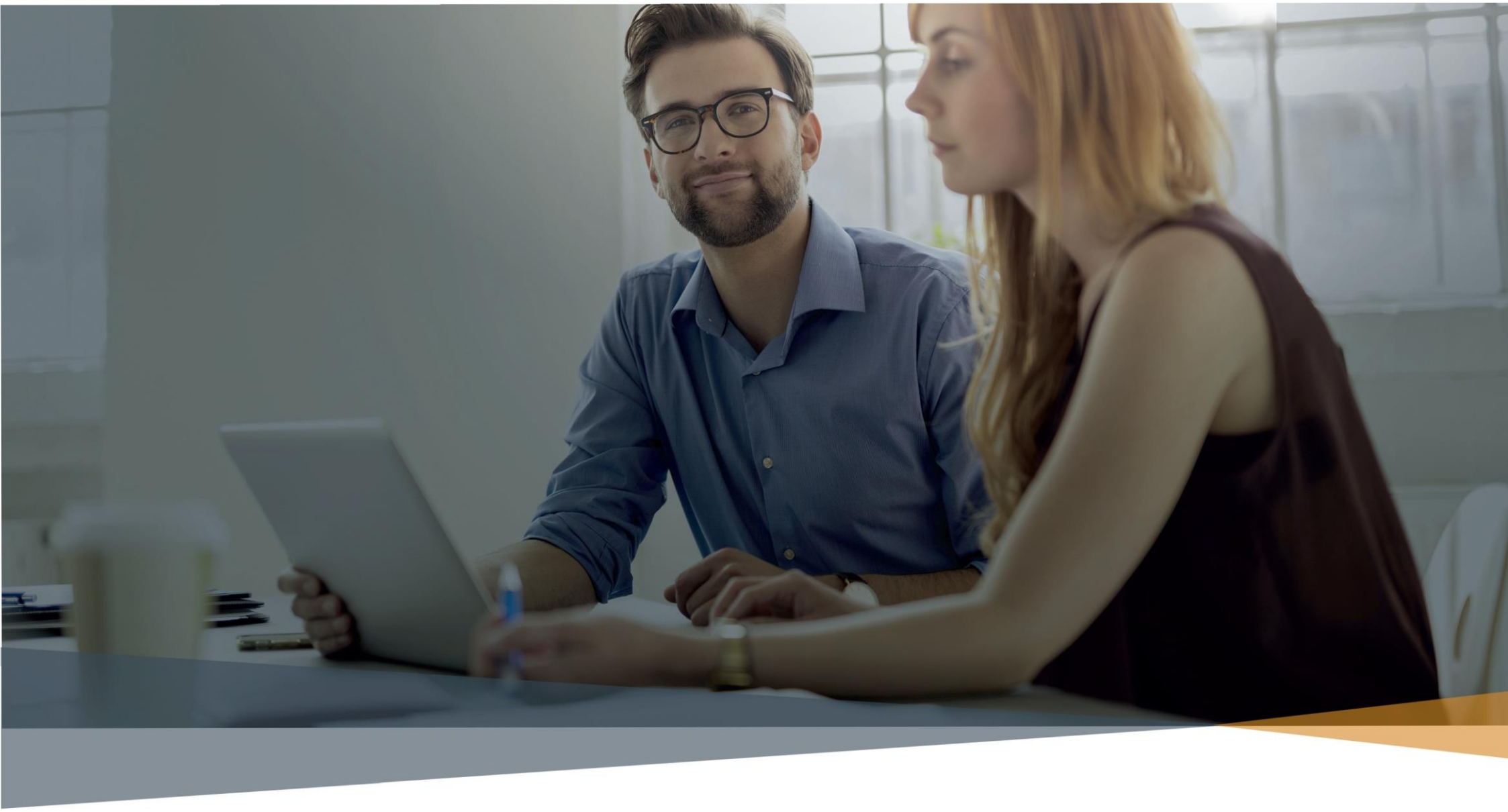
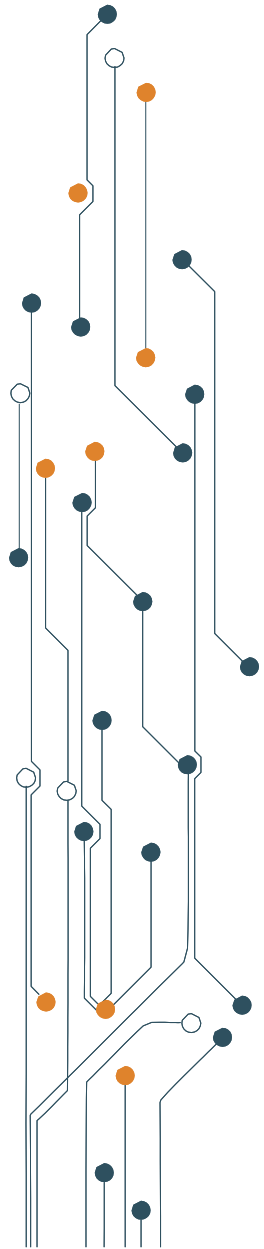


