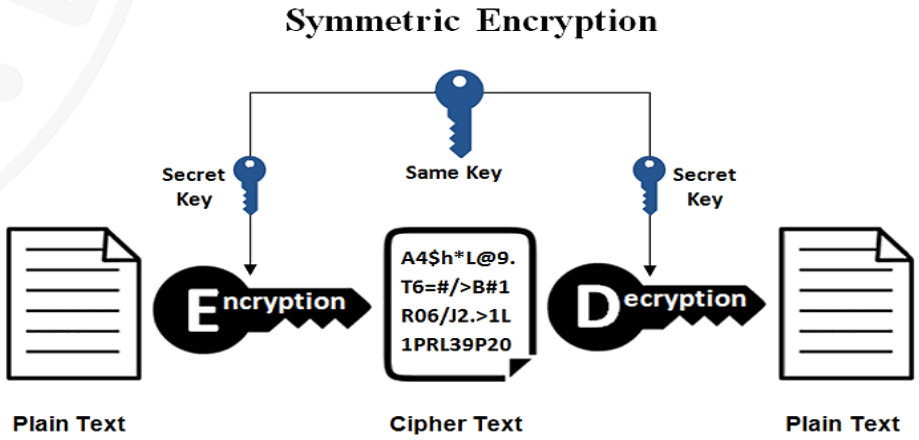
Cifrando solo con una clave

*Iconos por Bing Chat AI

Fuente: Bing Chat AI

Algoritmos de clave simétrica (aka Algoritmos de clave privada)

- **La función de descifrado es la inversa de la de cifrado**
  - Estos algoritmos están diseñados para maximizar el "**efecto avalancha**"
    - Solo cambiar un bit en la entrada produce un gran cambio en la salida (cuanto más, mejor)
  - Se usan funciones matemáticas para dificultar el descifrado
    - Funciones no lineales, funciones XOR... recuerda CN y Álgebra
    - Están diseñados utilizando un "núcleo" mínimo de funciones, que se repiten varias **rondas** (rounds)
    - Las rondas son parametrizables, aumentando la seguridad y el coste computacional (+ rondas -> + CPU)
- **El emisor y el receptor comparten la misma clave K**
  - Esta clave no se usa internamente "tal cual"; se usan "subclaves" generadas por el algoritmo
    - Favorece el "efecto avalancha"
  - Tienen el **problema de la distribución de la clave**
    - Los emisores y receptores deben compartir K
    - Debe **transmitirse, protegerse y almacenarse de forma segura**
    - ¡Cualquiera con una copia de K puede descifrar la información!



Fuente: https://cryptobook.nakov.com

ALGORITMOS DE CLAVE SIMÉTRICA SEGUROS

- Un algoritmo se considera **computacionalmente seguro** si un atacante no puede romper su cifrado más rápido que un ataque de fuerza bruta
  - ¡Llevaría más tiempo que probar todas las posibles claves!
  - Lo que puede llevar millones de años, dependiendo del tamaño de la clave ...
- Ejemplos de algoritmos de cifrado por bloques considerados seguros son
  - **AES** (**Advanced Encryption Standard**) (2001)
    - Adoptado como **estándar de cifrado** por el gobierno de EEUU en 2006, reemplazando a DES y 3DES
    - Divide los datos en bloques de 128 bits y utiliza claves de 128, 192 o 256 bits
    - Tiene varios **modos de operación**: AES-GCM-SIV, AES-GCM, AES-CTR y AES-CBC
    - Actualmente no existe ningún ataque práctico conocido que permita a alguien sin la clave leer datos cifrados, cuando AES se implementa correctamente (**es un cifrado fuerte**)
  - **IDEA** (**International Data Encryption Algorithm**) **NXT**
  - **Twofish**
  - **Serpent**

AES se usa en todas partes para cifrado simétrico

PayPal

Detalles técnicos
Conexión cifrada (TLS_AES_256_GCM_SHA384, claves de 256 bits, TLS 1.3)
La página que está viendo fue cifrada antes de transmitirse por Internet.

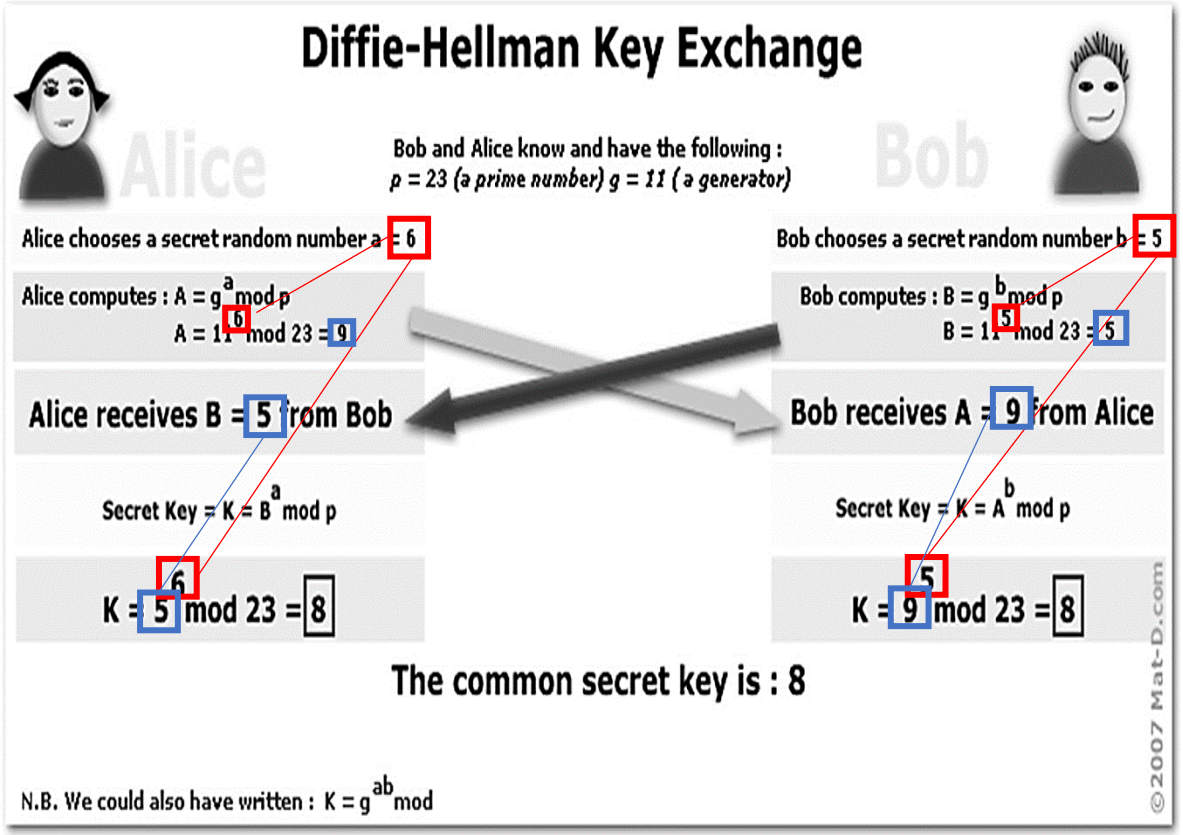
Google

Detalles técnicos
Conexión cifrada (TLS_AES_128_GCM_SHA256, claves de 128 bits, TLS 1.3)
La página que está viendo fue cifrada antes de transmitirse por Internet.

➢ Ejercicio **L3B2_SYMCRYPT**, **L3B2_SYMDECRYPT**, **L3B2_CIPHMODES**

Resolviendo el problema de la distribución de claves: Algoritmo de intercambio de claves de Diffie-Hellman

- **Establece una clave simétrica entre 2 máquinas para cifrar las comunicaciones**
  - Usable en **canales de comunicación inseguros**
  - La clave resultante no puede ser obtenida por un atacante, ¡incluso si captura los mensajes!
- **Utiliza los pasos siguientes**
  - Una máquina elige dos números (p, g) y los envía a la otra máquina de manera **pública**
  - Cada uno elige otro número **secreto** <p
  - Usando una fórmula matemática, cada máquina realiza una serie de operaciones
    - **Con los dos números públicos y el secreto**
  - Esta es la clave secreta usada para cifrar el mensaje
- **Revertir el cálculo para obtener la clave secreta compartida es muy caro**
  - **Si la clave es lo bastante grande**: **algoritmo seguro**



Fuente: https://stackoverflow.com/questions/28015433/is-it-possible-to-hack-diffie-hellman-by-knowing-the-prime-number-and-the-gene

¿Cuánto es "bastante grande"?: Las implementaciones reales usan nºs de 2048 bits (en 2016, la NSA recomendó 3072+ bits)

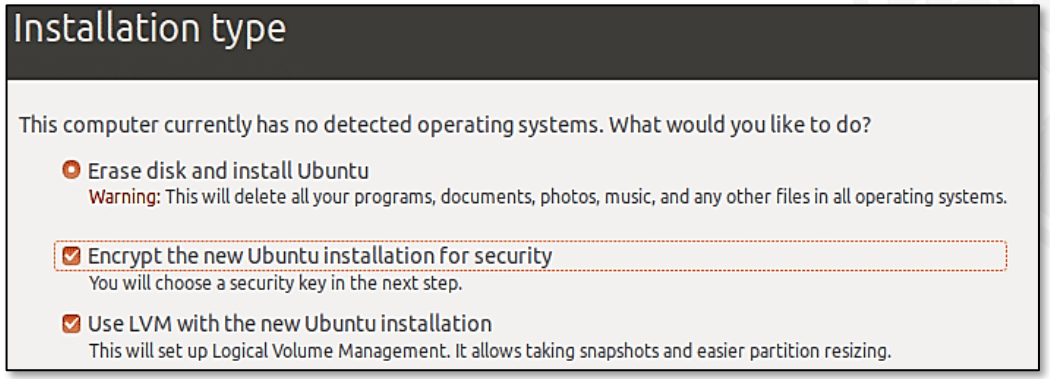
http://www.slideshare.net/samehelhakim/introduction-to-vpn-47256821

Resolviendo el problema de la distribución de claves: Forward Secrecy

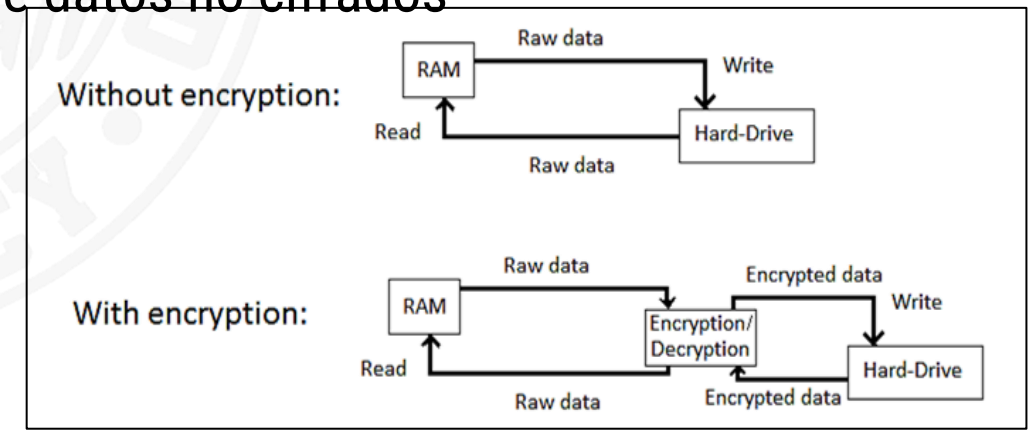
- **Es una propiedad de los algoritmos que deben intercambiar claves que dota de más seguridad a las comunicaciones de la siguiente forma**
  - Imagina que guardas gran cantidad de tráfico cifrado de una comunicación entre dos entidades
  - Ahora imagina que consigues de alguna forma **la clave de cifrado simétrica usada** para hacerla
  - Tendrías acceso a **todas las comunicaciones** futuras...¡y a las pasadas!
- **Si el algoritmo de cifrado tiene la propiedad de forward secrecy, un compromiso de una clave de cifrado solo afecta a esa sesión de comunicación comprometida**
  - Es decir, esta clave **no te va a servir para descifrar** tráfico pasado (o futuro) entre ambas entidades
  - Porque se genera **una clave simétrica única** a intercambiar para cada sesión de comunicación
  - Eso implica que hay que tener cuidado con el algoritmo simétrico elegido
    - No porque sea débil, sino por el impacto de un compromiso de una clave de cifrado
  - Por ejemplo, **GPG** (usado en gran cantidad de sistemas de email cifrado) **no lo tiene**
    - Si te comprometen tu clave de cifrado podrían leerte todo tu correo ☹
    - https://alexgaynor.net/2017/apr/26/forward-secrecy-is-the-most-important-thing/
  - **OpenSSL** (librería que implementa el cifrado usado en Internet) **sí lo tiene**

Aplicaciones prácticas: Cifrado de disco

- **Tecnología para cifrar la información, evitando el acceso no autorizado**
  - El ejemplo de uso más común es la prevención de fugas de datos debido al robo físico del disco duro
  - Los datos de un volumen cifrado no se pueden leer (descifrar) sin la contraseña correcta
- **Se usa software / hardware de cifrado de clave simétrica para cifrar cada bit**
  - Todo el sistema de archivos de un volumen o disco está cifrado
  - Nombres, contenidos y metadatos de archivos / carpetas
- **Es comúnmente transparente**
  - También se llama cifrado en tiempo real o **cifrado sobre la marcha** (OTFE)
  - Los datos se cifran o descifran automáticamente a medida que se cargan o guardan
  - Los usuarios no notan ninguna diferencia con el manejo de datos no cifrados



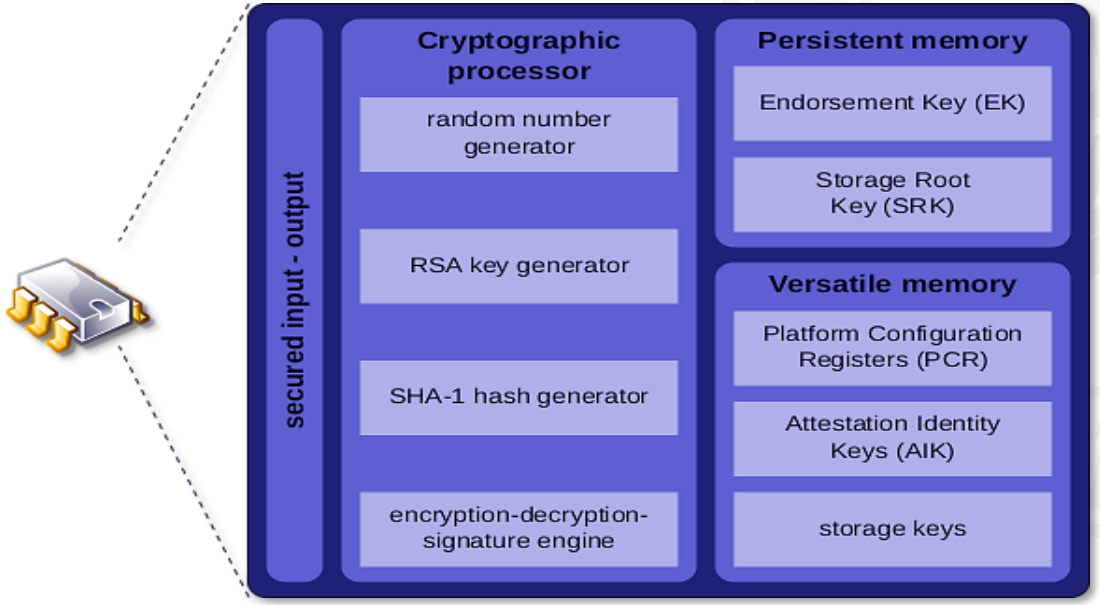
Fuente:
https://superuser.com/questions/448965/does-full-disk-encryption-on-ssd-drive-reduce-its-lifetime



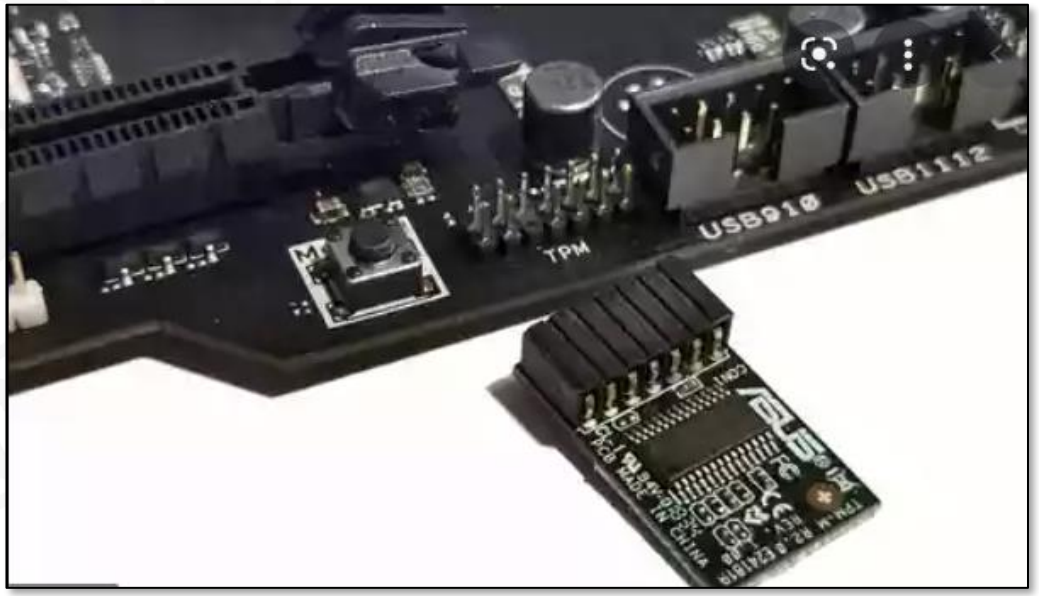
➢ Ejercicio **L3B2_DISKCIPH**

Aplicaciones prácticas de los algoritmos de clave simétrica: Autenticación de plataforma

- **Trusted Platform Module (TPM) es un procesador criptográfico seguro integrado en algunas placas base (o CPU) usado para autenticar un dispositivo hardware**
  - Cada chip TPM es único para cada dispositivo, por lo que puede autenticar la plataforma
  - Puede verificar si el sistema que accede a un dispositivo está autorizado
    - ¡"Trasplantar" una unidad de disco cifrada para leerla no funciona!
  - También es un requisito para instalar Windows 11 hoy en día



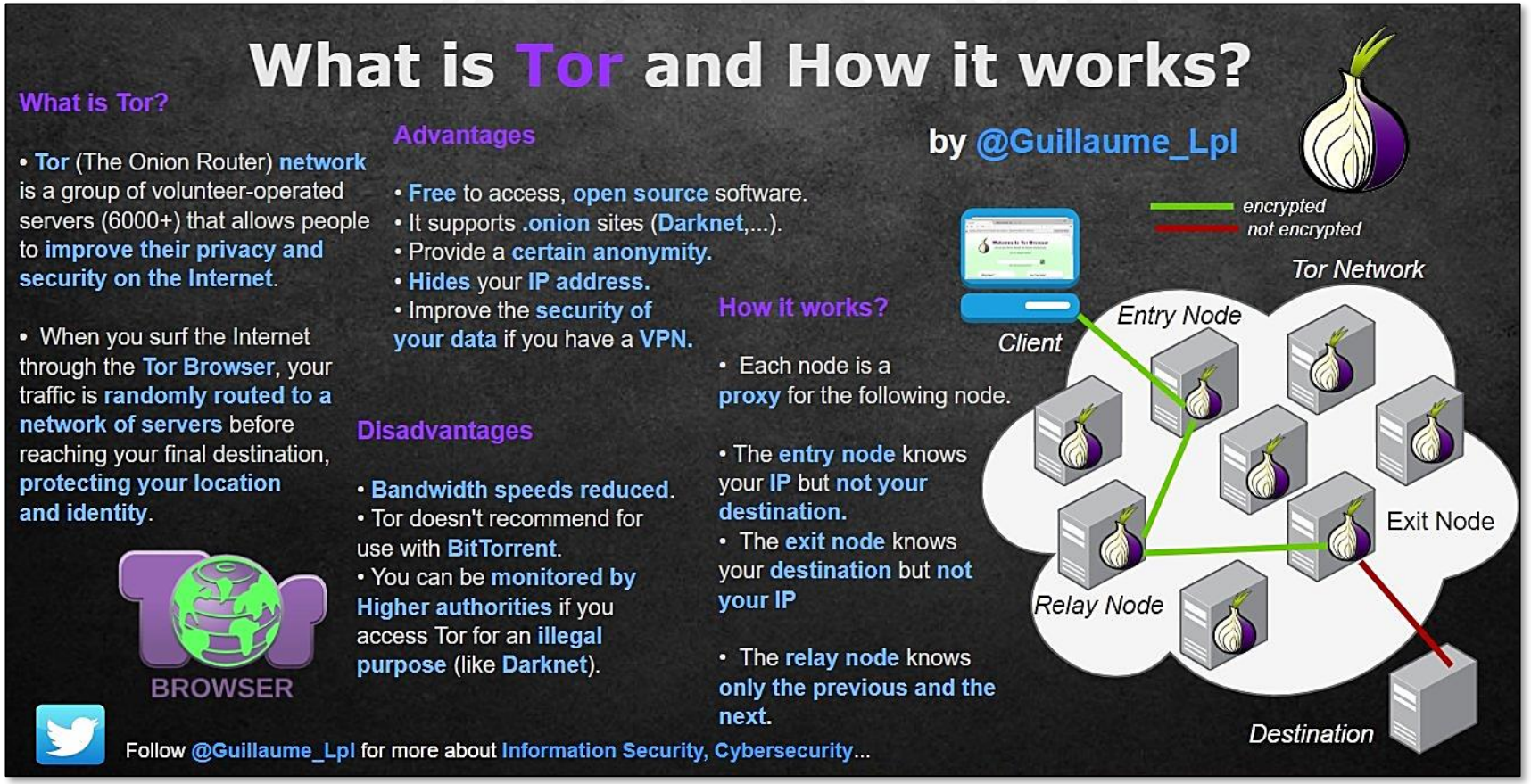
Fuente: https://sergioprado.blog/introduction-to-tpm-trusted-platform-module/



