**Universidad de Oviedo**

*Departamento de Informática*

*Campus de Gijón*

# Firma digital

*Presentación*

Daniel F. García

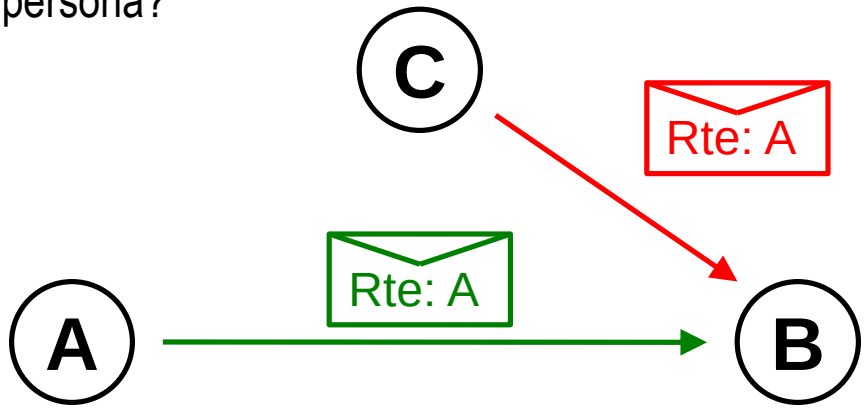
# Problemas de autenticación (1)

La firma digital resuelve problemas relacionados con la autenticación

Por ello se presentan primero este tipo de problemas

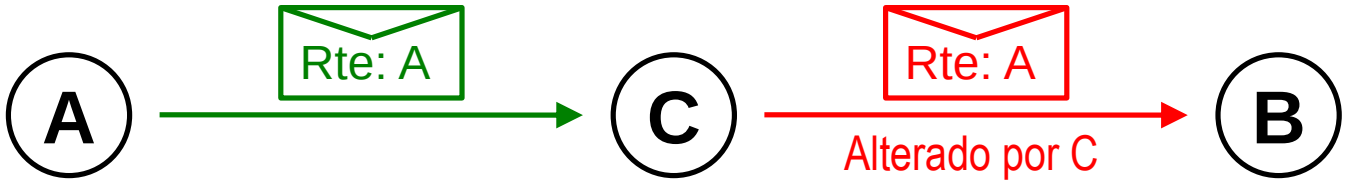
## Autenticidad del emisor

¿Cómo comprueba **B**enito que el mensaje recibido del emisor que dice ser **A**dela es efectivamente esa persona?




## Autenticidad del mensaje (≈Integridad del mensaje)

¿Cómo comprueba **B**enito que el mensaje recibido de **A**dela es el auténtico y no fue falseado?



# Problemas de autenticación (2)

## Repudio del emisor

¿Cómo prueba **B**enito que el mensaje recibido del emisor **A**dela –que niega haberlo enviado– ha sido realmente enviado por Adela?

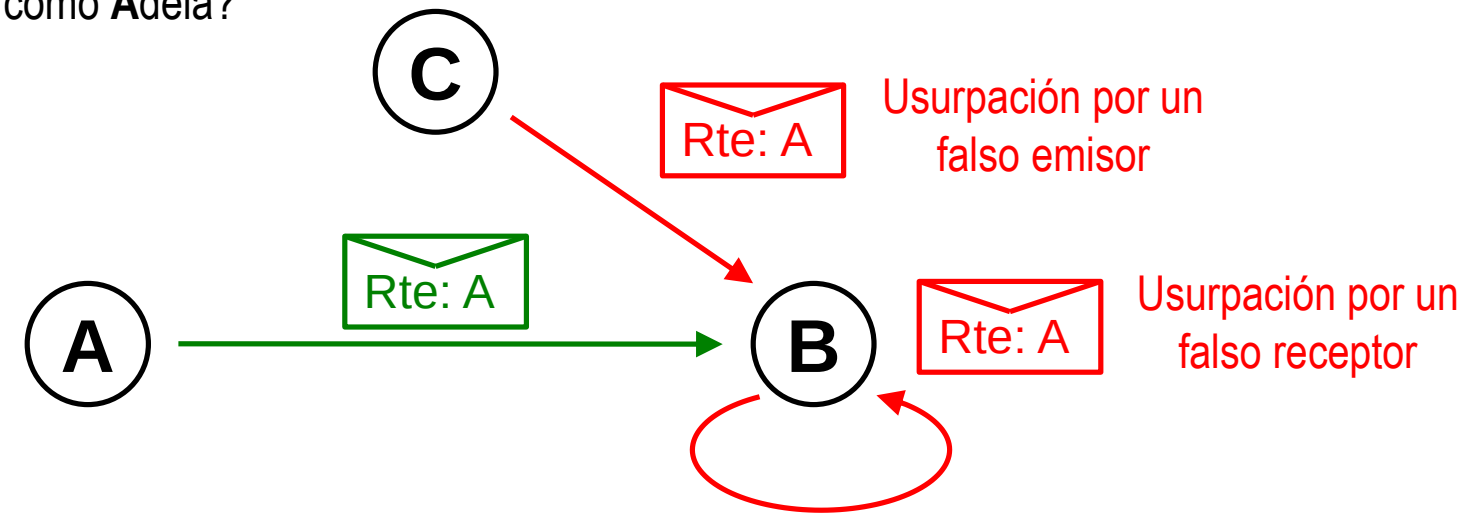

## Repudio del receptor

¿Cómo prueba **A**dela que el mensaje enviado al receptor **B**enito –que niega haberlo recibido– ha sido realmente recibido por Benito?

