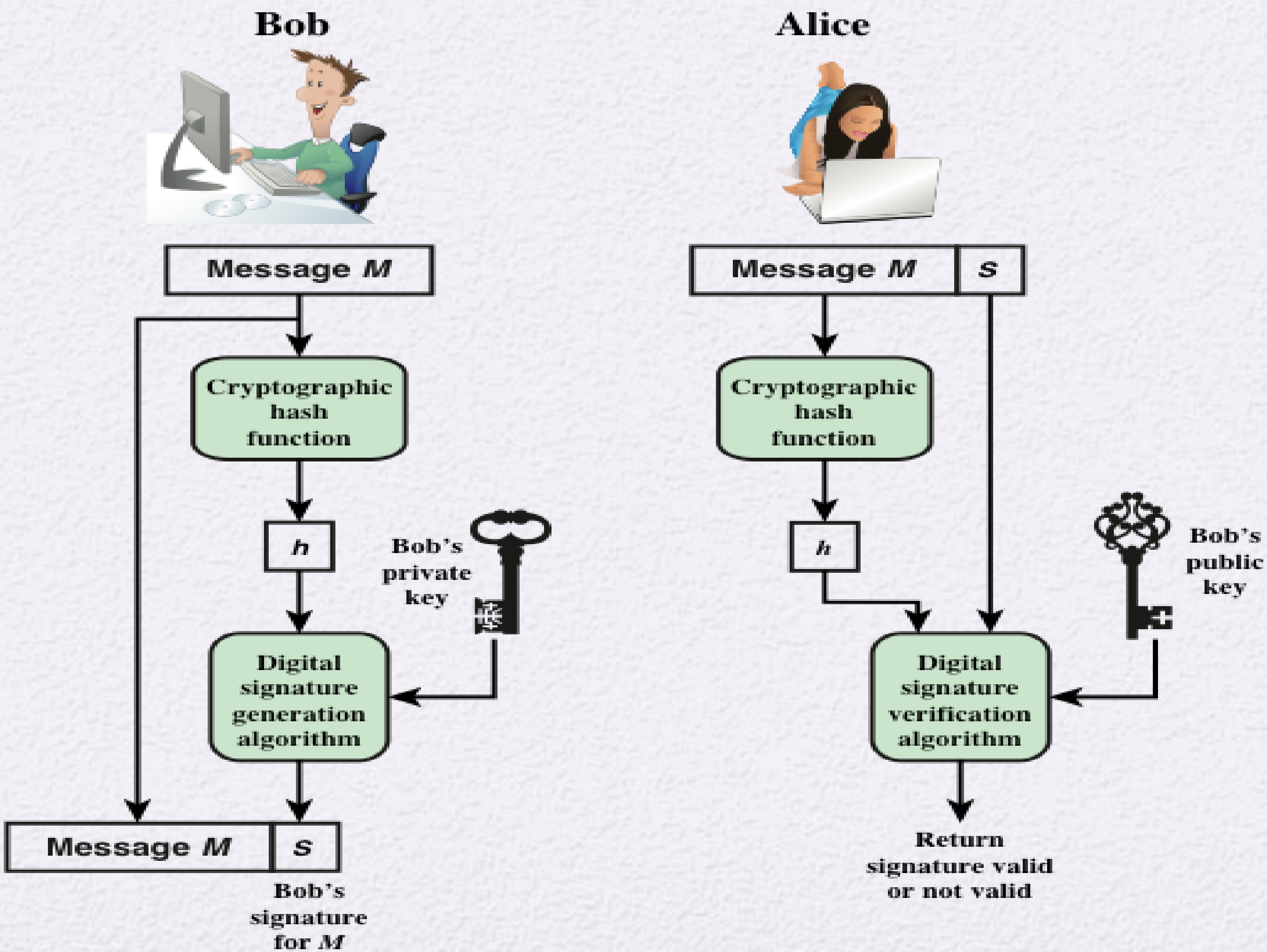
Chapter 13

Digital Signatures

Digital Signature Properties

- Verify the author and time of the signature
- Authenticate the contents at the time of the signature
- It must be verifiable by third parties to resolve disputes
- Note: design of digital signature is easier than public-key encryption

Digital Signature Model



Attacks

- Key-only attack
  - The attacker (C) only knows A's public key
- Known message attack
  - C is given a set of messages and their signatures
- (Adaptive) chosen message attack
  - C can request signatures of his chosen messages from A
  - It is adaptive if a new requested message depends previous requested results

Types of Forgery

- Total break
  - C determines A’s private key
- Universal forgery
  - C finds an efficient signing algorithm to construct signatures for arbitrary messages
- Selective forgery
  - C forges a signature for a particular message chosen by A
- Existential forgery
  - C forges a signature for at least one message; C has no control over the message

RSA Digital Signature

- No global parameters
- Each user A
  - Choose two large primes p and q and compute $n=pq$
  - Choose e with $\gcd(e, (p-1)(q-1))=1$
    - Verification (public) key: $PU_A = (e, n)$
  - Compute $d = e^{-1} \bmod (p-1)(q-1)$
    - Signing (private) key: $PR_A = (d, n)$
- RSA can be used for both encryption and digital signature. But, we must not use the same key pair for both

RSA: Sign and Verify

- Let H be a cryptographic hash function
- Sign M with $PR_A = (d, n) \rightarrow (M, s)$
  - Compute $m=H(M)$ and $s=m^d \bmod n$
- Verify (M, s) with $PU_A = (e, n)$
  - Compute $m = H(M)$ and $m' = s^e \bmod n$
  - Passed if and only $m == m'$

RSA: Toy Example

- $PU_A = (7, 143)$, $PR_A = (103, 143)$
- Let M="This is a test for RSA signature".
- Assume $H(M) = 35$
- Sign: $s = H(M)^{103} \bmod 143 = 74$
- Verify: $(M, 74)$
  - Pass: $74^7 \bmod 143 = 35$
  - Not pass (wrong signature): $\mathbf{73}^7 \bmod 143 = 83 \neq 35$
  - Not pass (wrong public key): $74^{\mathbf{17}} \bmod 143 = 68 \neq 35$

RSA: Large Numbers

- **p=**12131072439211271897323671531612440428472427633701410925634549312301964373042085619324197365322416866541017057361365214171711713797974299334871062829803541
- **q=**12027524255478748885956220793734512128733387803682075433653899983955179850988797899869146900809131611153346817050832096022160146366346391812470987105415233
- **n=**145906768007583323230186939349070635292401872375357164399581871019873438799005358938369571402670149802121818086292467422828157022922076746906543401224889672472407926969987100581290103199317858753663710862357656510507883714297115637342788911463535102712032765166518411726859837988672111837205085526346618740053
- ϕ(n)=145906768007583323230186939349070635292401872375357164399581871019873438799005358938369571402670149802121818086292467422828157022922076746906543401224889648313811232279966317301397777852365301547848273478871297222058587457152891606459269718119268971163555070802643999529549644116811947516513938184296683521280
- **e** = 65537
- **d=**9489425009274444368228545921773093919669586065884257445497854456487674839629818390934941973262879616797970608917283679875499331574161113854088813275488110588247193077582527278437906504015680623423550067240042466665654232383502922215493623289472138866445818789127946123407807725702626644091036502372545139713

RSA: Real

- **-----BEGIN PUBLIC KEY-----**
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCT21CXL6L/w4rXj2F9Yp+0bexZU7UGXkWcN
/mjApDjhx3xbHoJPbRCoanLzxCtYgU0Q/LiPOowntYDWtZisIMNRCb5OhrcW5gg+eoz3N836iYIhV
9HYbGBDeyd/Qvbu1foMWYgqLtEuFooI7DO+WL4FLABNDQRB8KZ1a1HZA+VZQIDAQAB
----END PUBLIC KEY-----

- **-----BEGIN RSA PRIVATE KEY-----**
MIICXQIBAAKBgQCT21CXL6L/w4rXj2F9Yp+0bexZU7UGXkWcN/mjApDjhx3xbHoJPbRCoanLzxCtY
gU0Q/LiPOowntYDWtZisIMNRCb5OhrcW5gg+eoz3N836iYIhV9HYbGBDeyd/Qvbu1foMWYgqLtEu
FooI7DO+WL4FLABNDQRB8KZ1a1HZA+VZQIDAQABAoGBAI0IyMG9lbxjmF9vi+2fEnl31NeMCgOu
DrYpinycKPpvWvd7opiH/BcGv8EBnYXVFuO44Mg+l30mNTvz/MUcW5PoWL2UjCljdyKHUHjFZNRr
+bQPiq6pKwscPXH/z9aOakPJg01URtFW4ecIc9AdRFngdllY4zRC2LVKMpMoEWYhAkEA3slUiX9IT
LpCZxezAX41Nqlf70wxkx3QB+KdTySMj80mxz8D7zwtHKGKhkzKADRtoM65NqHYY1Yrb0F8ucQ
QLQJBAKnmSEcPeKeldXMAwwTKbKoZFsX8GBz1kpVMIEOeZdG+jNV4SXD1X7u3Osc2AyBJ3wr1E
M6zL8SUK4XJTpiRpRkCQQCvKuueJIbrVTPqrRahIOOkM+5rT6RXAQTVflejp6AhXPOVi7WDP/RUY
6twRyY4nQBphMDZ9MoX5ePG2V3BDiiNAkBUUYo7Yg1CPlZsrcsbfI601Ye82GDrNmD6IV690EW9
83CXnOvt2Ikbc1MDfOXOR3sfSAKAYuNpDxQOgJq2Eoo5AkBX0YS05y5ii22BAljGrZ4v/RJ8K+oC6y5
oWBTKtoN8OR6tkpGLp/UCDMuRZJV6BvM3l77yBrsrAlzgzT/D5KrC
----END RSA PRIVATE KEY-----

RSA: RSA with SHA-256

- M = “Hello!”
- s =
YeLxe3GMCpxom65Gn/L3DURbPx/0myS/5kJmrw3t/x98
jgr8+2CaGxUsWUUZhPeRTI8PW82L58N7botehkHWRa
gxkVuh2G8xnZS2Py44ytntiwLCAl3ggBoqhtqcGxP1MiP7
frUBHupIh8/p9XGwd5v5cRNJ1KNjSbw0ppPkHO8=

ElGamal Digital Signature

- Global parameters
  - prime number q
  - α: a primitive root of q, *that is*, α's order is q
- Each user A generates their key
  - Chooses a secret key : $1 < X_A < q-1$
    - $PR_A = (q, \alpha, X_A)$
  - Compute $Y_A = \alpha^{X_A} \bmod q$
    - $PU_A = (q, \alpha, Y_A)$

ElGamal: Sign and Verify

- Let H be a cryptographic hash function
- Sign M with $PR_A = (q, \alpha, X_A) \rightarrow (M, s_1, s_2)$
  - Compute m=H(M)
  - Randomly choose k, 1<k<q and gcd(k, q-1)=1
  - Compute $s_1 = \alpha^k \bmod q$ and $s_2 = k^{-1}(m - X_A s_1)$ **mod q-1**
- Verify (M, s_1, s_2) with $PU_A = (q, \alpha, Y_A)$
  - Compute m = H(M)
  - Pass if and only if $\alpha^m \equiv Y_A^{s_1} s_1^{s_2} \pmod{q}$

ElGamal: Correctness

- $Y_A^{S_1} S_1^{S_2} \bmod q$
$$= \alpha^{X_A S_1} \alpha^{k[k^{-1}(m - X_A S_1) \bmod (q-1)]} \bmod q$$
$$= \alpha^m \bmod q$$

ElGamal: Toy Example

- $q=19$, $\alpha=10$
- PR=(19, 10, 16), PU = (19, 10, 4)
- H(M) = 14
- Sign(PR, M) → (10^5 mod 19, 5^{-1}(14-16x3) mod 18)=(3, 4)
  - k=5, 5^{-1} mod 18=11
- Verify(PU, M, s_1, s_2)
  - α^m mod q = 10^{14} mod q = 16
  - $Y_A^{s_1} s_1^{s_2} \bmod q = 4^3 3^4 \bmod 19 = 16$

ElGamal: Security

- Based on discrete logarithm problem
- Problem: find (α, s_1, s_2) for $\alpha^m \equiv Y_A^{s_1} s_1^{s_2} \pmod{q}$
  - Select (m, s_1) and solve s_2 for $\alpha^m = Y_A^{s_1} s_1^{s_2} \pmod{q}$
  - Select (m, s_2) and solve s_1 for $\alpha^m = Y_A^{s_1} s_1^{s_2} \pmod{q}$
  - Select (s_1, s_2) and solve m for $\alpha^m = Y_A^{s_1} s_1^{s_2} \pmod{q}$

Schnorr Signature

- Global parameters: (p, q, a)
  - Choose primes p and q, where q is a factor of p-1.
  - Typically, *p* is a 1024-bit number, and *q* is a 160-bit number
  - Choose $a \neq 1$, with $a^q \bmod p = 1$
- Private (signing) key: $PR_A = (p, q, a, s)$
  - Choose a number s, $1 < s < q-1$
- Public (verification) key: $PU_A = (p, q, a, v)$
  - Compute $v = a^{-s} \bmod p$

Schnorr: mathematics

- $p-1 = kq$, where p and q are both prime
- $G = Z_p^*$: a multiplicative group with the operation on "mod p". $|G|=p-1$
- G_q : a subgroup of G with $|G_q|=q$.
  - Operation: "mod p"
  - $G_q=\{g^k \bmod p: \ g\in G\}$
  - Every element 'a' in G_q
    - $a^q=1 \bmod p$
    - $a^b \bmod p=a^{b \bmod q} \bmod p$
  - Every element a, $a\neq 1$, is a generator of G_q.

Schnorr: Sign and Verify

- Let H be a cryptographic hash function
  - $\{0,1\}^* \rightarrow \{1, 2, \ldots, q\text{-}1\}$
- Sign M with $PR_A = (p, q, a, s) \rightarrow (e, y)$
  - Randomly choose r, $0 < r < q$
  - Compute $x = a^r \bmod p$
  - Compute $e = H(M||x)$ and $y = (r+se) \bmod q$
- Verify (M, e, y) with $PU_A = (p, q, a, v)$
  - Compute $x' = a^y v^e \bmod p$
  - Pass if and only if $e == H(M||x')$
- Shorter signature: $|e|+|y| = 2|q| = 320$ bits

Schnorr signature: Correctness

$a^y v^e \bmod p$

$=$

NIST Digital Signature: DSS

- NIST, FIPS 186
- A variant of Schnorr digital signature
  - Patent was given to Schnorr, but has expired now.
- The latest version, FIPS 186-3
  - Incorporates digital signature algorithms based on RSA and on elliptic curve cryptography

Global Public Key Components

p prime number where $2^{L-1} < p < 2^L$
for $512 \le L \le 1024$ and L a multiple of 64
i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$
i.e., bit length of N bits

g $= h^{(p-1)/q} \bmod p$
where h is any integer with $1 < h < (p - 1)$
such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

y $= g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$$r = (g^k \bmod p) \bmod q$$

$$s = \left[k^{-1}\left(\mathrm{H}(M) + xr\right)\right] \bmod q$$

$$\text{Signature} = (r, s)$$

Verifying

$$w = (s')^{-1} \bmod q$$

$$\mathrm{u}_1 = \left[\mathrm{H}(M')w\right] \bmod q$$

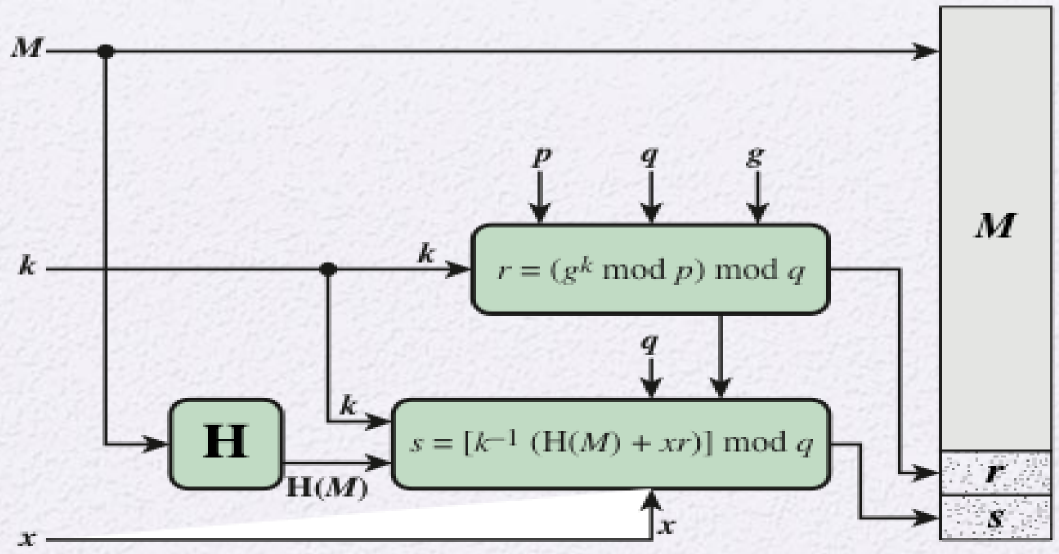
$$\mathrm{u}_2 = (\mathrm{r}')w \bmod q$$

$$v = \left[\left(g^{u1} y^{u2}\right) \bmod p\right] \bmod q$$

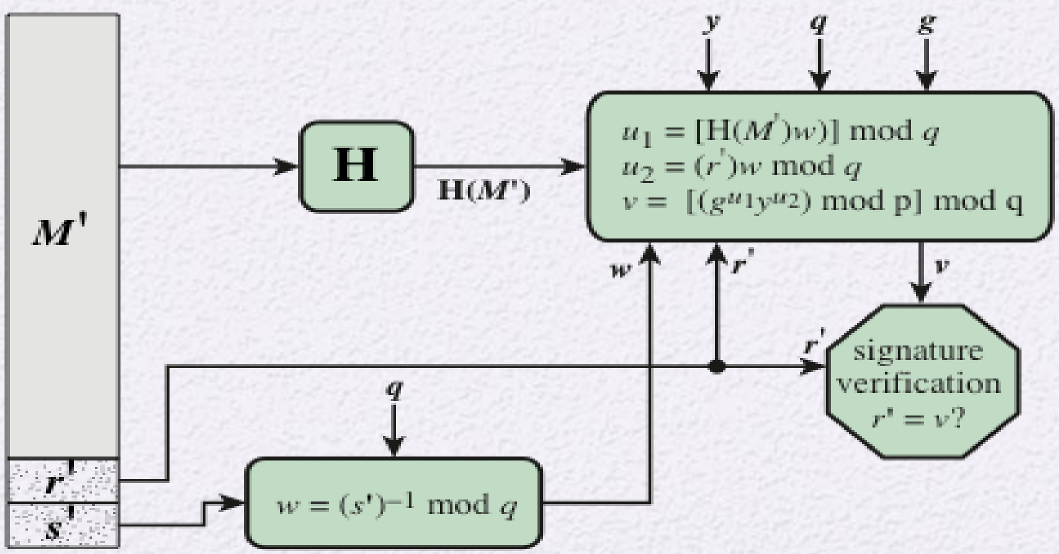
TEST: $v = r'$

M	=	message to be signed
$\mathrm{H}(M)$	=	hash of M using SHA-1
M', r', s'	=	received versions of M, r, s

Figure 13.3 The Digital Signature Algorithm (DSS)



(a) Signing



(b) Verifying

DSS: Correctness

- Given (M, r, s), check

$(g^{u1}y^{u2} \bmod p) \bmod q$

$= (g^{H(M)w}\, g^{xrw} \bmod p) \bmod q$

$= (g^{\mathbf{w(H(M)+xr)}} \bmod p) \bmod q$

$= (g^{\mathbf{k \bmod q}} \bmod p) \bmod q$

$= (g^{k} \bmod p) \bmod q$

$= r$

DSS: Example

- Key generation:
  - $p=67=6\times 11+1$, $q=11$
  - $g=2^{(p-1)/11} \bmod p=3^6 \bmod 67=59$
  - $x=5$, $y=g^x \bmod p=62$
  - $PU=(p, q, g, y) = (67, 11, 59, 62)$
  - $PR=(p, q, g, x) = (67, 11, 59, 5)$
- Signing
  - Let $H(M)=4$, $k=3$
  - $r=g^k \bmod p \bmod q=59^3 \bmod 67 \bmod 11=2$
  - $s=k^{-1}(H(M)+rx) \bmod q=3^{-1}(4+2\times 5) \bmod 11=1$
  - $(r,s)=(2,1)$

- Verification $(r', s')=(2, 1)$
  - $w=s'^{-1} \bmod q = 1^{-1} \bmod 11 = 1$
  - $u_1 = H(M)\times w \bmod q = 4\times 1 \bmod 11 = 4$
  - $u_2 = r'\times w \bmod q = 2\times 1 \bmod 11 = 2$
  - $v=g^{u1} \times y^{u2} \bmod p \bmod q$
    $= 59^4\times 62^2 \bmod 67 \bmod 11$
    $= 2$
  - Since v=r', (2,1) is a signature to H(M)

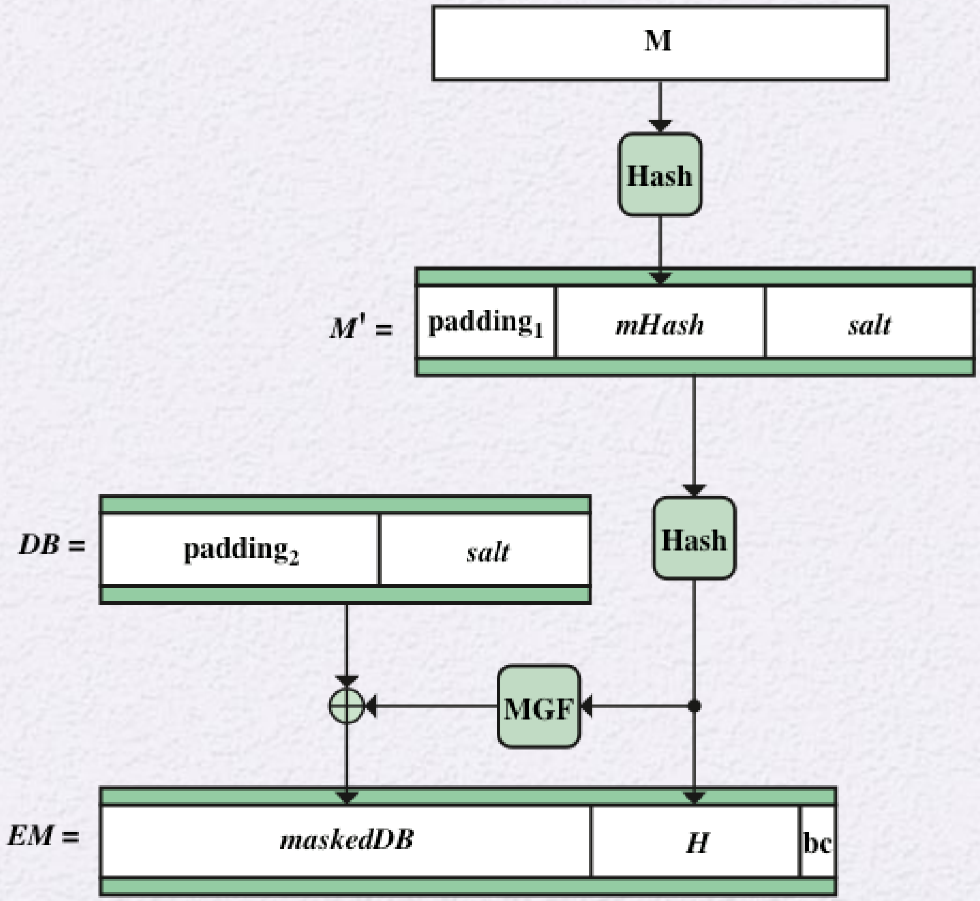
DSS: Security

- Based on computing discrete logarithm over a subgroup of size q: $\log_g y \bmod p$.
  Note: $ord_p(g)=q$
- The per-message secret k cannot be used twice. Otherwise, given two signatures (r_1,s_1) for M_1 and (r_2,s_2) for M_2, we have
  - $s_1=k^{-1}(H(M_1)+r_1x) \bmod q$
  - $s_2=k^{-1}(H(M_2))+r_2x) \bmod q$
  - Solve $\mathbf{x=(s_2H(M_1)-s_1H(M_2))/(r_2s_1-r_1s_2) \bmod q}$

RSA-PSS

- RSA drawback: no randomization in signature
  - For a signing key PR_A: One message → one signature
- RSA Probabilistic Signature Scheme, 2009, FIPS 186-3
- Introduce a randomization process
- Security is shown to be closely related to the security of the RSA algorithm itself

Message Encoding



RSA-PSS: Sign and Verify

- Treat EM as un-signed binary integer m
- Sign m with PR=(d, n) → s
  - Compute $s = m^d \bmod n$
- Verfiy (M, s) with PU = (e, n)
  - Compute $m' = s^e \bmod n$
  - Check whether $m'=m$

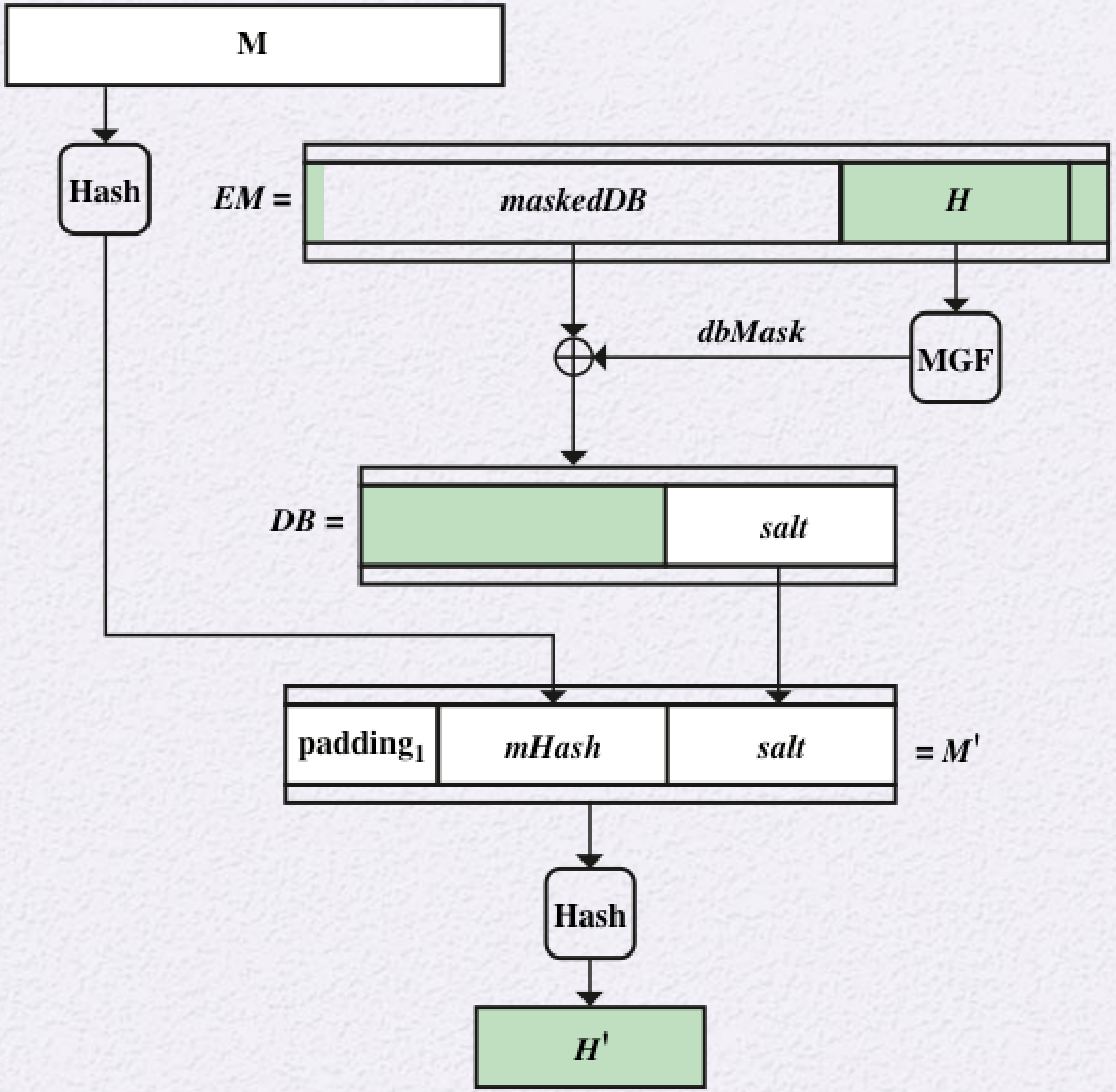


Figure 13.7 RSA-PSS EM Verification

Summary

- Digital signatures
  - Properties
  - Attacks and forgeries
  - Digital signature requirements
  - Direct digital signature
- RSA digital signature
- ElGamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature algorithm
  - The DSA approach
  - The digital signature algorithm
- RSA-PSS Digital Signature Algorithm
  - The signing operation
  - Signature verification