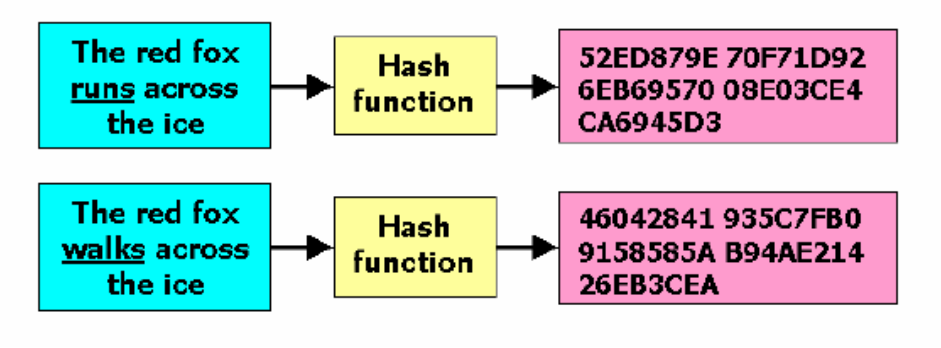
Fundamentos de criptografía
Criptografía moderna

Garantía y Seguridad de la Información.

Transformaciones irreversibles

- Transformaciones **irreversibles**:
  - a partir de un bloque de datos se produce otro que proviene *inequívocamente* del primero
    - a partir de él resulta imposible conocer el original.
  - mediante algoritmos Hash.



Algoritmos de transformación Hash

- *Propiedad de compresión*
  - El resultado (*resumen, digest, valor hash*) tiene siempre el mismo tamaño, independientemente del tamaño de los datos de entrada.
- *Propiedad de facilidad de cómputo*
  - Debe ser poco costoso de aplicar, computacionalmente.

Propiedad de irreversibilidad

- Dado un *hash*, es "imposible" encontrar un mensaje que produzca el mismo hash.
  - Dado y, no se puede (computacionalmente) encontrar un M tal que $h(M) = y$
- Resistente a la preimagen (one way hash function)

Propiedad débil de resistencia a la colisión

- Dados un mensaje y su *hash*, es "imposible" encontrar otro mensaje que produzca el mismo *hash*.
  - Dado M, no se encuentra M' / $h(M) = h(M')$
- Resistente a la segunda preimagen (weak one way hash function, OWHF)

Propiedad fuerte de resistencia a la colisión

- Es "imposible" encontrar dos mensajes que produzcan el mismo *hash*.
  - No se encuentra $(M, M') \,/\, h(M) = h(M')$
- Resistente a colisiones (strong one way hash function, CRHF)
- *Ataque por la "Paradoja del cumpleaños"*

Propiedades criptográficas

- Para cualquier valor hash dado y, es computablemente inviable encontrar M tal que $h(M) = y$
- Para cualquier M dado, es computacionalmente inviable encontrar $M \neq M'$ con $h(M) = h(M')$
- Es computacionalmente inviable encontrar cualquier par (M, M') con $M \neq M'$ tal que $h(M) = h(M')$

Fortaleza de una función Hash

- Una función hash que devuelve un resumen de 4 bits (0000, 0001, 0010, …, 1111)
  - Dado M, ¿cuál es la probabilidad de que M' tenga el mismo resumen?
  - ¿Cuál es la probabilidad de que dos mensajes distintos, M y M', tengan igual función hash?
- Con un resumen de 128 bits, la probabilidad disminuye a $1/2^{128}$

Paradoja del cumpleaños

- ¿Cuál es la probabilidad de que en un grupo de n personas haya dos personas que cumplen los años el mismo día del mismo mes?

- ¿Cuántas personas hace falta meter en una sala para que la probabilidad de que dos de ellas cumplan años el mismo día sea superior al 50%?
  - Basta con 23 personas.

Paradoja del cumpleaños

- Sea $n \leq 365$ (en grupos mayores de 365 la probabilidad es 1).
  - Sea p la probabilidad de que no haya dos personas que cumplen los años en el mismo día.
  - $1 - p$ es la probabilidad de que haya al menos dos personas

Paradoja del cumpleaños

- La probabilidad de que una persona no coincida con otra en su cumpleaños es casos favorables (todos los días del año excepto uno) entre casos posibles:

$$364/365$$

- Si incluimos otra persona, la probabilidad de que no coincida es

$$363/365$$

Paradoja del cumpleaños

- Para n personas (sucesos independientes):

$$\frac{364}{365} \cdot \frac{363}{365} \cdots \cdot \frac{365 - n}{365}$$

- Expresado en factoriales

$$\frac{365!}{365^n (365 - n)!}$$

Paradoja del cumpleaños

- Probabilidad de que no coincida el cumpleaños de dos personas, en grupo de n

$$\frac{365!}{365^n(365-n)!}$$

Paradoja del cumpleaños

- La probabilidad de que haya al menos dos personas que cumplan el mismo día, en grupo de n

$$1 - \frac{365!}{365^n (365-n)!} \approx \sqrt{n}$$

- Para n=15, es 0,245
- Para n=23, pasamos el 50%, es 0,507
- Para n=80, el resultado es 0,99991, más del 99%