# Problemas de autenticación (3)

## Usurpación de la identidad

¿Cómo comprueba **A**dela que **B**enito, **C**armen u otros están enviando mensajes firmados como **A**dela?




La suplantación de identidad es una estafa común en Internet
Ej. Un hacker se hace pasar por un banco y solicita datos bancarios al que recibe el mensaje

## Actualidad del mensaje

¿Cómo comprueba **B**enito, que el mensaje recibido del emisor **A**dela es actual, y no es un mensaje de fecha anterior reenviado?

# Soluciones para la autenticación (repaso)

1. Autenticación mediante cifrado de mensajes con un algoritmo simétrico

   El receptor al descifrar el mensaje y ver que es inteligible puede autenticar:
   - El mensaje (integridad) pues si fuese modificado el descifrado sería ininteligible
   - El emisor, pues solo él o el receptor pueden haberlo encriptado (solo ellos tienen la clave)

2. Autenticación mediante funciones hash

   Una función pública genera un resumen del mensaje de longitud fija que permite autenticar el mensaje (mensaje auténtico ≈ mensaje íntegro)

3. Autenticación con un MAC

   Una función pública y una clave secreta generan un código de longitud fija a partir del mensaje que se puede utilizar para autenticar el mensaje

4. Autenticación mediante una firma digital

   Una función pública y una pareja de claves (pública y privada) permiten la autenticación del mensaje y del emisor

# La firma digital

## Definición

Una firma digital es una información (secuencia de bits) que se añade a un documento electrónico que permite autenticar al documento y a su autor de forma inequívoca y segura

## Requisitos

1) Será **única**: solo la podrá generar su propietario

2) Será **irrevocable**: el propietario no podrá rechazarla

3) Debe ser **fácil** de **generar**

4) Debe ser **fácil** de **usar** por { Su propietario (emisor de información) / Los usuarios (receptores de información)

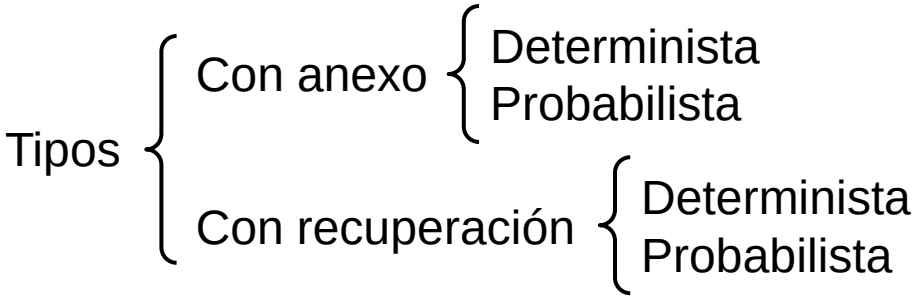
5) Debe **depender** del:

- Documento
- Autor

⇦ ¡ REQUISITO ESENCIAL !

**¡ Estos requisitos son mas fuertes que los de una firma manuscrita !**

# Clasificación de los esquemas de firma digital




- Esquema con anexo (del mensaje)

  Requiere el mensaje original como entrada al algoritmo de verificación

- Esquema con recuperación (del mensaje)

  NO Requiere el mensaje original como entrada al algoritmo de verificación

  El mensaje original se recupera de la propia firma

- Esquema Probabilista

  Introduce información generada aleatoriamente en el proceso de construcción de la firma

