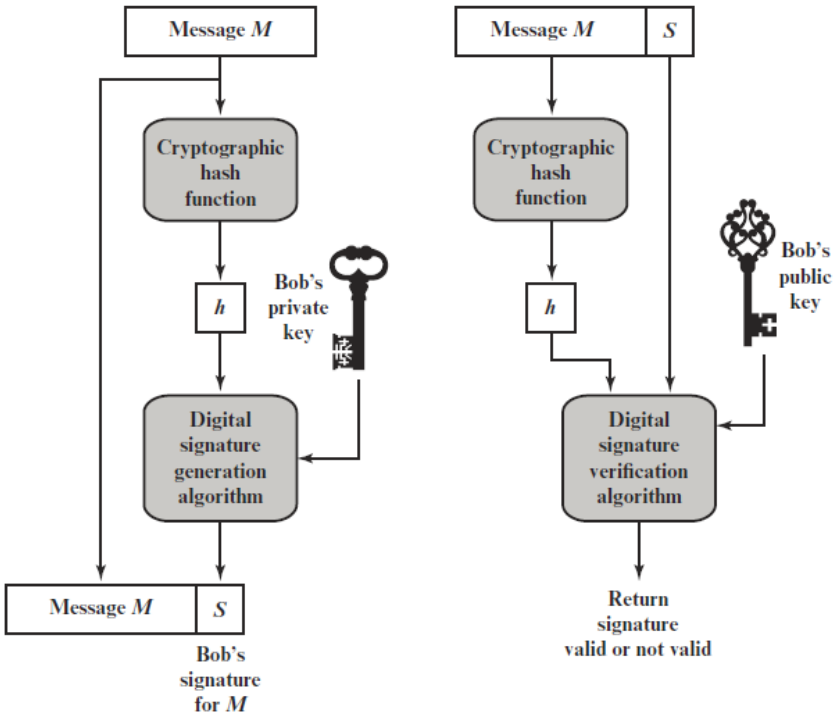
Chapter 2: Digital Signature

2.1 Introduction

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. For this purpose, Bob uses a secure hash function, such as SHA-512, to generate a hash value for the message. That hash value, together with Bob's private key serves as input to a *digital signature generation algorithm*, which produces a short block that functions as a **digital signature.** Bob sends the message with the signature attached.

When Alice receives the message plus signature, she (1) calculates a hash value for the message; (2) provides the hash value and Bob's public key as inputs to a digital signature verification algorithm. If the algorithm returns the result that 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.



The signature scheme involves the use of the private key for digital signature generation and the public key for digital signature verification.

2.2 ELGAMAL DIGITAL SIGNATURE SCHEME

As with Elgamal encryption, the global elements of Elgamal digital signature are a prime number q and a, which is a primitive root of q. User A generates a private/public key pair as follows.

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = a^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, a, Y_A\}$.

To sign a message M, user A first computes the hash m = H(M), such that m is an integer in the range $0 \leq m \leq q - 1$. A then forms a digital signature as follows.

1. Choose a random integer K such that $1 \leq K \leq q - 1$ and gcd(K, q - 1) = 1. That is, K is relatively prime to q - 1.
2. Compute $S_1 = a^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo q - 1.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

Any user B can verify the signature as follows.

1. Compute $V_1 = a^m \bmod q$.
2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.

The signature is valid if $V_1 = V_2$

For example, let us start with the prime 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 $Y_A = a^{X_A} \bmod q = a^{16} \bmod 19 = 4$.

3. Alice's private key is 16; Alice's pubic key is $\{q, a, Y_A\} = \{19, 10, 4\}$.

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. $S_1 = a^K \bmod q = 10^5 \bmod 19 = 3$

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. $V_1 = a^m \bmod q = 10^{14} \bmod 19 = 16$.

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 V1 = V2.

2.3 SCHNORR DIGITAL SIGNATURE SCHEME

As with the Elgamal digital signature scheme, the Schnorr signature scheme is based on discrete logarithms The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.

The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.