Ataque del cumpleaños

- Para tener confianza en encontrar dos mensajes con el mismo resumen, bastará una búsqueda en un espacio de tamaño $2^{n/2}$

$$1 - \frac{2^n!}{2^{nn}\,(2^n - k)!} \geq 0{,}5$$

$$k \approx 2^{n/2}$$

- Confianza: probabilidad $\geq 0{,}5$
- La búsqueda no se hace en un espacio de tamaño 2^n

Ataque del cumpleaños

- Si disponemos de $2^{n/2}$ bloques de datos al azar, la probabilidad de tener dos bloques con el mismo valor *hash* (una colisión) es superior al 50%
- Si el *hash* tiene una longitud de 160 bits, un ataque de fuerza bruta exige probar en promedio 2^{80} posibles valores de hash para encontrar una colisión.
  - Es como reducir la longitud efectiva del *hash* a la mitad.
  - Por lo menos *hashes* de 256 bits.

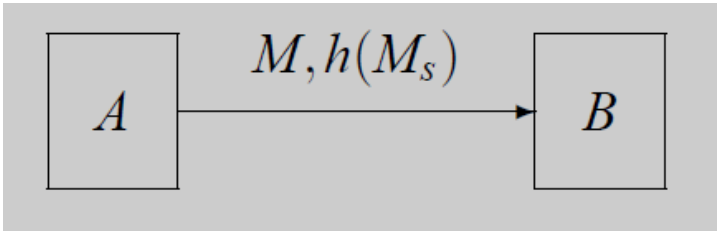
Utilidad de las transformaciones Hash

- Nos permiten **verificar** la fuente del mensaje y su **integridad**.

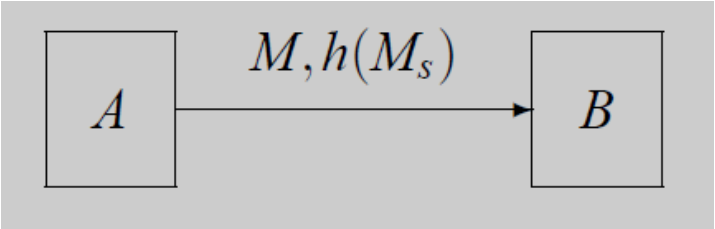
- Garantizan la **autenticidad** de los mensajes.
  - Un buen resumen hash es una "huella" infalsificable de un bloque de datos

Servicio de autenticidad (I)

- A quiere mandar un mensaje M a B
  - concatena el original con la clave s compartida: $M_s = (M, s)$
  - calcula $h(M_s)$
- Manda el mensaje M y el *resumen* de M_s



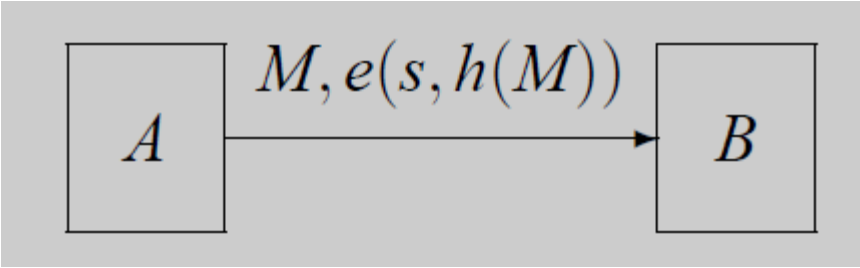
Servicio de autenticidad (y II)



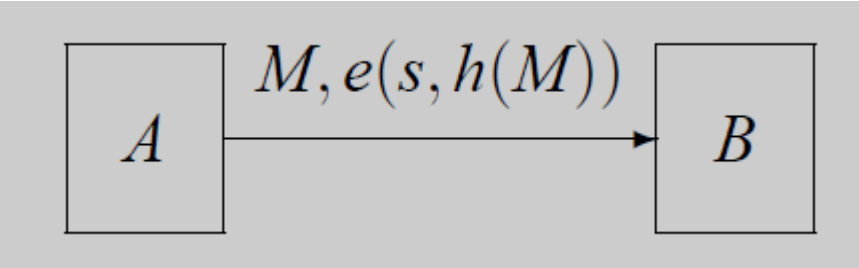
- B verifica que el *resumen* corresponde a M_s
  - lo ha generado alguien que conoce la clave secreta s (que debería ser A),
  - nadie ha modificado el mensaje.