Funciones hash

7ife{iinict¡ EDUCACIÓN DIGITAL

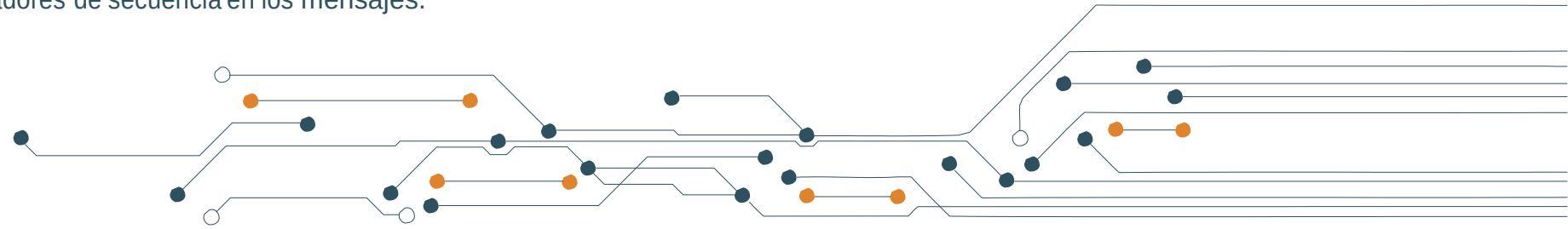
Índice



1. Esquemas de autenticación

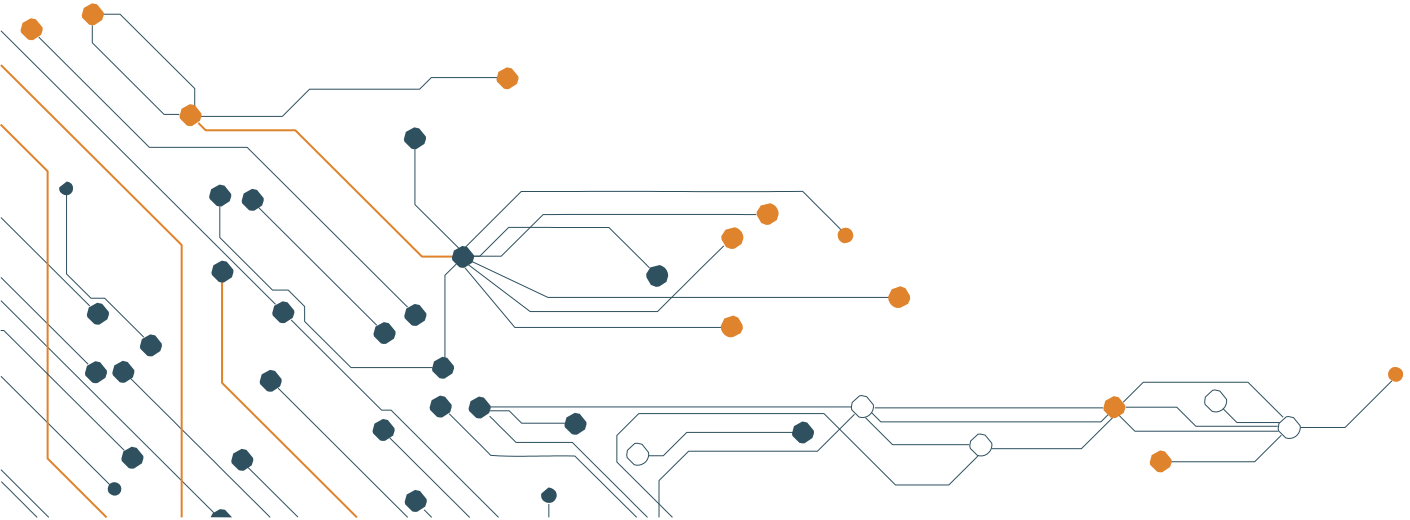
Tanto para verificar la identidad de un usuario como para comprobar la integridad de un mensaje, se utilizan una serie de mecanismos de autenticación. En el caso de un usuario, dicha autenticación persigue comprobar ante el sistema u otros usuarios que su identidad es verdadera. En el caso de un mensaje, se persigue que éste se reciba exactamente igual a como fue emitido o bien se almacene sin modificaciones, en el fondo que sea íntegro. Por lo tanto, los mecanismos de autenticación tratarán de evitar los siguientes cinco ataques:

1. La usurpación de identidad del usuario. Para que no se suplante la identidad del usuario, se aplicarán métodos de autenticación basados en algo secreto que se conoce (por ejemplo una password), algo personal que se posee (por ejemplo una tarjeta de coordenadas con números) o bien algo que se es (por ejemplo la huella dactilar). Ejemplos cotidianos de estos métodos son el acceso a una cuenta de correo electrónico vía login y password, las operaciones de la banca online vía tarjeta de coordenadas y el desbloqueo de algunos teléfonos móviles mediante la lectura de huellas dactilares. Obviamente existen muchos otros sistemas de autenticación. Es menester indicar aquí que autenticar es la acción de comprobar que la identidad aportada es la correcta e identificar consiste en aportar las pruebas necesarias que permitan dicha comprobación.
2. La modificación de contenidos del mensaje. Para tener certeza de que el mensaje es íntegro, es decir exactamente igual al que ha enviado o creado su autor, se aplican métodos de autenticación basado en el uso de las funciones hash.
3. La inserción de mensajes de un emisor fraudulento. Para evitarlo se usan los certificados digitales, documentos digitales que permiten comprobar ante una tercera parte de confianza que uno es quien dice ser.
4. La modificación de la secuencia de mensajes, incluyendo el borrado y la inserción de contenidos. Para evitar este ataque, pueden usarse indicadores de secuencia en los mensajes.



5. La modificación del momento de envío o de llegada de los mensajes. Para evitar que un mensaje se envíe o se haga llegar al destino más tarde de lo que le correspondería, es decir se retrasa el envío la solución está en el uso del sellado de tiempo mediante una autoridad que determina que los tiempos son los correctos. Se trata de una técnica criptográfica que se usa para proporcionar información temporal sobre los documentos electrónicos, permitiendo validar el instante en el que un documento ha sido creado y comprobar que dicho documento no ha sido alterado desde entonces.

En este capítulo se tratará solamente de la autenticación de los mensajes, es decir de su integridad, mediante el uso de funciones hash.



2. Propiedades de las funciones hash

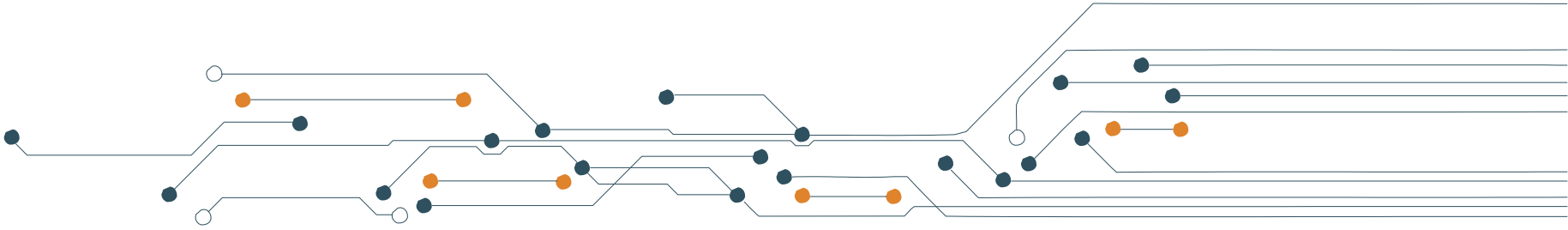
Las funciones hash son algoritmos que al aplicarlos sobre un mensaje, archivo o texto M, entregan un resumen de x bits conocido como h(M). Se trata de un número que a modo de huella digital representará a dichos documentos de forma supuestamente única. A diferencia de la popular familia de aplicaciones tipo zip para la compresión de archivos, las funciones hash entregan siempre un resumen de un número fijo de bits, independientemente del tamaño de ese archivo. Al carecer de claves, las funciones hash no pueden considerarse como algoritmos de cifra.

Las funciones hash juegan dos papeles muy importantes en la criptografía:

a. Como reducen el tamaño de un documento a una huella digital de un conjunto pequeño de bits, normalmente unas centenas de bits, permiten adaptar ese documento para su firma digital, en tanto los algoritmos de clave pública que se usan para la firma digital son muy lentos y no sería por tanto eficiente firmar documentos de miles o millones de bytes.

b. Como una de las propiedades de estas funciones es que si se altera algún bit del mensaje, accidentalmente durante el tránsito o bien de forma deliberada, el resumen de ese otro mensaje será completamente distinto, permitirá autenticar el mensaje de la siguiente manera:

- Se envía al destinatario el mensaje M (supuestamente no es secreto) junto al hash calculado en origen.
- El destinatario recibe el mensaje, que puede o no ser el mismo enviado y que llamaremos M', y calcula en destino el hash del mensaje recibido h(M').
- Si el valor h(M) recibido coincide con el valor h(M') calculado, se acepta el mensaje como íntegro.