1. Choose primes p and q, such that q is a prime factor of p - 1.
2. 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 users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

A user with private key s and public key v generates a signature as follows.

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

Any other user can verify the signature as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that e = H (M || x').

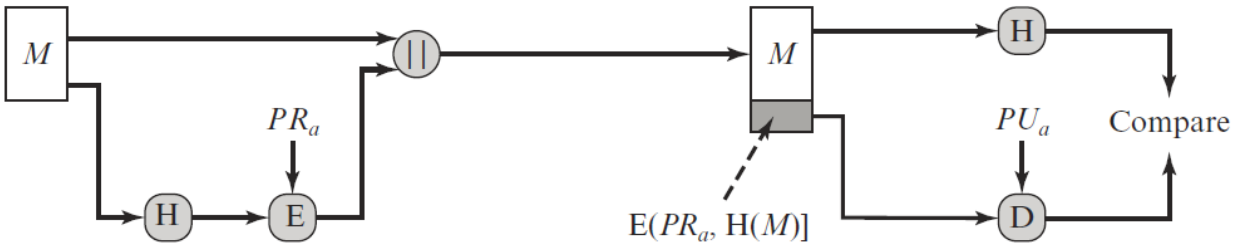
To see that the verification works, observe that

$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$

Hence, H (M || x')= H (M || x).

2.4 DIGITAL SIGNATURE Using RSA

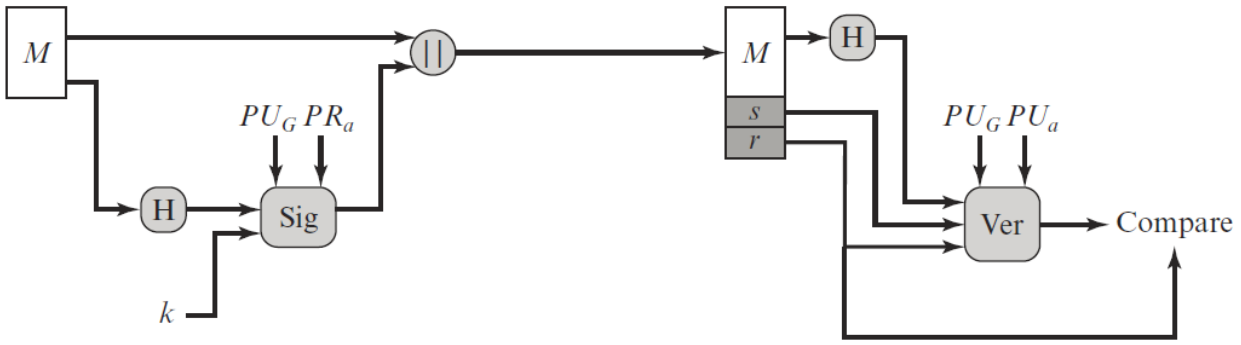
The message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.



2.5 NIST DIGITAL SIGNATURE ALGORITHM

The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G). The result is a signature consisting of two components, labeled s and r.

At the receiving end, the hash code of the incoming message is generated. The hash code and the signature are inputs to a verification function. The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key. The output of the verification function is a value that is equal to the signature component r if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.



DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by Elgamal and Schnorr.

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length L 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 $= h(p-1)/q$ is an exponent mod 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