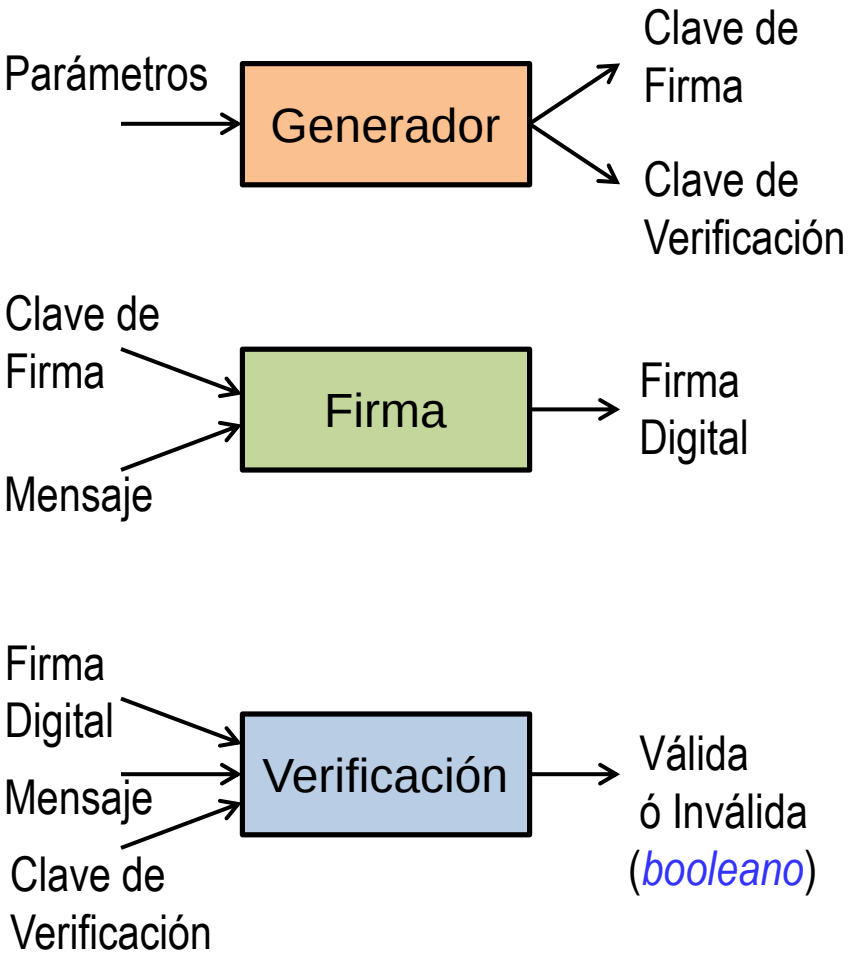
- Esquema Determinista

  NO Introduce información generada aleatoriamente en el proceso de construcción de la firma

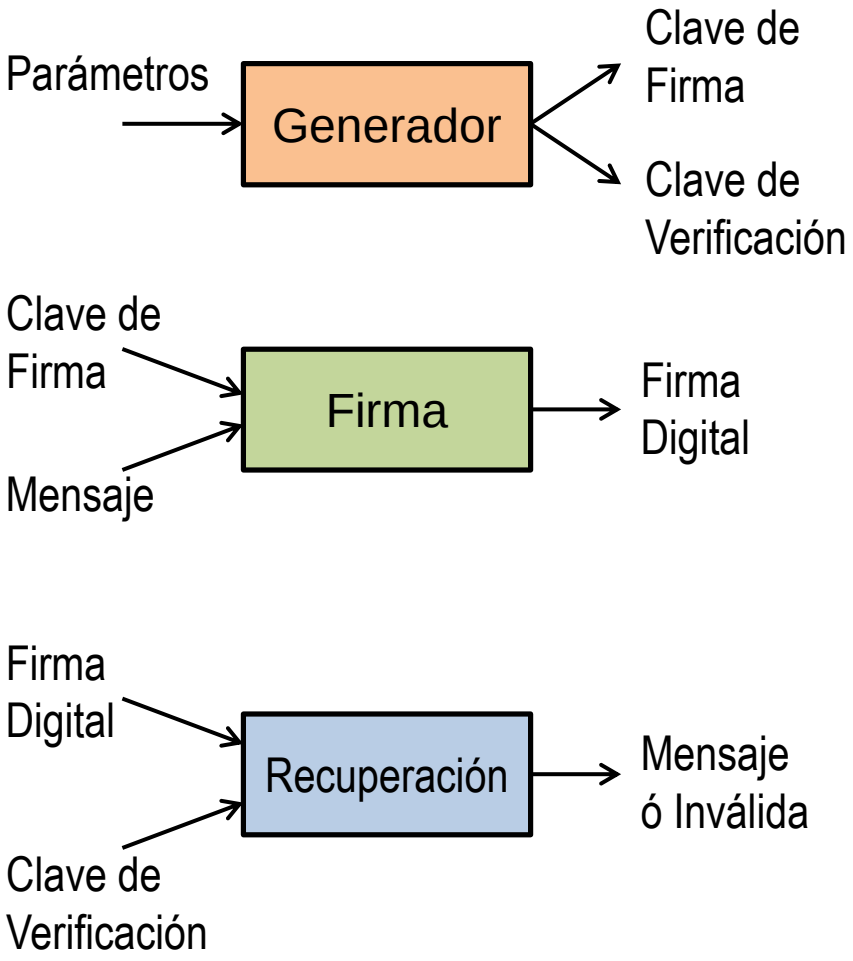
# Componentes de un esquema de firma digital

