교과목 : 정보보호

9. Digital Signature

2023학년도 2학기
Suk-Hwan Lee

- 참고자료
  - ✓ [교재] William Stallings, Cryptography and Network Security, 7th edition
  - ✓ (교재) Mark Stamp, Information Security: Principles and Practice, 2nd edition
  - ✓ 순천향대, 서울과학기술대 강의자료 참조
  - ✓ http://wiki.hash.kr/index.php/디지털서명

- 강의내용
  - ✓ Digital Signature
    - ✓ Digital Signature
    - ✓ Services of Digital Signature
    - ✓ Attacks on Digital Signature
    - ✓ Requirements of Digital Signature
    - ✓ Direct Digital Signature
  - ✓ Digital Signature Algorithms
    - ✓ Elgamal Digital Signature Scheme
    - ✓ Schnorr Digital Signature Scheme
    - ✓ NIST Digital Signature Algorithm
    - ✓ RSA-PSS Digital Signature Scheme

Definition of Digital Signature [Wikipedia, 해시넷 참조]

- A mathematical scheme for verifying the authenticity (진위) of digital messages.
  - 송신자(A)는 A의 개인 키로 문서를 암호화하여 수신자(B)에게 보내면, B는 A의 공개키로 복호화한 후, 그것이 정말 A가 보낸 것이 맞는지 확인하는 방식
  - 송신자는 디지털 서명을 통해 본인임을 인증하고, 수신인은 해당 메시지가 위, 변조 되지 않았음을 확인한다.
  - ❖ 블록체인에 기록되는 데이터의 보안 및 무결성을 보장하는 주요 측면 중 하나이다.

- Digital Signature – Properties
  - It must verify the author and the date and time of the signature.
  - It must authenticate the contents at the time of the signature.
  - It must be verifiable by third parties, to resolve disputes.

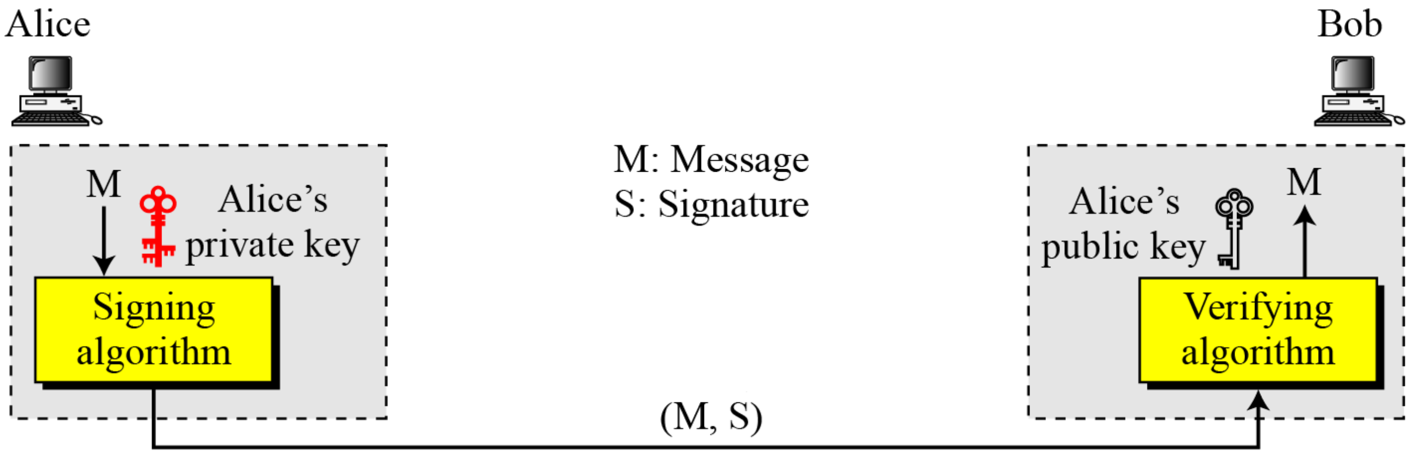
Thus, the digital signature function includes the authentication function.

Definition of Digital Signature [Wikipedia, 해시넷 참조]

- (Typically) 3개 알고리즘으로 구성
  - Key Generation : Selects a *private key* uniformly at random from a set of possible private keys. This algorithm outputs the *private key* and a corresponding *public key*.
  - Signing (서명 생성) : Given a message and a private key, produces a *signature*.
  - Signature verifying (서명 검증) : given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

- Require Two Properties
  1. 고정된 메시지와 고정된 개인 키로부터 생성된 서명의 신뢰성은 대응하는 공개키를 사용하여 검증될 수 있다
  2. 당사자의 개인 키를 알고 있다 하더라도 유효한 서명을 생성하는 것은 불가능하다.

Definition of Digital Signature [Wikipedia, 해시넷 참조]



Generic Model of Digital Signature Process

디지털 서명에서는 공개 키 시스템이 필요하다. : 서명자는 자신의 개인 키로 서명을 하고, 검증자는 서명자의 공개 키로 서명을 검증한다.

- 암호화 시스템 : 수신자의 개인 키와 공개 키가 활용
- 디지털 서명 : 송신자의 개인 키와 공개 키가 사용

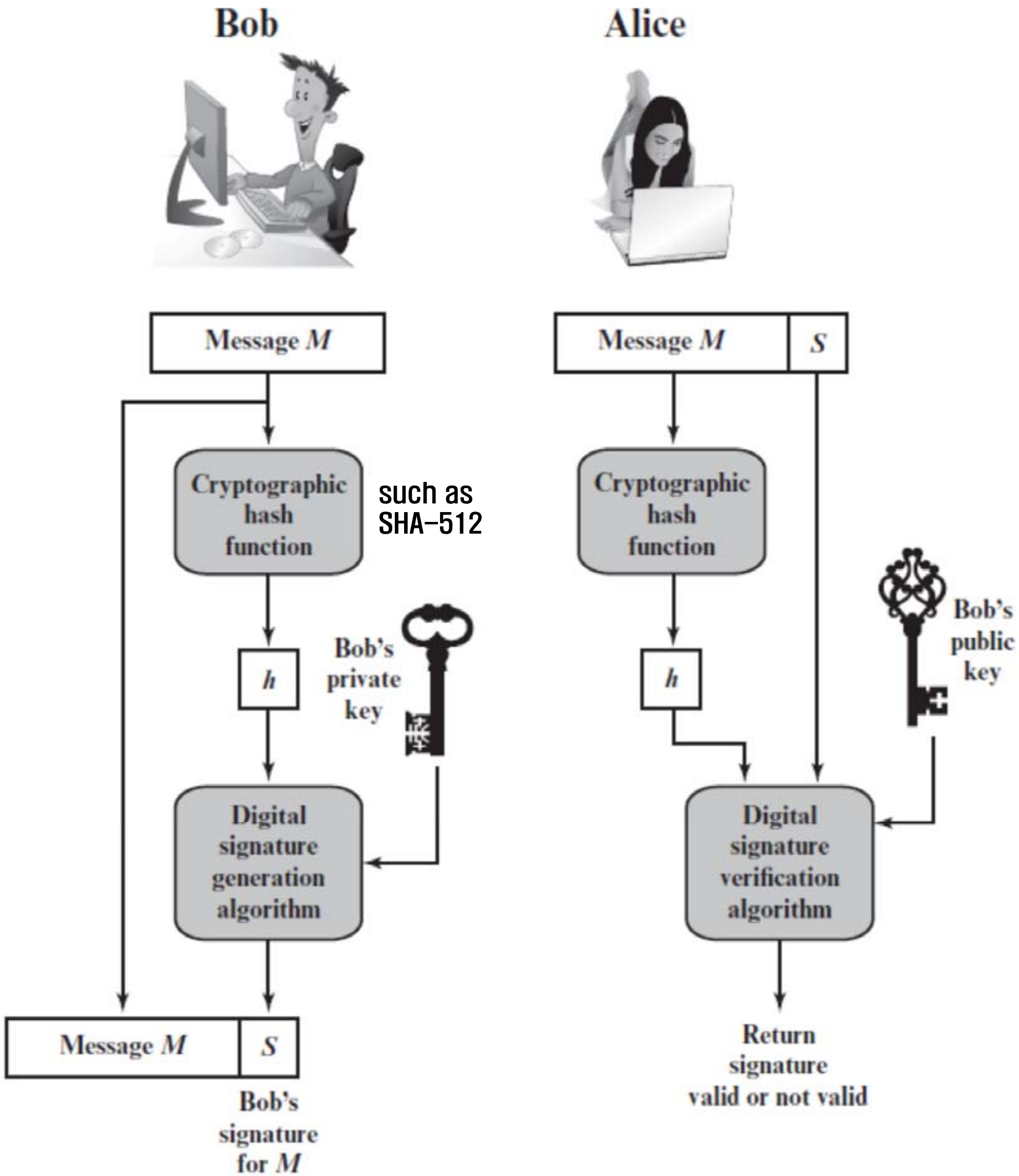
Digital Signature

- Simplified depiction of essential elements of digital signature process

  ✓ Suppose that Bob wants to send a message to Alice. Although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him.

  ✓ If the signature is valid, Alice is assured that the message must have been signed by Bob. No one else has Bob' s private key and therefore no one else could have created a signature that could be verified for this message with Bob' s public key.

  ✓ In addition, it is impossible to alter the message without access to Bob' s private key, so the message is authenticated both in terms of source and in terms of data integrity.



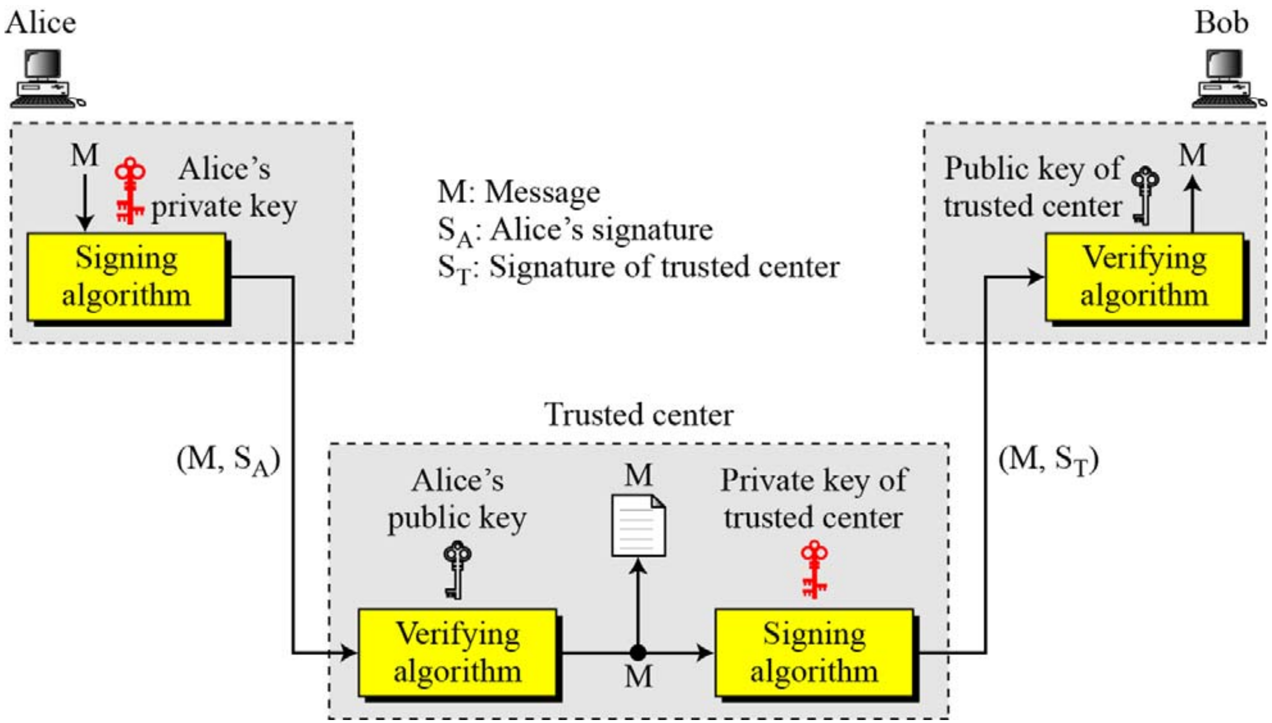
(a) Bob signs a message

(b) Alice verifies the signature

Services of Digital Signature

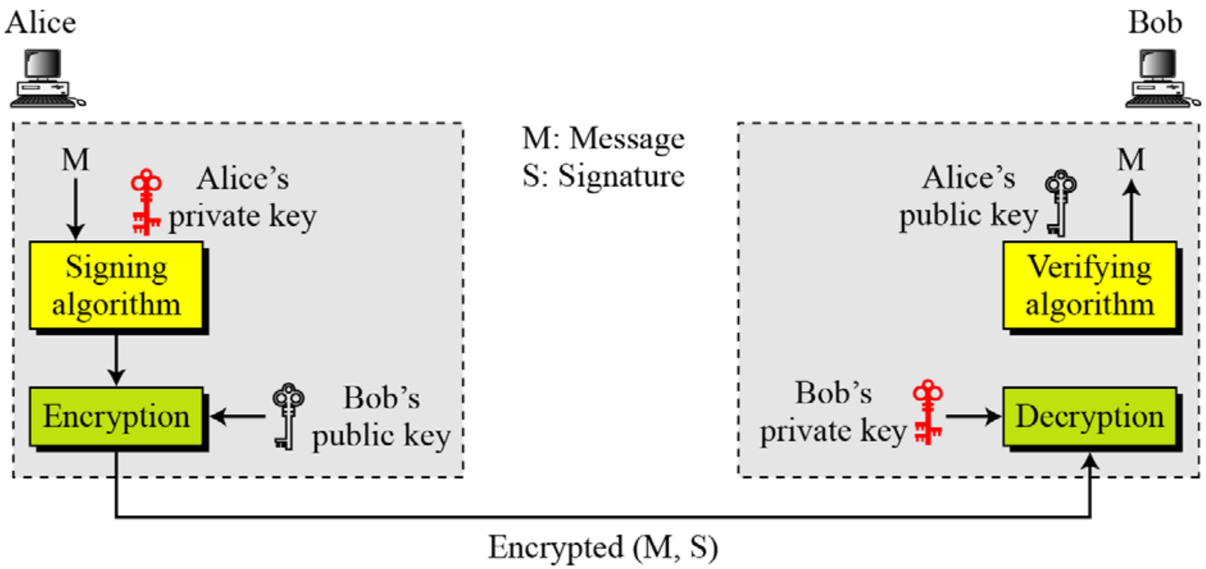
❖ Digital Signature 통해 기밀성(Confidentiality)은 보장할 수 없지만, 메시지 인증(Message Authentication), 메시지 무결성(Message Integrity), 부인봉쇄(Nonrepudiation)는 보장함. 기밀성도 보장하려면 암호확/복호화가 필요

- Message Authentication
  - ✓ 디지털 서명 구조는 메시지 인증(데이터 근원 인증) 보장
- Message Integrity
  - ✓ 메시지가 변경되면 서명이 달라짐
- Nonrepudiation
  - ✓ 신뢰받는 제 3자를 이용하면 부인봉쇄를 할 수 있다.



부인 방지를 위한 Trusted center 활용

- Confidentiality
  - ✓ Digital Signature은 Privacy를 보장해주지 못한다.
  - ✓ Privacy가 필요하다면 암호화/복호화를 할 수 있는 또 다른 수단이 적용되어야 한다.



Digital Signature 구조에 기밀성 추가

Attacks on Digital Signature

Attacks :

- ✓ A : user whose signature method is being attacked
- ✓ C : attacker

- **Key-only attack** : C only knows A' s public key.
- **Known message attack** : C is given access to a set of messages and their signatures.
- **Generic chosen message attack** : C chooses a list of messages before attempting to breaks A' s signature scheme, independent of A' s public key. *C then obtains from A valid signatures for the chosen messages.* The attack is generic, because it does not depend on A' s public key.
- **Directed chosen message attack** : Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A' s public key but before any signatures are seen.
- **Adaptive chosen message attack** : C is allowed to use A as an "oracle." This means that C may request from A signatures of messages that depend on previously obtained message-signature pairs.

Attacks on Digital Signature

Forgery :

✓ C can do any of the following with a non-negligible probability.

- **Total break** : C determines A’ s private key.
- **Universal forgery** : C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
- **Selective forgery** : C forges a signature for a particular message chosen by C.
- **Existential forgery** : C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.

Requirements

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information only known to the sender to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

➢ Secure hash function provides a basis for satisfying these requirements.

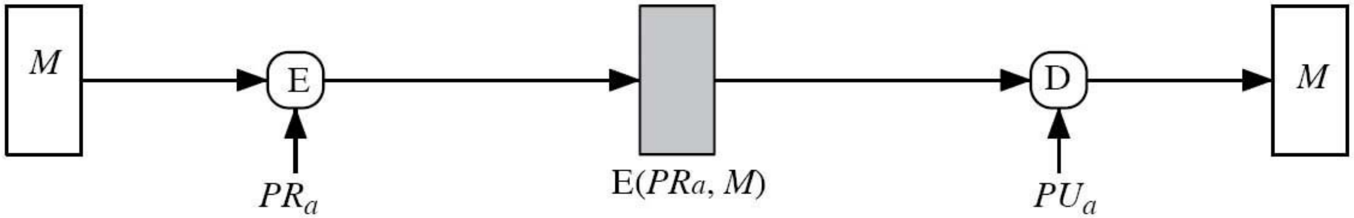
Direct Digital Signature

- 공개키 암호 알고리즘과 hash function을 사용한 모델
  - ✓ Involve only Sender & Receiver
  - ✓ Assumed receiver has sender's public-key, digital signature made by sender signing entire message or hash with private-key
  - ✓ Can encrypt using receiver's public-key
  - ✓ important that sign first then encrypt message & signature
  - ✓ 대표적인 예 : ELGamal Digital Signature, Schnoor Digital Signature

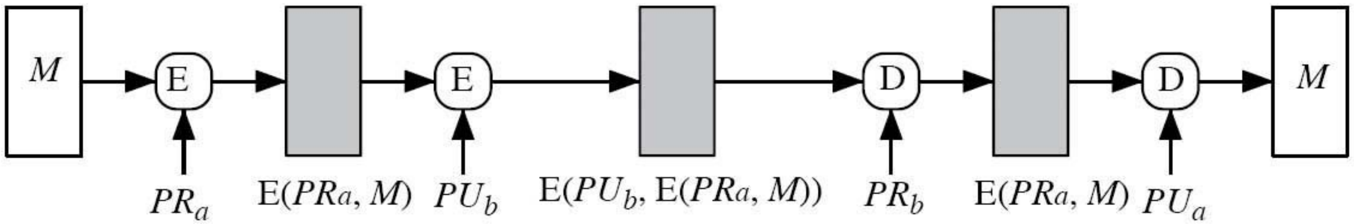
- Weakness
  - ✓ Sender의 private-key 안전성에 따라 유효성이 달라질수 있음
  - ✓ Sender가 private-key를 분실하거나 도난당했다고 거짓주장 하거나, 실제로 Sender의 private-key가 도난당할 수 있으며, 제3자의 개입이 필요하다는 문제점이 있음
  - ➢ 독립적인 검증 프로세스가 없기 때문에 발신자와 수신자 간의 신뢰가 필요함.
  - ➢ 이 프로세스에서는 보낸 사람에게 private-key가 있어야 하고 받는 사람에게만 public-key가 있어야 함

Types of Digital Signature

Direct Digital Signature



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Weakness: Security depends on sender's private-key

Arbitrated Digital Signature

- Involves use of arbiter A
  - ✓ validates any signed message
  - ✓ then dated and sent to recipient
- Requires suitable level of trust in arbiter
- Can be implemented with either private or public-key algorithms
- Arbiter may or may not see message

➢ Arbiter이 one-way이나 상대방에 대한 편견을 보여줄 수 있는 가능성이 있음

Signature Signature

(1) $X \rightarrow A$: $ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$
(2) $A \rightarrow Y$: $E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])$

(c) Public-Key Encryption, Arbiter Does Not See Message

Notations:

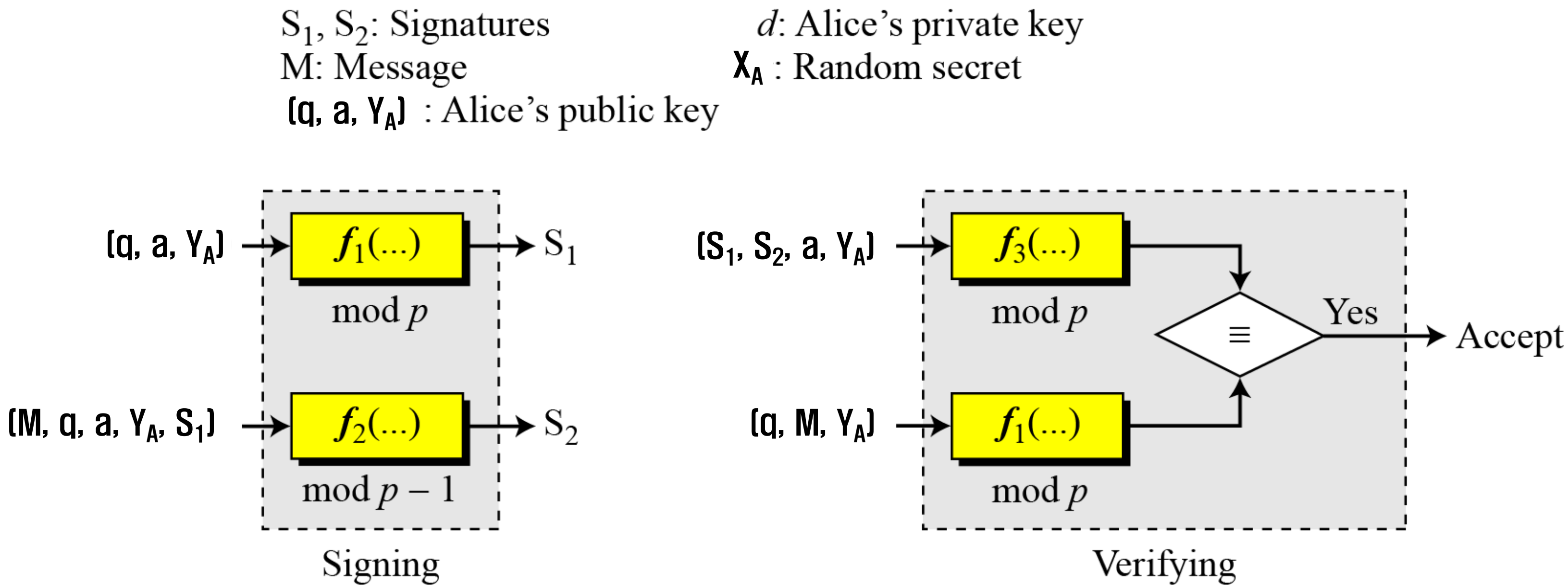
X=sender	M=message
Y=recipient	T=time stamp
A=Arbiter	PR_X=X's private key
ID_X=ID of X	PU_Y=Y's public key
	PR_A=A's private key

Weakness: twice public-key encryptions on the message

Digital Signature Algorithms

1. Elgamal Digital Signature Scheme

- Elgamal signature is designed to enable encryption by user s private key, and decryption by the user s public key
  - ✓ involves the use of the private key for digital signature generation and the public key for digital signature verification



Elgamal Digital Signature에 대한 일반적 아이디어

1. Elgamal Digital Signature Scheme

- First, there are prime number q and its primitive root a.
- Key generation
  - ✓ Generate a random integer X_A , such that $1 < X_A < q-1$.
  - ✓ Compute $Y_A = a^{X_A} \bmod q$
  - ✓ A' s private key is X_A ; A' s public key is $\{q, a, Y_A\}$
- (User A) form digital signature as
  - ✓ Compute the hash $m=H(M)$, which m is an integer in $0 \le m \le q-1$
  - ✓ Choose a random integer K such that $1 \le K \le q-1$ and $\gcd(K, q-1) = 1$. That is, K is relatively prime to $q-1$.
  - ✓ Compute $S_1 = a^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
  - ✓ Compute $K^{-1} \bmod (q-1)$. That is, compute the inverse of K modulo $q - 1$.
  - ✓ Compute $S^2 = K^{-1} (m-X_A S_1) \bmod (q-1)$.
  - ✓ The signature consists of the pair (S_1, S_2).

1. Elgamal Digital Signature Scheme

- (User B) Verify the signature as
  - ✓ Compute $V_1 = a^m \bmod q$
  - ✓ Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$
- The signature is valid if $V_1 = V_2$

❖ Let us demonstrate that this is so. Assume that the equality is true. Then we have

$\alpha^m \bmod q = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$	assume $V_1 = V_2$
$\alpha^m \bmod q = \alpha^{X_A S_1}\alpha^{KS_2} \bmod q$	substituting for Y_A and S_1
$\alpha^{m-X_A S_1} \bmod q = \alpha^{KS_2} \bmod q$	rearranging terms
$m - X_A S_1 \equiv KS_2 \bmod (q-1)$	property of primitive roots
$m - X_A S_1 \equiv KK^{-1}(m - X_A S_1) \bmod (q-1)$	substituting for S_2

- Example
  - ✓ Prime field GF(19); q=19. It has primitive roots {2, 3, 10, 13, 14, 15}.
  - ✓ We choose a=10.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^{16} \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S_1 = \alpha^K \bmod q = 10^5 \bmod 19 = 3$ (see Table 2.7).
3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
4. $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1) = 11(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

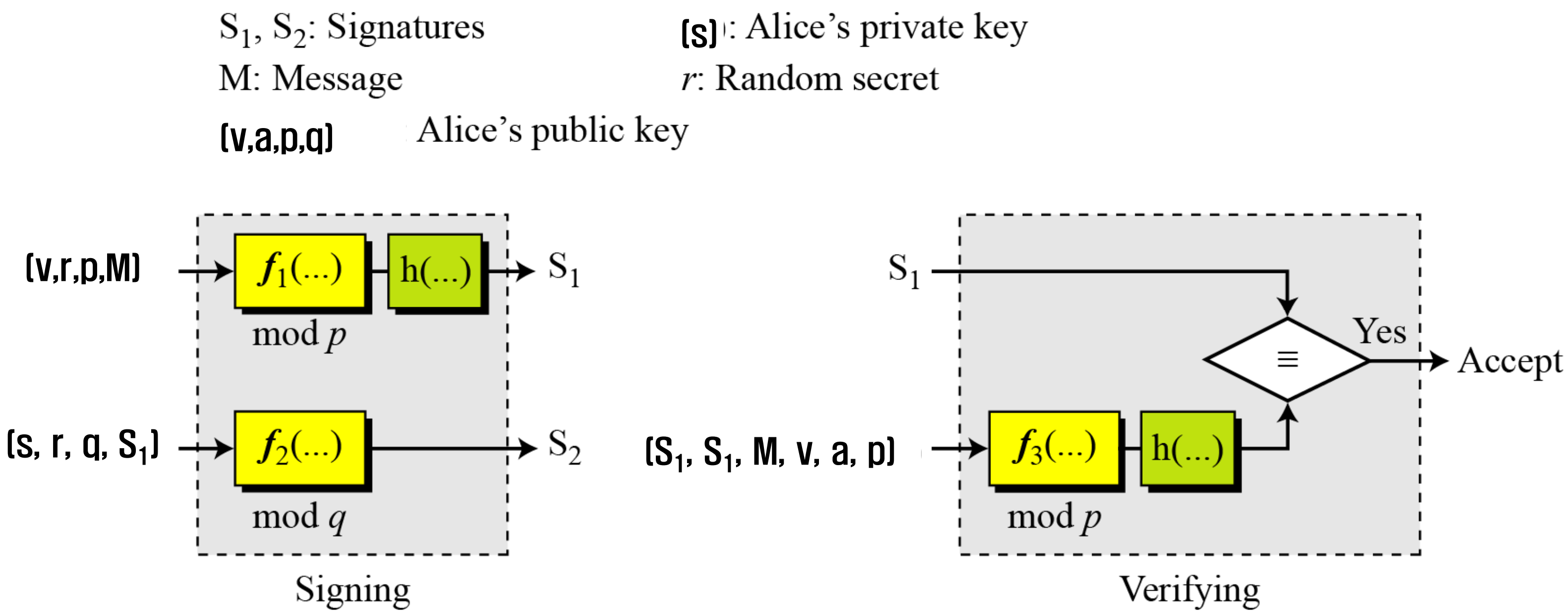
Bob can verify the signature as follows.

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$.
2. $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$.

Thus, the signature is valid because $V_1 = V_2$.

2. Schnorr Digital Signature Scheme

- It is based on discrete logarithms
- Minimizes the message-dependent amount of computation required to generate a signature



Schnorr Digital Signature에 대한 일반적 아이디어

- (private/public) Key generation
  - ✓ Choose primes p and q, such as q is a prime factor of p-1
    - p is 1024 bit , q is 160bit num (as SHA-1)
  - ✓ Choose an integer a, such that $a^q = 1 \bmod p$. the values a, p, and q comprise a global public key that can be common to a group of user
  - ✓ Choose a random integer s with $0 < s < q$. this is the user s private key
  - ✓ Calculate $v = a^{-s} \bmod p$, this is the user public key.
  - ✓ a,p,q : global public key
  - ✓ s: user s private key
  - ✓ v: user s public key

2. Schnorr Digital Signature Scheme

- Signature generation
  - ✓ Choose a random integer r with $0<r<q$ and compute $x= a^r \bmod p$.
    - This computation is a preprocessing state independent of the message M to be signed
  - ✓ Concatenate the message M with x and hash the result to compute the value e: $e = H(M||x)$
  - ✓ Compute $y = (r+se) \bmod q$ the signature consists of the pair (e,y).

- Signature verification
  - ✓ Compute $x' = a^y v^e \bmod p$.
  - ✓ Verify that $e = H(M||x')$

- To see the verification works, observe that
  - ✓ $X' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$
  - ✓ Hence, $H(M||x') = H(M||x)$

3. NIST Digital Signature Algorithm

- DSA is designed to provide only the digital signature by the National Institute of Standards and Technology. DSA make use of the Secure Hash Algorithm (SHA).
- Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, It is a public-key technique.
- It is FIPS 186 proposed in 1991 and revised in 1993 → FIPS 186-2 in 2000, → FIPS 186-3 in 2009 and FIPS 186-4 in 2013 (based on RSA and elliptic curve cryptography)

- DSA is based on the difficulty of computing DLP (Discrete Logarithm Problem)
- Also it is based on the Elgamal and Schnorr scheme.

- It’ s main processes are independent on Message.

3. NIST Digital Signature Scheme (DSA)

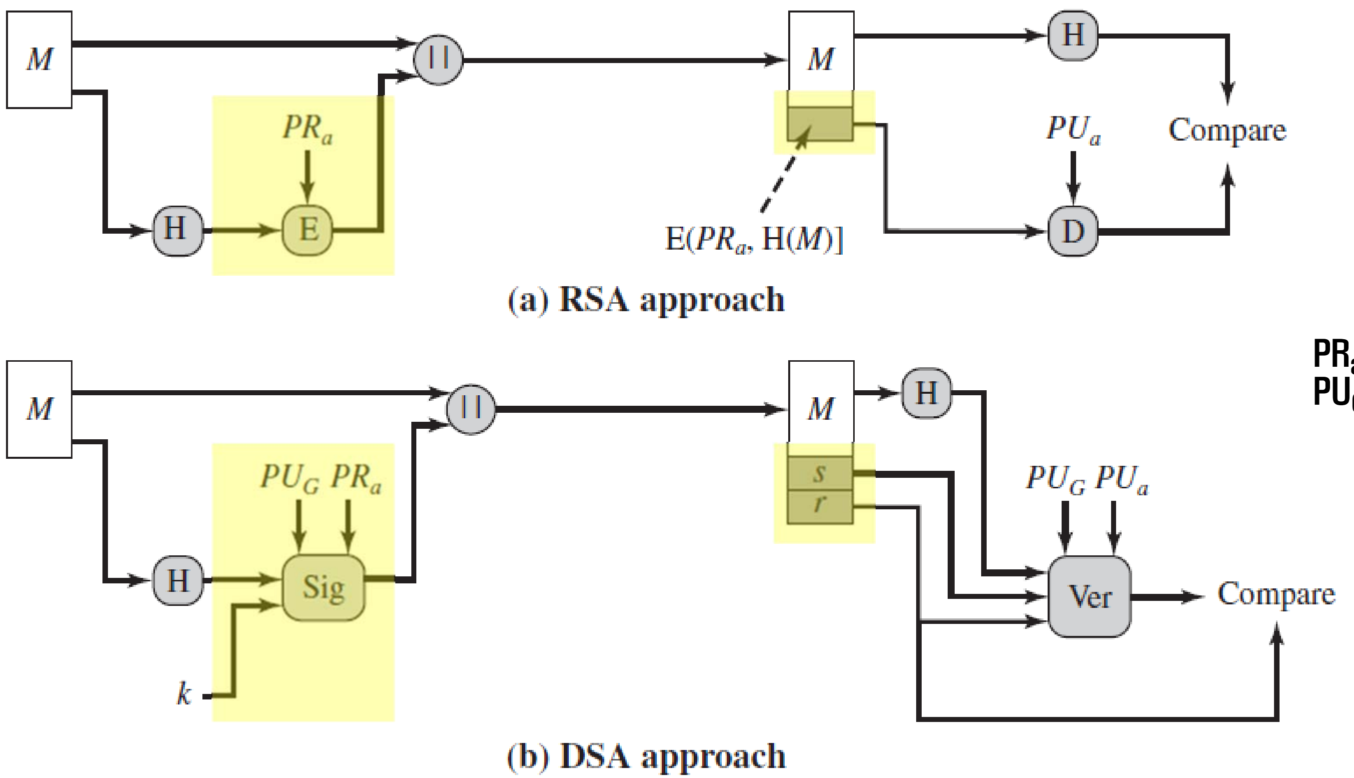


Figure 13.3 Two Approaches to Digital Signatures

PR_a : Private key of sender a
PU_G : Global Public key

3. NIST Digital Signature Scheme (DSA)

- Key generation

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

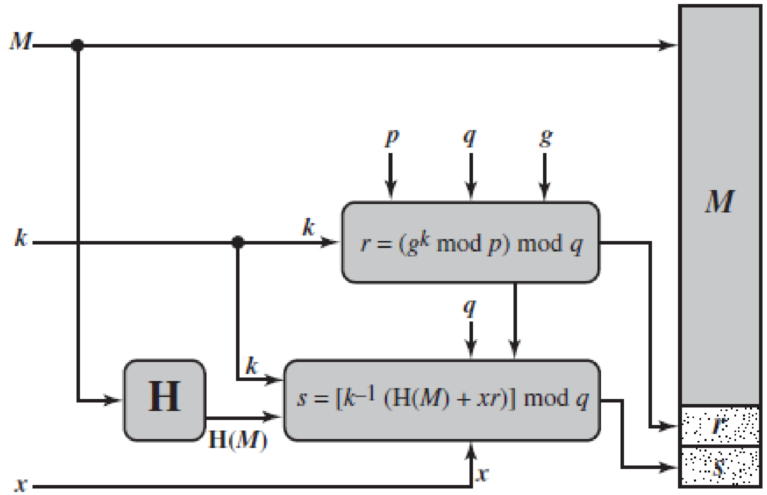
$y = g^x \bmod p$

User's Per-Message Secret Number

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

Figure 13.4 The Digital Signature Algorithm (DSA)

- Signature generation



(a) Signing

Signing

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

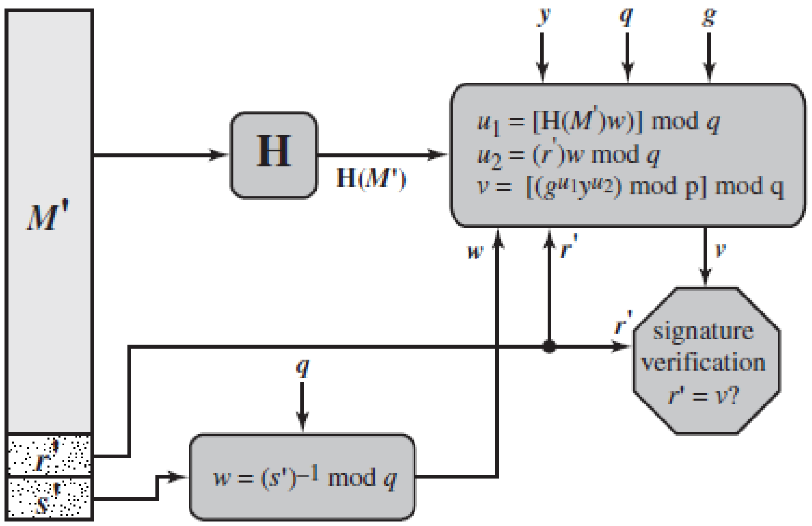
$s = [k^{-1} (H(M) + xr)] \bmod q$

Signature $= (r, s)$

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

3. NIST Digital Signature Scheme (DSA)

- Signature Verification



(b) Verifying

Figure 13.5 DSA Signing and Verifying

Verifying

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

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

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

TEST: $v = r'$

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

- RSA-PSS (RSA Probabilistic Signature Scheme), which is the latest of the RSA schemes and the one that RSA Laboratories recommends as the most secure of the RSA schemes.
- RSA-based schemes differ mainly in the padding format, and in how the verification operation determines that the hash and message representative are consistent
- Before PSS, there was not been possible to develop mathematical proof that scheme is secure as RSA encryption/decryption.