Las figuras 7.1 y 7.2 muestran el uso de la función hash para autenticar o comprobar la integridad de los mensajes. La primera sin uso de cifrado, si bien emisor y receptor deben compartir un secreto, y la segunda mediante el uso de un algoritmo de cifra de clave pública, por lo que además de comprobar que el mensaje es íntegro, se podrá comprobar que el emisor es quien dice ser por el tipo de cifra utilizado.

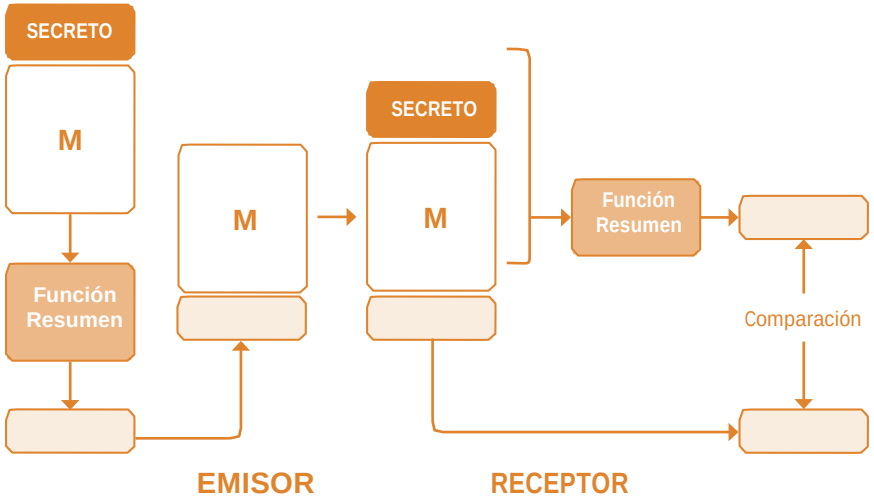


Figura 7.1. Autenticación vía hash sin cifra.

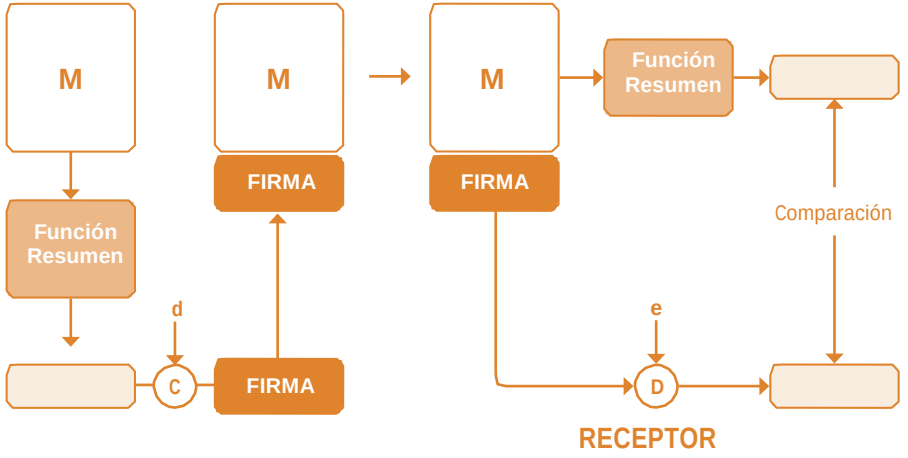
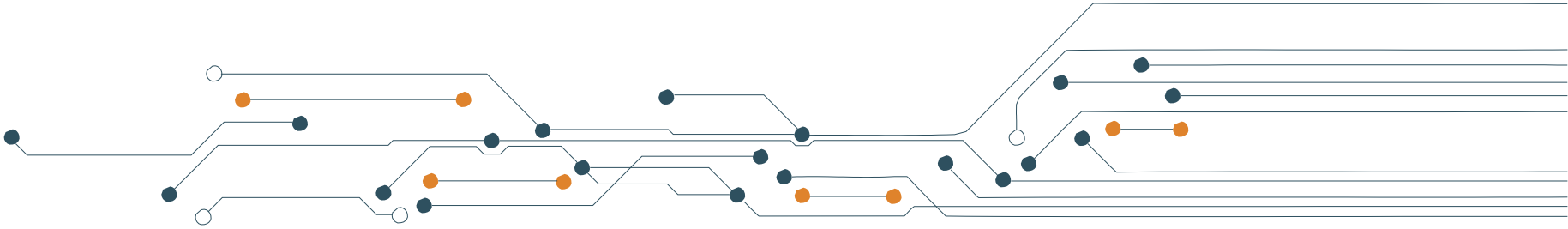


Figura 7.2. Autenticación vía hash con cifra asimétrica o clave pública.