Fuente: https://hardzone.es/reportajes/que-es/trusted-platform-module-tpm/

Aplicaciones prácticas de los algoritmos de clave simétrica: Onion routing (transmisiones anónimas cifradas)

- Los nodos de red de anonimización **ToR** intercambian claves simétricas (AES-128 en 2021) entre ellos, utilizando **Diffie-Hellman**
  - https://blog.insiderattack.net/deep-dive-into-tor-the-onion-router-6de4c25beba7?gi=2acdf11e842c
  - https://linuxconfig.org/install-tor-proxy-on-ubuntu-20-04-linux



Algoritmos asimétricos o de clave pública

Cifrando con un par de claves por usuario

Fuente: Bing Chat AI

*Iconos por Bing Chat AI

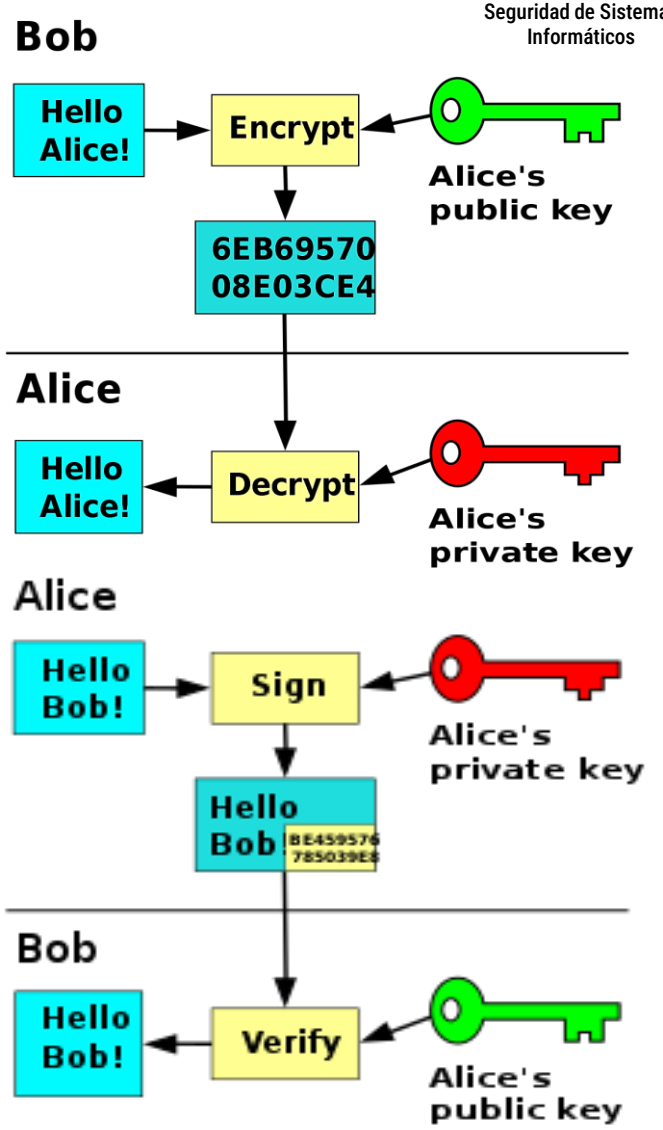
Algoritmos asimétricos (o de clave pública)

- **No tienen el problema del intercambio de claves de los cifrados simétricos**
  - Remitentes y receptores no necesitan acordar una clave, cada uno tiene su propio **par de claves**
  - **Las claves del par (DK, EK) se generan conjuntamente, no individualmente**
  - **Cada par es único para cada emisor / receptor**
  - Una clave del par es **privada** (DK, clave de descifrado) y **debe mantenerse en secreto**
  - La otra clave del par es **pública** (EK, clave de cifrado) y **se manda a todos los destinatarios posibles**
- **De esta forma**
  - Si un usuario desea **recibir mensajes cifrados**, envía la clave **pública** EK al remitente
    - O el remitente lo obtiene del receptor
    - Y almacena **de forma privada su clave de descifrado DK**
  - La información **cifrada con la clave privada solo se puede descifrar con la clave pública correspondiente** de su par, **y viceversa**
  - **Recuerda**: cada par de claves particular **solo se puede generar una vez**
  - **La clave privada no se puede inferir a partir de su clave pública correspondiente del par**
    - No hay peligro en la transmisión de claves públicas a través de la red

➢ Ejercicio **L4B1_KEYPAIR**

Algoritmos asimétricos (o de clave pública)

- **Bob usa la clave pública de Alice para enviarle información cifrada**
  - Entonces se asegura de que el destinatario sea realmente Alice
  - La información solo se puede descifrar con la clave privada de Alice
    - Si la clave privada de Alice descifra la información recibida, entonces debe haber sido cifrada por la clave pública de Alice
    - Un usuario U2 no puede leer (descifrar) mensajes cifrados destinados a U1
      - ¡A menos que U2 robe la clave privada de U1!
  - Por el contrario, los datos cifrados por la clave privada de Alice solo pueden ser descifrados por la clave pública de Alice
    - Por lo tanto, autentican a su remitente
    - Si la clave pública de Alice se puede obtener de manera fiable
- **RSA y ElGamal son algoritmos populares de este tipo**



Fuente: https://en.wikipedia.org/wiki/Public-key_cryptography

¡CLAVES CRIPTOGRÁFICAS Y CONTRASEÑAS DE USUARIO NO SON LO MISMO!

- Una clave criptográfica **NO ES como una contraseña**
- **Una clave simétrica AES**, por ejemplo, se expresa como un **string** hexadecimal de distintos tamaños
  - En función del nº de bits de la clave (**strength**)
    - https://www.ibm.com/docs/en/imdm/12.0?topic=encryption-generating-aes-keys-password
  - Ej.: AES-256
    - `salt=D495560961CCCFE0` (**un texto que se añade** a lo que se cifra para asegurarse aleatoriedad en el resultado, aunque la entrada sea la misma)
    - `key=4D92199549E0F2EF009B4160F3582E5528A11A45017F3EF8` (la clave en sí)
    - `iv=35B2FF0795FB84BBD666DB8430CA214E` (Un valor aleatorio usado para empezar a generar la clave)
- **Los pares de claves asimétricas** se pueden escribir en formato texto (**armored**) (mira las imágenes)
  - Como ves, son **GRANDES** ☺

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: BCPG v1.38

mQGiBEk5VoARBAC69sz9r6gm6fG7Y1rC4xxLOBVeQfTzp/3vXQGfnVOE13+1MNye
aQ4CITZ1/955HdHZvoDcF/IXEr+vhT7gMEMyHfGOdOaSQVJ2ObzeyTOYjrkOvzP6
5zJIOceJICzTsB1FsnWj5DZpTU4ycidc9cV+/lx1aiJLjXXShCqbMNq4RQCggx9m
6GxGERCzxP1rT2AyQpGoUdcykwMDAgo+Q1c6XYz7jVUQJuNZCYQoDovO7SbDW8Tu
O9/ZvEiU36YxqZw953K4zyVvlY8PB4f4AqlbO/MRQv14LG93OrreXmOa8c3iEBLg
Y1u8JYnEL3Tmkrb6xWfseDxaWwYsbvlD5QQAjfcazqjGpTxWYkaxb5WGeDu4Abzs
sDmYq1qGAAvMtAJFu5sW8FO+SahSdfAOpIyOaX6unlUXXIw6NoDI8vQLYGKjM7Om
sXwqlucY5SO/H+41PrV71LrHBWw+Xdutb+pPnFm61Q5unUxItkfYOOtgadDQvNuo
mco93MZv5AOmvzMEALl87JHmhmIOOfq8jZGfl1eWyeCz4SwNBgHBvC/tjWcKjpZE
fak1upvELMbjGvneQaXJtAXGnCUfDOuhZ8rBOkC9G/7EQJCmFy5iAIiSE2+NsJTZ
gn8K8swVEQxTcOk4y3ARP9UOzOSASgjOILPzf8FA+lChjyztcxypJE12imvFiGEE
GBECACEFAkk5VooFCQlopH4HCwkIBwIDBAMWAQIFFQIBAwUChAEACgkQBsNOcLRJ
/JJ4WwCeJ8sgSXKNDEN4BsB4oC9jfZSAQocAn31RINEpxXlPA6MY1463j/U3oaSQ
=OTAo
-----END PGP PUBLIC KEY BLOCK-----
```

Ejemplo de clave publica PGP. Fuente: https://docs.tibco.com/pub(mftis/8.3.0/doc/html/GUID-AB77E7DF-E9D8-4463-9582-19105FEE91A1.html

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lQOBBGC7U48RCACdbrfel9gzWMTvCDepXwpiCRlLyoUnJy0uCTpGPO+t0mQ1OUlD
Ahv1UpS6FSrqZOnMiM+QIOCTAJrzN+p6YFrvTcvQO9khsFzGgrGby29ZSTKzgOEa
Lr4aXv4IC0aON9splxb0340LDfrm261KL63g1cYP1RkWYw/Ch+dTGsq2Zjk21FEC
pS9kt9ZRMUPzMJJaCAQETkDJz+Mr3QEw+O5omb14C4HGUogCSatbv7gbpydp4SKf
J2B1zdNoTKKuQHcaScCHMElaNJbH1Du2Ri8/iiDeA0/U93JKbLpFmGZ/tsrBUxTv
dKk2v9jtu3QnYaBOX97Xf0HRpPQYXPFkd2bvAQCQFr3vNupu82PseR7kcIIdj7kd
JE9ZLiUujgD33MoknQf/SShh2UWK3Eaewq2ISHzMoZcGxUWHvK//ljZ/H9lnEI5t
sqxDfvAkDcmTRmbng4+D+V+k2GEP1a4ZtRa/KPSiZaEoP22NqS+uzNxoqcLgkRUd
```

Ejemplo de clave privada PGP. Fuente: https://blogs.sap.com/2021/06/10/step-by-step-procedure-generate-pgp-keys-and-end-to-end-iflow-to-encrypt-and-decrypt-with-signatures

RSA (Rivest, Shamir y Adleman, 1997)

- **El origen de los sistemas de clave pública**
  - Las claves se basan en dos números primos (`p` y `q`)
  - La clave pública es un par (`n`, `f(p,q)`), siendo `n = p*q`
  - La clave privada es un par (`n`, `g(p,q)`)
- **Es lento** (si se compara con los simétricos)
  - El cifrado / descifrado de bloques de datos requiere muchas operaciones con números muy grandes
- **Es inviable obtener la clave privada de la pública**
  - **La única forma de atacar este sistema es factorizar** `n`
    - Obtener qué dos números se han multiplicado para obtener `n`
    - La factorización **necesita mucha computación**
    - Las claves de 2048 bits se consideran actualmente seguras
    - Muchas organizaciones recomiendan migrar de RSA de 2048 bits a RSA de 3072 bits o más
      - Especialmente ahora, cuando hay investigadores que afirman haber roto RSA-2048 con un computador cuántico construible con la tecnología actual
  - Sin embargo, si es posible, es mejor migrar a la **criptografía de curva elíptica**
    - Esto garantiza una mejor resistencia a futuros ataques con ordenadores cuánticos y una mayor eficiencia

Elliptic-Curve Cryptography (ECC) (Criptografía de curva elíptica)

- **Criptografía de clave pública basada en la estructura algebraica de curvas elípticas sobre cuerpos finitos**
  - Es un concepto matemático que no revisaremos en este curso
  - https://en.wikipedia.org/wiki/Elliptic-curve_cryptography
- **Las curvas elípticas son aplicables para hacer acuerdos de claves, firmas digitales, generadores de nºs pseudoaleatorios y otras tareas**
  - **Ej.: Diffie-Hellman de curva elíptica (Elliptic-Curve Diffie–Hellman , ECDH)**
    - Variante del protocolo de intercambio de claves Diffie-Hellman, pero usando criptografía de curva elíptica
  - También se pueden utilizar para el cifrado
    - Combinando el acuerdo de clave ECC con un esquema de cifrado simétrico
  - Existen diferentes tipos de curvas elípticas para ser utilizadas con este tipo de algoritmos ECC
    - Como las curvas **secp256r1** y **X25519** (https://en.wikipedia.org/wiki/Curve25519) creadas para ser usadas con ECDH

Elliptic-Curve Cryptography (ECC): Ejemplo

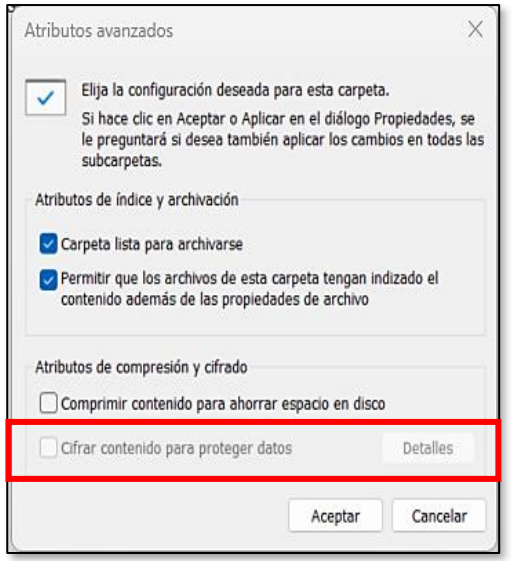
- **Los certificados (que se verán más adelante) utilizados para establecer conexiones SSL/TLS utilizan criptografía asimétrica**
- **Hay dos tipos**
  - **Los que usan RSA para sus claves y los que usan criptografía ECC**
  - Los ECC requieren **claves más pequeñas** que los no ECC para alcanzar el mismo nivel de seguridad
  - Esto significa que **las claves ECC son más rápidas y fáciles de manejar** que las RSA equivalentes
    - ¡Y escalan mejor!
  - Por lo tanto, **deberíamos utilizar soluciones basadas en ECC**

Tamaño de clave ECC	Tamaño equivalente con RSA
160 bits	1024 bits
224 bits	2048 bits
256 bits	3072 bits
384 bits	7680 bits
521 bits	15360 bits

Fuente: https://reanimandowebs.com/articulos-sobre-seguridad-web/certificado-ssl-tls/

Algoritmos asimétricos: Aplicaciones prácticas

Cifrado de archivos y directorios



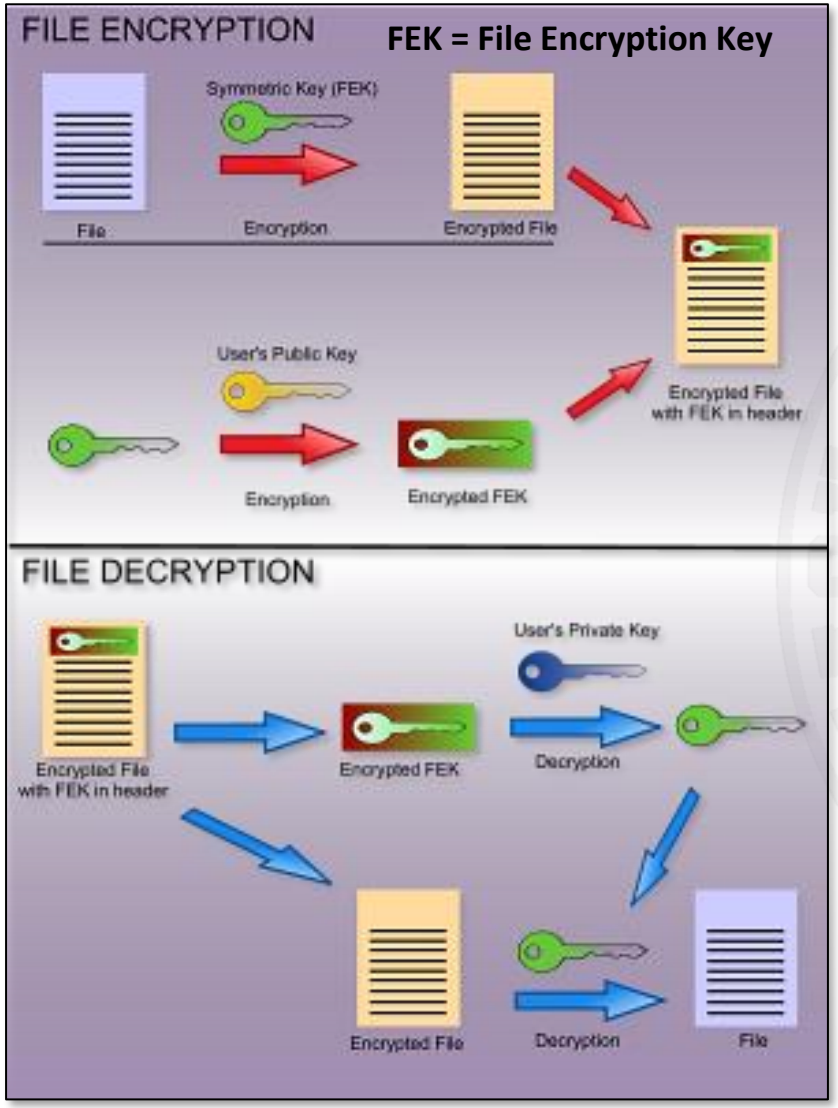
- **Archivos / directorios individuales son cifrados** por el sistema de archivos
  - Se denomina file-based encryption (FBE) o cifrado de archivos/carpetas
  - A diferencia del cifrado de disco completo, **donde se cifra toda la partición o disco**
- Los **tipos de cifrado** a nivel de sistema de archivos incluyen
  - Sistemas de archivos criptográficos '**apilables**', en capas sobre el principal
  - Sistemas de archivos de uso general con cifrado añadido
- Las **ventajas** del cifrado a nivel de sistema de archivos son
  - **Gestión flexible de claves basada en archivos**: cada archivo se puede cifrar con una clave independiente
  - **Gestión individual de archivos cifrados**: Ej.: copias de seguridad incrementales de archivos modificados individualmente incluso cifrados, en lugar de copia de seguridad de todo el volumen cifrado
- **Combina criptografía de clave pública y criptografía de clave simétrica**
  - El control de acceso se puede aplicar mediante criptografía de clave pública