Todos los esquemas de firma digital se basan en tres algoritmos fundamentales

## Esquema con anexo

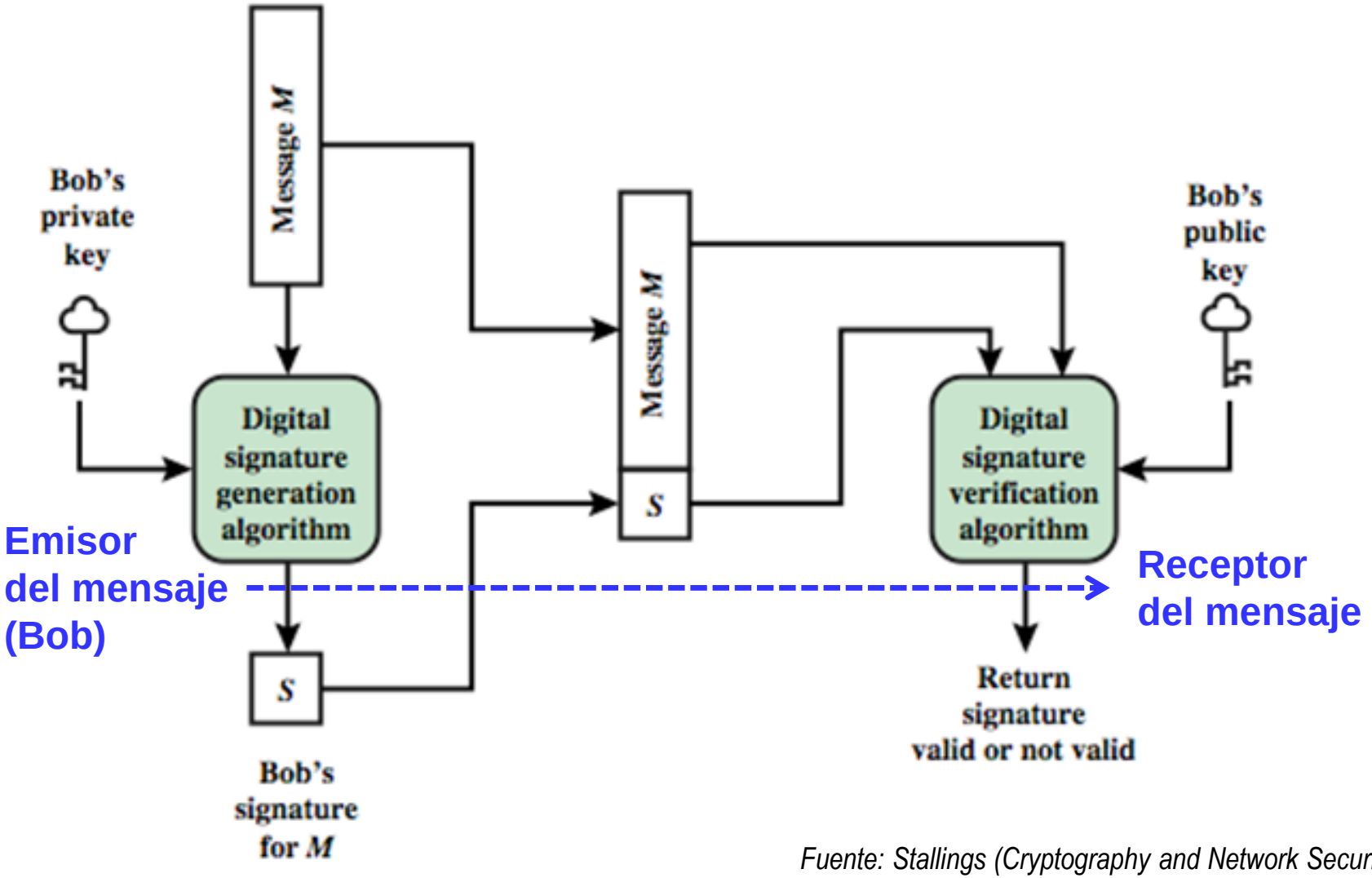



## Esquema con recuperación



# Esquema general de firma digital con anexo

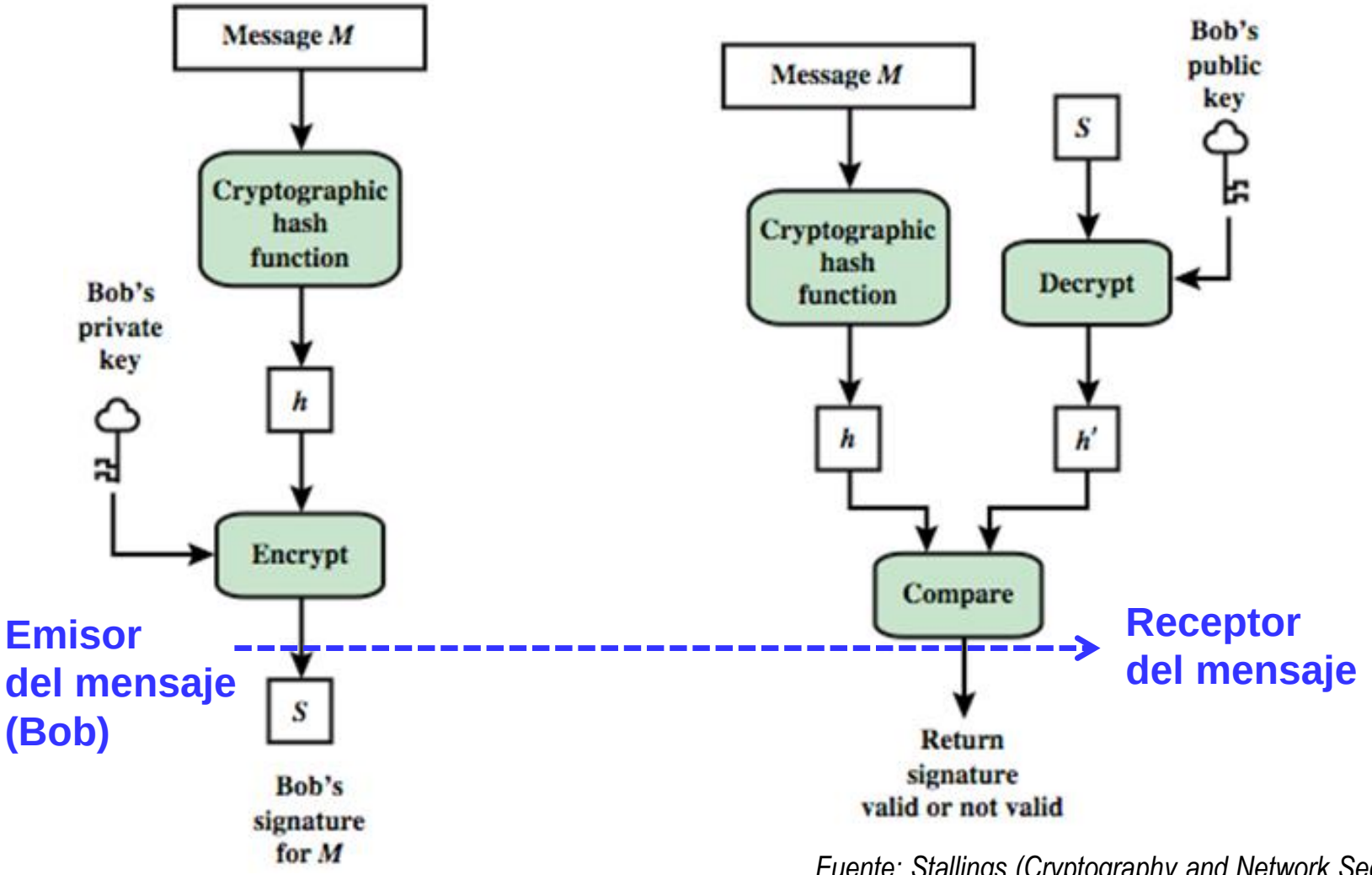
Los esquemas de firma digital con anexo son los más comunes:




*Fuente: Stallings (Cryptography and Network Security)*

# Esquema de firma digital con anexo basado en hash

Los esquemas de firma con anexo más comunes se basan en una F. de hash + Alg. de Cifrado/Descifrado




*Fuente: Stallings (Cryptography and Network Security)*

# El estándar de firma digital DSS

**DSS** = **D**igital **S**ignature **S**tandard

DSS es un estándar de firma digital propuesto por el NIST

- Básicamente el estándar DSS utiliza el algoritmo DSA (*Digital Signature Algorithm*)
  Es una variante del Alg. Schnorr basado en una modificación del Alg. ElGammal
- Pero DSS también soporta: RSA y ECDSA (*Elliptic Curve Digital Signature Algorithm*)

Evolución

1994 NIST Integra el DSA en el estándar DSS publicado en FIPS 186 (19-May-1994)

1998 Primera revisión menor de DSS publicada en FIPS 186-1 (15-Dic-1998)

2000 Se expande el estándar DSS publicado en FIPS 186-2 (27-Ene-2000)

2009 Nueva expansión del estándar DSS (Junio-2009)

2013 Revisión del estándar DSS (Julio-2013)

2023 Nueva expansión del estándar DSS (Febrero-2023 Borrador)
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

Partes DSA

Para usar DSA hay que considerar 3 partes
1) Generación de claves
2) Firmar un mensaje
3) Verificar la firma de un mensaje

# DSA: Generación de claves 1

La generación de claves tiene dos fases:

**FASE-1:** Elección de los **parámetros** del algoritmo DSA
que tienen que ser compartidos por todos los usuarios de un dominio

**FASE-2:** Cálculo de una pareja de **claves** (pública y privada)
para cada uno de los usuarios

## FASE-1 Generación de parámetros

1. Elegir una función de hash H aprobada por el estándar DSS
   - La especificación original de DSS usaba la función SHA-1
   - En la actualidad hay que utilizar una función SHA-2

2. Elegir la longitud de los parámetros L y N → *Posibilidades*
   - L (longitud del parámetro p) N (longitud del parámetro q)

| L | N |
|---|---|
| 1024 | 160 |
| 2048 | 224 |
| 2048 | 256 |
| 3072 | 256 |

3. Elegir el parámetro p, primo de L bits ( $2^{L-1} < p < 2^{L}$ ) p = 223

4. Elegir el parámetro q, primo de N bits y divisor de (p-1) ( $2^{N-1} < q < 2^{N}$ )

   223-1= 222 = 2 · 3 · 37 → q = 37

# DSA: Generación de claves 2

**5** Elegir el parámetro g, generador de orden q del grupo p ( 1 < g < p )

Un generador de orden q, es una raíz g en el cuerpo $Z^*_p$, que verifica → $g^q \bmod p = 1$

[ Números primos < p ]

Se puede encontrar g con esta ecuación:

$$g = \beta^{(p-1)/q} \bmod p$$

Eligiendo aleatoriamente un $\beta$ tal que $1 < \beta < p-1$ →
- Si $g \neq 1$ el valor de g es válido
- Si $g = 1$ elegir otro valor de $\beta$, etc.

$g = \beta^{(223-1)/37} \bmod 223 = \beta^6 \bmod 223$

Con $\beta=19$ $g = 19^6 \bmod 223 = 17$ (es válido)

## FASE-2 Generación de claves

**1** Elegir una clave secreta X ( $1 \leq X \leq q-1$ )

Usar un "buen" método de generación de aleatorios

Elegir X= 25 verifica: $1 \leq X \leq 36$

**2** Calcular la clave pública Y

$Y = g^X \bmod p$

$Y = 17^{25} \bmod 223 = 30$

**RESUMEN**

Datos públicos:

p = 223 q = 37

g = 17 Y =30

Datos secretos:

X = 25

# DSA: Firmar un mensaje

Para firmar un mensaje se dan los pasos siguientes:

1. Generar un número secreto K, PARA CADA MENSAJE ( $1 \le K \le q-1$ )
   La generación debe ser aleatoria
   Ejemplo: K = 12 < 37-1