Hay muchas aplicaciones que requieren contraseñas, como pueden ser los sistemas de foros o servidores de correo, donde lo que se almacena en el servidor no es la contraseña sino el hash de esa contraseña. Esto no se hace con el propósito de comprimir la contraseña, que seguramente será más corta que el hash generado, sino con el propósito de que si alguien tiene acceso al almacén de contraseñas, no pueda ver la contraseña como texto en claro y, por tanto, no pueda tener acceso al sistema, a no ser que rompa ese contraseña por fuerza bruta u otro tipo de ataque si conoce qué función hash se ha empleado.

Las funciones hash deberán cumplir las siguientes 6 propiedades:

1. La unidireccionalidad. Esto quiere decir que, conocido un resumen h(M), debe ser computacionalmente imposible encontrar el mensaje M a partir de dicho resumen.
2. La compresión. Como lo normal es que el mensaje M tenga una longitud de bits mayor que la que entrega el hash, la función hash actuará normalmente como un compresor. Si el mensaje tuviese menos bits que el hash, el resumen siempre tendrá ese valor fijo de bits.
3. La facilidad de cálculo. Debe ser fácil calcular h(M) a partir de un mensaje M, y así sucede en la práctica. Además, el hash es una función muy rápida.
4. La difusión de bits. Esta propiedad, también conocida como efecto de avalancha, indica que el resumen h(M) debe ser una función compleja de todos los bits del mensaje M. Por lo tanto, si se modifica tan sólo un bit del mensaje M, el nuevo hash debería cambiar aproximadamente en la mitad de sus bits con respecto al anterior.
5. La resistencia débil a colisiones. Esta propiedad se cumplirá si es computacionalmente imposible que, conocido un mensaje M, podamos encontrar otro mensaje M' tal que h(M) = h(M').
6. La resistencia fuerte a colisiones. En este caso, nos indica que será computacionalmente imposible encontrar un par aleatorio de mensajes (M, M') de forma que h(M) = h(M'). Si el hash no cumple con esta propiedad, se facilitará su ataque por la paradoja del cumpleaños.

Esta última es una condición muy importante porque se trata justamente del talón de Aquiles de estos algoritmos, como veremos en el apartado quinto de este capítulo dedicado a los ataques a las funciones hash.

3. Función hash MD5

Diseñada en 1992 por Ron Rivest, esta función resumen está obsoleta desde mediados de 2005 al detectarse fallos en su diseño. No obstante, se siguió utilizando hasta bien entrada la década de 2010, especialmente en aplicaciones y para comprobar integridad de archivos y en redes P2P, monedas, comercio y banca online.

Procesa mensajes en bloques de 512 bits y produce una salida de 128 bits. En primer lugar expande el mensaje M hasta una longitud exactamente 64 bits inferior a un múltiplo de 512 bits. Para ello añade un 1 seguido de tantos ceros como sean necesarios y reserva los últimos 64 bits para añadir la información sobre la longitud del mensaje en bits.

Como se observa en la figura 7.3, el algoritmo comienza con cuatro vectores iniciales A, B, C, D de 32 bits, cuyo valor no es secreto. A estos vectores y al primer bloque de 512 bits se le aplicarán 64 operaciones de 32 bits que se muestran en la figura 7.4.

Terminadas esas 64 operaciones con el primer bloque de texto en claro, se obtienen cuatro nuevos vectores A'B'C'D' que serán los vectores de entrada para el siguiente bloque de 512 bits, acción que repite con los restantes bloques del documento encadenando los sucesivos vectores A, B, C, D. El último bloque dará el resumen final h(M) del mensaje M.

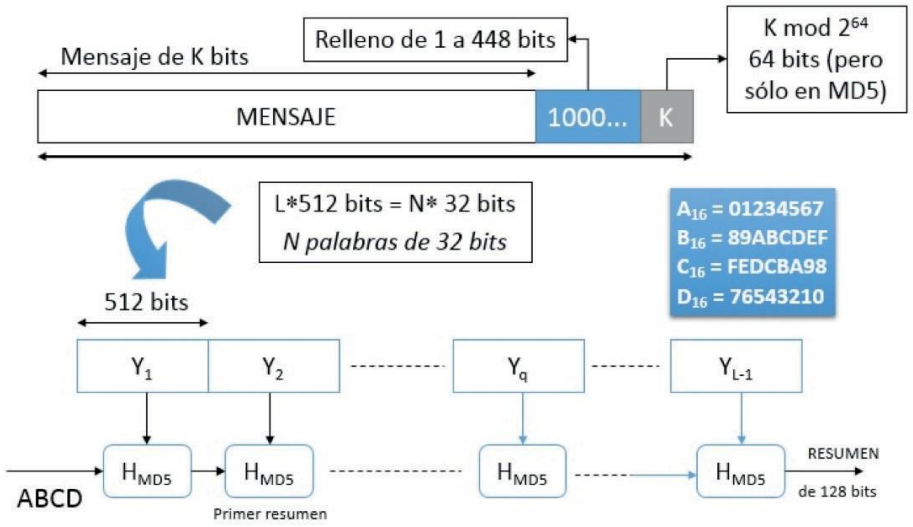


Figura 7.3. Esquema de la función hash MD5.