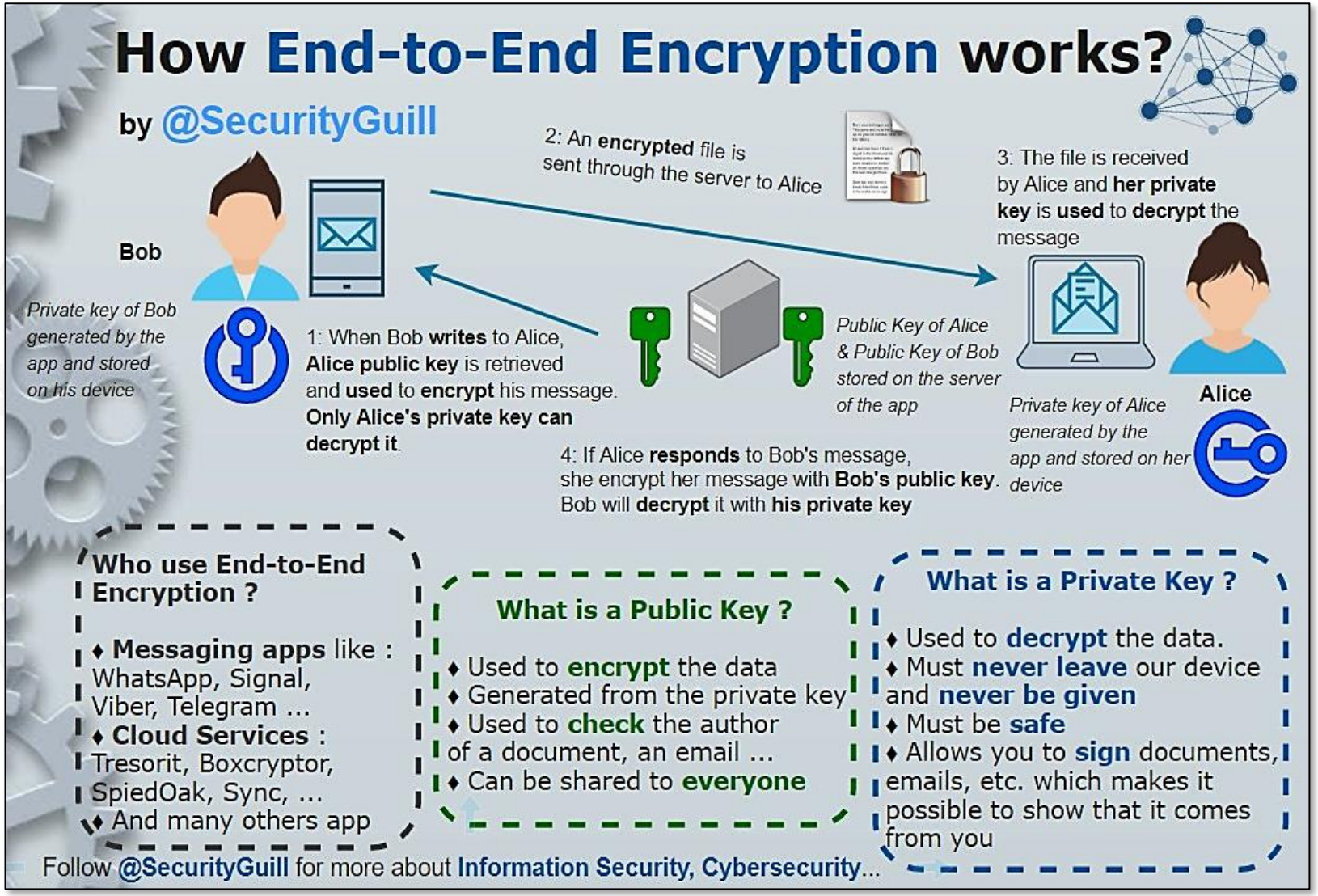
Cifrado de extremo a extremo

Fuente: WhatsApp

- Comunicaciones seguras entre usuarios, típicas de muchas aplicaciones de mensajería y en la nube

ALGORITMOS ASIMÉTRICOS: APLICACIONES PRÁCTICAS





Fuente: https://en.wikipedia.org/wiki/Encrypting_File_System

Usado por WhatsApp, Telegram, Signal...

¿Crees que lo has entendido todo? ¡Autoevalúate!

- **Eres capaz de hacer lo siguiente**
  - *Entender qué es el cifrado por bloques y de flujo*
  - *Entender la diferencia entre un cifrado con y sin clave*
  - *Comprender qué principios hacen una buena clave criptográfica y cómo se mide su fortaleza*
  - ***Cifrado simétrico***
    - *Ser consciente de que, si se conoce la clave criptográfica usada, toda la seguridad del cifrado se pierde y que, por tanto, distribuir la clave es el mayor problema del mismo*
    - *Tener claro qué algoritmo simétrico deberías usar si tienes que tomar una decisión*
    - *Comprender qué es el algoritmo de intercambio de claves **Diffie-Hellman***
    - *Entender el concepto y la importancia del **Forward Secrecy** en el cifrado simétrico*
    - *Ser capaz de identificar aplicaciones comunes del cifrado simétrico en tu día a día*
  - ***Cifrado asimétrico***
    - *Entender claramente qué clave se distribuye a otros y cual no del par de claves*
    - *Entender las especiales propiedades de un par de claves publica, privada, la relación entre ellas a la hora de cifrar/descifrar información y que no es posible obtener una a partir de la otra*
    - *Comprender el aspecto de una clave criptográfica al uso*
    - *Entender por qué el cifrado ECC supone una ventaja sobre el cifrado "tradicional"*
    - *Ser capaz de identificar aplicaciones comunes del cifrado asimétrico en tu día a día*

< Ir al Índice

INTEGRIDAD

¿Ha modificado alguien esto?

*Iconos por Bing Chat AI

Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

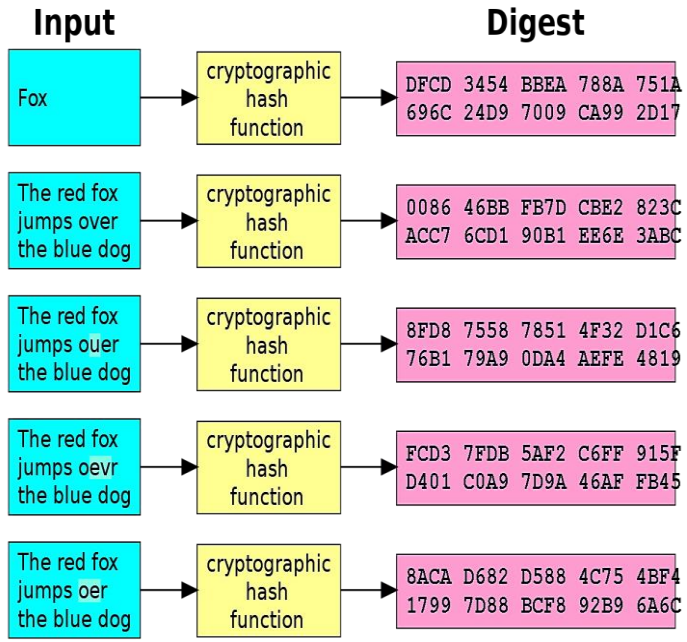
¿QUÉ TE VAS A ENCONTRAR EN ESTE BLOQUE?

En este bloque aprenderás

- Qué es una **función hash**, sus propiedades principales y para qué sirve
- Las **principales aplicaciones** actuales de una función hash
- Qué es una **KDF** y para qué se usa, así como la importancia de lo que hacen

FUNCIÓN HASH O RESUMEN

- **Función matemática que transforma un conjunto de datos M (texto, archivo...) en una salida H de longitud fija (n)**
  - Por lo tanto, un mensaje de entrada de longitud variable **M** se representa por una salida de **longitud fija** (n) de la función Hash(M), llamada **H**
  - H se denomina **resumen criptográfico, huella digital o resumen**
- **Propiedades**
  - **Fácil de calcular**: es computacionalmente barato obtener H=Hash(M)
  - **Unidireccional**: Es imposible obtener el mensaje M a partir de H
  - **Compresión**, **difusión** y **no previsibilidad**
    - Favorecen el "efecto avalancha"
    - Una salida de longitud fija implica que **varios mensajes M diferentes pueden generar el mismo valor H=Hash(M)**
    - Estas 3 propiedades aseguran que sea **muy difícil predecir** qué mensajes tienen el mismo hash
    - Y además, que cambiar un solo bit en M genera una H muy diferente



El tamaño de la entrada cambia, ¡pero el de la hash siempre es el mismo!

Fuente:
https://es.wikipedia.org/wiki/Funci%C3%B3n_hash_criptogr%C3%A1fica

Función hash o resumen

- **Primera resistencia a la preimagen**
  - Si conoces un valor hash H, encontrar o crear un mensaje que lo genere debe ser muy difícil
  - Depende del tamaño n de la H calculada por el algoritmo hash que utilicemos
- **Segunda resistencia a la preimagen** (resistencia a colisiones simple y fuerte)
  - **Simple**: Es computacionalmente difícil que, dada una M, podamos encontrar una M' que **Hash(M)=Hash(M')**
    - Hacer variaciones de M no debe generar el mismo hash (efecto avalancha, evita la falsificación de mensajes)
  - **Fuerte**: Es computacionalmente difícil encontrar un par aleatorio (M, M') que **Hash(M)=Hash(M')**
    - Los mensajes aleatorios que generan el mismo hash no deben encontrarse fácilmente (falsificación)

- **Hay muchos algoritmos hash**
  - **MD5** (Message Digest Algorithm, 128 bit) y **SHA-1** (Secure Hash Algorithm, 160 bits): **Inseguros**
  - Los algoritmos **SHA-2 son los que se recomienda**n hoy en día
    - **SHA-256**, **SHA-384** o **SHA-512**
  - **SHA-3**. Estándar NIST actual que podría ser más preparado para el futuro
    - La longitud del hash varía: 224, 256, .., 512 bits (http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf)
  - **BLAKE2** es otro algoritmo hash preparado para el futuro

APLICACIONES DE LOS ALGORITMOS DE HASH

- **Almacenamiento de contraseñas**: aumenta la dificultad de averiguarlas
  - Obliga a los atacantes a usar técnicas de fuerza bruta, computacionalmente costosas
  - Evita obtener fácilmente contraseñas si se roba una BD de contraseñas (ataque común)
  - Almacenar contraseñas en texto plano es señal de que una empresa **no se preocupa por la seguridad**
  - La fuerza bruta puede no ser factible **si la contraseña es lo suficientemente compleja**
  - Se usa una KDF (Key Derivation Function) para garantizar máxima resistencia a ataques (vista luego)

```
gates:$1$vjFpMmig$2yJJhqiYADW0w03tPQZB9.:50:0:99999:7: ::
elon:$1$0QT3fPmk$UHSi7AcfaLvXP49P3gbvb/:100:0:99999:7:::
```

- **Huella digital (fingerprinting) de archivos**: asegura de que un archivo es el original
  - Comprueba que no se han modificado en la transmisión / almacenamiento (troyanizado, dañado...)
  - Ej.: un archivo puede publicar su hash calculado utilizando un determinado algoritmo
    - Se calcula el hash del ejecutable localmente y así se sabe si el archivo ha sido alterado o dañado
    - El simple hecho de modificar un solo bit genera un resultado de hash diferente
- **Firmas digitales**: asegura que un archivo fue creado por quien dice (visto luego)
- **Blockchain**: Usa SHA-256 para detectar modificaciones fraudulentas de la cadena

Fingerprinting de archivos: Comprobando hashes de archivos

- **Los hashes se pueden usar para detectar modificaciones no autorizadas del software**
- **Los Host-Based Intrusion Detection Systems (HIDS, como Tripwire o AIDE) precalculan y almacenan de forma segura los hashes de los archivos críticos del sistema**
  - También comprueban periódicamente estos hashes
  - La falta de coincidencia entre el hash almacenado y el calculado puede significar
    - **Una nueva versión del ejecutable**: actualización de software; el hash almacenado debe ser recalculado
    - **Alteración maliciosa de archivos**: hay un malware en el sistema
- **Los ejecutables pueden comprobar su propio hash al iniciarse, pero es menos seguro**
  - El código que realiza la comprobación puede ser deshabilitado por un malware
  - El hash correcto debe almacenarse / obtenerse de un sitio remoto, y puede ser manipulado
  - **Microsoft SmartScreen utiliza este enfoque**: el SO analiza el archivo, evalúa su "reputación" y envía un hash a un servicio en la nube para decidir si se ejecutará (o advierte al usuario sobre posibles peligros)

```
6de6036ef0dc8a908e4cc248ef1d8aab87172e722d8c5bad9e137fd43994e0fe  ./netboot/debian-installer/amd64/grub/x86_64-efi/parttool.lst
13886a4e494c33bcf387210f6c6910bc4a84db81d931b63ecc2f61ad722fb483  ./netboot/debian-installer/amd64/grub/x86_64-efi/parttool.mod
7aa3c84b739ed7920e21de6ef6013c50e797adb4aa5d86e92cceeac3536bb0e5  ./netboot/debian-installer/amd64/grub/x86_64-efi/password.mod
3c0ff52b79422033b3aa3153be7f67473dcec3e34155c1c725daba8abb96a6ec  ./netboot/debian-installer/amd64/grub/x86_64-efi/password_pbkdf2.mod
dc9bdda70583365f2cc7e63417d1e056f876067cb3bf67fa43c5ae51f112ea36  ./netboot/debian-installer/amd64/grub/x86_64-efi/pata.mod
dd91bce28a6e9b7c801e38ed2a8ab9ce1aed8970b9880054c22f438a16f9d30f  ./netboot/debian-installer/amd64/grub/x86_64-efi/pbkdf2.mod
5a6300a6b576afa0a96685571c0af13bd69041b248a7494db825e083dfa0106c  ./netboot/debian-installer/amd64/grub/x86_64-efi/pbkdf2_test.mod
50eec78eb440928415493c0c59cf11ce2d909c582b3beea58c590894f437fd32  ./netboot/debian-installer/amd64/grub/x86_64-efi/pcidump.mod
```

Fuente:
https://deb.debian.org/debian/dists/bullseye/main/installer-amd64/current/images/SHA256SUMS

Guardar claves: Key Derivation Function (KDF)

- **Técnicas típicas para adivinar contraseñas a partir de datos almacenados**
  - **Fuerza bruta pura**: cubrir todo el espacio de claves posible y normalmente con un coste prohibitivo
  - **Diccionarios** (EDI) de claves potenciales que pueden corresponder a datos almacenados
    - Comprobamos menos valores, pero consume más memoria y puede no tener éxito (la clave no está en los diccionarios usados)
  - **Tablas Rainbow**: estructuras de datos que permiten almacenar grandes cantidades de contraseñas
    - Consume menos memoria, pero requiere más potencia computacional que los diccionarios simples
- **¿Podemos resistir estos ataques?**
  - Sí, pero no usando algoritmos hash simples
  - ¡Sino las funciones de derivación de claves especializadas (Key Derivation Functions, KDF) que mencionamos antes!

➢ Ejercicios **L3B3_SSLCRACK**, **L3B3_PASS**, **L3B3_BRUTE**

Guardar claves: Key Derivation Function (KDF)

- **Los KDF obtienen una o más claves de un valor secreto**
  - El valor secreto es la **clave maestra**, la contraseña real que introduce el usuario ☺
  - Su salida se utiliza para almacenar contraseñas
    - Hace más computacionalmente costosas las técnicas de adivinación de contraseñas
  - ¿Cómo? Usando técnicas de "**key stretching** " para mejorar la seguridad de los datos almacenados
    - **Sal** (salt, o semilla): datos aleatorios concatenados a las contraseñas de texto sin formato
      - Se utiliza para dificultar los ataques basados en diccionarios o tablas rainbow
      - Las salts aseguran que una misma contraseña no genere siempre el mismo hash
      - Único para cada contraseña, y normalmente almacenado junto con el hash de la contraseña
      - Un **pepper** (o **secret salt**) también es un secreto agregado a una entrada durante el hashing
        - Pero se almacenan por separado en un medio externo, como un módulo de seguridad de hardware
    - **Iteraciones/rondas**: aplicar la función hash varias rondas (>=1)
      - Obtener el texto original con fuerza bruta es más costoso, retrasando cada intento

```
DK (Clave de derivación: datos que deben almacenarse realmente) = KDF (contraseña o clave
                    proporcionada por el usuario, Salt, Iteraciones)
```

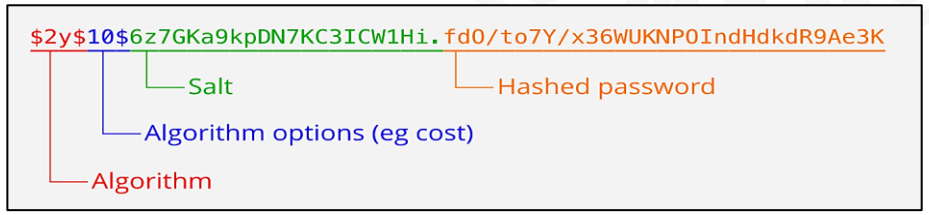
➢ Ejercicio **L4B2_KDFPASS**

Guardar claves: Key Derivation Function (KDF)

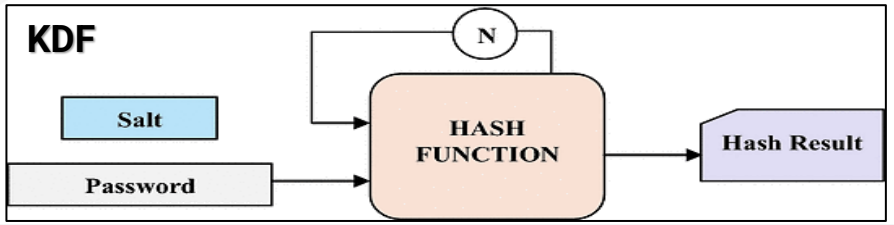
- **La resistencia a este tipo de ataque depende de la longitud de la clave y la cantidad de entropía computacional que podría generarse al generar el DK**
  - Cuanto más, mejor seguridad
  - Esta es la aleatoriedad recopilada por un SO / aplicación para usar en criptografía, entre otros usos
- **Algoritmos KDF modernos conocidos**
  - **PBKDF2**: longitud de 160 bits, utiliza salt e iteraciones recomendadas de 1000 a 10000000
    - Utilizado por redes Wifi, Truecrypt, Veracrypt, el gestor de arranque GRUB, Android...
  - **scrypt**: obliga a usar mucha memoria, por lo que implementar ataques contra sus valores cuesta más
    - Utilizado como prueba de trabajo en algunas criptomonedas
  - **bcrypt**: El algoritmo de hash de contraseñas de varias distribuciones de Linux, usa salt e iteraciones
  - **Argon2id**: Cuenta con tres versiones específicamente parametrizadas contra diferentes ataques

Fuente: https://stackoverflow.com/questions/40993645/understanding-bcrypt-salt-as-used-by-php-password-hash

Bcrypt stored string example



Fuente: https://en.bitcoinwiki.org/wiki/Key_derivation_function



Guardar claves: Ejemplo práctico de Key Derivation Function (KDF)

- **Esta imagen muestra el contenido del archivo `/etc/shadow`**
  - Las contraseñas de Linux se almacenan en él
  - `*` significa que el usuario no tiene una cuenta interactiva (bash): es una **cuenta de un servicio**
  - ¡Los usuarios `vagrant`, `redondo`, y `vinuesa` tienen la misma contraseña! ("`test123…`")
- **¿Por qué su hash es tan diferente?**
  - **Recuerda**: pepper y salt; la misma contraseña no genera el mismo hash
    - Si roban el hash de `vagrant`, no sabrán la password de `vinuesa` automáticamente, ¡aun siendo la misma!
    - Si se roba `/etc/shadow` y se rompe una hash, no sabrán inmediatamente las contraseñas de otros usuarios que puedan tener la misma contraseña
    - ¿Entiendes la importancia de almacenar contraseñas así para minimizar los efectos de una fuga de datos?
    - ¿Y `redondo`? Usa un algoritmo de hash más débil (y más corto), pero siguiendo los mismos principios

```
vagrant:$6$zGFNTxdh$RZWHA/hxHzu/CNAnZyJxBiKGZB0UpZwU0OpwDSpI/vOOGm8hgArmO1FC9/xLV6pe6xLvuGB5EJ.WyzguLukDY1:18983:0:99999:7:::
vboxadd:!:18123::::::
rtkit:*:18983:0:99999:7:::
usbmux:*:18983:0:99999:7:::
pulse:*:18983:0:99999:7:::
redondo:$1$c26Kbia1$d/7sXVp9im42R/w.oqU2U/:18983:0:99999:7:::
vinuesa:$6$Ikh3d1YG$RyOzFnaoxtqAZMqLyy3a0ftVI.B5EkUEHCeGSuJxNM2u5FIy46XYQ1IeeLR4BTlwhydw11sfnwwRUjqVsPM.5/:18988:0:99999:7:::
```

Romper una password puede ser imposible si es lo suficientemente compleja...