2. Calcular $r = (g^K \bmod p) \bmod q$
   $r = ( 17^{12} \bmod 223 ) \bmod 37 = 23$

3. Calcular $s = ( K^{-1} (z + X \cdot r) ) \bmod q$
   $s = ( 34 (30 + 25 \cdot 23) ) \bmod 37 = 35$

   z = los n bits más-a-la-izquierda de H(M)

   ↓

   n = min( Lq, Lh ) { Lq = Longitud de q ; Lh = Longitud del hash

   Sea Lh<Lq → Usamos todos los bits del hash
   Entonces z = H(M)=30 (ejemplo)

   $K^{-1}$ es el inverso multiplicativo de K con respecto a la multiplicación en módulo q

   Se verifica → $0 < K^{-1} < q$ ; $(K^{-1} \cdot K) \bmod q = 1$

   $(K^{-1} \cdot 12) \bmod 37 = 1$
   $K^{-1}=34$ obtenido con un bucle ⋯

   Hay un método para calcular $K^{-1}$ en el Apéndice C1 de FIPS186-3
   *(Es muy típico usar el Algoritmo de Euclides Extendido)*

4. Re-calcular la firma en el caso improbable de que $r = 0$ ó $s = 0$

La firma digital DSA de H(M) es el par (r,s)

A envía a B → (M,r,s) = (M,23,35)

# DSA: Verificar una firma

Antes de verificar una firma hay que obtener los datos públicos necesarios de una entidad de confianza

Cualquiera que tenga acceso a los datos públicos puede verificar la firma digital

El propio firmante puede verificar su firma antes de enviarla con el documento

Datos públicos

| p | q | g | Y |
|---|---|---|---|
| 223 | 37 | 17 | 30 |

El receptor dispone del mensaje firmado recibido M' r' s'

M' 23 35

Para verificar una firma se dan los pasos siguientes:

1. Comprobar que $0 < r' < q$ Y $0 < s' < q$ — $0<23<37$ Y $0<35<37$

   Si alguna de estas condiciones no se cumple, la firma NO es válida

2. Calcular $w = s'^{-1} \bmod q$ — $(s'^{-1} \cdot 35) \bmod 37 = 1$, $s'^{-1} = 18$ — $w = 18 \bmod 37 = 18$

3. Calcular $u1 = ( z \cdot w ) \bmod q$ — $u1 = (30 \cdot 18) \bmod 37 = 22$

   **H(M') = z**

4. Calcular $u2 = ( r' \cdot w ) \bmod q$ — $u2 = (23 \cdot 18) \bmod 37 = 7$

5. Calcular $v = [ (g^{u1} \cdot Y^{u2}) \bmod p ] \bmod q$ — $v = [ (17^{22} \cdot 30^{7}) \bmod 223 ] \bmod 37 = 23$

Si v = r' la firma es válida
Si v ≠ r' la firma NO es válida

# Firma Digital basada en RSA

Todo el proceso de firma se describe en el estándar **PKCS#1**:

1998 Mar V1.5 RFC-2313 RSA Encryption

1998 Oct V2.0 RFC-2437 RSA Cryptography Specifications

2003 Feb V2.1 RFC-3447 RSA Cryptography Specifications

https://www.rfc-editor.org/rfc/rfc3447

2016 Nov V2.2 RFC-8017 RSA Cryptography Specifications

https://www.rfc-editor.org/rfc/rfc8017

El estándar describe dos esquemas de firma con anexo (*appendix*)

Determinista → RSASSA-PKCS1-V1.5

Probabilista → RSASSA-PSS

**SSA** = **S**ignature **S**cheme with **A**ppendix

**PPS** = **P**robabilistic **S**ignature **S**cheme

El estándar de firma digital DSS del NIST soporta ambos esquemas RSA

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf (Ver sección 5)

Pero impone restricciones adicionales a las indicadas en PKCS#1 v2.2

# RSASSA – PKCS1: Generación de la firma

## Fase 1: Codificación del mensaje

Aplicar una función de hash para obtener H = Hash(M)

Codificar el ID de la función de hash y el valor del hash como un tipo Digest Info de ASN.1 → **T**

ASN.1 = Abstract Syntax Notation 1

Generar un **PS** (*Padding String*) de la longitud apropiada con el byte 0xFF

Construir la codificación del mensaje EM por concatenación:

EM = 0x00 || 0x01 || **PS** || 0x00 || **T**

## Fase 2: Cifrado de la codificación del mensaje

Convertir la codificación del mensaje EM (cadena de bytes) en un número m

Cifrar con RSA usando la clave privada (d,n) → $s = m^d \bmod n$

Convertir el número s a bytes para obtener la firma S (cadena de bytes)

# RSASSA – PKCS1: Verificación de la firma

## Fase 1: Descifrado de la firma

Convertir la firma S (cadena de bytes) en un número s

Descifrar con RSA usando la clave pública (e,n) → $m = s^e \bmod n$

Convertir el número m a bytes para obtener la codificación del mensaje EM' (cadena de bytes)

## Fase 2: Verificación de la codificación del mensaje

### Método 1:

Codificar el mensaje recibido $M_R$

$EM'_R = \text{EMSA-PKCS1-Encode}(M_R)$

Comparar ($EM' == EM'_R$) ?