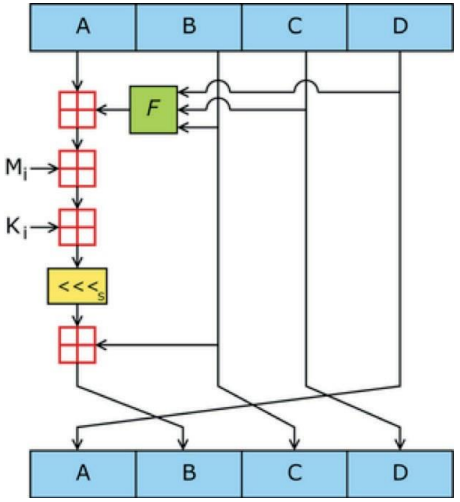


Figura 7.4. Operaciones en MD5.

Dependiendo de la ronda, la función F puede ser F, G, H, I:

1ª ronda	F = (B AND C) OR (NOT B AND D)	3ª ronda	H = (B XOR C XOR D)
2ª ronda	G = (B AND D) OR (C AND NOT D)	4ª ronda	I = (C XOR (B OR NOT D))

La suma es mod 232, Mi son 16 palabras de 32 bits del bloque de Yi de 512 bits de entrada por ronda, Ki es una constante de 32 bits, diferente en cada operación, que viene dada en tablas, y s es un desplazamiento diferente en cada ronda y operación, que también viene dado en tablas. Todos los valores son públicos y opera los bytes en formato little endian.

4. Función hash SHA-1

SHA-1, Secure Hash Algorithm, propuesto inicialmente en 1993 por la NSA y adoptado como estándar en 1995 por el NIST, sustituye a SHA-0 en el que se habían detectado problemas. SHA-1 es muy similar a MD5; también trata bloques de 512 bits de mensaje, si bien genera un resumen de 160 bits pues cuenta con 5 vectores de 32 bits, ABCDE, en vez de los 4 de MD5. En este caso, para cada bloque se realizan un total de 80 operaciones con palabras de 32 bits, organizadas en 4 rondas de 20 vueltas cada una.

Terminadas las primeras 80 vueltas, los vectores iniciales ABCDE habrán cambiado varias veces de valor, y serán los nuevos 5 vectores A'B'C'D'C' que se mezclarán con el segundo bloque de 512 bits de entrada. Esta acción se va encadenando con los siguientes bloques, hasta que el último valor de los vectores ABCDE es el resumen de 160 bits de todo el documento.

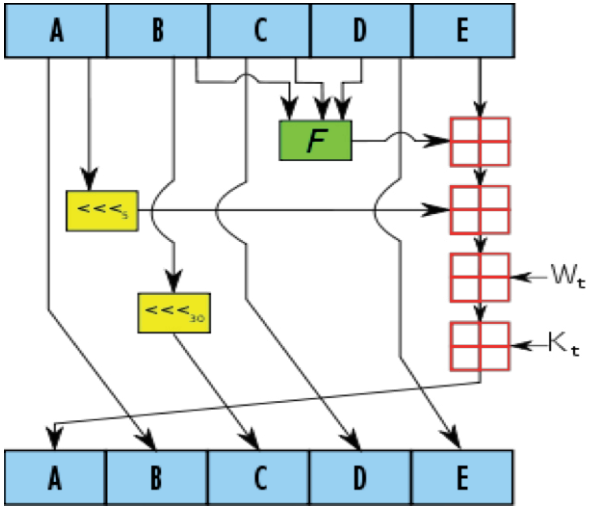


Figura 7.5. Operaciones en SHA-1.

Los vectores son: A = 67452301; B = EFCDAB89; C = 98BADCFE; D = 10325476; E = C3D2E1F0. En este caso los bytes se tratan en formato big endian.

Dependiendo de la ronda, la función F puede ser F, G, H, I, en donde:

Los vectores son: A = 67452301; B = EFCDAB89; C = 98BADCFE; D = 10325476; E = C3D2E1F0. En este caso los bytes se tratan en formato big endian.