Mask Generation Function (MGF)

- MGF is used as a building fixed length output.
- MGF(X, maskLen) : A pseudo random function that has as the input of a string X in any length and desired length L (maskLen) in octets of the ouput.
- Typically based on hash function as SHA-1
- In RSA-PSS, MGF1 is used with the parameters

Option	Hash:	Hash function with output hLen octets
Input	X; maskLen:	Octet string to be masked Length in octets of mask
Output	mask:	An octet string of length maskLen

➢ MGF1 is defined as follows:

- Initialize variables
  T = empty string
  K = [maskLen / hLen] −1

- Calculate intermediate values
  for *counter* = 0 to k
  Represent *counter* as a 32-bit string C
  T = T || Hash(X || C)

- Output results
  Mask = the leading maskLen octets of T

- If maskLen = hLen,
  the output is the hash of the X concatenated with 32-bit counter value of 0.
- If maskLen ≥ hLen,
  MGF1 iterate by hashing X concatenated with the counter and appending that to the current string T.

- output : Hash(X||0) || Hash(X||1) || ··· ||Hash(X||K)

- This is repeated until T is greater or equal with maskLen, at which point the output is the first maskLen octets of T

The Signing Operation

➢ Message Encoding

- Generate from a message M a fixed-length message digest, (encoded message, EM)

Definitions of parameters and functions

Options	Hash	hash function with output *hLen* octets. The current preferred alternative is SHA-1, which produces a 20-octet hash value.
	MGF	mask generation function. The current specification calls for MGF1.
	sLen	length in octets of the salt. Typically $sLen = hLen$, which for the current version is 20 octets.
Input	*M*	message to be encoded for signing.
	emBits	This value is one less than the length in bits of the RSA modulus *n*.
Output	*EM*	encoded message. This is the message digest that will be encrypted to form the digital signature.
Parameters	*emLen*	length of *EM* in octets $= \lceil emBits/8 \rceil$.
	padding_1	hexadecimal string 00 00 00 00 00 00 00 00; that is, a string of 64 zero bits.
	padding_2	hexadecimal string of 00 octets with a length $(emLen - sLen - hLen - 2)$ octets, followed by the hexadecimal octet with value 01.
	salt	a pseudorandom number.
	bc	the hexadecimal value BC.

Message Encoding Process

1. Generate the hash value of M:
   mHash = Hash(M)
2. Generate a pseudorandom octet string salt and form block M′
   M′ = $padding_1$ || mHash || salt
3. Generate the hash value of M′:
   H = Hash(M′)
4. Form data block DB
   DB = $padding_2$ || salt
5. Calculate the MGF value of H:
   dbMask=MGF(H, emLen−hLen−1)
6. Calculate maskedDB
   maskedDB = DB XOR dbMask
7. Set the leftmost 8emLen−emBits bits of the leftmost octet in masked DB to 0
8. EM = masked DB || H || bc

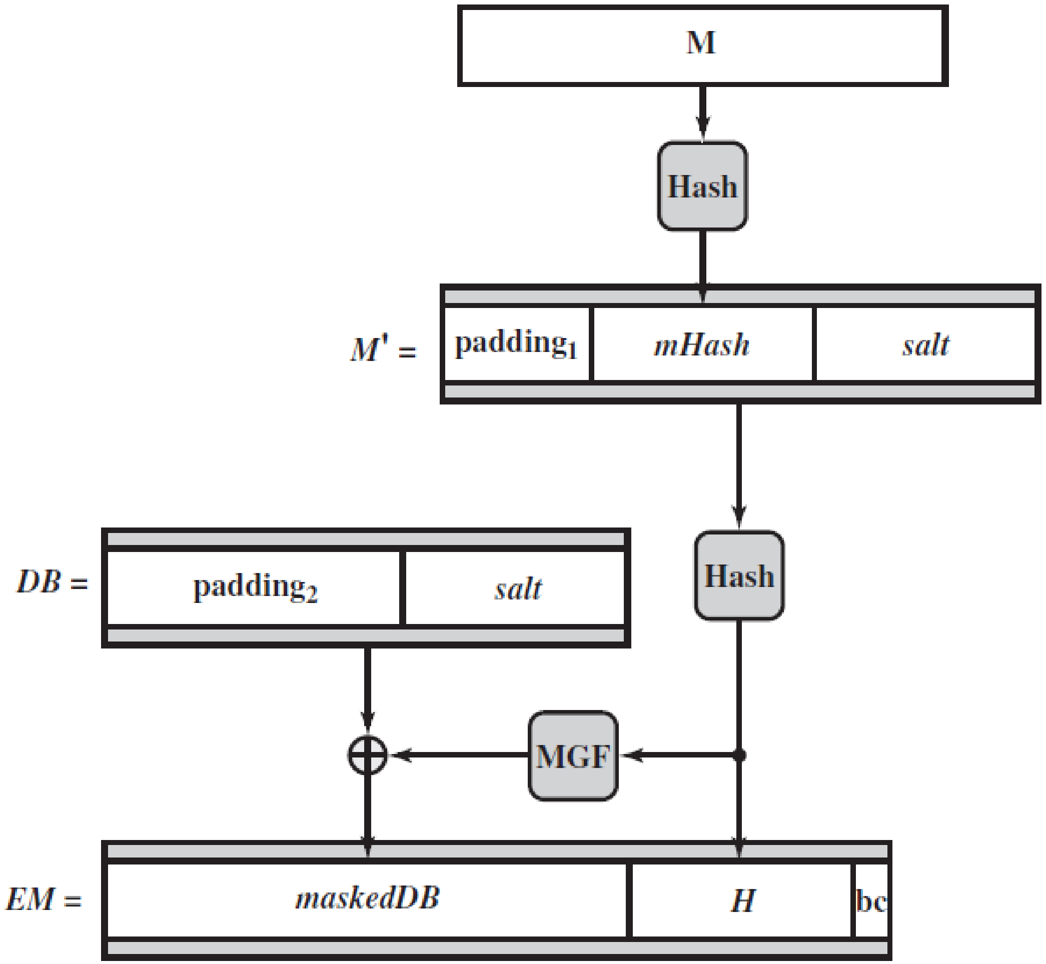


Figure 13.7 RSA-PSS Encoding

4. RSA-PSS Digital Signature Scheme

➢ Forming the Signature

- Private key {d, n}, public key {e, n} – (RSA algorithm)
- Treat the octet string EM as an unsigned, nonnegative binary integer m. the signature s is formed by encrypting octet string S of length k octets.

$$s = m^d \bmod n$$

- Let k be the length in octets of the RSA modulus n.
- If key size is 2048bits, then k = 2048/8=256.
- Then convert the signature value s into the octet string S of length k octets.

The Signing Verification

➢ Decryption

- The message digest m is recovered by decrypting s

$$m = s^e \bmod n$$

- Then, convert m to EM of length emLen = [modBits −1]/8]octets, where modBits is the length in bits of the modulus n.

The Signing Verification

- **EM Verification**

1. Generate the hash value of M: mHash = Hash(M)
2. If emLen<hLen+sLen+2, output "inconsistent" and stop
3. If the rightmost octet of EM does not have hexadecimal value BC, output "inconsistent" and stop
4. Let maskedDB be the leftmost emLen − hLen − 1 octets of EM, and let H be the next hLen octets
5. If the leftmost 8emLen − emBits bits of the leftmost octet in maskedDB are not all equal to zero, output "inconsistent" and stop
6. Calculate dbMask = MGF (H, emLen − hLen − 1)

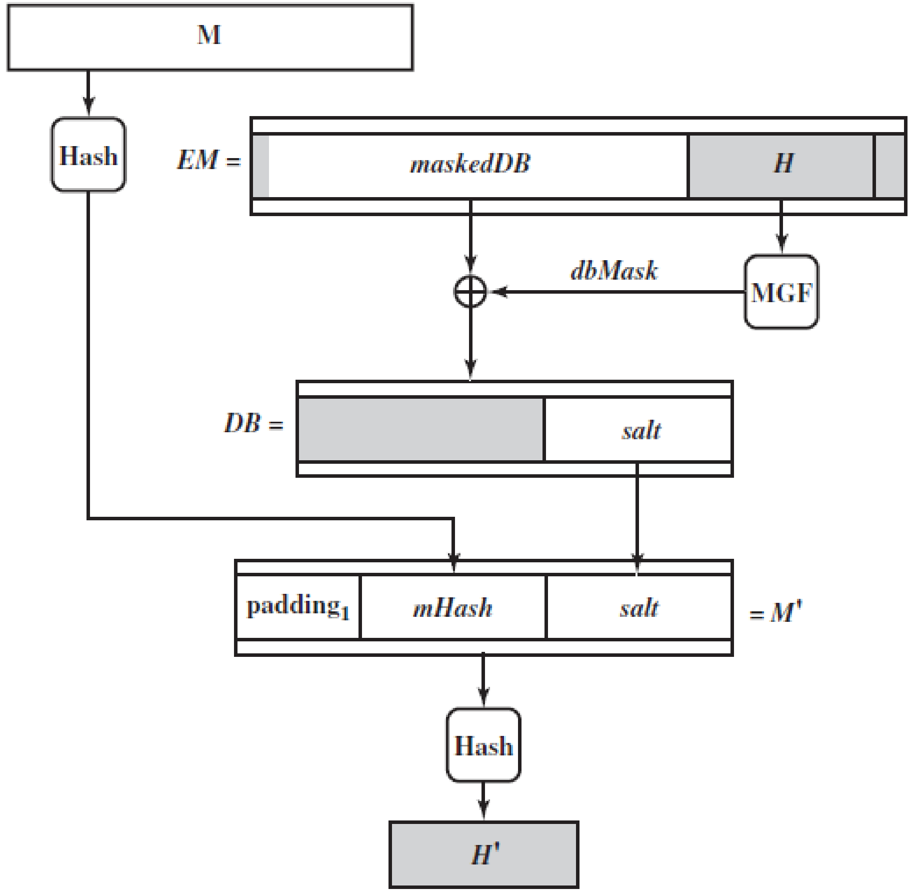


Figure 13.8 RSA-PSS EM Verification

The Signing Verification

- **EM Verification**

7. Calculate DB = maskedDB XOR dbMsk
8. Set the leftmost (8emLen–emBits) bits of the leftmost octet in DB to zero
9. If the leftmost (emLen – hLen – sLen – 1) octets of DB are not equal to $padding_2$, output "inconsistent" and stop
10. Let salt be the last sLen octets of DB
11. Form block M′ = padding1 || mHash || salt
12. Generate the hash value of M' : H′ = Hash(M′)
13. If H = H', output "consistent." Otherwise, output "inconsistent"

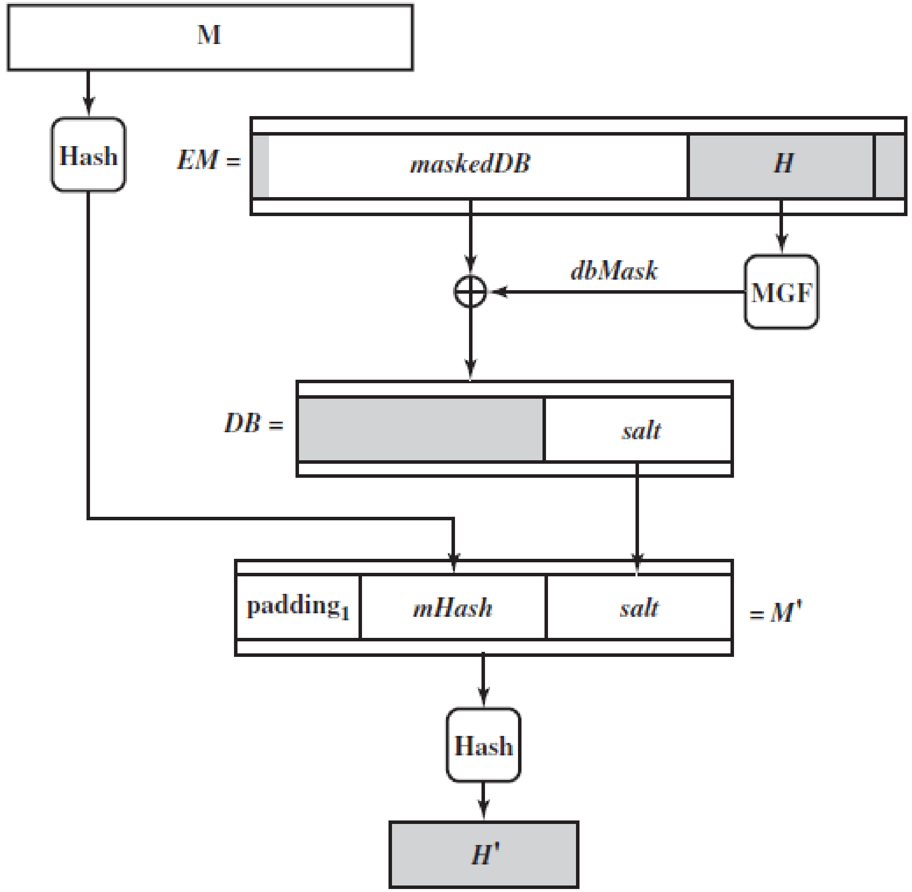


Figure 13.8 RSA-PSS EM Verification