Fuente:
https://www.reddit.com/r/dataisbeautiful/comments/322lbk/time_required_to_bruteforce_crack_a_password/

Time until *guaranteed* brute-force password crack

Based on attacker's guesses per second vs. password strength

Formula: (*Seconds to guaranteed crack*) = ($2^{(entropy)}$) ÷ (guesses per second)

Result then converted from *seconds* to more reasonable units of time such as years

Note: By definition, it takes **half** of the *guaranteed* crack time on *average* to crack a password

Length of password approximately equivalent to a given entropy

Note: First three columns are rounded to a max of +/- 0.2
Fourth column is rounded to a max of +/- 0.05

Entropy (in bits) Entropy = $\log_2(S^L)$ L is pass length S is symbol pool (*i.e. if your password has lowercase and numbers, your **symbol pool** is 26+10=36*) (*If entropy confuses you, just focus on the rest of the chart*)	Lowercase (26 symbols, 4.7 bits ea)	UPPER + lower + 0-9 (62 symbols, 5.95 bits ea)	UPPER + lower + 0-9 + special characters (94 symbols, 6.55 bits ea)	Pass **phrase** with an average of **4.3** letters per word (12.93 bits per word) Column's value = words in phrase	Attacker's brute force guesses per second: 10,000	1,000,000	1,000,000,000	100,000,000,000	1,000,000,000,000	100,000,000,000,000	Entropy (in bits) Entropy = $\log_2(S^L)$ L is pass length S is symbol pool (*i.e. if your password has lowercase and numbers, your **symbol pool** is 26+10=36*) (*If entropy confuses you, just focus on the rest of the chart*)
120		20			∞	∞	∞	421 quadrillion years	42 quadrillion years	421 trillion years	**120**
118	25		18		∞	∞	∞	105 quadrillion years	11 quadrillion years	105 trillion years	118
116				9	∞	∞	∞	26 quadrillion years	2.6 quadrillion years	26 trillion years	116
114		19			∞	∞	658 quadrillion years	6.6 quadrillion years	658 trillion years	6.6 trillion years	114
112	24		17		∞	∞	165 quadrillion years	1.6 quadrillion years	165 trillion years	1.6 trillion years	112
110					∞	∞	41 quadrillion years	411 trillion years	41 trillion years	411 billion years	**110**
108	23	18			∞	∞	10 quadrillion years	103 trillion years	10.3 trillion years	103 billion years	108
106			16		∞	∞	2.6 quadrillion years	26 trillion years	2.6 trillion years	26 billion years	106
104	22			8	∞	643 quadrillion years	643 trillion years	6.4 trillion years	643 billion years	6.4 billion years	104
102		17			∞	161 quadrillion years	161 trillion years	1.6 trillion years	161 billion years	1.6 billion years	102
100					∞	40 quadrillion years	40 trillion years	402 billion years	40 billion years	402 million years	**100**
98	21		15		∞	10 quadrillion years	10 trillion years	100 billion years	10 billion years	100 million years	98
96		16			251 quadrillion years	2.5 quadrillion years	2.5 trillion years	25 billion years	2.5 billion years	25 million years	96
94	20				63 quadrillion years	628 trillion years	628 billion years	6.3 billion years	628 million years	6.3 million years	94
92			14		16 quadrillion years	157 trillion years	157 billion years	1.6 billion years	157 million years	1.6 million years	92
90	19	15		7	3.9 quadrillion years	39 trillion years	39 billion years	392 million years	39 million years	392 millennia	**90**
88					981 trillion years	9.8 trillion years	9.8 billion years	98 million years	9.8 million years	98 millennia	88
86			13		245 trillion years	2.5 trillion years	2.5 billion years	24.5 million years	2.5 million years	25 millennia	86
84	18	14			61 trillion years	613 billion years	613 million years	6.1 million years	613 millennia	6.1 millennia	84
82					15.3 trillion years	153 billion years	153 million years	1.5 million years	153 millennia	1.5 millennia	82
80	17				3.8 trillion years	38 billion years	38 million years	383 millennia	38 millennia	383 years	**80**
78		13	12	6	958 billion years	9.6 billion years	9.6 million years	96 millennia	9.6 millennia	96 years	78
76	16				239 billion years	2.4 billion years	2.4 million years	24 millennia	2.4 millennia	24 years	76
74					60 billion years	599 million years	599 millennia	6 millennia	599 years	6 years	74
72		12	11		15 billion years	150 million years	150 millennia	1.5 millennia	150 years	1.5 years	72
70	15				3.7 billion years	37 million years	37 millennia	374 years	37 years	4.5 months	**70**
68					935 million years	9.4 million years	9.4 millennia	94 years	9 years	34 days	68
66	14	11	10		234 million years	2.3 million years	2.3 millennia	23 years	2.3 years	8.5 days	66
64				5	58 million years	584 millennia	584 years	5.8 years	7 months	2.1 days	64
62	13				14.6 million years	146 millennia	146 years	1.5 years	1.8 months	13 hours	62
60		10	9		3.7 million years	37 millennia	37 years	4.4 months	13 days	3.2 hours	**60**
58					913 millennia	9.1 millennia	9 years	33 days	3.3 days	48 minutes	58
56	12				228 millennia	2.3 millennia	2.3 years	8.3 days	20 hours	12 minutes	56
54		9			57 millennia	571 years	6.8 months	2.1 days	5 hours	3 minutes	54
52	11		8	4	14.3 millennia	143 years	1.7 months	12.5 hours	1.3 hours	45 seconds	52
50					3.6 millennia	36 years	13 days	3.1 hours	19 minutes	11 seconds	**50**
48	10	8			892 years	8.9 years	3.3 days	47 minutes	4.7 minutes	2.8 seconds	48
46			7		223 years	2.2 years	20 hours	12 minutes	1.2 minutes	0.7 seconds	46
44					56 years	6.7 months	4.9 hours	2.9 minutes	18 seconds	0.18 seconds	44
42	9	7			14 years	51 days	1.2 hours	44 seconds	4.4 seconds	0.044 seconds	42
40			6	3.1	3.5 years	13 days	18 minutes	11 seconds	1.1 seconds	0.011 seconds	**40**

Entropy (in bits) Entropy = $\log_2(S^L)$ L is pass length S is symbol pool (*i.e. if your password has lowercase and numbers, your **symbol pool** is 26+10=36*) (*If entropy confuses you, just focus on the rest of the chart*)

Lowercase (26 symbols, 4.7 bits ea) | UPPER + lower + 0-9 (62 symbols, 5.95 bits ea) | UPPER + lower + 0-9 + special characters (94 symbols, 6.55 bits ea) | Pass **phrase** with an average of **4.3** letters per word (12.93 bits per word) Column's value = words in phrase

Attacker's brute force guesses per second: 10,000 | 1,000,000 | 1,000,000,000 | 100,000,000,000 | 1,000,000,000,000 | 100,000,000,000,000

Time until *guaranteed* brute-force password crack

Based on attacker's guesses per second vs. password strength

Formula: (*Seconds to guaranteed crack*) = ($2^{(entropy)}$) ÷ (guesses per second)

Result then converted from *seconds* to more reasonable units of time such as years

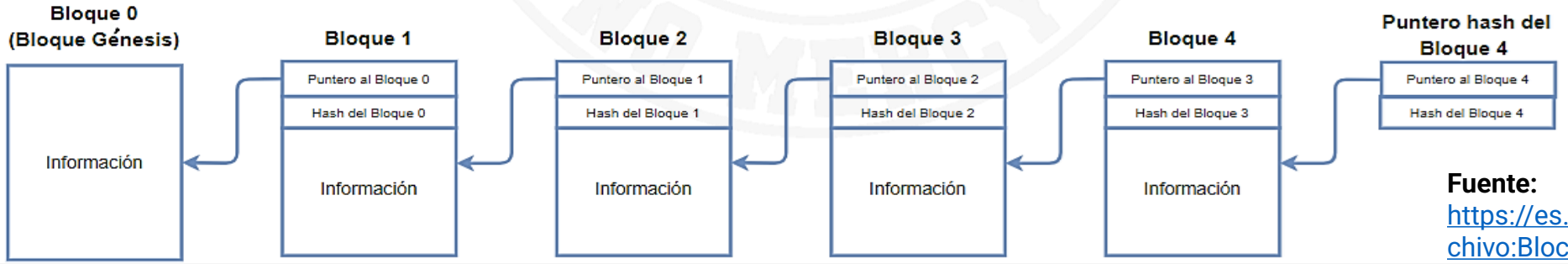
Note: By definition, it takes **half** of the *guaranteed* crack time on *average* to crack a password

Length of password approximately equivalent to a given entropy

Note: First three columns are rounded to a max of +/- 0.2
Fourth column is rounded to a max of +/- 0.05

BLOCKCHAIN

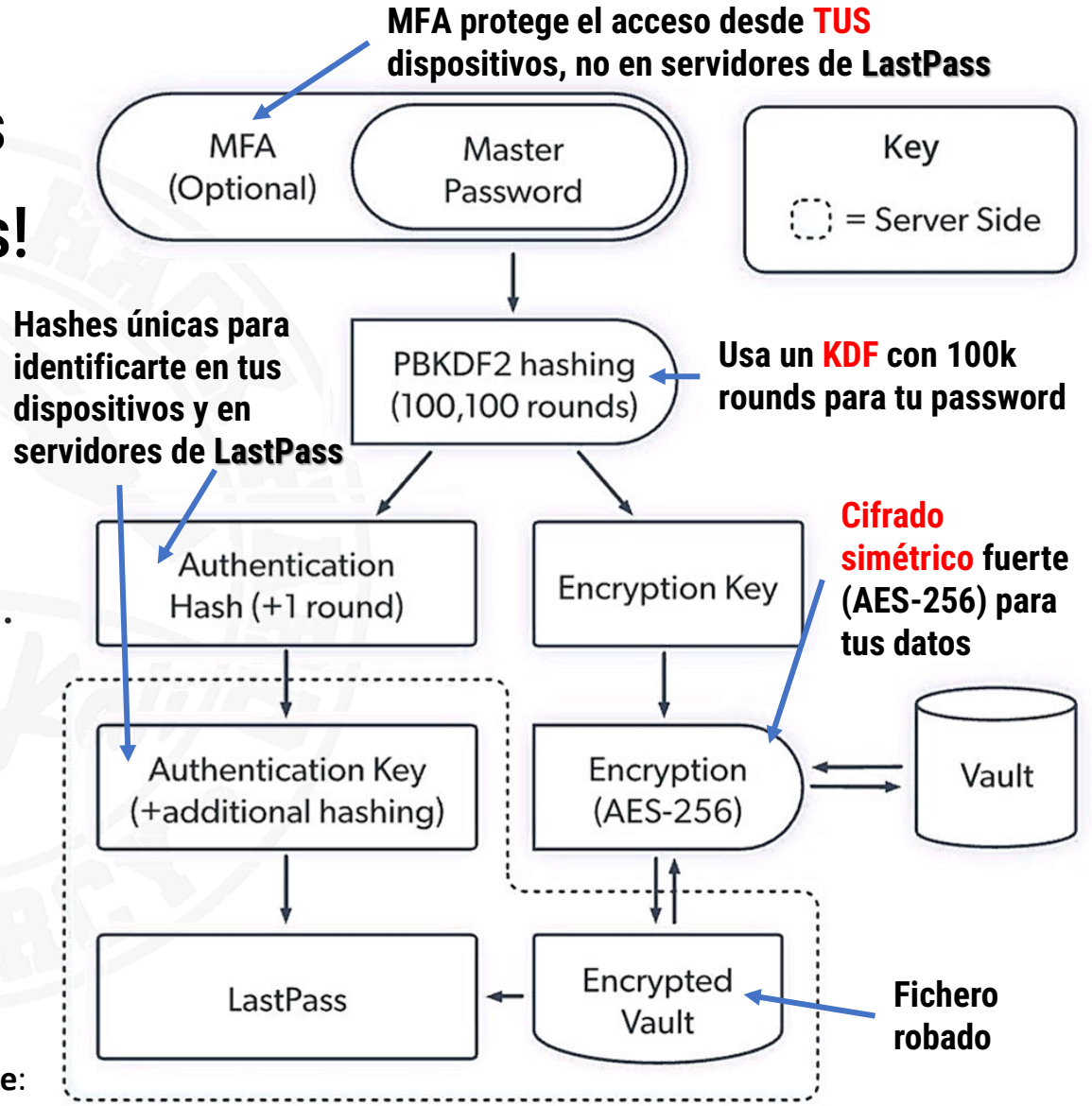
- **El primer bloque (bloque 0) de la cadena se conoce como el bloque génesis**
  - Cada otro bloque almacena un puntero hash al anterior, su valor hash y su información
- **Es imposible cambiar un bloque de la cadena sin ser detectado por los demás**
  - Si un atacante cambia el contenido del bloque 1, debido a la resistencia a la colisión no es factible encontrar otra información que tenga el mismo hash que la que había: el hash del bloque 1 cambia
  - **El bloque 2 ahora podrá detectar este cambio en el bloque 1**, ya que el hash que guarda no coincidirá
  - Para evitar esto, el atacante tendría que cambiar el valor del hash del bloque 1 en el bloque 2
  - Pero eso modificará el hash del bloque 2, por lo que ahora el bloque 3 podría detectar el problema
  - Esta situación se repite a lo largo de toda la cadena, hasta el final
  - Los usuarios guardan el hash del bloque final, por lo que si se modifica detectarán la inconsistencia
- **Por eso blockchain es ideal para almacenar transacciones no falsificables**



Fuente: https://es.wikipedia.org/wiki/Archivo:Blockchain_Hash.png

Ejemplo práctico: La filtración de LastPass de 2022

- En 2022 LastPass informó de que se habían copiado los encrypted vaults de sus usuarios
- ¡Eso potencialmente los compromete a todos!
  - Los atacantes pueden averiguar master password débiles **offline por fuerza bruta**
    - https://www.tomshardware.com/news/eight-rtx-4090s-can-break-passwords-in-under-an-hour (¡**288.500.000.000 hashes/s!**)
  - ¡Sabrán todas las passwords de sus usuarios!
  - **Que tendrán que cambiar TODAS ellas**, por si acaso...
- LastPass usó bien el cifrado…
  - KDF con muchos rounds, AES-256, no se guardan las master passwords en el servidor, hashing...
- ¡Pero falló en autenticación!
  - Alguien accedió a ficheros de otros usuarios
  - ¡Aunque esté cifrada, es información sensible!



Fuente: https://www.linkedin.com/feed/update/urn:li:activity:7012507887449116672/

¿Crees que lo has entendido todo? ¡Autoevalúate!

?

- **Eres capaz de hacer lo siguiente**
  - *Tener claro que el tamaño de una hash solo depende del algoritmo que uses para crearla, y no del tamaño de entrada*
  - *Entender claramente las principales propiedades de una hash, especialmente su unidireccionalidad y que, a igual entrada, igual hash*
  - *Entender que las hashes tienen una serie de propiedades que hacen que cambien mucho tan solo por cambiar un bit de la entrada*
  - *Comprender por qué las hash son ideales para guardar de forma segura contraseñas y para comprobar que un archivo no se ha modificado*
  - *Entender qué es un KDF y para que se usa, así como las operaciones que hacen que sean la forma ideal de hash para guardar claves*
  - *Comprender los distintos métodos usados para averiguar claves y la protección contra ellos*
  - *Entender los principios básicos de funcionamiento de una blockchain y como usa las hash para ello*

< Ir al Índice

Fuente: Bing Chat AI

Firma digital >

Cifrado Autenticado >

Certificados >

AUTENTICACIÓN

¿Quién creo que eres?

*Iconos por Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿QUÉ TE VAS A ENCONTRAR EN ESTE BLOQUE?

En este bloque aprenderás

- Qué es exactamente una **firma digital**
  - Y el **procedimiento** tradicional de firma digital de un documento
- Para que se usa el **cifrado autenticado** y qué problema resuelve
- Qué es exactamente un **certificado digital** y para qué se usa
- Como se resuelve el **problema de confianza** existente entre un certificado y quien dice que es su propietario
- Los **usos principales** de los certificados digitales

Firmas digitales

¡No es lo mismo que la foto de tu firma en papel! ☺

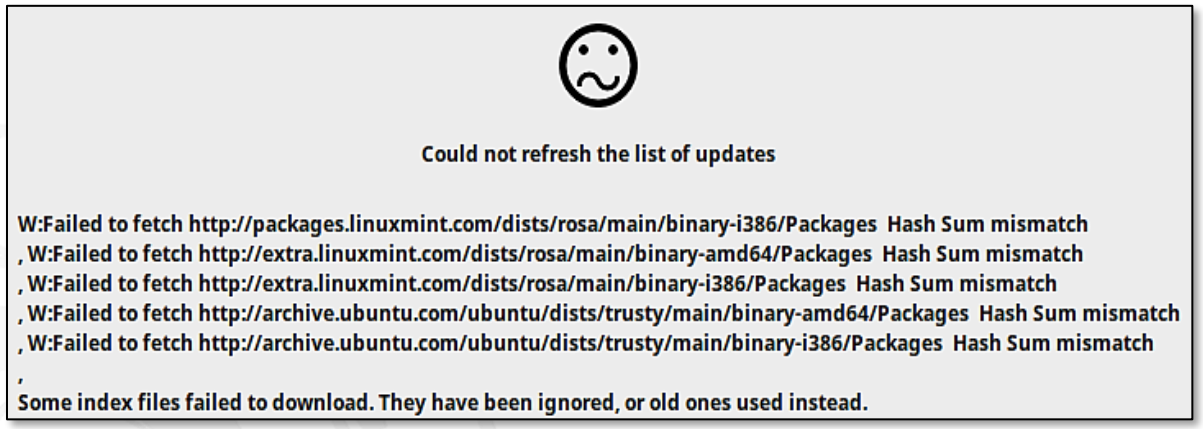
***Iconos por Bing Chat AI**

FIRMAS DIGITALES

- **Resultado de una operación criptográfica que une un documento firmado a un elemento personal, el Certificado Digital**
- **Son el resultado de firmar criptográficamente el hash de un archivo con la clave privada de su autor**
  - Las firmas suelen ser archivos con una extensión `.sig` o `.asc`
- **Un archivo firmado se puede verificar**
  - El hash del archivo original se obtiene (junto con la clave pública del usuario) del fichero firmado
  - Si el documento se ha modificado de alguna manera, este hash será diferente del que se calcule a partir del contenido actual del archivo
  - Se producirá un error al intentar verificar la firma y **se detectará la manipulación de archivos**
- **Por lo tanto, certifican un documento y le agregan una marca de tiempo**
- **Un ejemplo de implementación es el Digital Signature Standard (DSS):**
  - https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

Firmas digitales. Objetivos

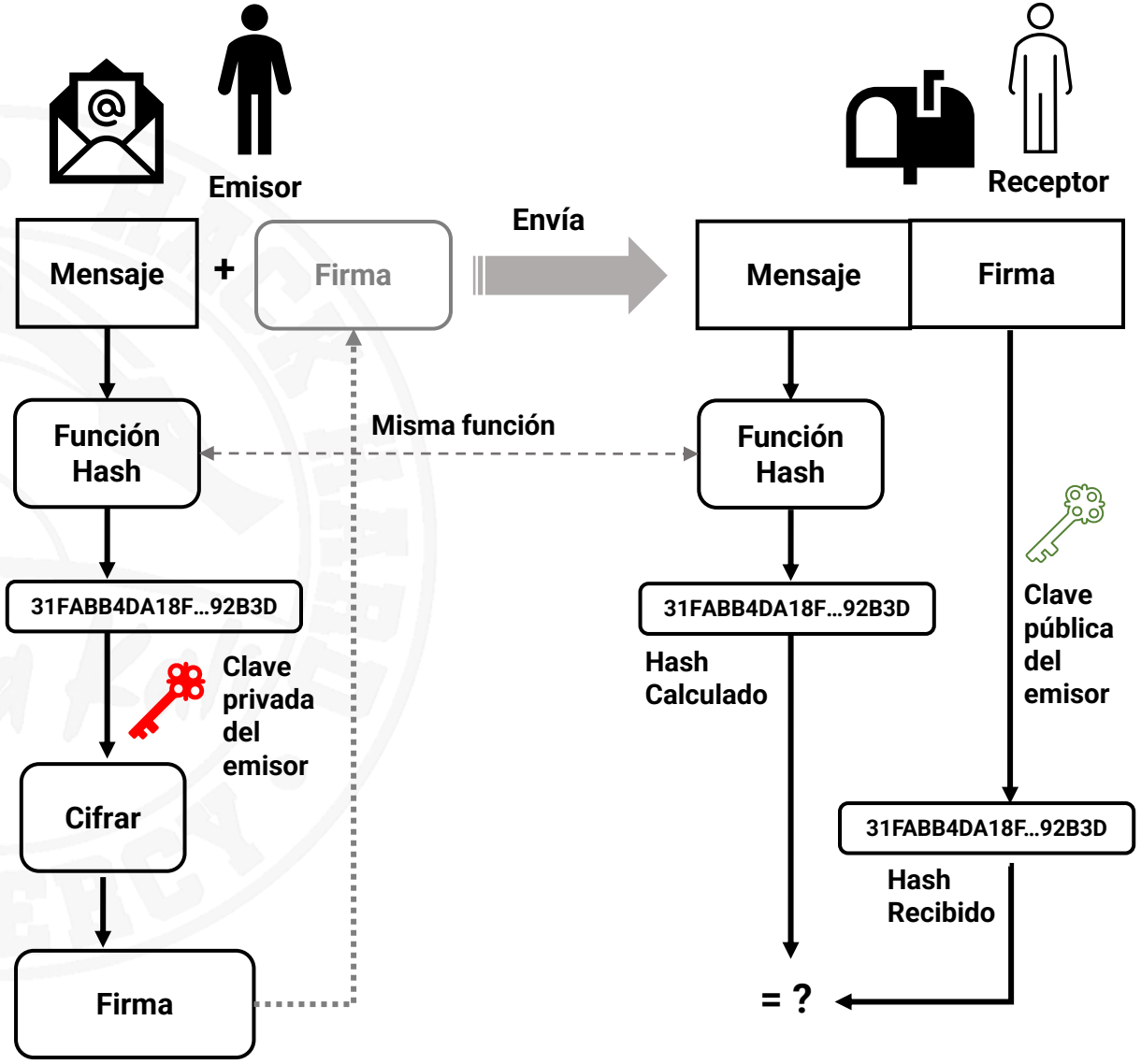
- **Autenticación del origen**
  - La firma debe convencer al receptor de que **el remitente ha firmado deliberadamente el documento**
- **Integridad**
  - Debe asegurarse de que el **documento / mensaje no ha sido modificado**
- **No repudiar**
  - Al garantizar los dos anteriores, **el autor no puede negar el contenido del mensaje**
- **Debe ser imposible de falsificar, de reutilizar y, si el documento se altera, debe poder detectarse**