Dependiendo de la ronda, la función F puede ser F, G, H, I, en donde:

1ª ronda F = (B AND C) OR (NOT B AND D)

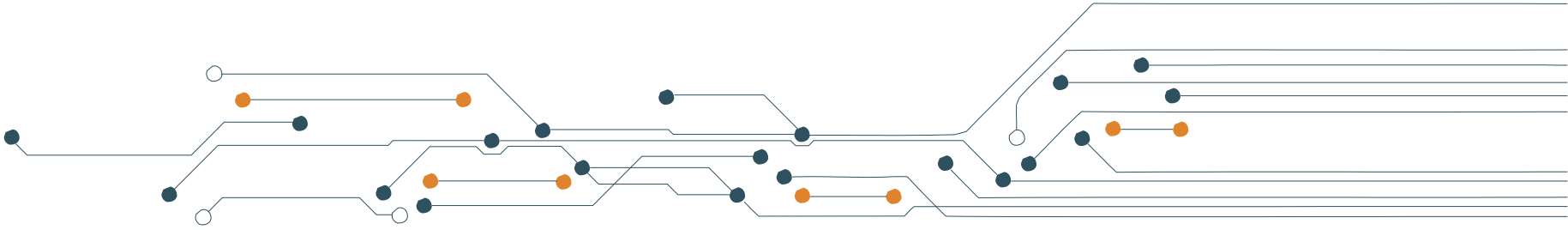
2ª ronda G = (B XOR C XOR D)

3ª ronda H = (B AND C) OR (B AND D) OR (C AND D).

4ª ronda I = (B XOR C XOR D)

La suma es mod 232, Wt son 80 palabras de 32 bits del bloque de Y de 512 bits de entrada, Kt es una constante de 32 bits y existen 4 valores diferentes, una para cada ronda. Por último, se aplican desplazamientos de 5 bits en el vector A y 30 bits en el vector B.

A cada bloque del texto de entrada se le aplicarán 20 vueltas con 4 funciones distintas, de forma tal que el número total de vueltas por bloque será igual a 20x4 = 80. Para obtener 20 palabras de 32 bits desde una cadena de tan sólo 512 bits, se procede de la siguiente manera: cada bloque de 16 palabras de 32 bits del mensaje se expandirá en 80 palabras, de forma que las palabras 0 a 15 se corresponden a los 512 bits del bloque, y las 64 palabras restantes, de la 16 hasta la 79, se obtienen mediante una operación lógica entre las palabras anteriores.



5. Funciones hash SHA-2, SHA-3 y ataques

A la vista del carisma que tomaban los ataques a las funciones hash, especialmente contra MD5 y en parte contra SHA-1, voces reconocidas a nivel mundial pedían en 2005 un concurso similar al del algoritmo AES para sustituir al DES (situación en la que el NIST estuvo atento) para encontrar un nuevo estándar de funciones resumen. Finalmente el NIST publica una nota de prensa el 23 de enero 2007 con un borrador para el nuevo estándar hash que se llamará SHA-3. El 30 de abril se cierra el plazo para el envío de comentarios.

En diciembre de 2010 el NIST selecciona en la tercera ronda a cinco finalistas, BLAKE, Ghøstl, JH, Keccak y Skein, y en octubre de 2012 se elige como ganador de SHA-3 al algoritmo Keccak, declarado como estándar mundial en agosto de 2015 por ese mismo organismo.

Como desde que un algoritmo se declara como estándar mundial hasta que su uso puede considerarse masivo, pasan fácilmente 10 años (e.g., el AES sólo comienza a usarse de manera masiva en comunicaciones SSL/TLS, diez años después de ser declarado nuevo estándar mundial de cifra simétrica), la NSA desarrolla en 2001 una familia de algoritmos hash conocida como SHA-2, que consiste en un conjunto de cuatro funciones hash de 224, 256, 384 y 512 bits, en este último caso con palabras de 64 bits.

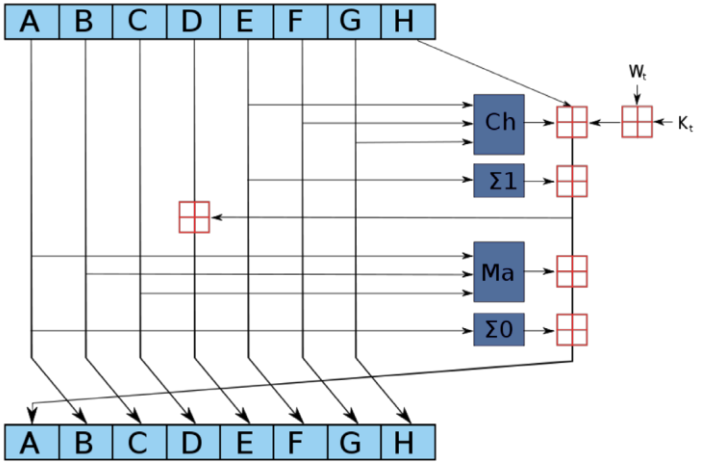


Figura 7.5. Operaciones en SHA-1.