SI → Firma verificada
No → Firma no verificada

### Método 2:

Decodificar EM' (no especificado en el estándar)

Extraer el Hash(M) = H' de la decodificación

Calcular el hash del mensaje recibido $H'_R$

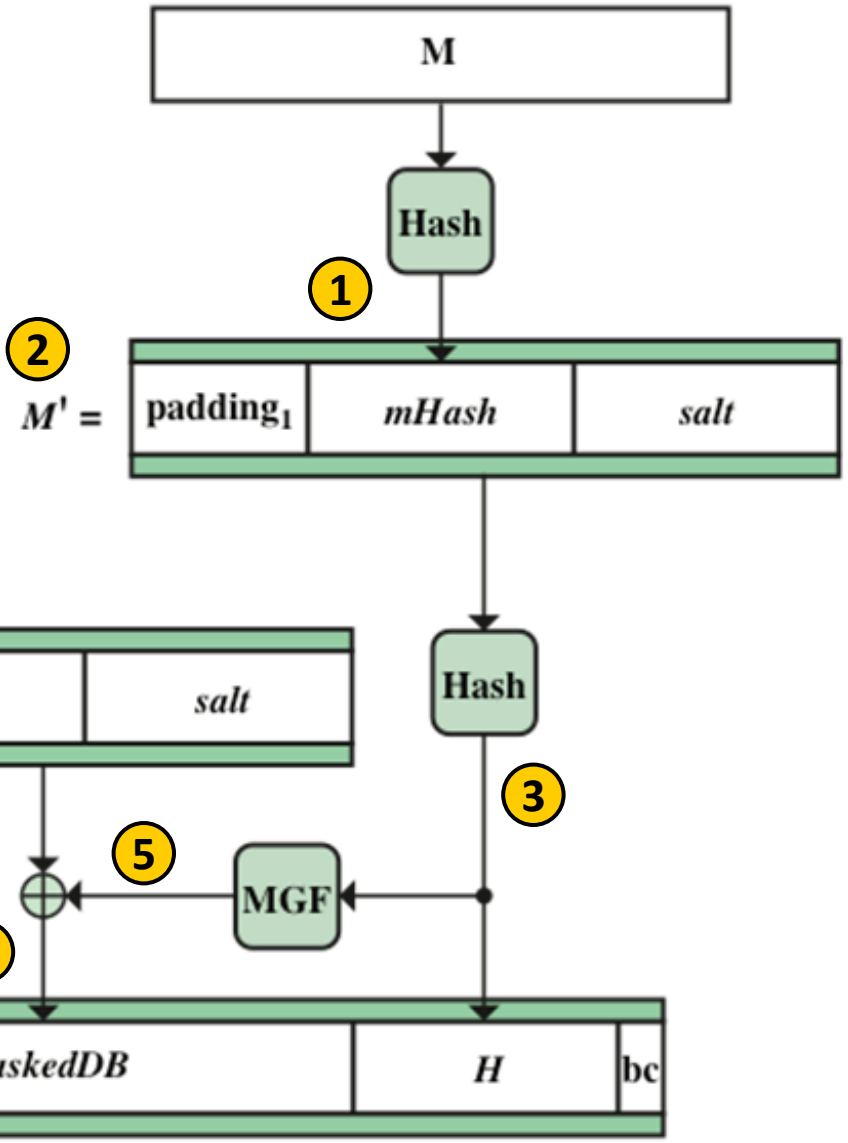
Comparar ($H' == H'_R$) ?

SI → Firma verificada
No → Firma no verificada

# RSASSA – PSS: Generación de la firma

## Fase 1: Codificación del mensaje

1. Generar el hash de M: mHash = Hash(M)
2. Generar salt (cadena de bytes aleatoria)
   componer M' = $padding_1$ || mHash || salt
3. Generar el hash de M': H = Hash(M')
4. Formar el Data Block
   DB = $padding_2$ || salt
5. Calcular dbMask = MGF(H)
6. Calcular maskedDB = DB ⊕ dbMask
7. Poner bits +Izda de maskedDB a 0
8. EM = maskedDB || H || bc




*Fuente: Stallings (Cryptography and Network Security)*

# RSASSA – PSS: Cifrado y Descifrado de EM

## Fase 2 Generación: Cifrado de la codificación del mensaje

Convertir la codificación del mensaje EM (cadena de bytes) en un número m

Cifrar con RSA usando la clave privada (d,n) → $s = m^d \bmod n$

Convertir el número s a bytes para obtener la firma S (cadena de bytes)

Ejemplo: Si el tamaño de la clave RSA es 2048 bits → Tamaño de S = 2048/8 = 256 bytes

---

## Fase 1 Verificación: Descifrado de la firma

Convertir la firma S (cadena de bytes) en un número s

Descifrar con RSA usando la clave pública (e,n) → $m = s^e \bmod n$

Convertir el número m a bytes para obtener la codificación del mensaje EM (cadena de bytes)