Fuente: https://www.comoinstalarlinux.com/como-solucionar-problemas-al-instalar-o-actualizar-paquetes-en-ubuntu-o-linux-mint/

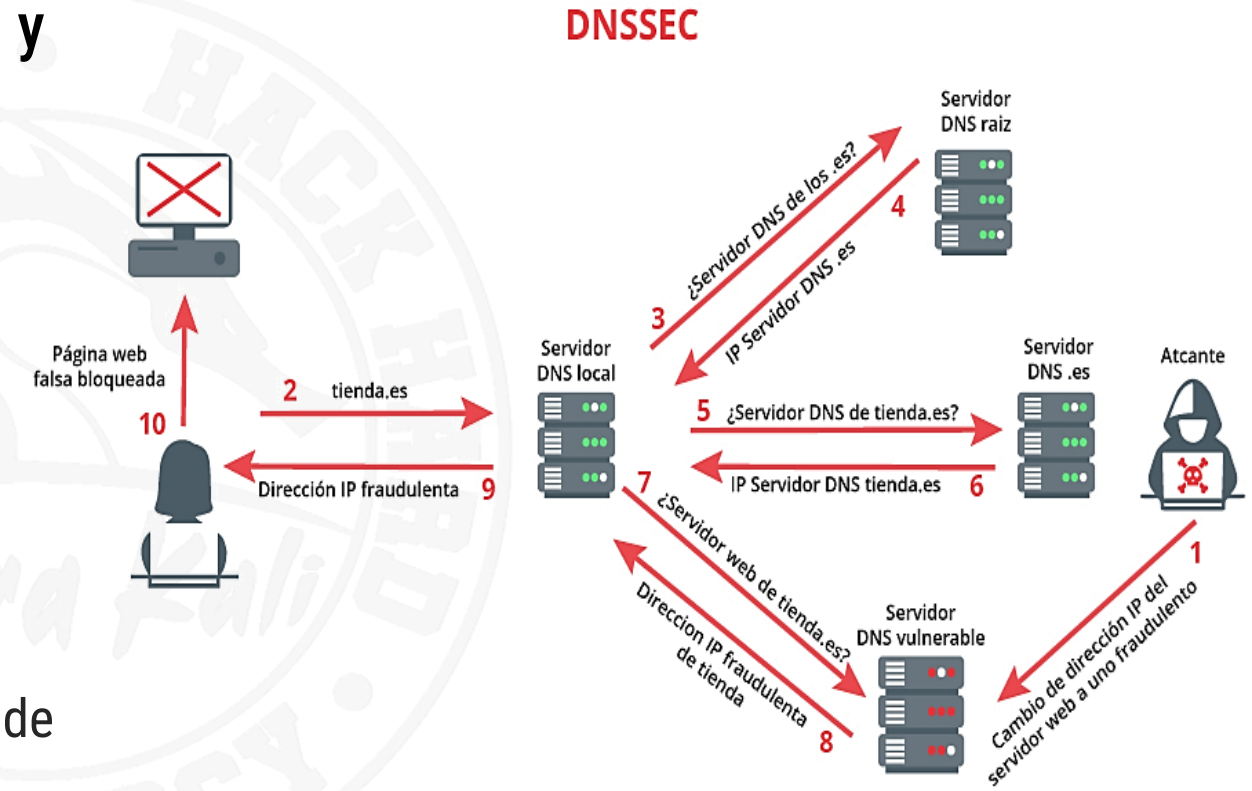
Firmas digitales & funciones hash: Procedimiento de firma de mensajes

- **Un hash de mensaje se calcula y se cifra con la clave privada del usuario**
  - El costo computacional de cifrar un hash es mucho menor que cifrar el mensaje completo
    - El tamaño del hash es fijo, más pequeño que el mensaje, y lo representa de forma única
    - Cifrar el hash equivale a cifrar su mensaje correspondiente
- **El hash se descifra una vez recibido**
  - Con la **clave pública** del usuario
  - El resultado se puede comprobar con el hash calculado del mensaje recibido
- **Si son iguales, el mensaje no ha sido alterado**
  - ¡Y sabemos quién es el remitente!



APLICACIONES DE LAS FIRMAS DIGITALES: DNSSEC

- **El tema 1 describió la "cadena" de resolución DNS**
- **Pero si uno de esos servidores DNS es atacado y sus datos comprometidos...**
  - ¡La IP de cualquier nombre de dominio en Internet podría falsificarse!
  - **Internet no sería fiable** y se desmoronaría ☹
- **DNSSEC utiliza firmas digitales para garantizar que el contenido (direcciones DNS) de los servidores no se altere**
  - Utiliza una "cadena de custodia" que los clientes pueden verificar
  - Cualquier modificación no autorizada de un nombre de dominio será detectada y rechazada
- **Más información**
  - https://www.infoblox.com/dns-security-resource-center/dns-security-solutions/dns-security-solutions-dns-security-extensions-dnssec/



Fuente: https://www.incibe.es/protege-tu-empresa/blog/dnssec-asegurando-integridad-y-autenticidad-tu-dominio-web

Cuando las firmas digitales fallan: El caso de Sony PS3®

- **Los juegos legales en la consola PS3® deben estar firmados por una clave privada de Sony para evitar la corrupción de datos y garantizar la ejecución de juegos auténticos**
  - Las consolas se venden con la clave pública correspondiente en su firmware
  - La consola validará el hash de los archivos del juego con su clave pública, negándose a ejecutar software que no pase la validación
- **Pero esta clave privada fue robada y publicada en Internet en 2011**
  - Y, por lo tanto, cualquiera podía firmar software con ella...
  - ... y la gente podría crear ejecutables capaces de ejecutarse en el hardware de PS3® (homebrew)
  - ¡Sony Mario Bros podría ser una posibilidad! ☺
- **La escena de PS3® era entonces una carrera entre modders y Sony para evitar la ejecución de software no autorizado**
  - Mediante el desarrollo de nuevas comprobaciones de firmware y en la red PSN
- **Conclusión: La publicación de la clave privada causó millones de pérdidas a Sony**

Cifrado autenticado (Authenticated Encryption)

MACs sin manzanas ☺

*Iconos por Bing Chat AI

¿Por qué se usa este tipo de cifrado?

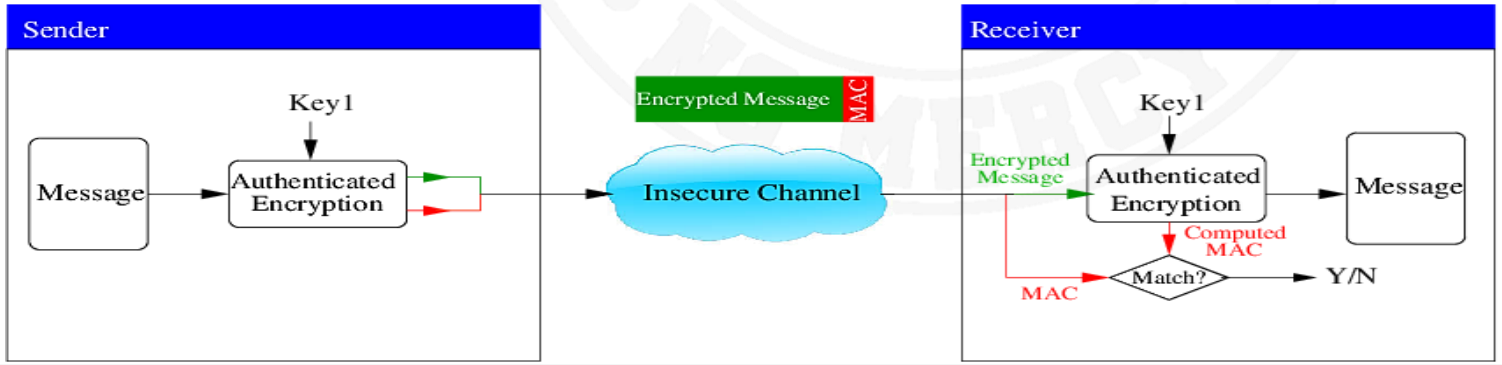
- **Vimos que para garantizar la CIA en una transmisión se usa el cifrado asimétrico**
  - Pero es lento, especialmente al aumentar el volumen de datos transmitidos
- **También que el simétrico es rápido, pero solo garantiza confidencialidad**
  - Los datos cifrados se pueden corromper, y tiene el problema de la distribución de claves
- **Para lograr CIA eficientemente, en el pasado se combinaban ambos cifrados**
  - Pero históricamente eso produjo vulnerabilidades, porque era complejo hacerlo correctamente
- **Por eso se dio un paso más: El cifrado autenticado (Authenticated Encryption, AE)**
  - **Cifrado simétrico** que detecta si el mensaje ha sido alterado (integridad)
  - Usa **una clave compartida** solo por un emisor (E) y receptor (R) concreto, por lo que hay autenticación
    - Solo R puede conocer lo que envía E y viceversa
  - *¿Y si usamos cifrado asimétrico solo para compartir una clave secreta simétrica aleatoria en una conexión de manera segura, y luego AE para el resto del tráfico con dicha clave compartida?*
  - ¡Tenemos **lo mejor de ambos mundos**! Cifrado rápido y sin el problema de la distribución de claves ☺

Cifrado autenticado

- **Por tanto, muchos protocolos modernos de seguridad (TLS, WPA2, WireGuard, Signal…) usan AE**
  - Que opcionalmente puede llevar Authenticated Data (**AEAD**)
    - El Authenticated Data no se cifra, para que el receptor pueda leerlo sin la clave
    - Pero si se altera, **el algoritmo lo detectará**
- **AEAD es un cifrado simétrico pero que además proporciona autenticación**
  - Por tanto, **ambos interlocutores deben conocer una misma clave simétrica compartida**
  - Por ejemplo, el texto cifrado de las transmisiones TLS se construye con AEAD
    - Está compuesto por una cabecera sin cifrar (AD) y un payload cifrado (AE)
    - Los dos están autenticados, pero **solo el payload está cifrado**
    - La cabecera sin cifrar tiene el tipo de contenido y la longitud del texto cifrado, necesarios para descifrar el payload cuando se reciba
- **Más información**
  - https://textbook.cs161.org/crypto/macs.html
  - https://www.cryptography-primer.info/encryption/

MAC (Message Authentication Code)

- **Usar AE o AEAD añade al mensaje un Message Authentication Code o MAC**
  - Un MAC es un **checksum** cifrado del mensaje, enviado junto a él
    - **Un checksum es un tipo de hash,** pero especializado en comprobar la integridad de un mensaje
    - https://www.baeldung.com/cs/hash-code-vs-checksum
  - El MAC se cifra usando una clave simétrica compartida entre ambos interlocutores
  - EL MAC resultante es de longitud fija y pequeña
  - Un MAC asegura que cualquier cambio en el mensaje **hace que el checksum ya no sea válido**
- **Hay 3 formas de crear MACs**
  - **Basado en hash** (HMAC, recomendado), **basado en cifrado por bloques** (CBC-MAC) y **Galois Counter Message Authentication Code-based** (GMAC)



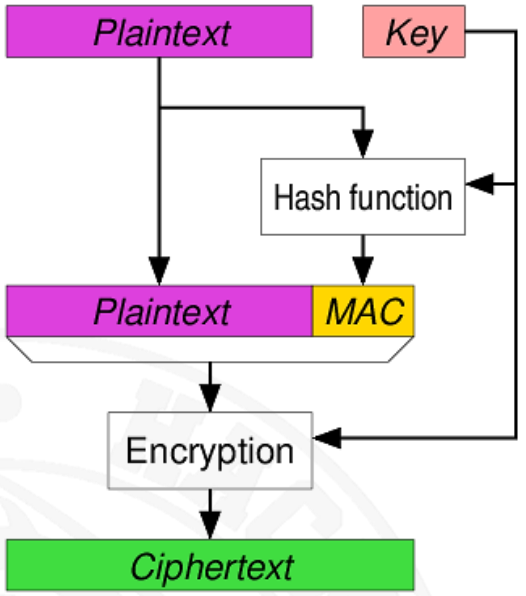
Fuente:
https://www.researchgate.net/publication/279264224_Authenticated_Encryption_on_FPGAs_from_the_Reconfigurable_Part_to_the_Static_Part/figures?lo=1

Cálculo de MAC

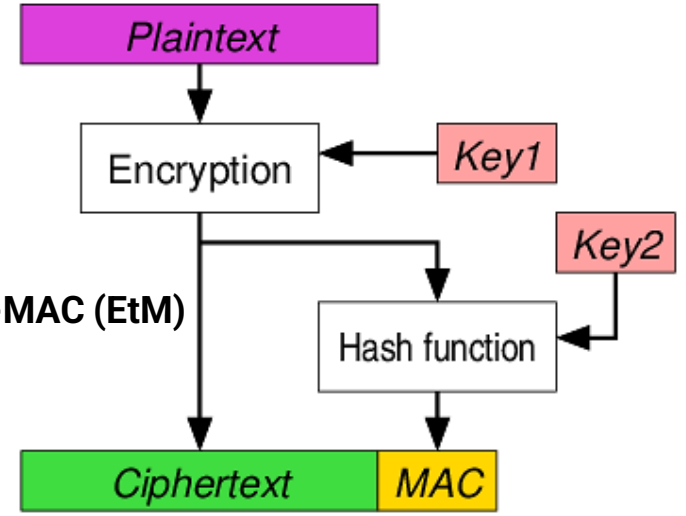
Seguridad de Sistemas Informáticos

- **Comenzamos con 2 usuarios S y R compartiendo una clave secreta K**
  - S crea una MAC a partir del mensaje M con una función f
    - MAC = f(K, M)
    - Es imposible obtener K a partir de MAC y M, **f no es reversible**
  - S envía M+MAC a R
  - R calcula MAC' con K y M
  - Si MAC' = MAC, el mensaje es auténtico
- **Hay tres maneras de hacer esto**
  - Mac-then-Encrypt (MtE): usado por SSL
  - Encrypt-then-MAC (EtM): utilizado por IPSec, el método recomendado
  - Encrypt-and-MAC (E&M): utilizado por SSH

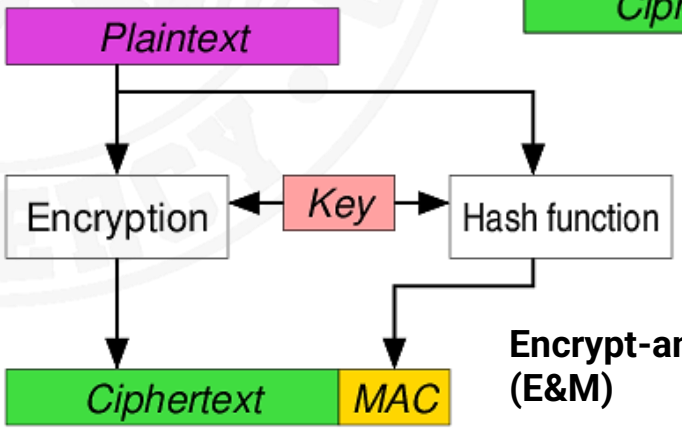
Fuente:
https://en.wikipedia.org/wiki/Authenticated_encryption



Mac-then-Encrypt (MtE)



Encrypt-then-MAC (EtM)



Encrypt-and-MAC (E&M)

¿Crees que lo has entendido todo? ¡Autoevalúate!

?

- **Eres capaz de hacer lo siguiente**
  - ***Firmas digitales***
    - *Entender bien cómo se usan las hashes y el cifrado asimétrico para firmar un documento*
    - *Lo que garantiza una firma de esta clase*
    - *Comprender cómo el firmado permite asegurar la confianza en el funcionamiento de un sistema crítico, como un DNS o un sistema de autenticación de software en un hardware cerrado*
    - *A consecuencia de lo anterior, comprender que un filtrado de una clave privada usada en una firma compromete todo el sistema de firmas*
  - ***Cifrado autenticado***
    - *Entender el problema que resuelve el cifrado autenticado respecto al cifrado simétrico típico*
    - *Entender qué es un MAC*

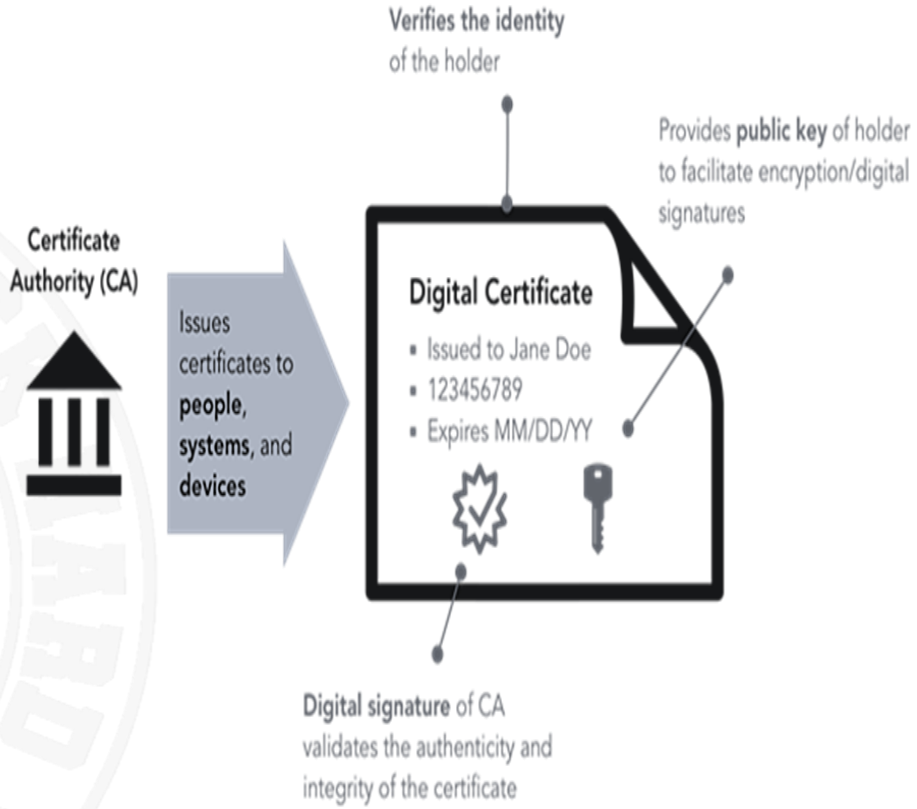
Certificados digitales

Un documento digital para identificarlos a todos

*Iconos por Bing Chat AI

CERTIFICADOS DIGITALES

- **Se utilizan para autenticar claves públicas que se transmiten entre entidades**
  - La criptografía asimétrica no funciona si no podemos garantizar que una clave pública dada pertenece a alguien
- **Un certificado es como una tarjeta de identidad: identifica entidades (máquinas, usuarios...)**
  - Son documentos que contienen **información sobre su propietario y su clave pública**
  - Parte del par de claves de la criptografía asimétrica
- **Por lo tanto, un certificado es la asociación entre una entidad física y una clave pública transmitida**
  - Y validado por una entidad de confianza (Ej. : CA)
- **¿Cómo se garantiza la confianza de la asociación?**
  - Garantizar identidades reales
  - Proporcionar claves que no están dañadas