Como se observa en la figura 7.6, tenemos 8 vectores de 32 bits cada uno, por lo que el resumen será de 8x32 = 256 bits.

Las operaciones en color azul son similares a las de MD5 y SHA-1, W_t son 64 palabras de 32 bits que se obtienen de una forma similar a como lo hace la expansión en SHA-1, se cuenta con 64 constantes K_t y la suma sigue siendo mod 2^{32}.

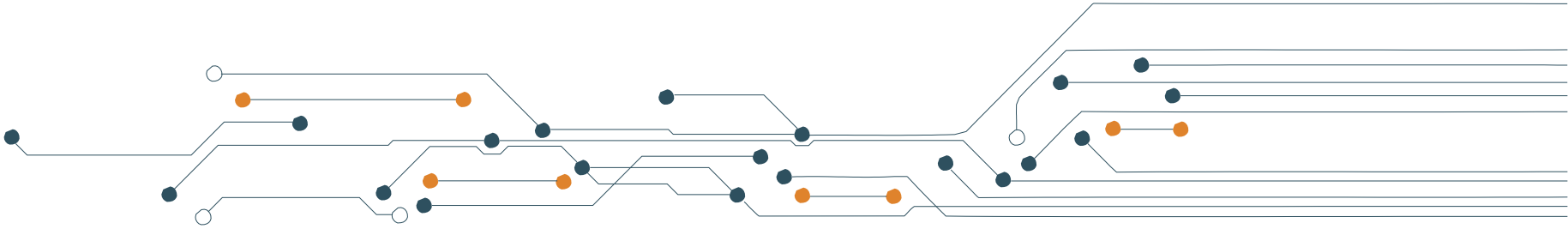
SHA-224 y SHA-256 usan bloques de entrada de 512 bits, con 64 vueltas. En el caso de SHA-384 y SHA-512, se trabaja con palabras de 64 bits, la entrada es de 1.024 bits y se realizan 80 vueltas. Los cuatro algoritmos usan las mismas funciones lógicas que SHA-1, a las que se añade la función Shr, que es un desplazamiento a la derecha no circular.

En cuanto al algoritmo SHA-3, comentar que no tiene un esquema como los de la familia SHA y MD5, que todos eran muy similares en su concepción, sino que se trata de un sistema conocido como esponja en el que la entrada es una cadena de bits de cualquier longitud, que son absorbidos por esa esponja, y se obtiene una salida del tamaño de bits que se desee exprimiéndola.

Los algoritmos de resumen deben ser capaces de resistir un ataque basado en la paradoja del cumpleaños, cuya definición se da a continuación.

La paradoja del cumpleaños indica que dos personas al azar pueden estar de cumpleaños en la misma fecha con una probabilidad mayor que el 50%, si ese grupo lo forman 23 personas como media. Este problema matemático, que en realidad no es ninguna paradoja, permite hacer ataques criptográficos a las funciones hash y también a RSA.

Así, para tener confianza, es decir una probabilidad mayor o igual que el 50%, en encontrar dos mensajes distintos con el mismo resumen h(M) de n bits, no habrá que buscar en el espacio 2^n, que sería lo esperado, sino que bastará con una búsqueda en el espacio $2^{n/2}$, un espacio muchísimo menor, por lo que la complejidad algorítmica se reduce de forma drástica.



En la tabla 7.1 se muestra la fortaleza de las funciones hash más conocidas, basadas en este ataque, sus valores teóricos y los que en realidad hoy se han alcanzado. Observa que MD5 ha podido ser atacado en junio de 2007 con tan sólo 2^{24} intentos en un Pentium 4 de 2,6 GHz encontrando una colisión en pocos segundos, y SHA-1 con 2^{57} en noviembre de 2010 y 2^{52} según unos investigadores australianos en 2009.

Función hash	Fortaleza teórica	Fortaleza real (2016)
MD5	2^{64}	2^{24}
SHA-1	2^{80}	2^{57}
SHA-256	2^{128}	2^{128}
SHA-512	2^{256}	2^{256}
SHA-3 256	2^{128}	2^{128}
SHA-3 512	2^{256}	2^{256}

Tabla 7.1. Fortaleza de las funciones hash a los ataques por paradoja del cumpleaños.

A diferencia de MD5, definitivamente roto, el hash SHA-1 podría considerarse hoy en día medianamente seguro. De hecho, se sigue usando en certificados digitales X.509, aunque tenga fecha de caducidad a partir del 1 de enero de 2017. Por ello, es recomendable usar ya la familia SHA-2 o bien el nuevo estándar SHA-3.


Telefónica