La longitud en bytes de EM es emLen = $\lceil (modBits-1)/8 \rceil$

*modBits es la longitud en bits del módulo n de RSA*

# RSASSA – PSS: Verificación de la firma

1. Generar *mHash* = Hash(M)

2. SI emLen < (hLen + sLen +2)
   Hay inconsistencia → Stop

3. SI Byte Dcho EM ≠ bc
   Hay inconsistencia → Stop

4. Descomponer EM en *maskedDB* y *H*

5. Si Bits +Izda de *maskedDB* ≠ 0
   Hay inconsistencia → Stop

6. Calcular *dbMask* = MGF(H)

7. Calcular DB = maskedDB ⊕ dbMask

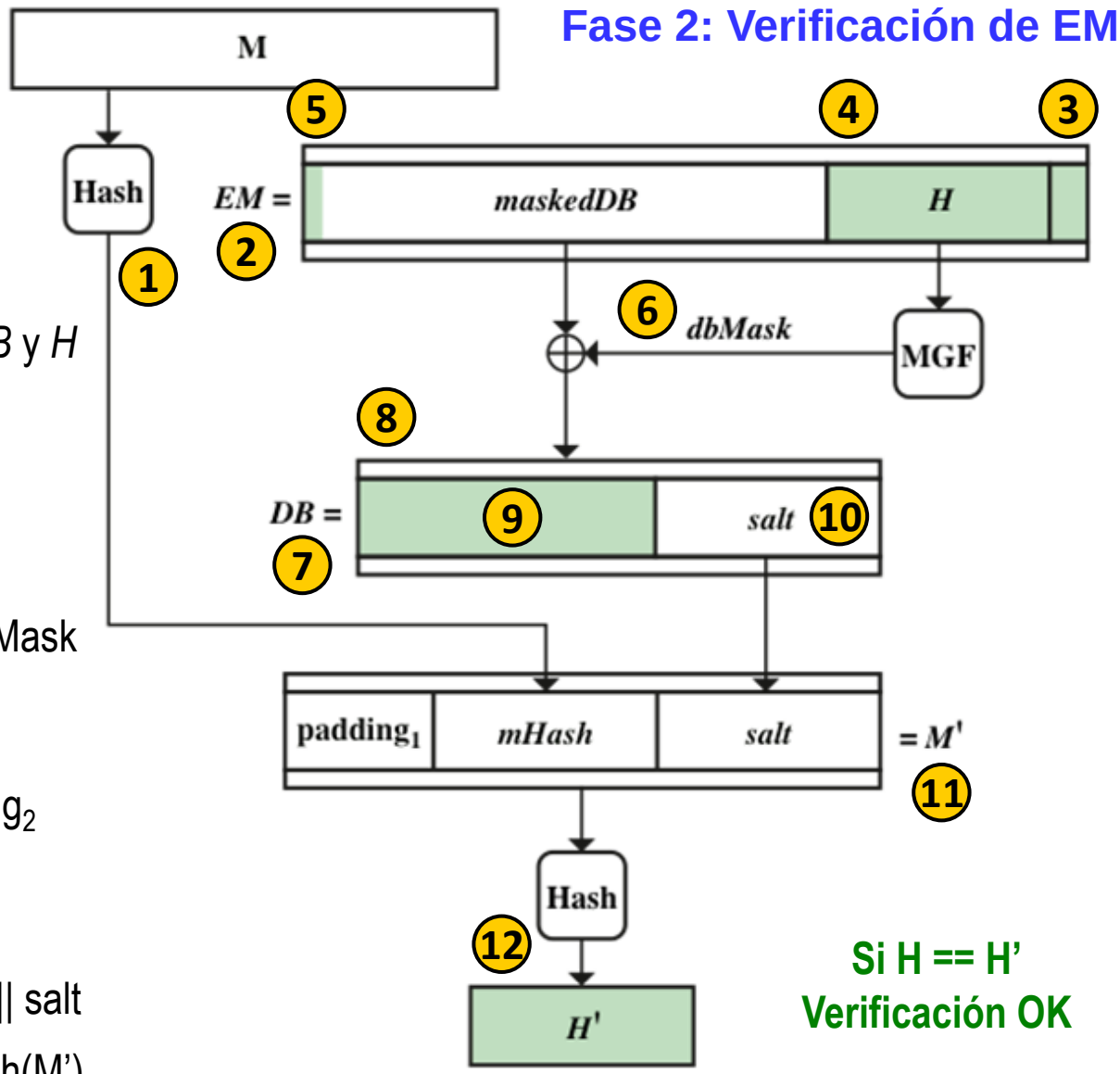
8. Poner bits +Izda de DB 0

9. Si Bits en verde de DB ≠ $padding_2$
   Hay inconsistencia → Stop

10. Salt = sLen bits +Dcha de DB

11. Formar M' = $padding_1$ || mHash || salt

12. Generar el hash de M': H' = Hash(M')




*Fuente: Stallings (Cryptography and Network Security)*

# RSASSA – PSS: Función de generación de máscaras

## MGF = Mask Generation Function

Función que genera una máscara de bits (*mask*) de la longitud deseada (*maskLen*) a partir de Una cadena de bits de cualquier longitud (*X*)

*mask* = *MGF (X, maskLen*)

MGF es una función pseudoaleatoria, típicamente basada en una función hash como SHA-1

MGF basada en Hash ←→ Método para generar un resumen de longitud variable a partir de una función que genera un resumen de longitud fija

## Algoritmo

T ← Cadena vacía (a transformar progresivamente)
k = ⌈ maskLen / hLen ⌉ –1
for C=0 to k
  T = T || Hash(X || C)
mask = Los primeros maskLen bytes de T

## Funcionamiento

SI ( masklen = hLen ) mask = Hash(X||0)

SI ( masklen > hLen ) mask = Hash(X||0) || Hash(X||1) || · · · || Hash(X||k)

# RSASSA – PSS: Generación + Verificación de la firma