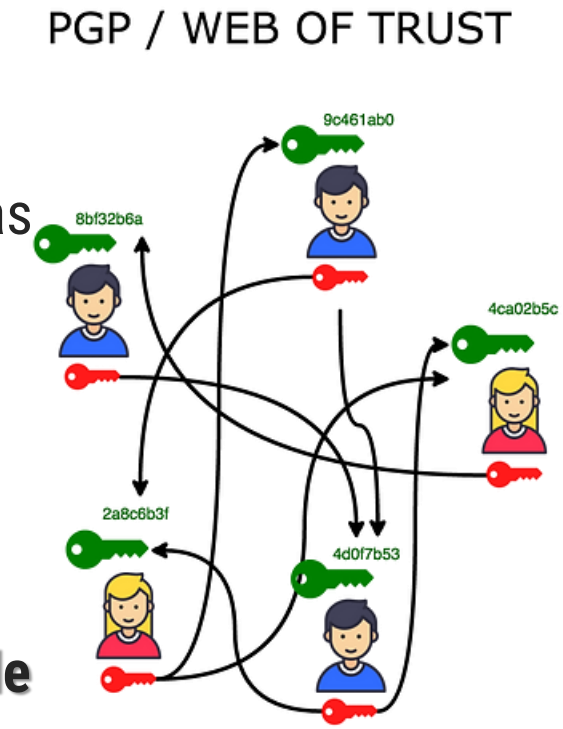
Hay certificados para transmitir claves públicas que solo contienen ésta (Ej.: .crt), pero hay otros que se usan para firmar dentro de una máquina, y que contienen ambas (Ej.: .pfx). Por tanto, no deben transmitirse a un tercero a la ligera porque ¡le estaríamos dando nuestra clave privada!. Fuente: https://id4d.worldbank.org/guide/digital-certificates-and-pki

AUTENTICACIÓN DE CLAVES DE USUARIO

- **Necesitamos un mecanismo adicional (un tercero) para asegurarnos de que el titular de una clave es quien dice ser**
- **Hay dos, y ambos siguen un enfoque basado en la confianza**
  - Una entidad U1 **firma digitalmente** (como se vio antes) los certificados (claves) de otra entidad U2
  - Confiamos en U1, por lo que aceptamos que el certificado U2 es válido (la confianza es "transitiva")
- **Estos mecanismos pueden ser**
  - **Anillos de confianza (Trusted Rings)** (p. ej. GnuPG): **Otros usuarios** confían en los certificados
    - Cuantos más usuarios confíen en un certificado, más "confiable" será
      - Como las publicaciones de Twitter / videos de YouTube ... pero también tiene sus mismos problemas
    - Es un modelo de confianza **descentralizado** y tolerante a fallos
  - **Autoridades de certificación** (**CA**): entidades responsables de establecer la validez de un certificado
    - Funcionan como un **notario**
    - Comprometer o deshabilitar una CA invalida o impide la verificación de sus certificados asociados

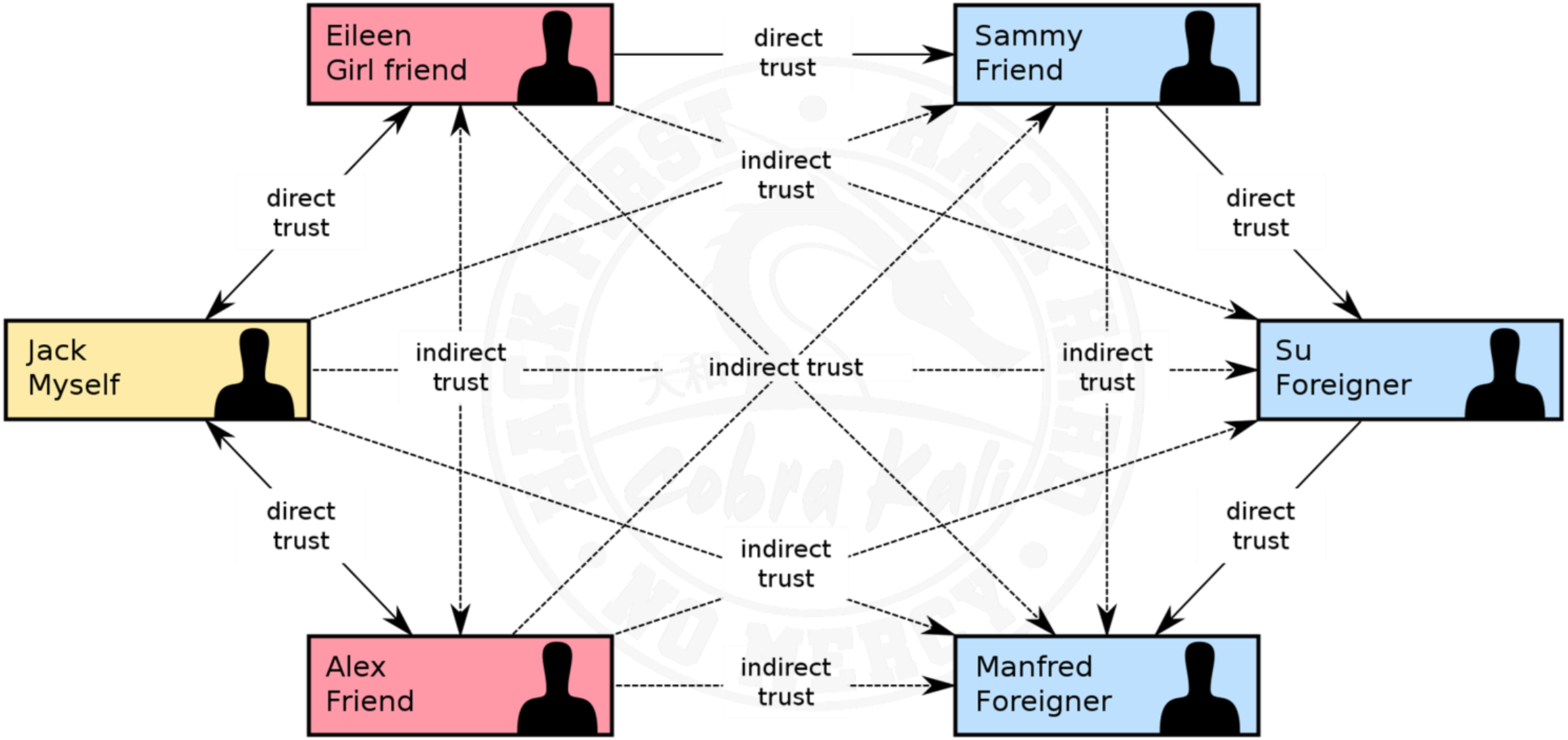
Trusted rings (GnuPG)

- **Cuando obtenemos una clave pública de un usuario nuevo, podemos asociarle un nivel de confianza**
  - **Desconocido**. No sabemos qué nivel de confianza dar a una clave
  - **Ninguno**. La clave no nos da ningún nivel de confianza
  - **Marginal**. No estamos seguros, conocemos los riesgos, pero la aceptamos
  - **Completo**. Las firmas hechas con esa clave son tan buenas como las nuestras
    - Confiamos en la clave
- **¿Cómo se le asigna un nivel a esa clave?**
  - La clave puede estar firmada por otros usuarios
  - El nuevo usuario será de confianza según nuestra confianza en esas firmas
    - Incluso sin conocerlo de antes
  - Las relaciones de confianza entre un conjunto de usuarios forman un **anillo de confianza** (**trusted ring**)
  - **Más información**: https://cran.r-project.org/web/packages/gpg/vignettes/intro.html



Fuente: https://cran.r-project.org/web/packages/gpg/vignettes/intro.html

Trusted ring (Web of trust)

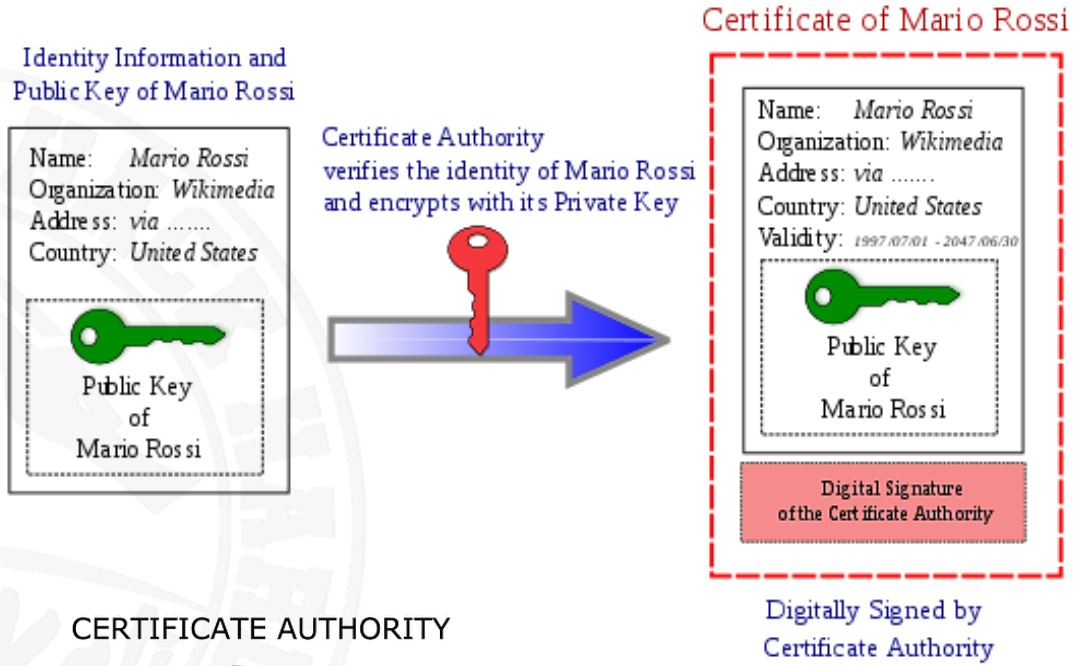


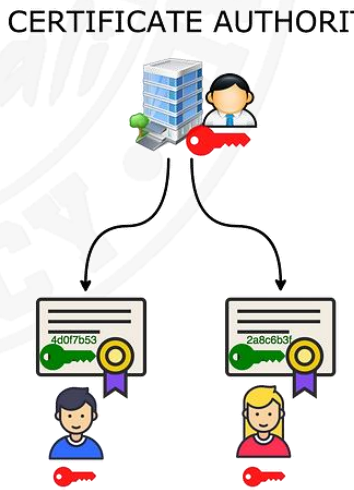
https://en.wikipedia.org/wiki/Web_of_trust#/media/File:Web_of_Trust-en.svg

Certification Authority (CA) (Autoridad certificadora)

Fuente: https://en.wikipedia.org/wiki/Certificate_authority

- **Firman** sus certificados con su propia **clave privada** para garantizar su autenticidad
  - Actúa como un **notario**, certificando la validez de sus certificados
- **Si el destinatario confía en la CA**, se puede confiar en la información del certificado
  - Y que el remitente del mensaje es quien dice ser
- **Además, admiten aplicar una función hash al propio certificado**
  - Y al contenido del mensaje
  - Esto evita la falsificación de certificados y comprueba su integridad





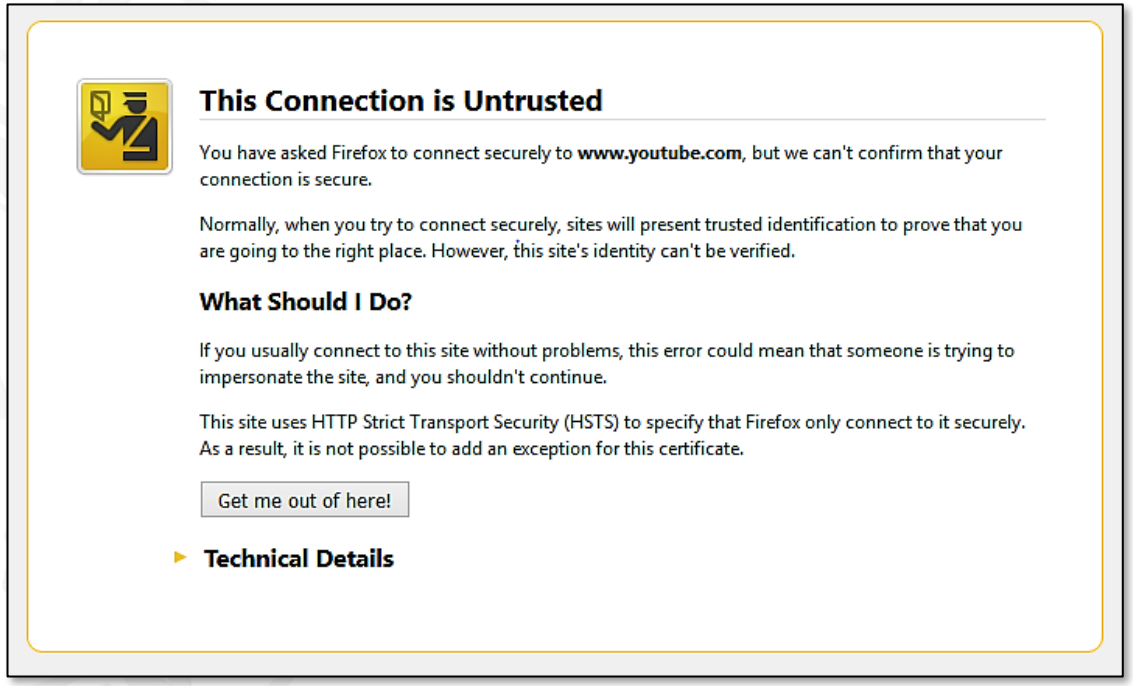
Fuente: https://cran.r-project.org/web/packages/gpg/vignettes/intro.html

Certification Authority (CA) (Autoridad certificadora)

- En este modelo de confianza, **un certificado será tan fiable como su autoridad emisora**
- Hay varios tipos de CAs
  - **CA conocidas** (Verisign, Thawte, ...)
    - Emiten certificados a petición del cliente, pero generalmente **de pago**
    - Sin embargo, hay CA conocidas gratuitas y fiables muy populares, como Let's Encrypt: https://letsencrypt.org/
  - **Autoridades certificadoras privadas de una organización**: con el coste añadido de mantenerlas
    - Las entidades ajenas a la empresa pueden no confiar en ella
  - **CA públicas estatales**: como el DNI español
    - https://www.dnielectronico.es/PortalDNIe/
- Cada certificado debe estar firmado por una sola entidad de certificación

CAs DESCONOCIDAS

- **Si usamos CA que no son de confianza veremos advertencias o errores**
  - Los navegadores incluyen su propia lista de CA de confianza
  - Usar una que no está en esta lista (como una propia) nos da esta advertencia
  - Típicamente esto pasa cuando se implementa HTTPS con un certificado autofirmado
  - **Común en entornos de prueba, pero no es adecuado para la producción en absoluto**
- **La información sigue firmada / cifrada**
  - El problema es que el certificado involucrado en el proceso no se puede verificar
  - ¿Confías en el emisor?



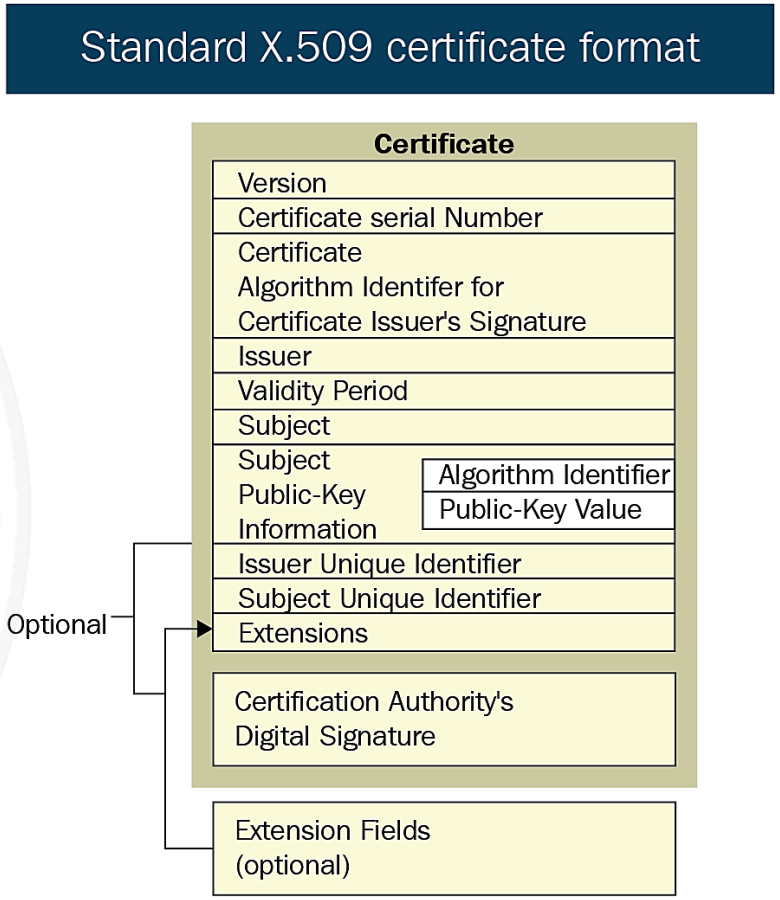
Fuente: https://support.mozilla.org/en-US/questions/1091062

Certificados digitales: Tipos

- **Personal**: se emite a usuarios para tareas como autenticación o firmas digitales
  - Un solo usuario puede tener varios, cada uno firmado por la CA que lo emitió
  - Los certificados personales generalmente pueden tener usos adicionales
    - **Pertenencia a empresa**: Además de identidad personal, certifica que se pertenece a una empresa dada
    - **Como representante de la empresa**: Certifica la membresía + poderes de representación de una empresa
    - **Como persona jurídica**: identifica a una empresa a la hora de realizar trámites administrativos
- **Servidor / máquina**: las máquinas también pueden necesitar demostrar su identidad a otras entidades
  - "Soy el servidor que estás buscando"
- **Software**: enviado a los desarrolladores para firmar su software
  - Antes de la producción o distribución
  - "Este software está firmado por Microsoft"
- **De CA**: la utilizan las CA para firmar los certificados que envían a otras entidades
  - "(Luke) Yo soy tu CA"

CERTIFICADOS DIGITALES: ESTRUCTURA

- **X.509** es un estándar que define el formato de los certificados de clave pública
  - Utilizado en muchos protocolos de Internet
  - Incluyendo TLS/SSL, que es la base de HTTPS
- **Todos los certificados, independientemente del tipo, tienen una misma estructura que incluye al menos**
  - Identidad del propietario del certificado
  - Clave pública del propietario
  - Propósito del certificado
  - Identidad del emisor del certificado (CA)
  - Fecha de caducidad
  - Algoritmos utilizados
  - Huella dactilar (para demostrar su integridad)
- **Veamos algunos usos de certificados**



Fuente:
https://subscription.packtpub.com/book/business/9781788832687/3/ch03lvl1sec31/pki-certificate-standards-for-iiot

IDENTIFICAR SERVIDORES

Seguridad de Sistemas Informáticos

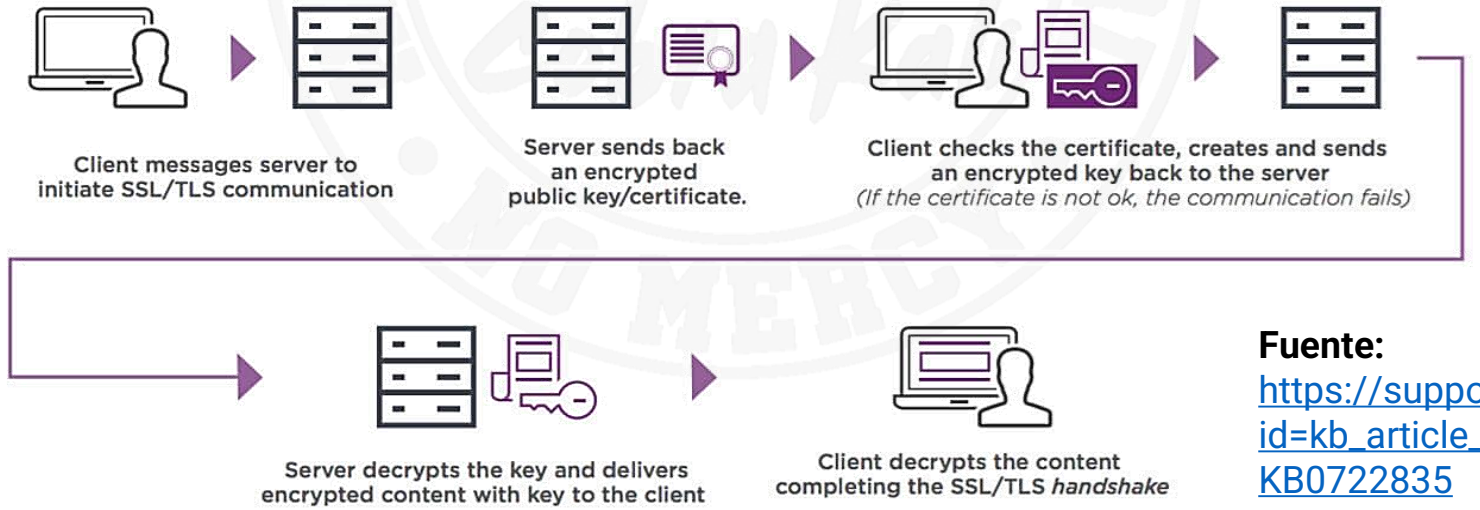
- Aplicación más frecuente
- Asegura que la máquina a la que nos conectamos es realmente quien dice ser
- El uso más popular es conectarse a servidores web a través del HTTP
  - **La transmisión está cifrada**: no se pueden obtener datos confidenciales por sniffing de red
- Es clave para garantizar
  - Privacidad de contraseñas, mensajería secreta, transacciones electrónicas...
  - **¡Toda nuestra vida digital!**

Google firma con el algoritmo SHA-256

Nombre del asunto	
Nombre común	www.google.com
Nombre del emisor	
País	US
Organización	Google Trust Services LLC
Nombre común	GTS CA 1C3
Validez	
No antes	Tue, 02 Jan 2024 13:09:58 GMT
No después	Tue, 26 Mar 2024 13:09:57 GMT
Nombres alternativos del sujeto	
Nombre de la DNS	www.google.com
Información de clave pública	
Algoritmo	Elliptic Curve
Tamaño de la clave	256
Valor público	04:62:ED:2D:2D:30:79:FE:94:F1:CC:C8:AE:5F:4F:86:D8:2E:79:AB:16:23:CF:4B:E7:B9:...