Otra implementación del servicio de autenticidad (I)

- A calcula el *resumen* del mensaje M y lo cifra utilizando s, clave compartida



Otra implementación del servicio de autenticidad (y II)

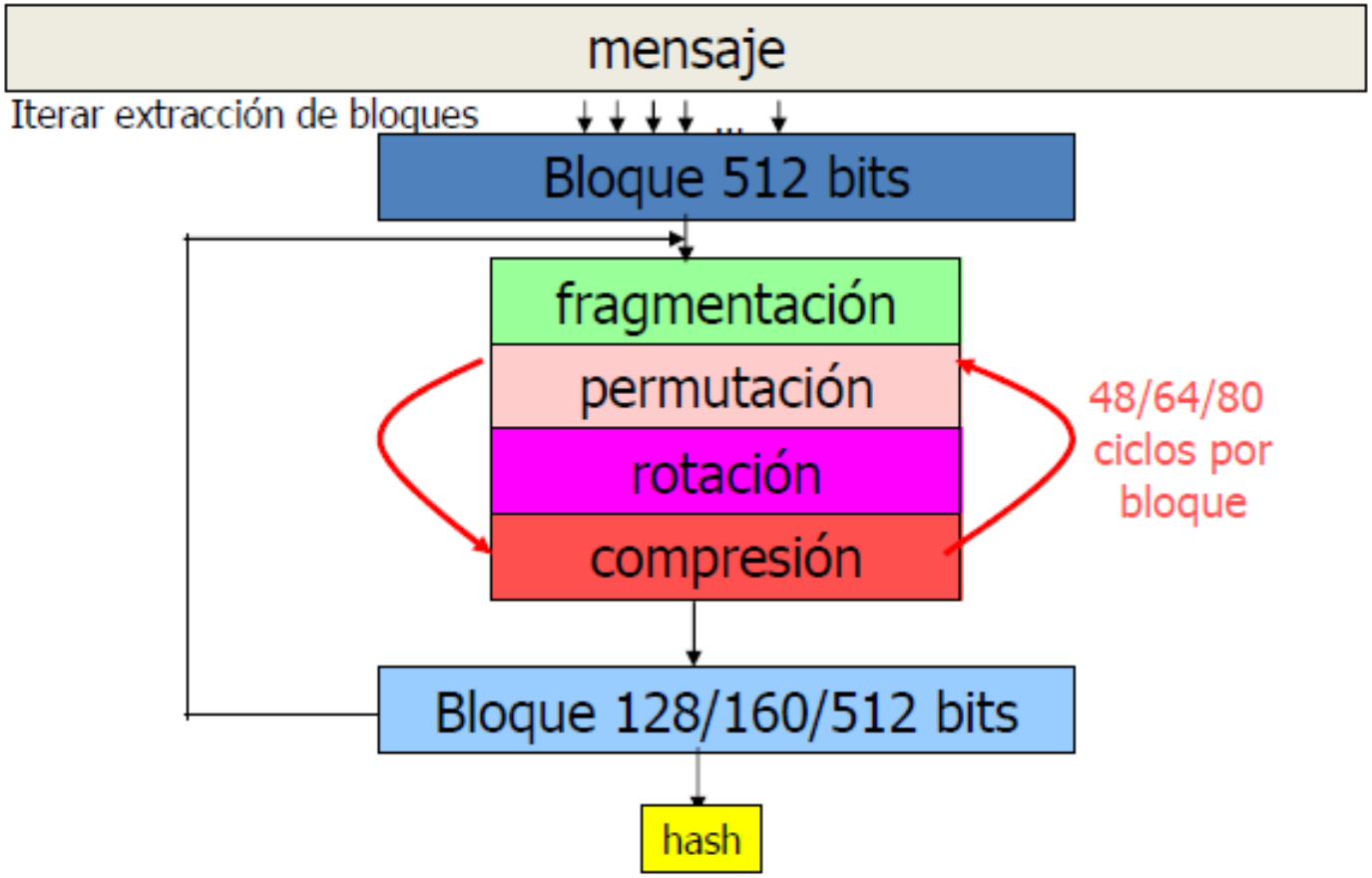


- B recupera el resumen enviado, descifrándolo con s y lo compara con el resumen correcto de M.