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

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

Signature $= (r, s)$

Verifying

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

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

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

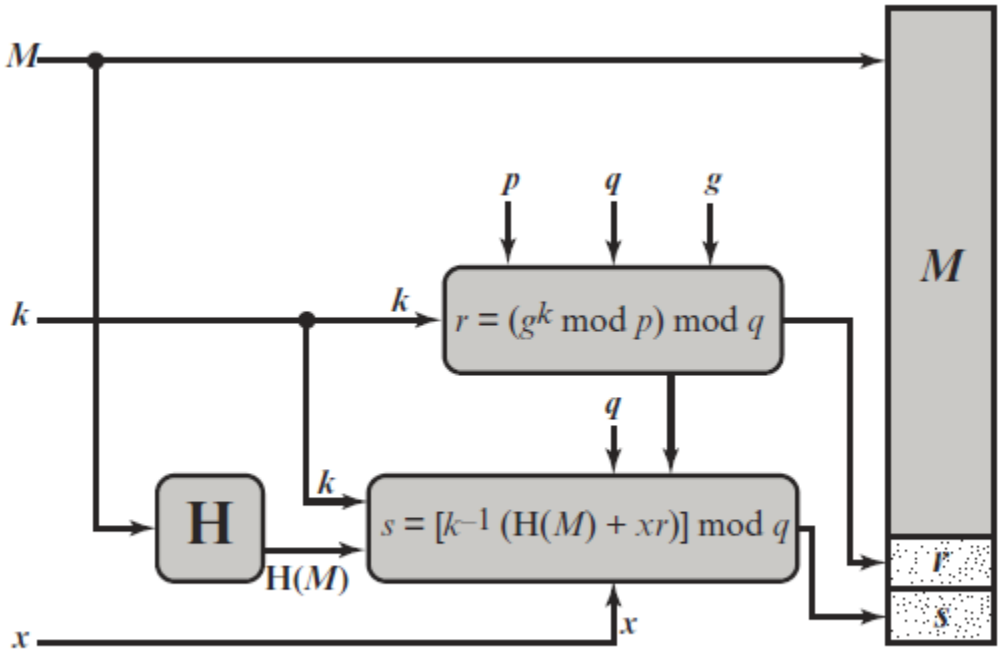
$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

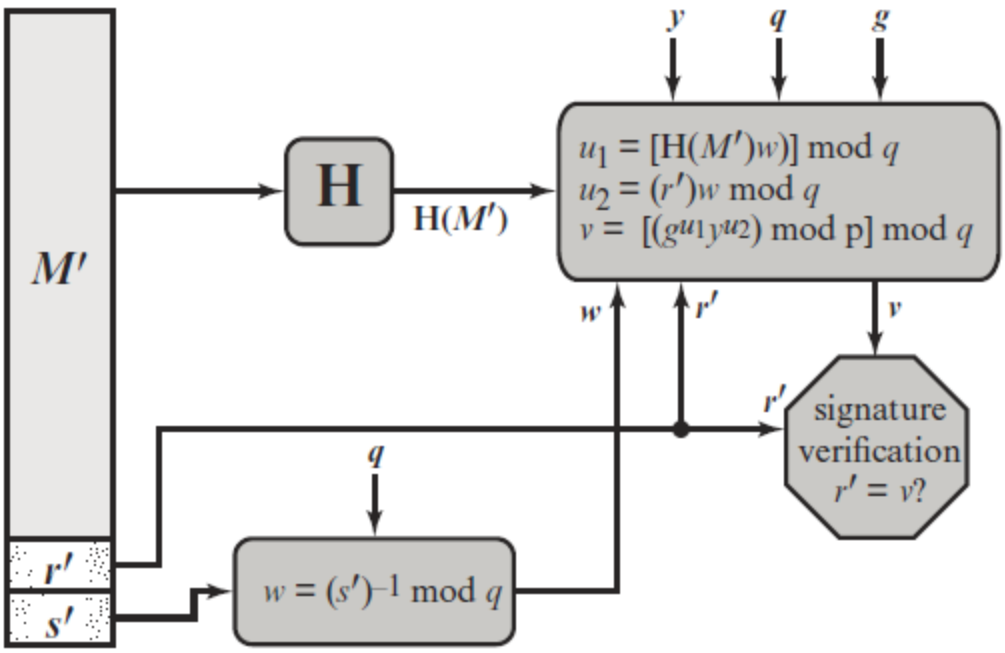
M = message to be signed

H(M) = hash of M using SHA-1

M', r', s' = received versions of M, r, s



(a) Signing


y
q
g
H
H(M′)
u1 = [H(M′)w)] mod q
u2 = (r′)w mod q
v = [(gu1yu2) mod p] mod q
M′
w
r′
v
r′
signature
verification
r′ = v?
q
r′
s′
w = (s′)–1 mod q

(b) Verifying