TRANSACCIONES REMOTAS: HTTPS

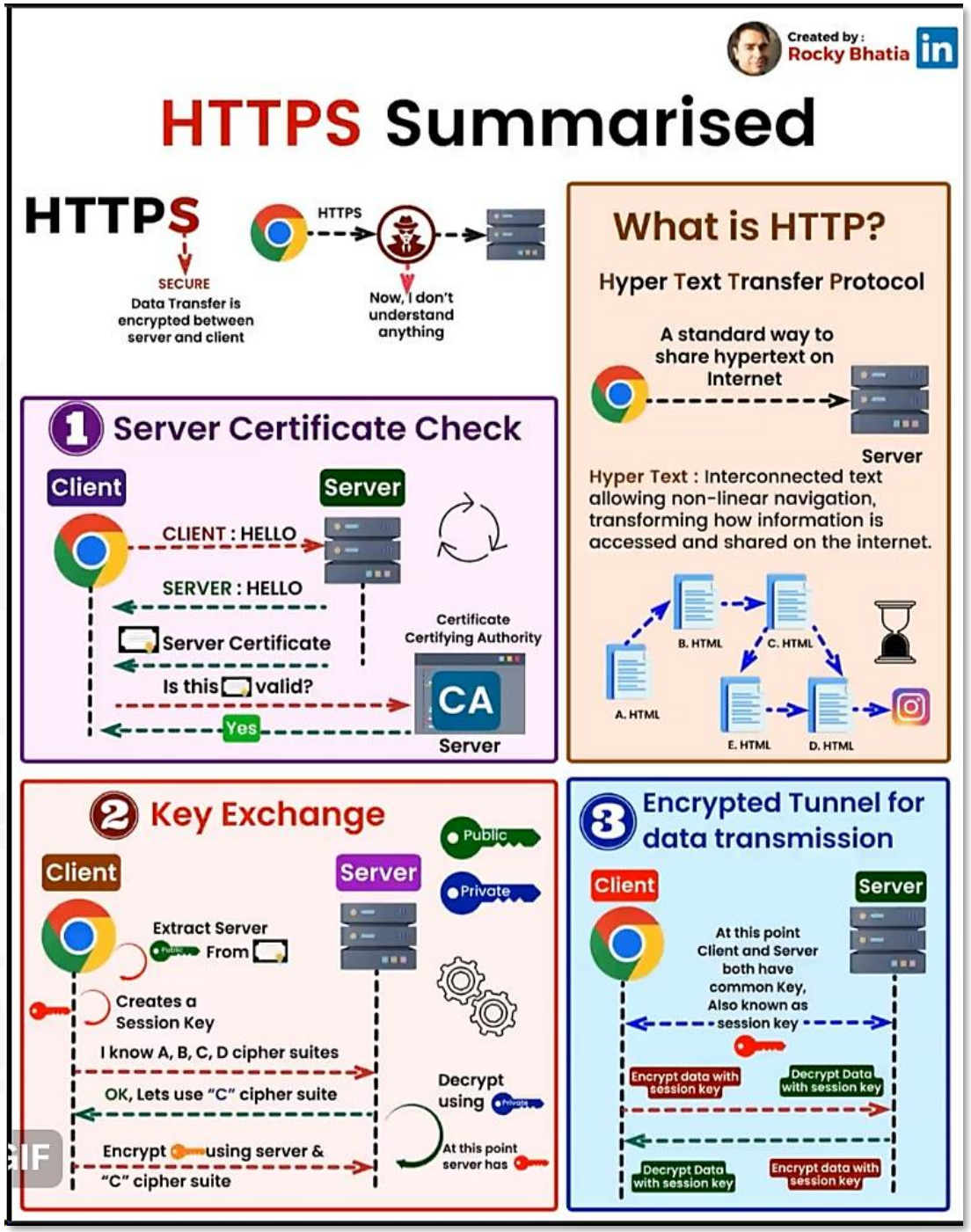
- Las conexiones **HTTP seguras** son un proceso transparente de negociación de certificados establecido entre navegadores y servidores web
- Garantiza una transferencia de datos segura si se negocia un cifrado seguro
  - No ocurre con **clientes antiguos** (Ej.: Windows XP)
  - O con **servidores web mal configurados** (ataques de degradación criptográfica)
- Configurar cualquier servidor web con características de protocolo HTTP adecuadas es fácil siguiendo esta guía: https://ssl-config.mozilla.org/



Fuente:
https://support.servicenow.com/kb?id=kb_article_view&sysparm_article=KB0722835

TRANSACCIONES REMOTAS: HTTPS

- **Como ves, muchos conceptos ya vistos forman parte de una conexión HTTPs**
  - **Certificados** para identificar servidores y transmitir su clave pública al cliente
  - **Cifrado asimétrico** para cifrar una clave simétrica para cada conexión
    - Que asegura su transmisión segura del cliente al servidor
  - **Cifrado simétrico** a todos los datos enviados
    - Más rápido que usar el cifrado asimétrico para eso
    - ¿Te das cuenta de que ahora **ambos comparten una clave simétrica**? (session key)
      - Aquí entra AEAD
      - ¡El tráfico está cifrado y no se puede alterar!

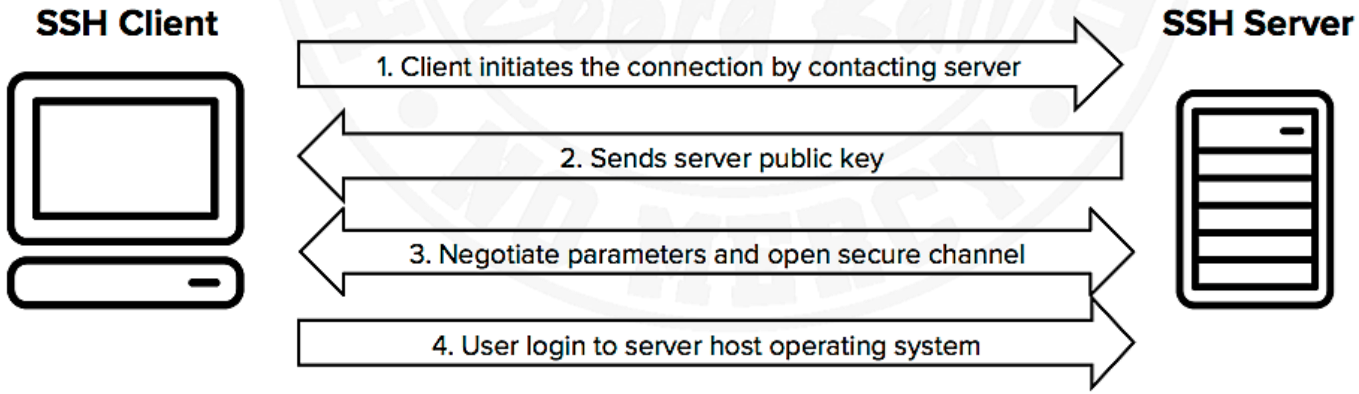


TRANSACCIONES REMOTAS: SSH

- **Puedes usar SSH para cifrar conexiones a un equipo remoto**
  - Al principio de la conexión, el servidor envía su clave pública al cliente
  - Si el cliente acepta esta clave, la guarda para futuras conexiones
  - El cliente genera una **clave de sesión aleatoria**, la cifra con la clave pública del servidor y se la envía
  - Esta clave se utilizará para la transmisión cifrada de los datos (**cifrado simétrico**)
    - **Recuerda**: es **más rápido** que el cifrado de clave pública

```
The authenticity of host '10.0.0.1 (10.0.0.1)' can't be established
but keys of different type are already known for this host.
RSA key fingerprint is f8:88:86:4f:d7:9f:8c:c2:a2:e9:0b:01:8f:5b:8e:fd.
Are you sure you want to continue connecting (yes/no)? _
```

- **Configurar un servidor SSH correctamente es fácil siguiendo estas pautas**
  - https://infosec.mozilla.org/guidelines/openssh

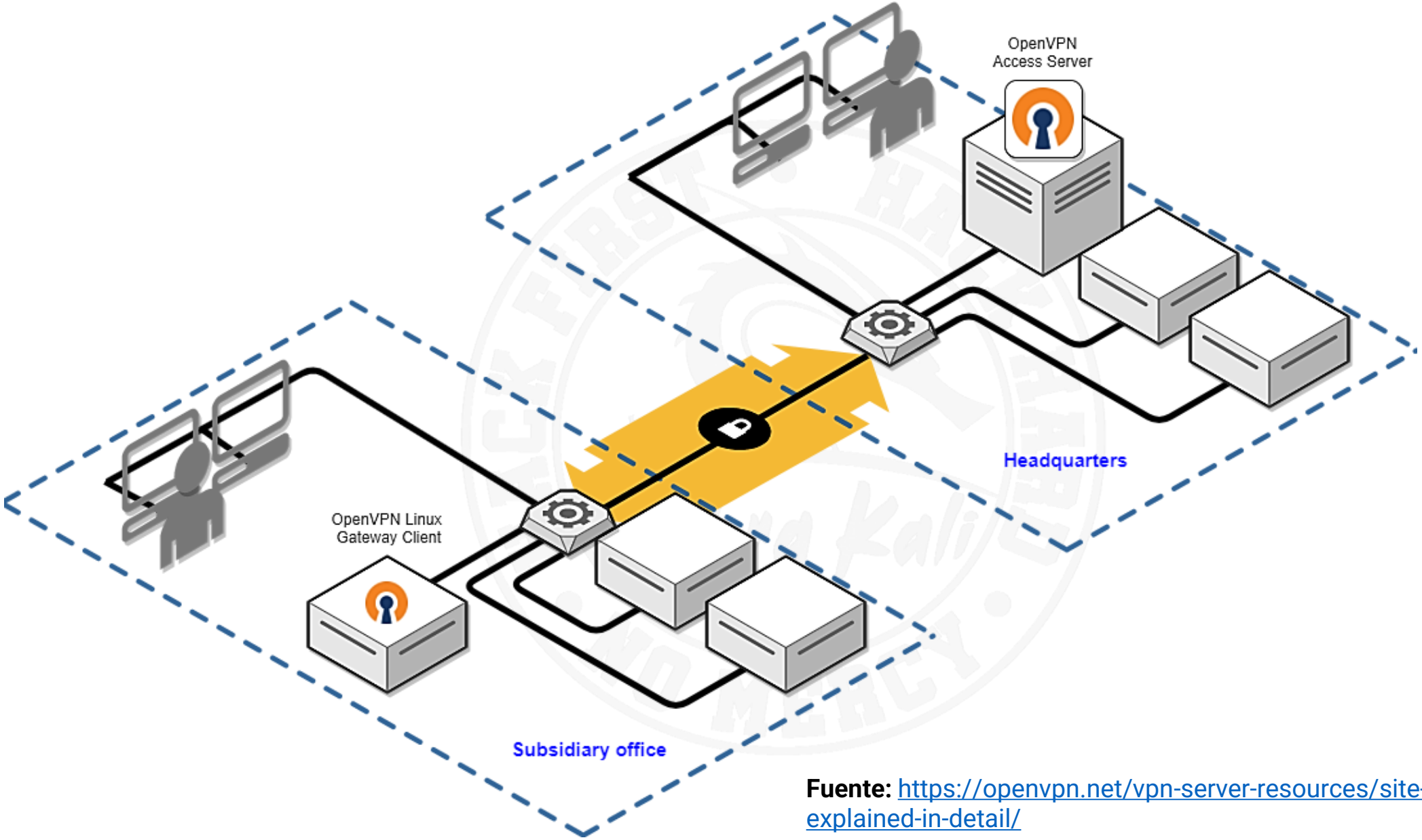


Fuente: https://www.ssh.com/academy/ssh

Virtual Private Networks (VPNs)

- **Extiende una red privada a través de una red pública**
  - Creada estableciendo una conexión virtual punto a punto utilizando circuitos dedicados o con protocolos de túnel sobre redes existentes (el Eurotunel puede ser una buena metáfora) ☺
  - Desde la perspectiva del usuario, se accede a los recursos de una red privada de forma remota
  - Permite a los usuarios intercambiar datos a través de redes compartidas o públicas **como si sus dispositivos estuvieran conectados directamente a la red privada**
- **La tecnología VPN se desarrolló para permitir que usuarios remotos y sucursales accedan a aplicaciones y recursos corporativos**
  - Para garantizar la seguridad, una red privada se crea usando un protocolo de túnel en capas cifrado
  - Los usuarios de una VPN se autentican con contraseñas o **certificados** para acceder a ella
- **Implementaciones populares**
  - WireGuard es un software VPN muy recomendable: https://www.wireguard.com/
  - ZeroTier, una VPN P2P (ya veremos qué es eso ☺): https://www.zerotier.com/
  - ExpressVPN: Es la elección de los editores de TechRadar 2020 y CNET: www.expressvpn.com

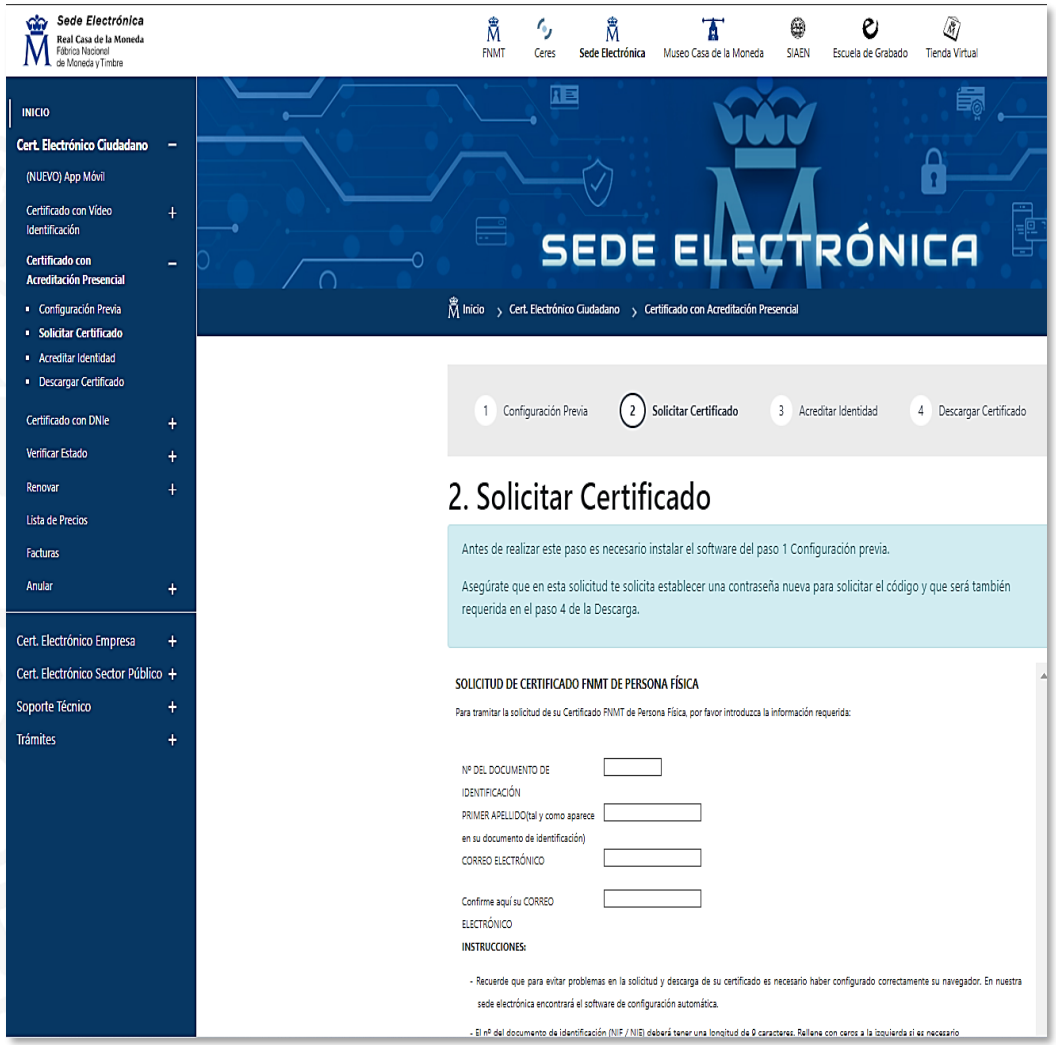
Virtual Private Networks (VPNs) tradicionales



Fuente: https://openvpn.net/vpn-server-resources/site-to-site-routing-explained-in-detail/