Algoritmos de transformación Hash

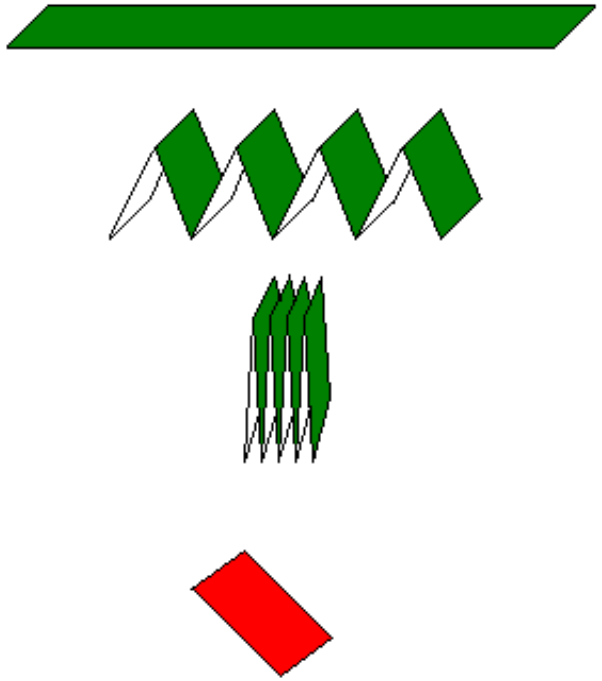
- MD5 (message digest v5)
  - bloques de 128 bits. Falla en la resistencia a colisiones (2008)
- SHA-1 (secure Hash Algorithm v1)
  - bloques de 160 bits. Falla en la resistencia a colisiones (2020)
- SHA-2 (familia de algoritmos: -256, -384, -512)
- SHA-3 (familia de algoritmos: -256, -384, -512)
- Whirlpool
  - bloques de 512
- RIPEMD-160
- Tamaño de *resumen* mayor; más resistente.

Modelo básico de transformaciones Hash



Modelo básico de transformaciones Hash

- De forma iterativa:
- La entrada en el paso i es función del
  - i-ésimo bloque del mensaje, M_i
  - la salida del paso $i-1$



Función MD5

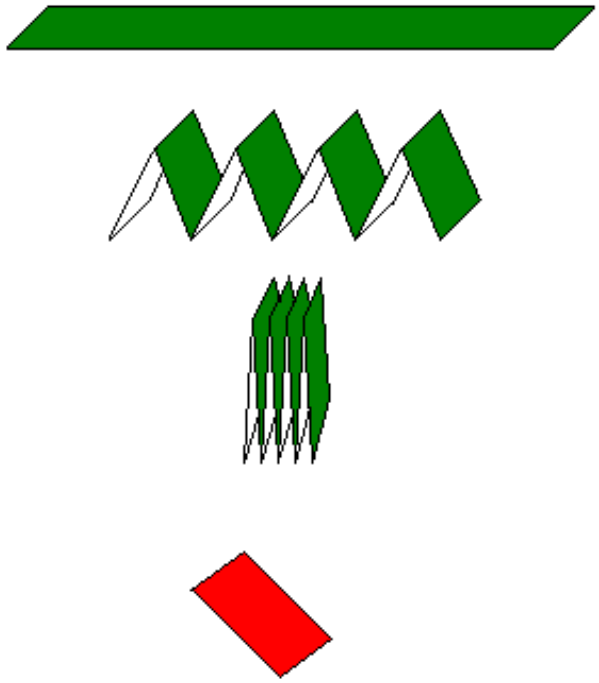
- El mensaje M se divide en bloques de 512 bits, añadiendo bits al final, si es necesario.
- Operaciones lógicas (*fragmentar, permutar, rotar, comprimir*)
  - Primer bloque del mensaje (512 bits)
  - Cuatro vectores iniciales ABCD de 32 bits
- La salida es de 128 bits
  - (nuevos ABCD)

Función MD5

- Nuevas operaciones lógicas (*fragmentar, permutar, rotar, comprimir*) con
  - Segundo bloque del mensaje (512 bits)
  - Cuatro vectores iniciales ABCD de 32 bits
- Y repetimos hasta el último bloque del mensaje.
  - Al terminar, tenemos un *resumen* que corresponde a los últimos 128 bits obtenidos

Función MD5

- En esencia…
  - Se toma el mensaje
  - Se parte en pedazos de longitud constante
  - Se combina pedazo por pedazo hasta obtener un solo mensaje de longitud fija.



Ver: ¿Cómo funciona el hash MD5?

Función SHA-1

- Es muy similar a MD5.
- Trata bloques de 512 bits con un total de 80 vueltas.
- El resumen es de 160 bits.
- La ejecución de SHA-1 es más lenta que la de MD5.

Ver: ¿Cómo funciona el hash SHA-1?

Ataques a funciones hash

- Encontrar y documentar colisiones
- Podemos encontrar un M' con el mismo hash que M.
- ¿M' tendrá un aspecto válido?
  - ¿Si M es un mensaje de texto, M' tendría un significado concreto para que la falsificación tuviera interés?