SEGURIDAD DE LA INFORMACIÓN

TEMA 2

TÉCNICAS CRIPTOGRÁFICAS BÁSICAS

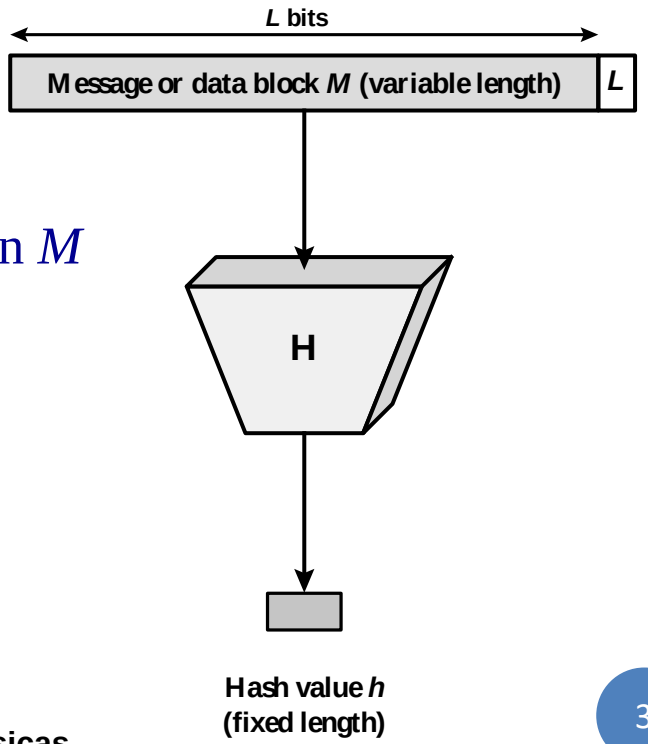
(Y SERVICIOS DE SEGURIDAD ASOCIADOS)

Otras primitivas criptográficas

Funciones Hash

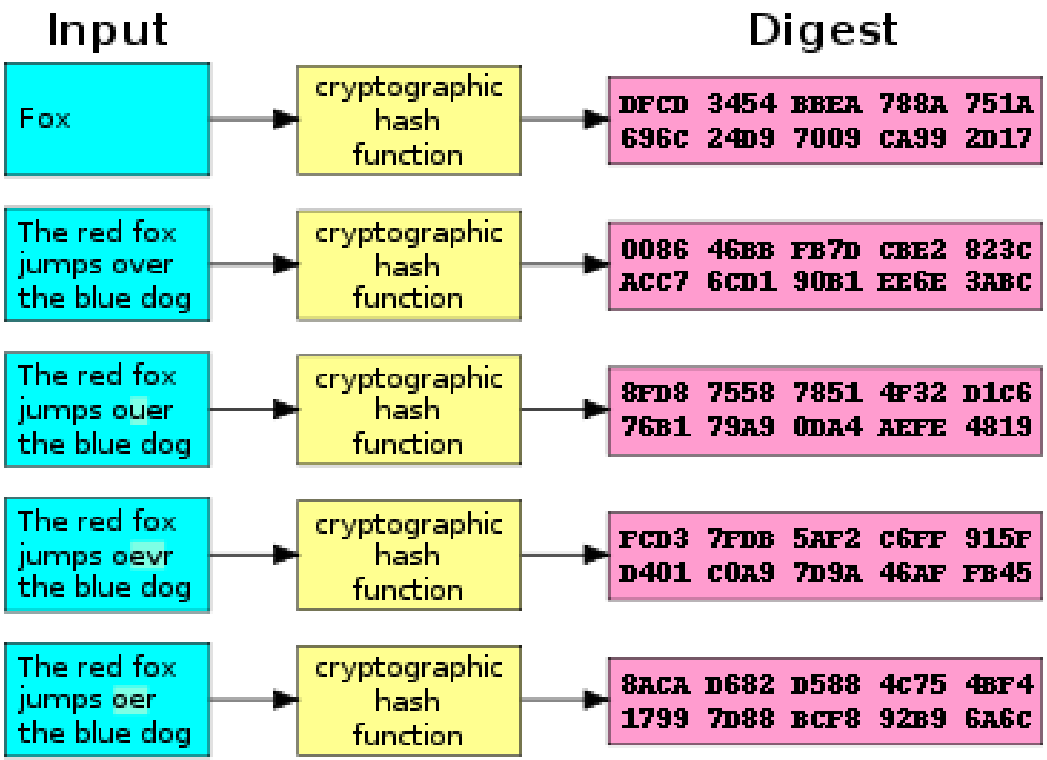
- Una función unidireccional (o de sentido único) es un bloque de construcción para muchos protocolos criptográficos
  - Son fáciles de computar, pero muy difíciles de invertir
  - Por lo tanto, no son útiles de cara al cifrado
    - porque un mensaje cifrado mediante una función unidireccional no se podría descifrar
  -

- Una función hash es una función que:
  - acepta como entrada un bloque de información *M* (preimagen) de longitud variable
  - produce un bloque de salida *h* (valor hash) de longitud fija
    - Ese valor se denomina huella digital de *M*

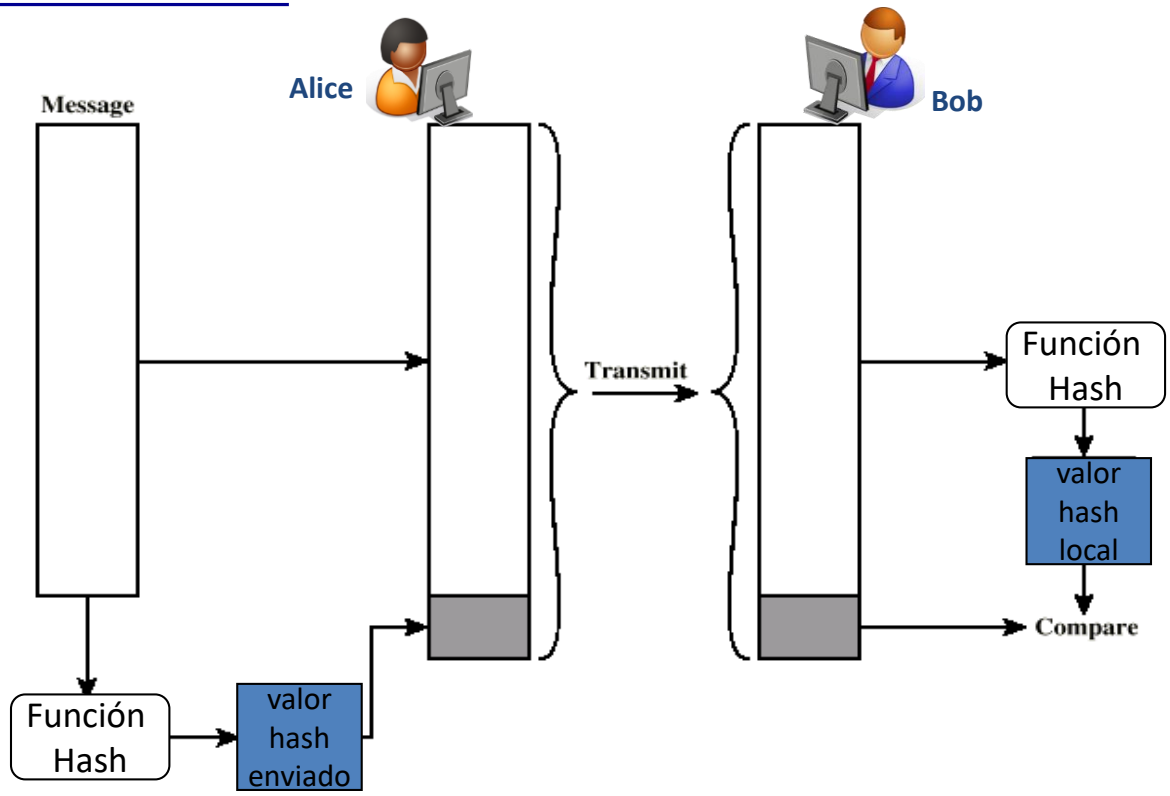


- Una función hash criptográfica es un algoritmo con el que es fácil computar el valor hash a partir de una preimagen, pero para el que resulta computacionalmente imposible:
  1. encontrar la preimagen *M* para un valor hash *h* predeterminado (propiedad: unidireccionalidad),
  2. encontrar dos bloques *M* y *M´* que produzcan el mismo valor hash *h* (propiedad: libre de colisiones)
- Para cualesquiera dos entradas *M* y *M'* a una función hash criptográfica, se producen dos salidas *h* y *h'* diferentes, pero no se puede discernir la relación entre esas variaciones
  - Concretamente, un cambio en un único bit de la preimagen da lugar a *un cambio de, como media, la mitad de los bits de salida*

- Efecto de una función hash (caso de *SHA-1*):

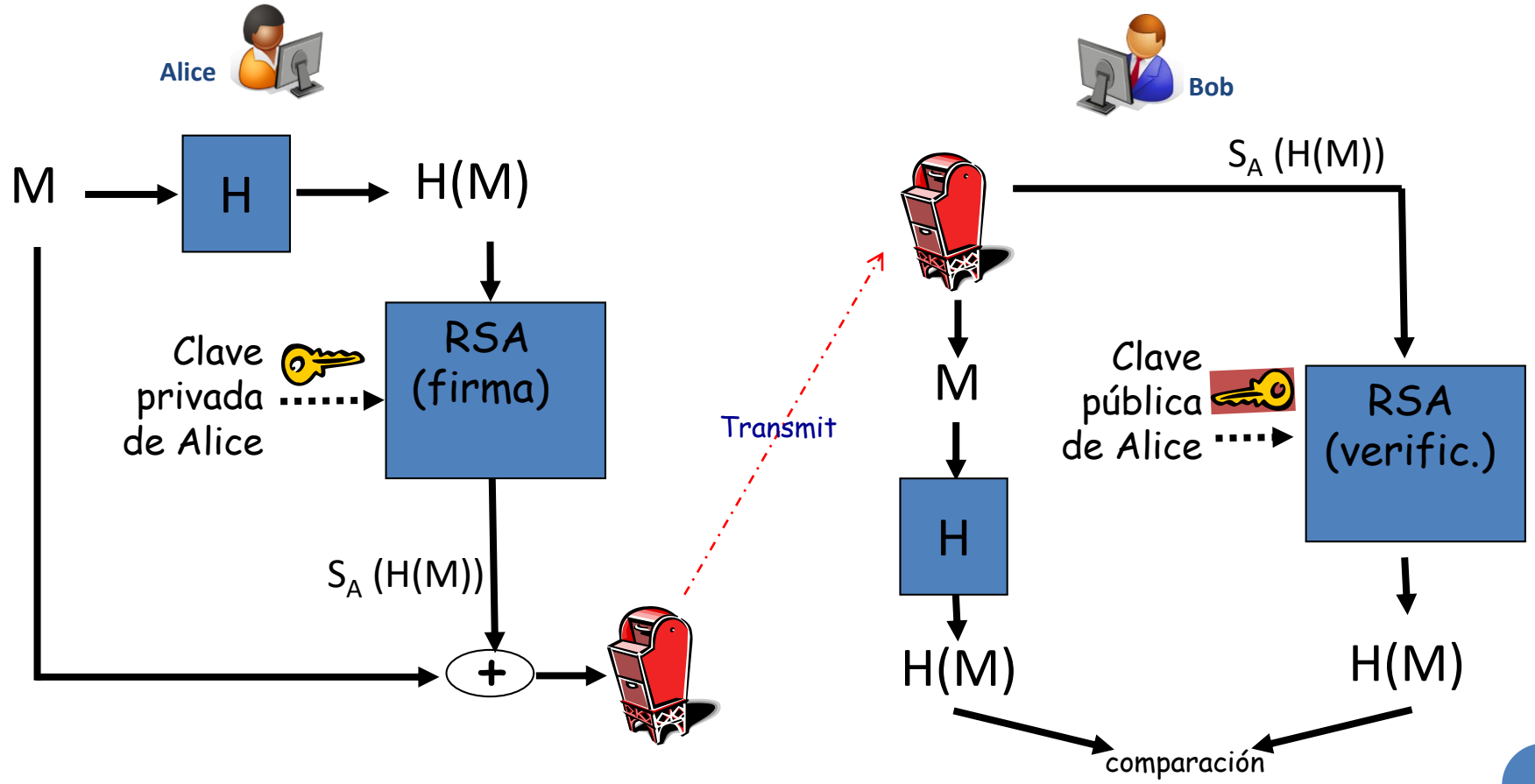


- La utilidad más inmediata de una función hash es proporcionar el servicio de integridad de datos
- Ejemplo de uso:

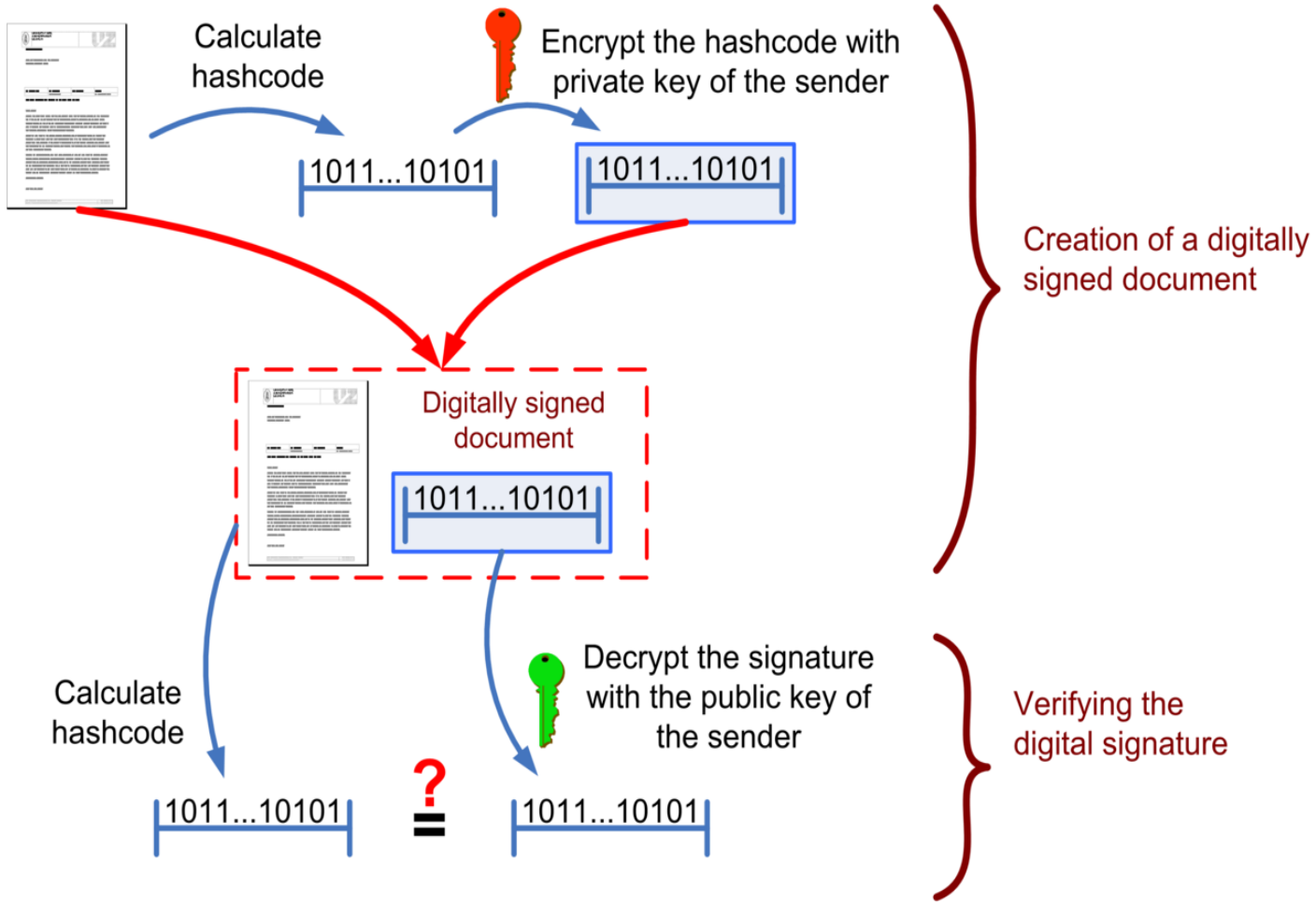


- Sin embargo, se observa que *no hay autenticidad* (*Bob* no tiene garantías de que *Alice* sea el origen)

- El uso combinado de las funciones hash con la criptografía de clave pública mejora sustancialmente la eficiencia de uso de esta última (en el procedimiento específico de firma digital)

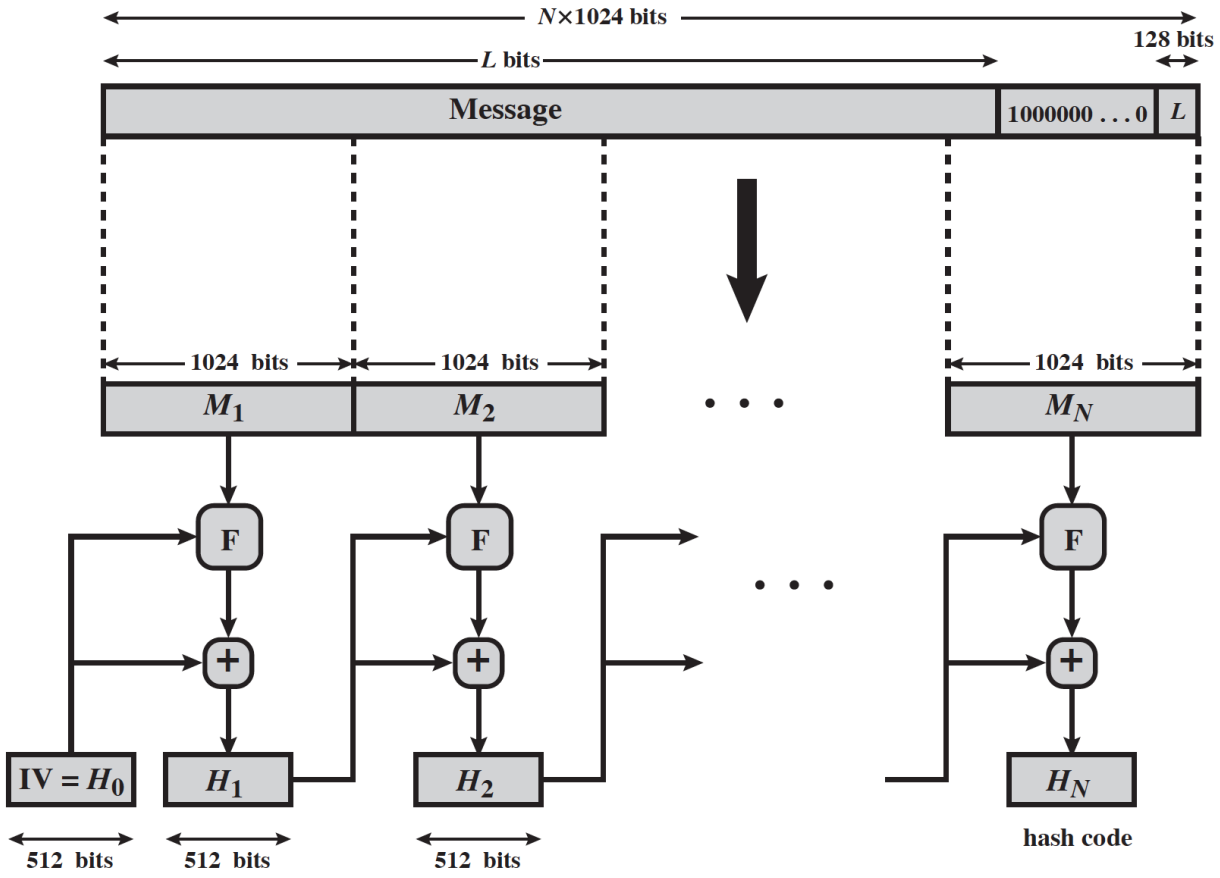


Creating and verifying a digital signature

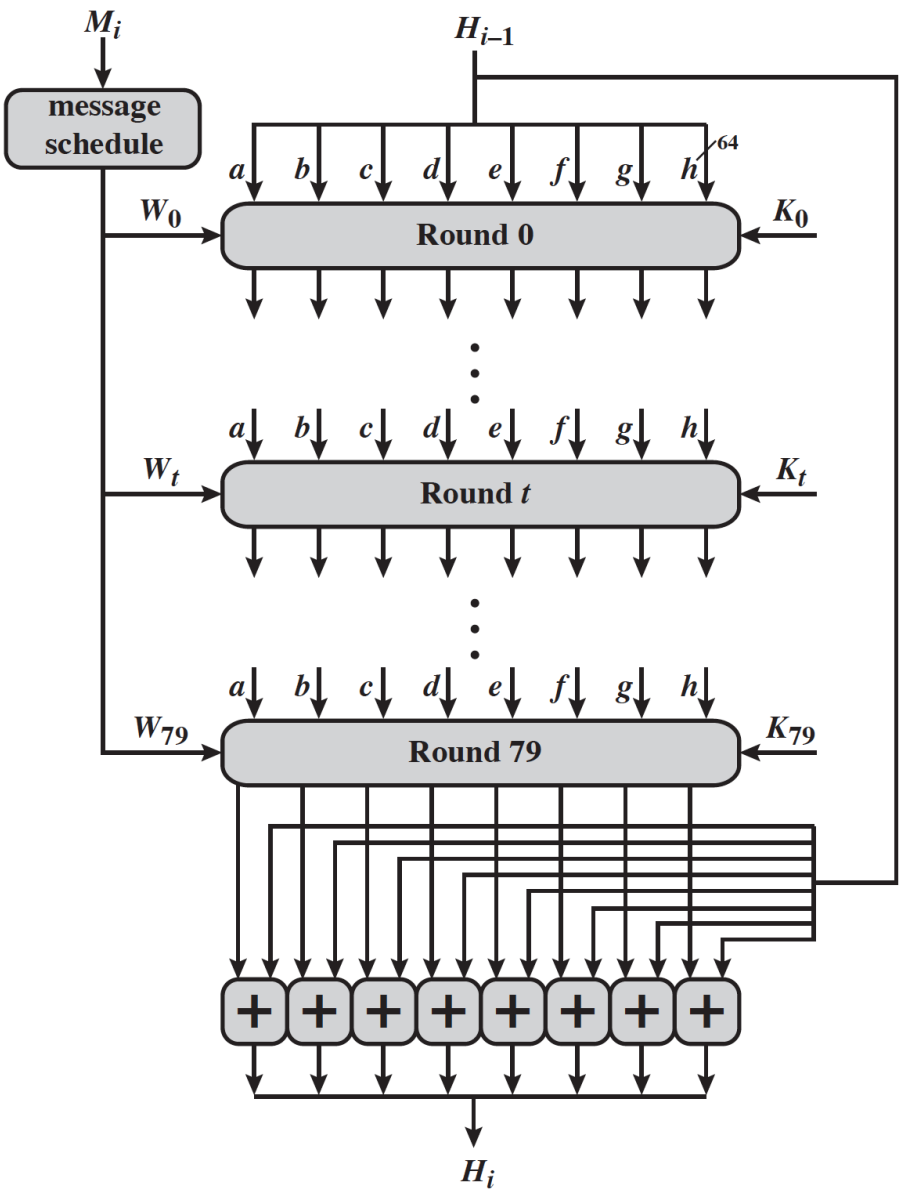


- Los beneficios del esquema anterior son:
  - la firma se puede mantener separada del documento
  - los requisitos de almacenamiento y firma son mucho menores
- Un sistema de archivos puede usar este tipo de esquema para verificar la existencia de documentos sin tener que almacenar sus contenidos

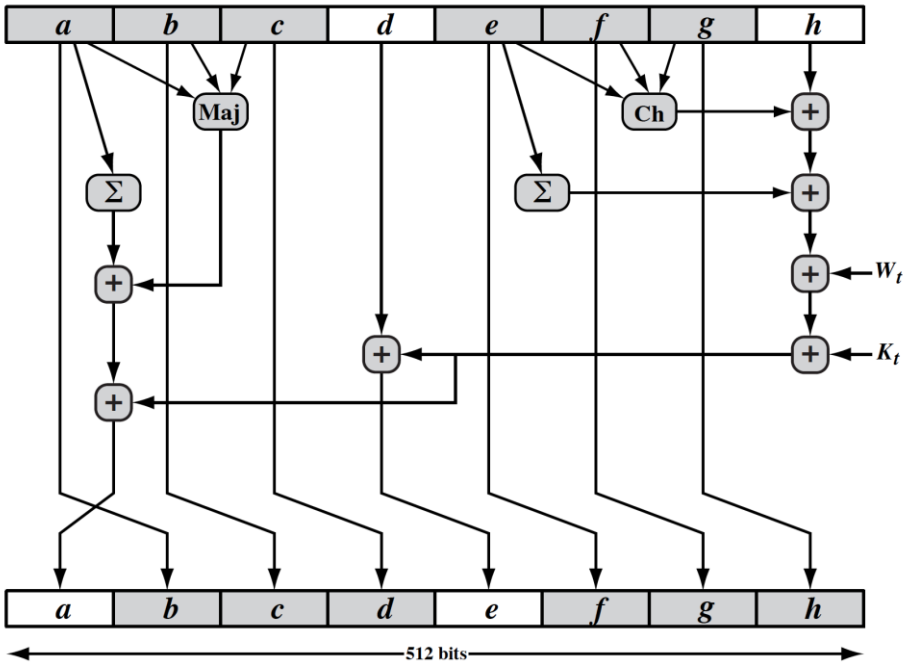
- Ejemplo de función hash: *SHA-512* (*SHA-2*)



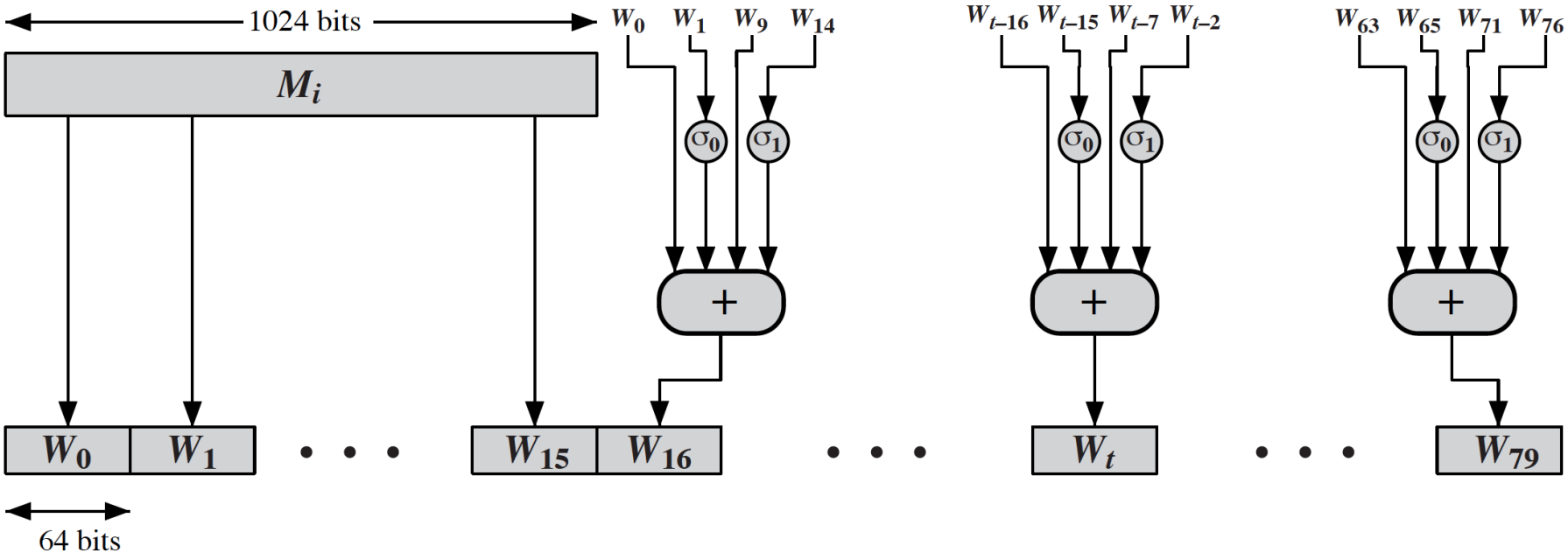
Esquema general del algoritmo SHA-512



Procesamiento de un bloque de 1024 bits en SHA-2



Operación elemental (una etapa) en SHA-2



Creación de secuencia de entrada de $W_0..W_{79}$ para procesamiento del bloque M_i en SHA-2

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

Constantes $K_0..K_{79}$ en SHA-2

Funciones hash

- Estas son algunas de las funciones hash más conocidas:
  - MD-5
    - A pesar de estar muy exendida, *MD-5* **no se debería considerar segura**, porque **se puede atacar criptoanalíticamente**, encontrando colisiones en un margen de varios segundos
  - RIPEMD-128
    - **No se puede considerar segura** en la actualidad con independencia de los **ataques criptoanalíticos** que puedan reducir la complejidad global
  - SHA-1
    - Se utiliza extensivamente pero **ya no se considera segura**
    - El criptoanálsis de *MD-5* se ha aplicado también a *SHA-1* (dado que son de la misma familia) y se han encontrado colisiones no sólo del punto de vista teórico sino también práctico

SHA-1

Criptoanalistas 'rompen' SHA1

theregister.co.uk 17-Feb-2005

Twittear

Share

El rumor que se ha estado corriendo, es ahora oficial, el algoritmo popular SHA-1 ha estado siendo atacado exitosamente por investigadores en China y Estados Unidos. Se ha descubierto una "colisión" en la versión completa en 2^{69} operaciones de hash, haciendo posible intentar un ataque de fuerza bruta exitoso con las computadoras más potentes de la actualidad.

Esto no significa desastre en términos prácticos, ya que la cantidad de poder computacional y conocimiento matemático necesario para poder llevar a cabo un ataque exitoso es elevado. Pero se ha demostrado que el algoritmo SHA-1 no está más allá del alcance de las supercomputadoras, como se había creído o por lo menos esperado. Teóricamente, se requeriría aproximadamente 2^{80} operaciones para poder encontrar una colisión.

Usando versiones de rendondeo-reducido del algoritmo, y la técnica del equipo, fue posible atacar el SHA-1 con menos de 2^{33} operaciones. Usando la misma técnica, el algoritmo completo SHA-0 pudo ser atacado en 2^{39} operaciones.

SHA-1 se había presumido ser más seguro que MD5, en donde las colisiones fueron encontradas el último año por algunas personas que reportaron el descubrimiento reciente. Además, en el último año, se han encontrado colisiones en SHA-0 por un equipo francés.

SHA-1

https://www.theregister.co.uk/2017/02/23/google_first_sha1_collision/

Security

'First ever' SHA-1 hash collision calculated. All it took were five clever brains... and 6,610 years of processor time

Tired old algo underpinning online security must die now

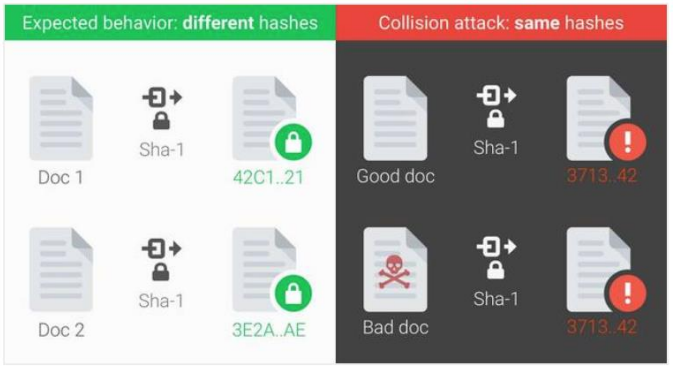
23 Feb 2017 at 18:33, John Leyden, Thomas Claburn and Chris Williams

Google researchers and academics have today demonstrated it is possible – following years of number crunching – to produce two different documents that have the same SHA-1 hash signature.

This proves what we've long suspected: that SHA-1 is weak and can't be trusted. This is bad news because the SHA-1 hashing algorithm is used across the internet, from Git repositories to file deduplication systems to HTTPS certificates used to protect online banking and other websites. SHA-1 signatures are used to prove that blobs of data – which could be software source code, emails, PDFs, website certificates, etc – have not been tampered with by miscreants, or altered in any other way.

Now researchers at CWI Amsterdam and bods at Google have managed to alter a PDF without changing its SHA-1 hash value. That makes it a lot easier to pass off the meddled-with version as the legit copy. You could alter the contents of, say, a contract, and make its hash match that of the original. Now you can trick someone into thinking the tampered copy is the original. The hashes are completely the same.

Specifically, the team has successfully crafted what they say is a practical technique to generate a SHA-1 hash collision. As a hash function, SHA-1 takes a block of information and produces a short 40-character summary. It's this summary that is compared from file to file to see if anything has changed. If any part of the data is altered, the hash value should be different. Now, in the wake of the research revealed today, security mechanisms and defenses still relying on the algorithm have been effectively kneecapped.



Google's illustration how changes made to a file can sneak under the radar by not changing the hash value

The gang spent two years developing the technique. It took 9,223,372,036,854,775,808 SHA-1 computations, 6,500 years of CPU time, and 110 years of GPU time, to get to this point. The team is made up of Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam), Ange Albertini (Google), and Yarik Markov (Google), and their paper on their work can be found here [PDF]. Its title is: "The first collision for full SHA-1."

For all the gory details, and the tech specs of the Intel CPU and Nvidia GPU number-crunchers used, you should check out the team's research paper. On a basic level, the collision-finding technique involves breaking the data down into small chunks so that changes, or disturbances, in one set of chunks is countered by twiddling bits in other chunks. A disturbance vector [PDF] is used to find and flip the right bits.

A description of Google's SHA-1 colliding PDFs can be found here. We note that the files essentially each contain a large JPEG, and the hash collision is focused on that image data. We also note that you don't have to burn another few thousand years of CPU and GPU time to create more SHA-1 collisions for simple files: thanks to Google's computations, and quirks of the PDF file format, you can from here on out produce PDFs that are visually different but still have the same SHA-1 hash value. This online tool that popped up today will easily help you create colliding PDF files.

In other words, it is now trivial for anyone to alter PDFs, webpages, and certain other simple documents, and keep the SHA-1 hash values the same, thanks to Google and co's research.

Spotlight

Speaking in Tech: A chat with Web 2.0 MySpace worm dude Samy Kamkar

The Register's guide to protecting your data when visiting the US

SHA1 collider

Quick-and-dirty PDF maker using the collision from the SHAttered paper.

Choose two image files (must be JPG, roughly the same aspect ratio). For now, each file must be less than 64kB.

Aspect ratio 512 x 512

Seleccionar archivo nada seleccionado

Seleccionar archivo nada seleccionado

Enviar

(this one is actually even simpler -- just a single comment with the entire first file in it, hence the 64k limit)

Complaints to @steike.

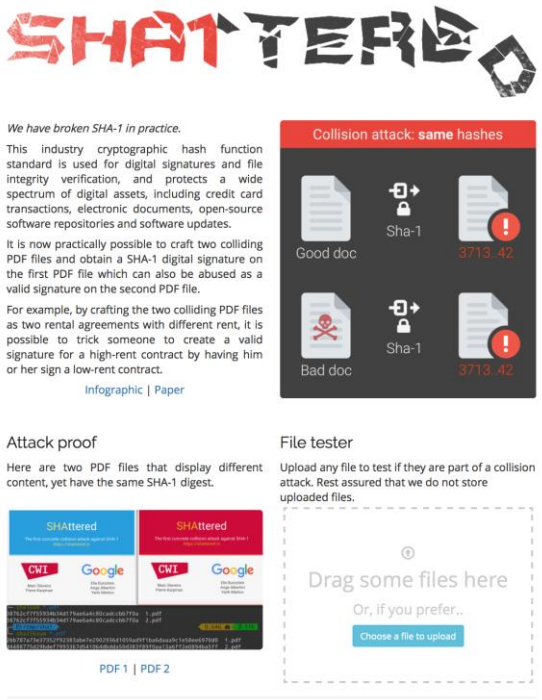
https://alf.nu/SHA1

Hola a tod@s

Hola a tod@s

2 PDFs (a.pdf y b.pdf) con el resultado de la colisión

https://shattered.io



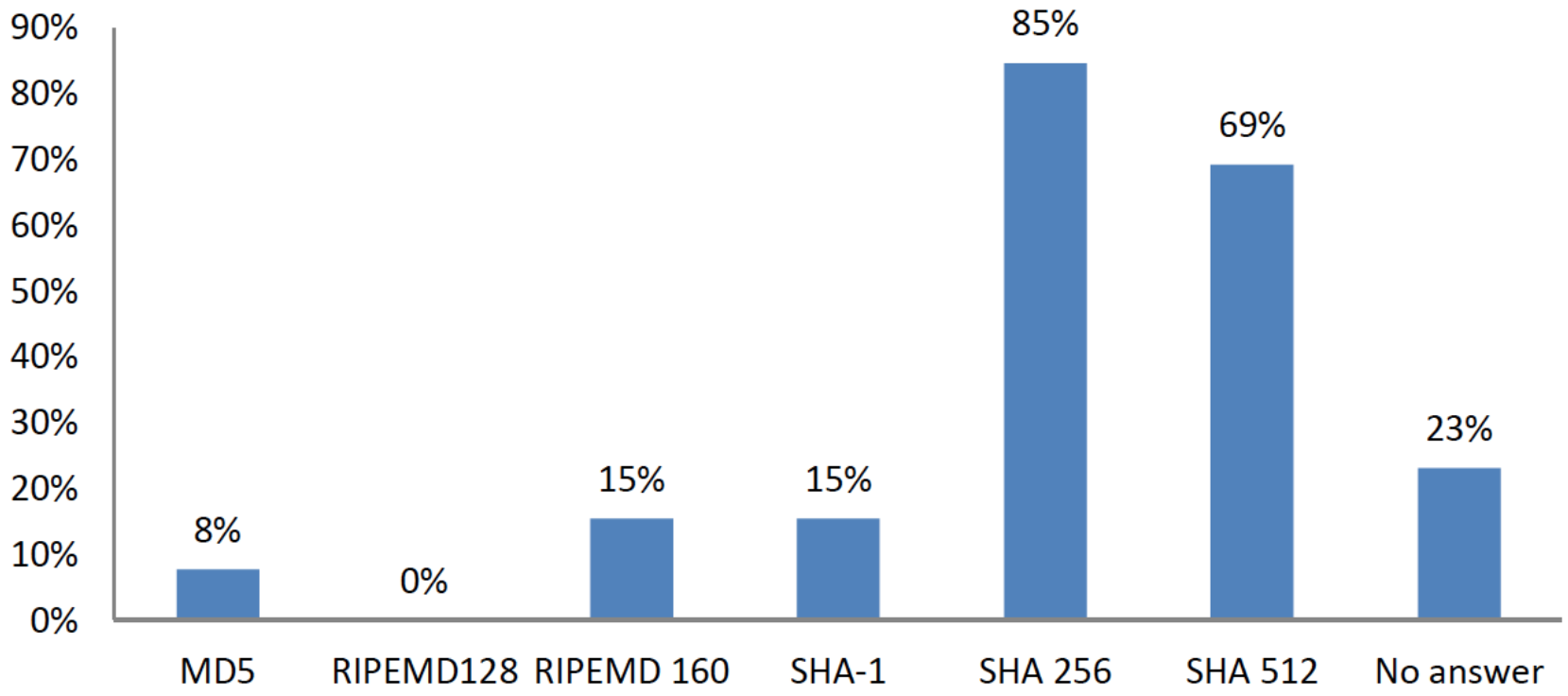
SHA-1

- Según https://shattered.io, se puede crear colisiones a diferentes tipos de artefactos:
  - Digital Certificate signatures
  - Email PGP/GPG signatures
  - Software vendor signatures
  - Software updates
  - ISO checksums
  - Backup systems
  - Deduplication systems
  - GIT (a version control system (VCS) for tracking changes in computer files and coordinating interactive work based on files)
  - ...

- SHA-2
  - En la actualidad hay una familia de cuatro algoritmos hash:
    - *SHA-224*, *SHA-256*, *SHA-384* y *SHA-512*
  - *SHA-224* (resp. *SHA-384*) es una variante de *SHA-256* (resp. *SHA-512*), en los que se trunca la salida
  - Para *SHA-224*/*SHA-256* (resp. *SHA-384*/*SHA-512*) se han encontrado algunas colisiones reducidas
- SHA-3
  - La competición organizada por *NIST* para elegir el estándar *SHA-3* finalizó en Octubre de 2012, seleccionando el algoritmo *Keccak*
- Whirlpool
  - Produce una salida de 512 bits, y no pertenece a la familia *MD-X*
  - Se construye a partir de métodos similares a los usados en *AES*, y es una buena alternativa de uso para garantizar la diversidad de algoritmos

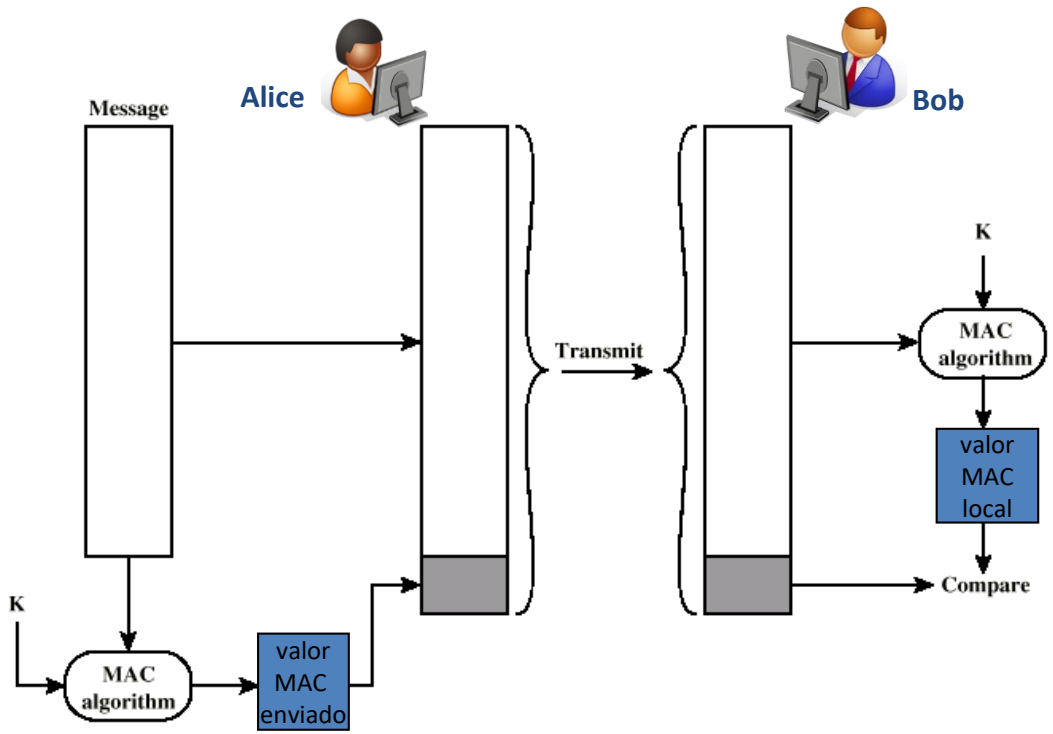
- Comentarios generales:
  - El desarrollo de la funciones hash es, probablemente, el área de la Criptografía que ha atraído más atención durante la pasada década
    - debido a las mejoras realizadas en el criptoanálisis de funciones hash, así como por la competición del *SHA-3* para diseñar un sustituto al conjunto de funciones existentes
  - La mayoría de las funciones hash existentes derivan mucho de los criterios de diseño de la función hash *MD-4* (familia *MD-X*)
    - Esta familia incluye *MD-4*, *MD-5*, *RIPEMD-128*, *RIPEMD-160*, *SHA-1* y *SHA-2*
  - Las salidas de las funciones hash deberían ser como mínimo de 160 bits de longitud para las aplicaciones heredadas, y de 256 bits para todas las aplicaciones nuevas

Primitive	Output Lenfth	Recommendation Legacy	Recommendation Future
SHA-2	256, 384, 512	✓	✓
SHA-3	?	✓	✓
Whirlpool	512	✓	✓
SHA-2	224	✓	✗
RIPEMD-160	160	✓	✗
SHA-1	160	✓	✗
MD-5	128	✗	✗
RIPEMD-128	128	✗	✗


90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
8%
0%
15%
15%
85%
69%
23%
MD5
RIPEMD128
RIPEMD 160
SHA-1
SHA 256
SHA 512
No answer

Funciones MAC

- Un código de autenticación de mensaje, o función MAC, es un concepto que evoluciona a partir del concepto de función hash
- Concretamente, una función MAC toma como entrada un mensaje *M* y una clave *K*, y produce un valor hash (que en este caso se denomina valor MAC)
- Ejemplo de uso:



- El esquema anterior soluciona el problema anteriormente mencionado de que la función hash, aunque proporciona integridad de datos, no proporcionan autenticación
  - Ahora la autenticación se consigue porque *Alice* y *Bob* comparten la clave simétrica *K*
  - *Bob* está convencido de que la información proviene de *Alice* porque es la única (además de él) que conoce *K*
- Las funciones MAC más conocidas entran en dos categorías:
  - Basadas en funciones hash (por ejemplo: *HMAC*, *UMAC*)
  - Basadas en algoritmos de cifrado en bloque (por ejemplo: *EMAC*, *AMAC*, *CMAC*)

Referencias bibliográficas

Bibliografía básica

- *"Cryptography and Network Security: Principles and Practice"*

  William Stallings

  Prentice Hall, 2014 (6ª edición)

- "Handbook of Applied Cryptography"

  Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone,

  CRC Press, 1996

- "Applied Cryptography: Protocols, Algorithms, and Source Code in C"

  Bruce Schneier

  Wiley, 1996 (2ª edición)

Referencias

- *"Data Encryption Standard (DES)"*, FIPS PUB 46-3, NIST, 1999
- *"Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher"*, NIST, 2012
- *"Advanced Encryption Standard (AES)"*, Federal Information Processing Standards Publication 197, NIST, 2001
- *"Recommendation for Block Cipher Modes of Operation Methods and Techniques"*, NIST, 2001
- *"Record Breaking DES Key Search Completed"*

http://www.cryptography.com/technology/applied-research/research-efforts/des-key-search.html

Otras referencias

- “The Washington-Moscow Hot Line. A Compilation of Extracts”, edited by Jerry Proc
  - http://www.jproc.ca/crypto/hotline.html