Transacciones remotas: Clientes / ciudadanos

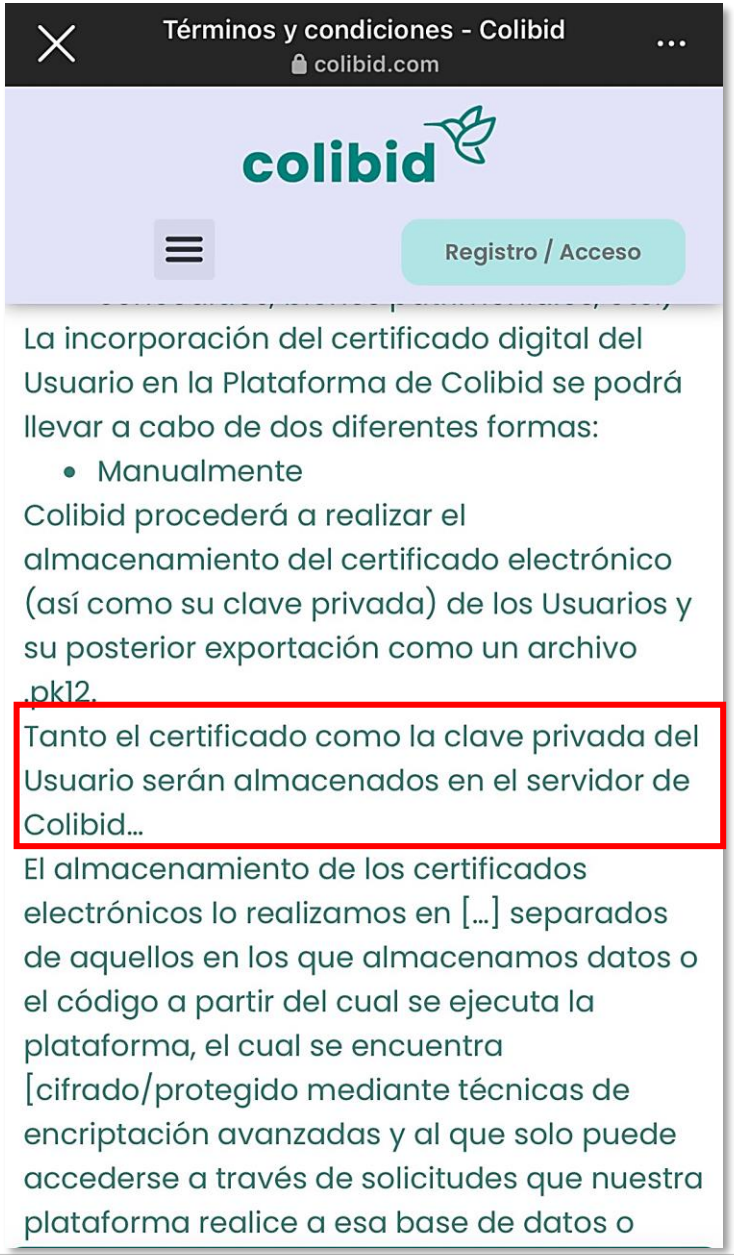
- **Ciertos procedimientos administrativos requieren que los usuarios se identifiquen**
  - ¡Una empresa / institución **DEBE** asegurarse de que la persona que realiza la operación es quien dice ser!
- **Esto es especialmente importante**
  - En trámites oficiales: Hacienda, Ayuntamientos...
  - Y para transacciones comerciales
- **Los certificados están reemplazando contraseñas y tarjetas de coordenadas en este tipo de operaciones**
- **Las instituciones oficiales Españolas reconocen certificados emitidos por el Ceres**
  - De la Fábrica Nacional de Moneda y Timbre
  - http://www.cert.fnmt.es/



Fuente: https://www.sede.fnmt.gob.es/certificados/persona-fisica/obtener-certificado-software/solicitar-certificado

Transacciones remotas: Clientes / ciudadanos

- **Hay que tener mucho cuidado con el uso de certificados que nos representen legalmente**
  - Dónde los guardamos y a quién se los cedemos
- **Por ejemplo, esta aplicación está pidiendo al usuario que le ceda un certificado de su identidad con ambas claves**
  - Ahora en la práctica la aplicación podrá hacer cualquier trámite electrónico en nombre del usuario sin restricciones
  - La aplicación es efectivamente para hacer dichos trámites...
  - ...pero está pidiendo al usuario que **confíe ciegamente en ella** a la hora de guardar los certificados y protegerlos
- **¡Si su almacén de certificados se vulnera, el atacante podría suplantar a todos sus usuarios en procedimientos de la administración electrónica directamente! ☹**

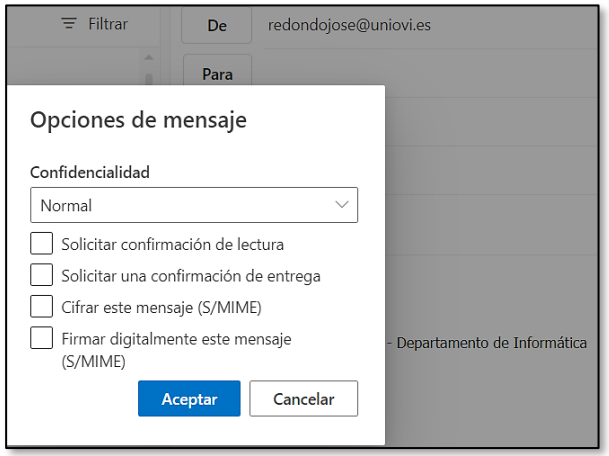


Estos términos y condiciones polémicos han sido modificados desde entonces para no requerir esta información sensible del usuario

AUTENTICACIÓN DE EMAIL / FICHEROS

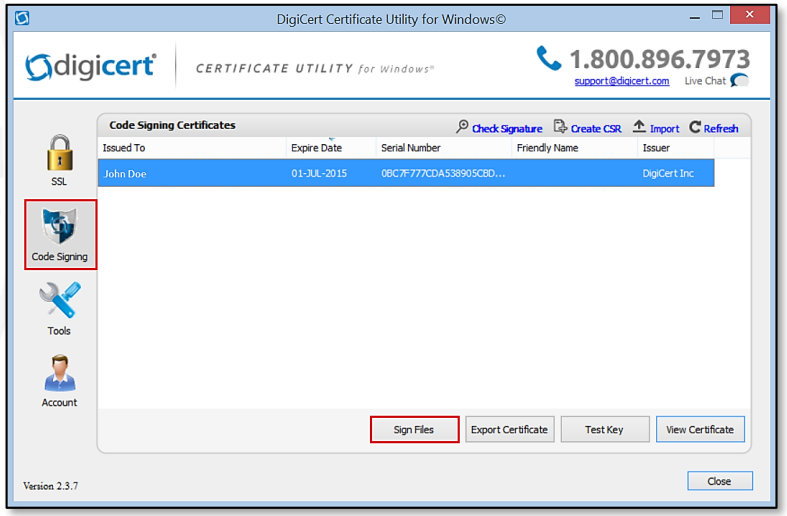
Correos electrónicos

- Ciertos clientes de correo permiten enviar y recibir correos electrónicos firmados y/o cifrados
  - Existen sistemas basados en certificados **RSA** o **GnuPG**
  - https://support.microsoft.com/en-us/office/encrypt-email-messages-373339cb-bf1a-4509-b296-802a39d801dc
  - https://support.microsoft.com/en-us/office/secure-messages-by-using-a-digital-signature-549ca2f1-a68f-4366-85fa-b3f4b5856fc6
- Soportados por ciertos servicios de **webmail** (Ej.: **ProtonMail**)
  - Los navegadores admiten la administración de certificados, por lo que el servidor puede solicitarles la identidad del usuario



Archivos

- Existen aplicaciones que permiten firmar y/o cifrar archivos con **RSA/GnuPG** o similar
- Permitir el uso de documentos electrónicos con la misma validez legal que documentos físicos firmados
- Existen muchos sistemas, pero debemos utilizar los aceptados por nuestras instituciones/empresas (**eCofirma**, **Adobe**...)

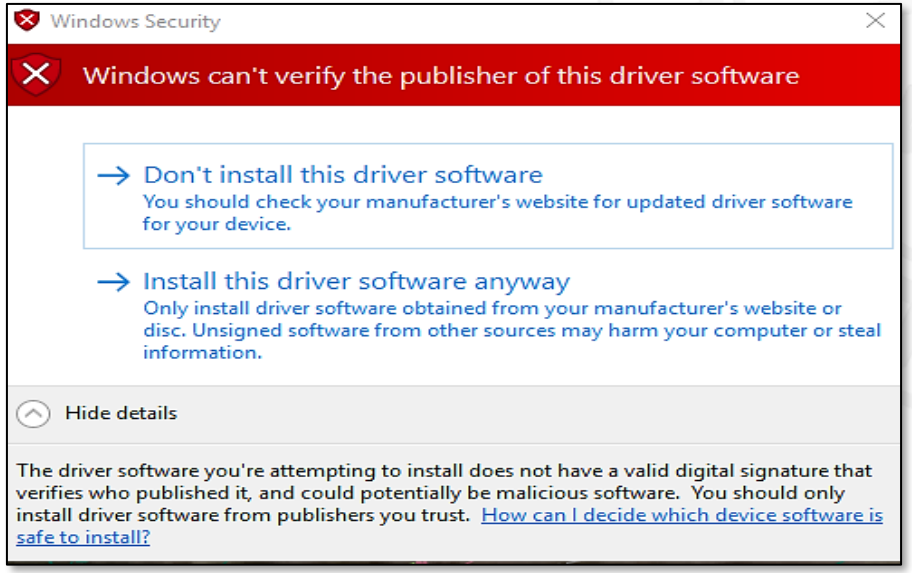


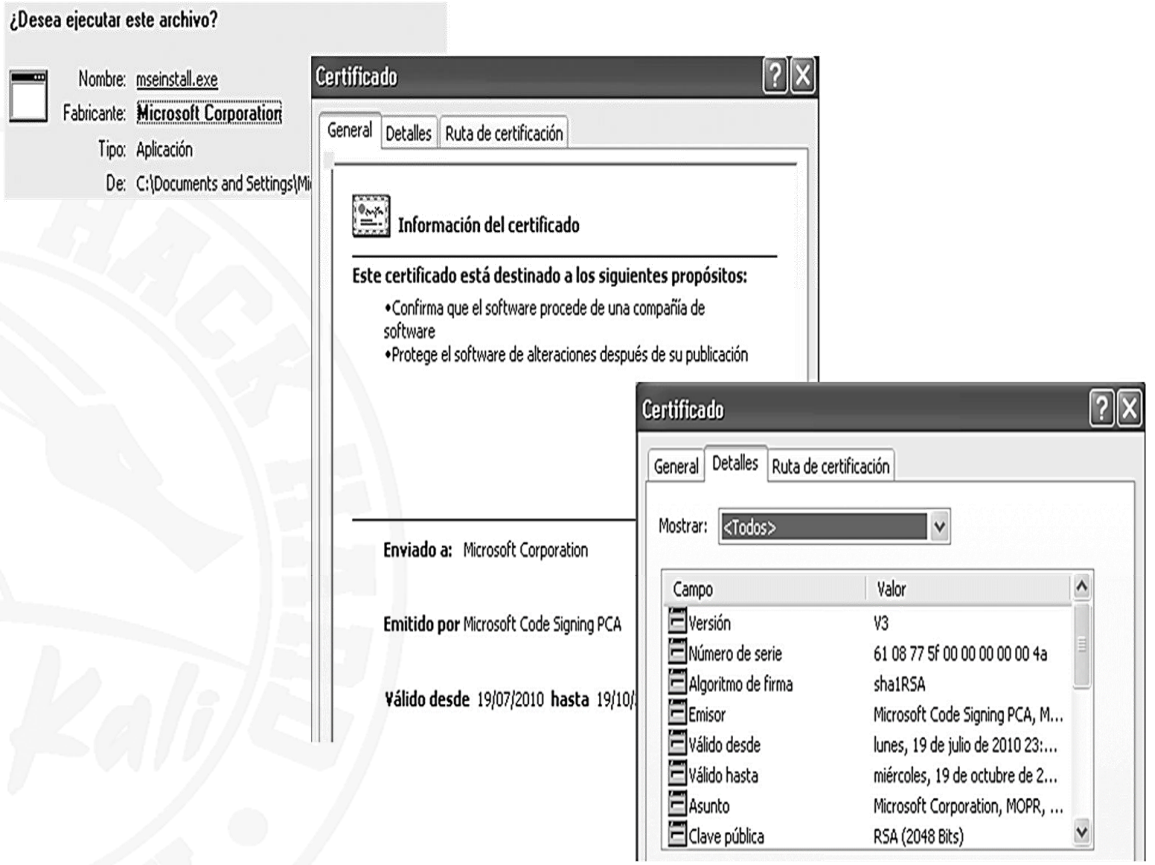
Fuente: https://www.digicert.com/kb/code-signing/digicert-certificate-utility-to-sign-code.htm

➢ Ejercicio **L4B3_CERTSIGNPDF**, **L4B4_CHECKSIGN**

AUTENTICACIÓN DE CÓDIGO

- Los programas se pueden firmar **para garantizar su autoría e integridad de contenido**
  - https://en.wikipedia.org/wiki/Code_signing
- Un programa puede tener 3 estados
  1. **Sin firmar**. El autor es desconocido
  2. **Firmado**, pero **sin** un certificado de confianza
  3. **Firmado con** un certificado de confianza





¿Crees que lo has entendido todo? ¡Autoevalúate!

?

Eres capaz de hacer lo siguiente

- *Entiendes que, en realidad, un certificado digital no es más que un documento con una estructura conocida que dice que una(s) determinada(s) claves pertenecen a una persona o entidad conocida*
- *Comprendes que hay certificados pensados para enviarse a cualquiera (con tu clave pública) y otros pensados para quedarse en una máquina concreta (con ambas claves),*
  - *Y que en caso de transmitir estos últimos debes hacerlo con mucho cuidado y a lugares de confianza (Ej.: otras máquinas de tu propiedad)*
- *Entiendes la función tan importante que cumplen los anillos de confianza y las autoridades certificadoras a la hora de trabajar con certificados*
  - *Y las diferencias entre ambas aproximaciones*
- *Puedes recordar las principales aplicaciones de los certificados digitales*
- *Entiendes como algunas de ellas combinan cifrado simétrico y asimétrico para aprovecharse de sus principales ventajas*

< Ir al Índice

APÉNDICE

Otros conceptos interesantes

*Iconos por Bing Chat AI

Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Técnicas de cifrado negable (Plausibly deniable encryption)

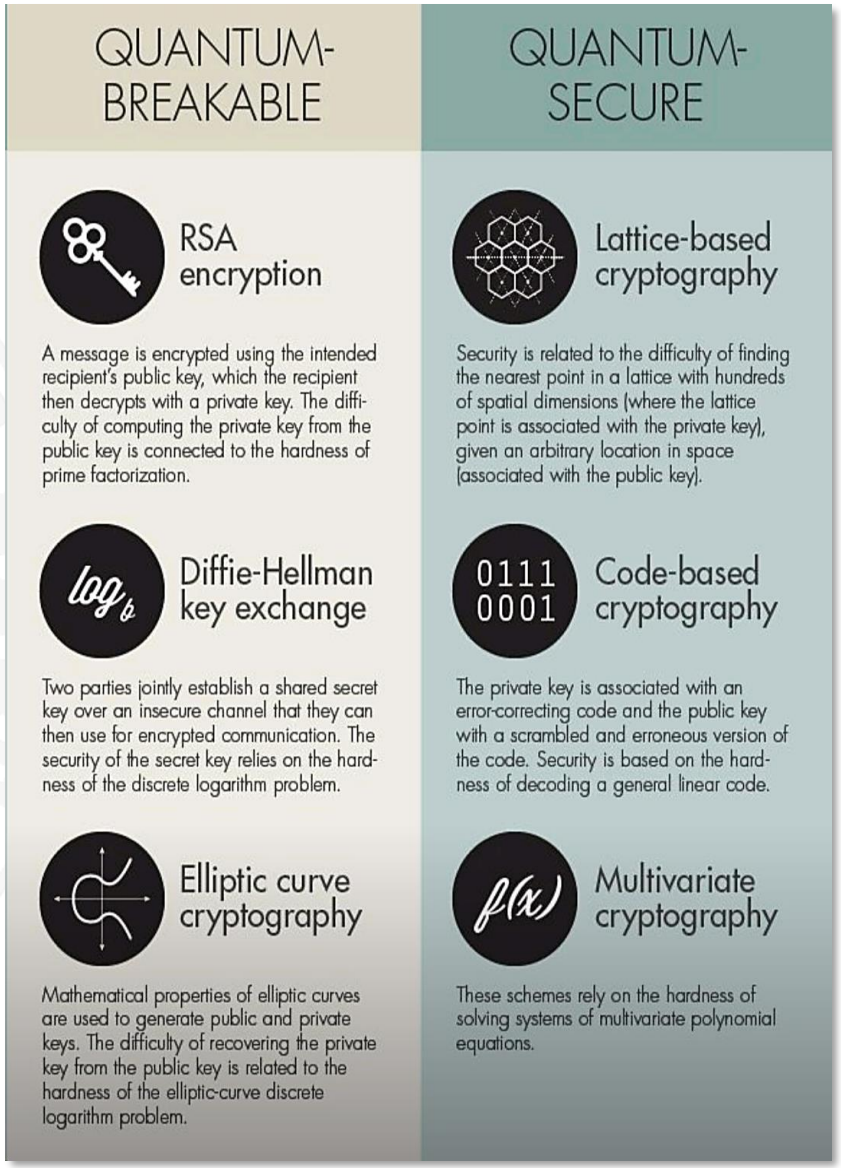
- **Se puede negar la existencia de un archivo o mensaje cifrado concreto**
  - Un adversario no puede probar que unos datos en texto plano existen
- **El cifrado negable hace que sea imposible probar la existencia de un mensaje de texto plano sin la clave de descifrado adecuada**
  - Esto se puede hacer **permitiendo que un mensaje cifrado se descifre en diferentes textos planos**, dependiendo de la clave utilizada
    - **Datos falsos** (decoy): que le den al adversario información falsa ante un descifrado aparentemente exitoso
    - **O aleatorios** (chaff): que le den al adversario la sensación de que no hay información cifrada realmente
  - Permite al remitente hacer una negación plausible si se ve obligado a dar una clave de cifrado
- **Este concepto fue utilizado por Julian Assange y Ralf Weinmann**
- **Hay algunas herramientas que facilitan el cifrado negable**
  - OpenPuff Steganography and Watermarking
  - FreeOTFE (On The Fly Encryption)
  - TrueCrypt, LibreCrypt y BestCrypt

Criptografía cuántica (Quantum cryptography)

- **Utiliza principios de la mecánica cuántica para intercambiar claves aleatorias en canales inseguros**
- **No es (actualmente) rompible, el intercambio de claves es seguro**
  - Si alguien escucha durante la creación de claves secretas, **el proceso se altera**
  - Alertando así a las partes que se comunican antes de que se transmita la información
  - Esto es una consecuencia del principio de incertidumbre de Heisenberg
    - La medición en un sistema cuántico lo cambia (una de sus propiedades más importantes)
- **Se han desarrollado protocolos de distribución de claves cuánticas**
  - Su implementación tiene un costo considerable...
  - Desde 2007, la transmisión del recuento de votos en las elecciones federales de Ginebra está protegida con él
  - Algunos prestigiosos bancos suizos también lo están utilizando
  - Se usan láseres para emitir a través de fibras ópticas información sobre el elemento constituyente de la luz (el fotón)

OTROS CONCEPTOS O APLICACIONES DE LA CRIPTOGRAFÍA

- **"Trending topics" de la criptografía**
  - **Cifrado homomórfico** (**Homomorphic encryption**): permite hacer cálculos sobre datos cifrados
  - **Cifrado ligero** (**Lightweight encryption**): para dispositivos con baja potencia computacional, como IoT
  - **Cifrado de caja blanca** (**Whitebox encryption**): protección de claves mientras residen en RAM
  - **Cifrado post-cuántico** (**Post-quantum encryption**): algoritmos que se pueden ejecutar en ordenadores tradicionales...
    - ¡Pero robustos contra los ordenadores cuánticos!
  - Cifrado con capacidad de búsqueda (**Searchable encryption**), Criptografía diferencial (**Differential cryptography**), Cifrado con conservación de formato (**Format-Preserving Encryption**, FPE)...
- **¿Quieres usar criptografía de manera efectiva?**
  - Prácticas criptográficas recomendadas: https://gist.github.com/atoponce/07d8d4c833873be2f68c34f9afc5a78a



Fuente: https://es.linkedin.com/in/ismaelr

PARA SABER MÁS…

- **Si te gusta este tema, aquí tienes estos recursos**
  - Practical Cryptography for Developers
    - https://cryptobook.nakov.com/
  - Handbook of Applied Cryptography
    - https://www.amazon.es/Handbook-Cryptography-Discrete-Mathematics-Applications/dp/0849385237
  - Real-World Cryptography
    - https://www.manning.com/books/real-world-cryptography
  - Foundations of Cryptography
    - Volume 1, Basic Tools: https://www.amazon.es/Foundations-Cryptography-Basic-Tools-Vol/dp/0521791723
    - Volume 2, Basic Applications: https://www.amazon.es/Foundations-Cryptography-Basic-Applications-Paperback/dp/052111991X
  - Serious Cryptography: A Practical Introduction to Modern Encryption
    - https://www.amazon.es/Serious-Cryptography-Practical-Introduction-Encryption/dp/1593278268
  - Recursos del experto de referencia en criptografía, Alfonso Muñoz
    - Libros "Seguridad en el protocolo SSL-TLS" y "Esteganografía lingüística y canales encubiertos" (gratis): https://github.com/mindcrypt/libros
    - Y si lo que te gusta es el cracking ☺…: https://github.com/mindcrypt/cracking
- **¿Quieres especializarte? Mira la certificación CriptoCert Certified Crypto Analyst**
  - ¡Pero tendrás que estudiar mucho!: https://www.criptocert.com/CriptoCert_Analyst.html

TEMA 2
APLICACIONES DE LA CRIPTOGRAFÍA

Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo