INFORMATION SECURITY (2170709)

Digital Signature

Digital Signature

- A Digital Signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature.

- Message authentication protects two parties who exchange messages from any third party.
- However, it does not protect the two parties against each other.
  - Receiver B may forge a different message and claim that it came from sender A.
  - Sender A can deny sending the message.

Requirements for Digital Signature

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique 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.

Requirements for 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.

Digital Signature

- A variety of approaches have been proposed for digital signature.

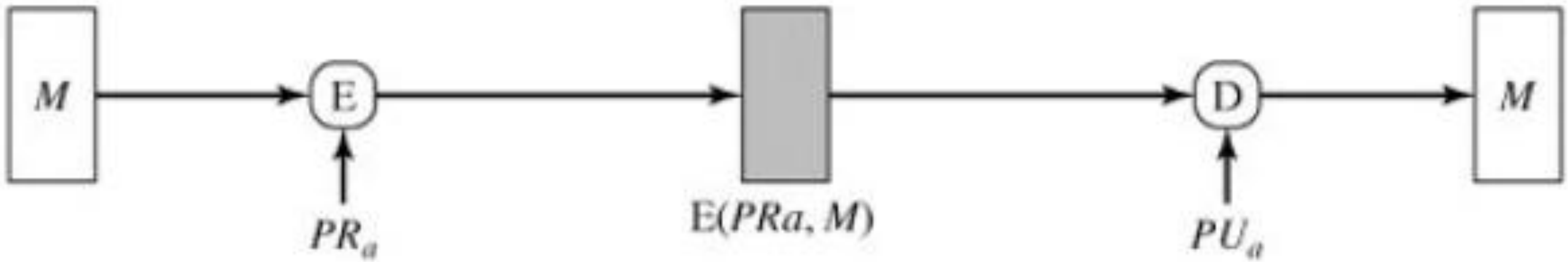
- These approaches fall into two categories:
  - Direct Digital Signature
  - Arbitrated Digital Signature

Direct Digital Signature

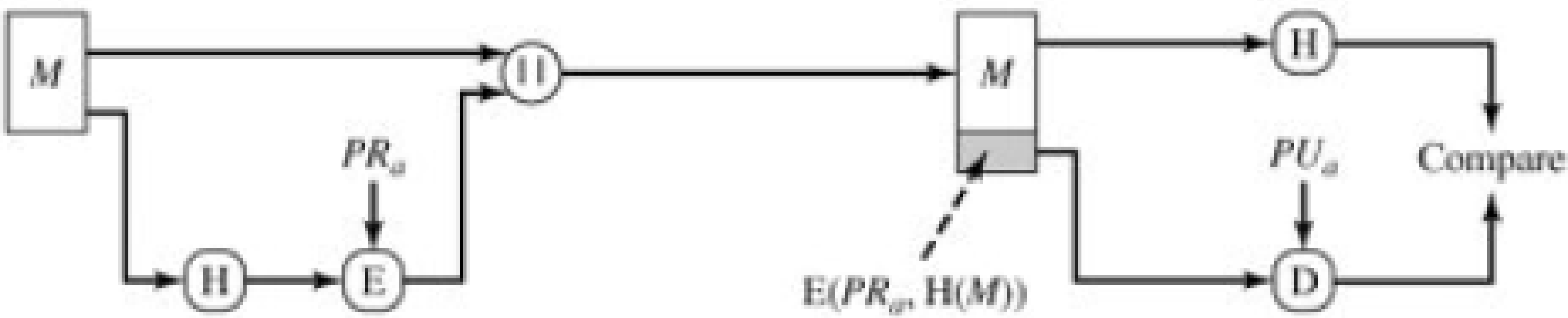
- Involves only the two communicating parties (source and destination)

- A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key.

Direct Digital Signature



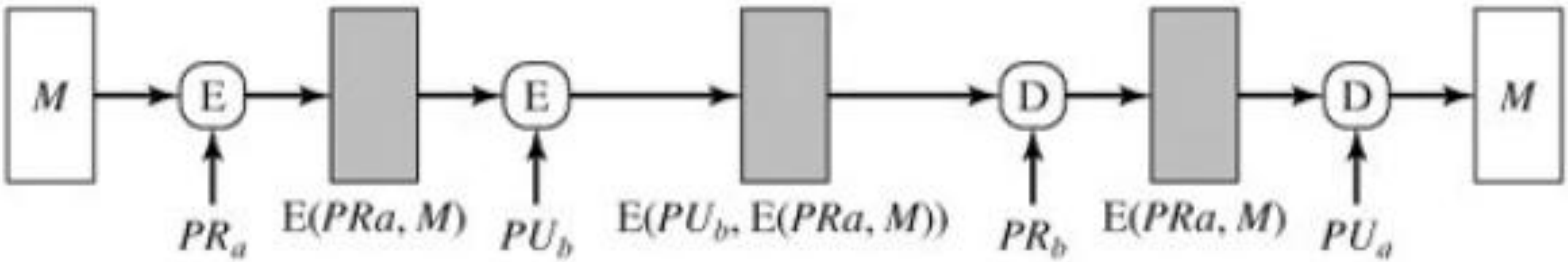
(c) Public-key encryption: authentication and signature



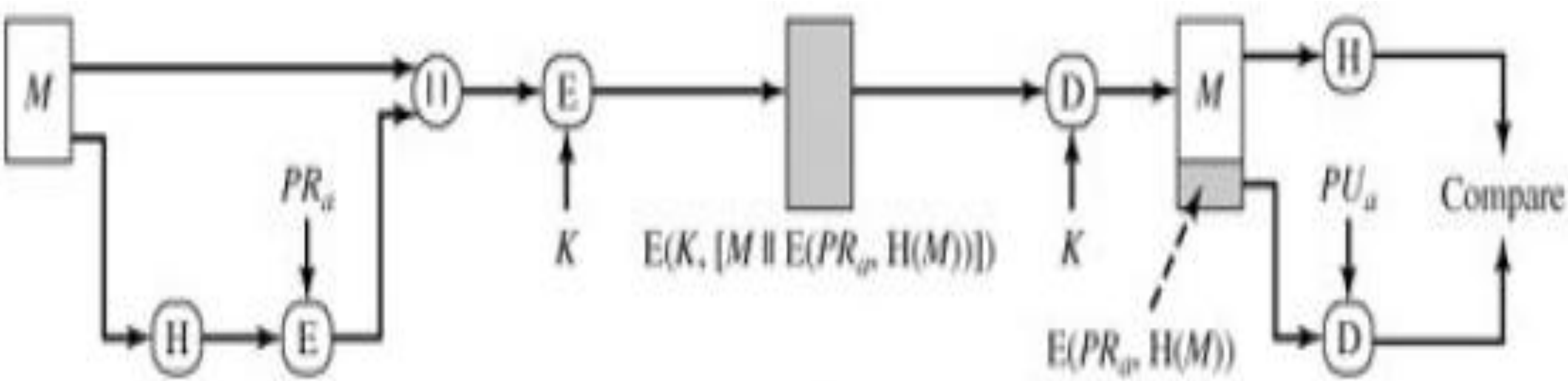
Direct Digital Signature

- Confidentiality can be provided by further encrypting the entire message plus signature with either the receiver's public key (public-key encryption) or a shared secret key (symmetric encryption);

Direct Digital Signature



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



Direct Digital Signature

- All direct schemes described so far share a common weakness. The validity of the scheme depends on the security of the sender's private key.

- If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen.

- Administrative controls relating to the security of private keys can be employed to thwart or at least weaken this ploy.

Arbitrated Digital Signature

- Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content.

- The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter.

Arbitrated Digital Signature

Table 13.1. Arbitrated Digital Signature Techniques

(1) $X \longrightarrow A$: $M \| E(K_{xa}, [ID_X \| H(M)])$

(2) $A \longrightarrow Y$: $E(K_{ay}, [ID_X \| M \| E(K_{xa}, [ID_X \| H(M)]) \| T])$

(a) Conventional Encryption, Arbiter Sees Message

(1) $X \longrightarrow A$: $ID_X \| E(K_{xy}, M) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M))])$

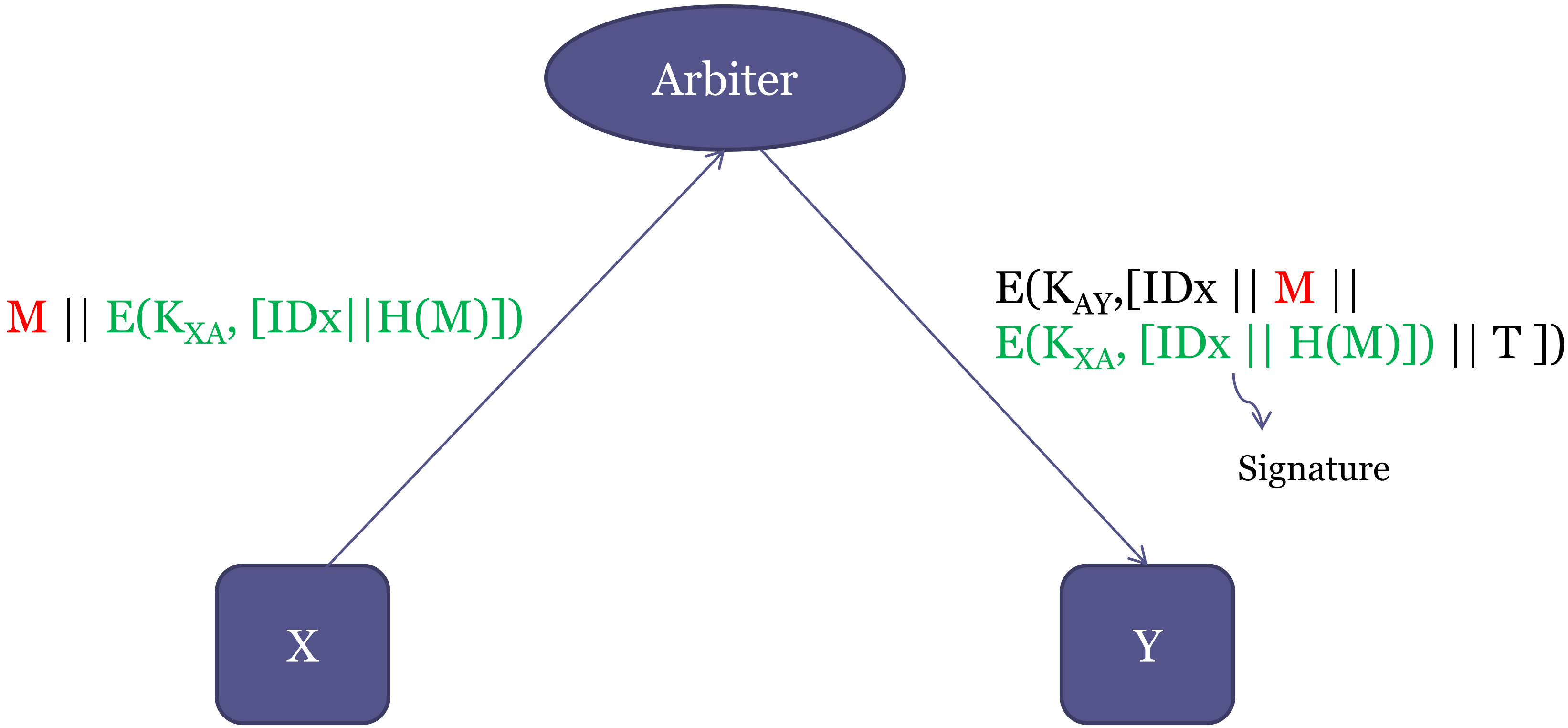
(2) $A \longrightarrow Y$: $E(K_{ay}, [ID_X \| E(K_{xy}, M)]) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M)) \| T])$

(b) Conventional Encryption, Arbiter Does Not See Message

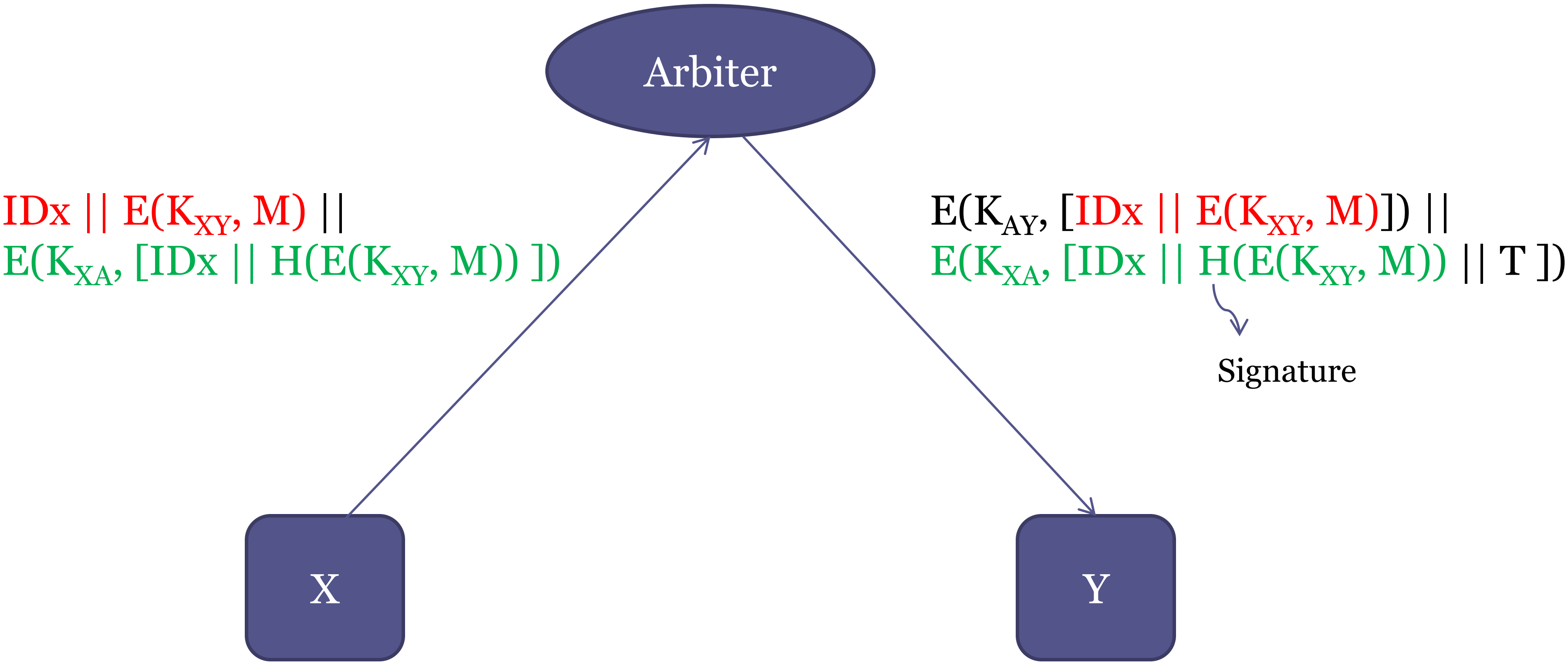
(1) $X \longrightarrow A$: $ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$

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

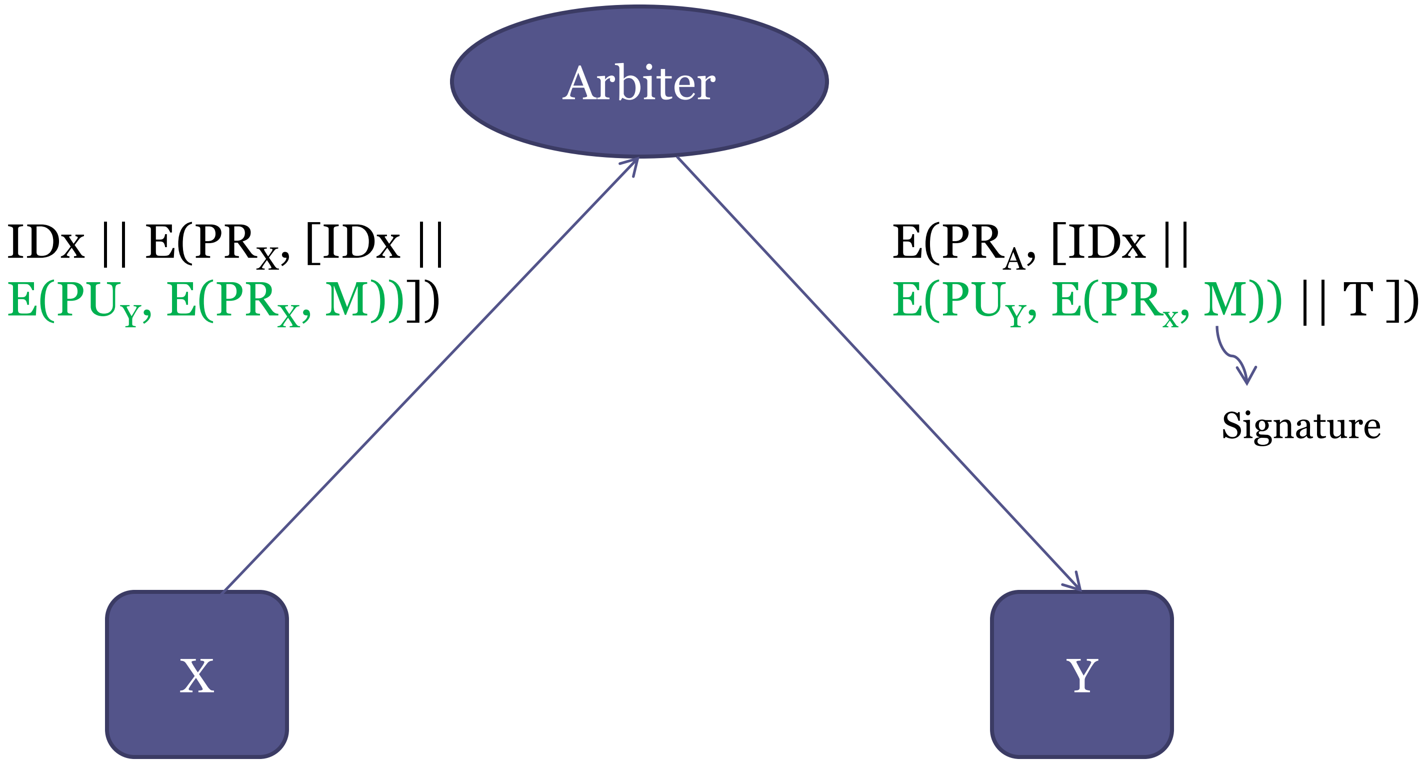
(c) Public-Key Encryption, Arbiter Does Not See Message



Arbitrated Digital Signature:
Conventional Encryption: Arbiter sees the Message



Arbitrated Digital Signature:
Conventional Encryption: Arbiter does not see the Message



Arbitrated Digital Signature:
Public Key Encryption: Arbiter does not see the Message

Arbitrated Digital Signature

- In the first scenario, symmetric encryption is used. It is assumed that the sender X and the arbiter A share a secret key K_{xa} and that A and Y share secret key K_{ay}.
- X constructs a message M and computes its hash value H(M). Then X transmits the message plus a signature to A. The signature consists of an identifier ID_X of X plus the hash value, all encrypted using K_{xa}.
- A decrypts the signature and checks the hash value to validate the message.
- Then A transmits a message to Y, encrypted with K_{ay}. The message includes ID_X, the original message from X, the signature, and a timestamp.
- Y can decrypt this to recover the message and the signature. The timestamp informs Y that this message is timely and not a replay. Y can store M and the signature.

Arbitrated Digital Signature

- Second scenario provides the arbitration as before but also assures confidentiality.
- In this case it is assumed that X and Y share the secret key K_{xy}.
- Now, X transmits an identifier, a copy of the message encrypted with K_{xy}, and a signature to A.
- The signature consists of the identifier plus the hash value of the encrypted message, all encrypted using K_{xa}.
- A decrypts the signature and checks the hash value to validate the message. In this case, A is working only with the encrypted version of the message and is prevented from reading it.
- A then transmits everything that it received from X, plus a timestamp, all encrypted with K_{ay}, to Y.

Arbitrated Digital Signature

- Third scenario uses a public-key scheme,
- In this case, X double encrypts a message M first with X's private key, PRx and then with Y's public key, PUy.
- This signed message, together with X's identifier, is encrypted again with PRx and, together with IDX, is sent to A.
- The inner, double-encrypted message is secure from the arbiter (and everyone else except Y). However, A can decrypt the outer encryption to assure that the message must have come from X (because only X has PRx).
- A checks to make sure that X's private/public key pair is still valid and, if so, verifies the message.
- Then A transmits a message to Y, encrypted with PRa. The message includes IDX, the double-encrypted message, and a timestamp.

Arbitrated Digital Signature

- This third scheme has a number of advantages over the preceding two schemes.
- First, no information is shared among the parties before communication, preventing alliances to defraud.
- Second, no incorrectly dated message can be sent, even if PRx is compromised, assuming that PRa is not compromised.
- Finally, the content of the message from X to Y is secret from A and anyone else.

Euclidean/Euclid's Algorithm

- An efficient way to find the Greatest Common Divisor GCD(a, b)
- Theorem:
  - For any non negative integers a and b,

$$GCD(a, b) = GCD(b, a \bmod b) \quad (a>b)$$

- GCD(a, 0) = | a |

Euclidean/Euclid's Algorithm

- Ex: GCD (1970, 1066)

gcd(1066, 904)	1970 = 1 x 1066 + 904
gcd(904, 162)	1066 = 1 x 904 + 162
gcd(162, 94)	904 = 5 x 162 + 94
gcd(94, 68)	162 = 1 x 94 + 68
gcd(68, 26)	94 = 1 x 68 + 26
gcd(26, 16)	68 = 2 x 26 + 16
gcd(16, 10)	26 = 1 x 16 + 10
gcd(10, 6)	16 = 1 x 10 + 6
gcd(6, 4)	10 = 1 x 6 + 4
gcd(4, 2)	6 = 1 x 4 + 2
gcd(2, 0)	4 = 2 x 2 + 0

= 2

Euclidean/Euclid's Algorithm

- Ex: GCD (55, 22)

= GCD(22, 55 mod 22) = GCD (22, 22*2 + 11)

≡ GCD (22, 11)

≡ GCD (11, 22 mod 11) = GCD (22, 11*2 + 0)

≡ GCD (11, 0)

= 11

Extended Euclidean/Euclid's Algorithm

- The extended Euclidean algorithm is an extension to the Euclidean algorithm. Besides finding the greatest common divisor of integers a and b, it also finds integers x and y that satisfy $ax + by = \gcd(a, b)$.

- The extended Euclidean algorithm is particularly useful when a and b are coprime, since x is the multiplicative inverse of $a \bmod b$, and y is the multiplicative inverse of $b \bmod a$.

Extended Euclidean/Euclid's Algorithm

EXTENDED EUCLID(*m*, *b*)

1. (A1, A2, A3)=(1, 0, *m*);
   (B1, B2, B3)=(0, 1, *b*)
2. if B3 = 0
   return A3 = gcd(*m*, *b*); no inverse
3. if B3 = 1
   return B3 = gcd(*m*, *b*); B2 = b^{-1} mod *m*
4. Q = A3 div B3
5. (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)
6. (A1, A2, A3)=(B1, B2, B3)
7. (B1, B2, B3)=(T1, T2, T3)
8. goto 2

Extended Euclidean/Euclid’s Algorithm

- Extended Euclid(26, 7)

Quotient of A3/B3

Q	A1	A2	A3	B1	B2	B3
—	1	0	26	0	1	7
3	0 B1	1 B2	7 B3	1 A1-QB1	—3 A2-QB2	5 A3-QB3
1	1	-3	5	-1	4	2
2	-1	4	2	3	-11	1

GCD(26,7)

x y Inverse of 7 mod 26

- $x = 3, y=(-11)$
- $ax+by$ = (26*3)+(7*-11) = 1 = gcd(26, 7) = B3
- Inverse of 7 mod 26: 7^{-1} mod 26 ≡ (-11) mod 26
- -11 is multiplicative inverse of 7 mod 26

Extended Euclidean/Euclid's Algorithm

- Extended Euclid(49, 37)

Q	A1	A2	A3	B1	B2	B3
—	1	0	49	0	1	37
1	0	1	37	1	-1	12
3	1	-1	12	-3	4	1

- x = -3, y=4
- $ax+by$ = (49*-3)+(37*4) = 1 = gcd(49, 37) = B3
- $37^{-1} \equiv 4 \bmod 49$ or $4 \equiv 37^{-1} \bmod 49$
- 4 is multiplicative inverse of 37 mod 49

Extended Euclidean/Euclid's Algorithm

- Extended Euclid(1759, 550)

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

- $x = -111, y=355$
- $ax+by$ = (1759*-111)+(550*355) = 1 = gcd(1759, 550) = B3
- $550^{-1} \equiv 355 \bmod 1759$ or $355 \equiv 550^{-1} \bmod 1759$
- 355 is multiplicative inverse of 550 mod 1759

NIST Digital Signature Algorithm

- National Institute of Standards and Technology (NIST)

- Digital Signature Approaches
  - RSA
  - Digital Signature Algorithm (DSA)

Digital Signature Approaches

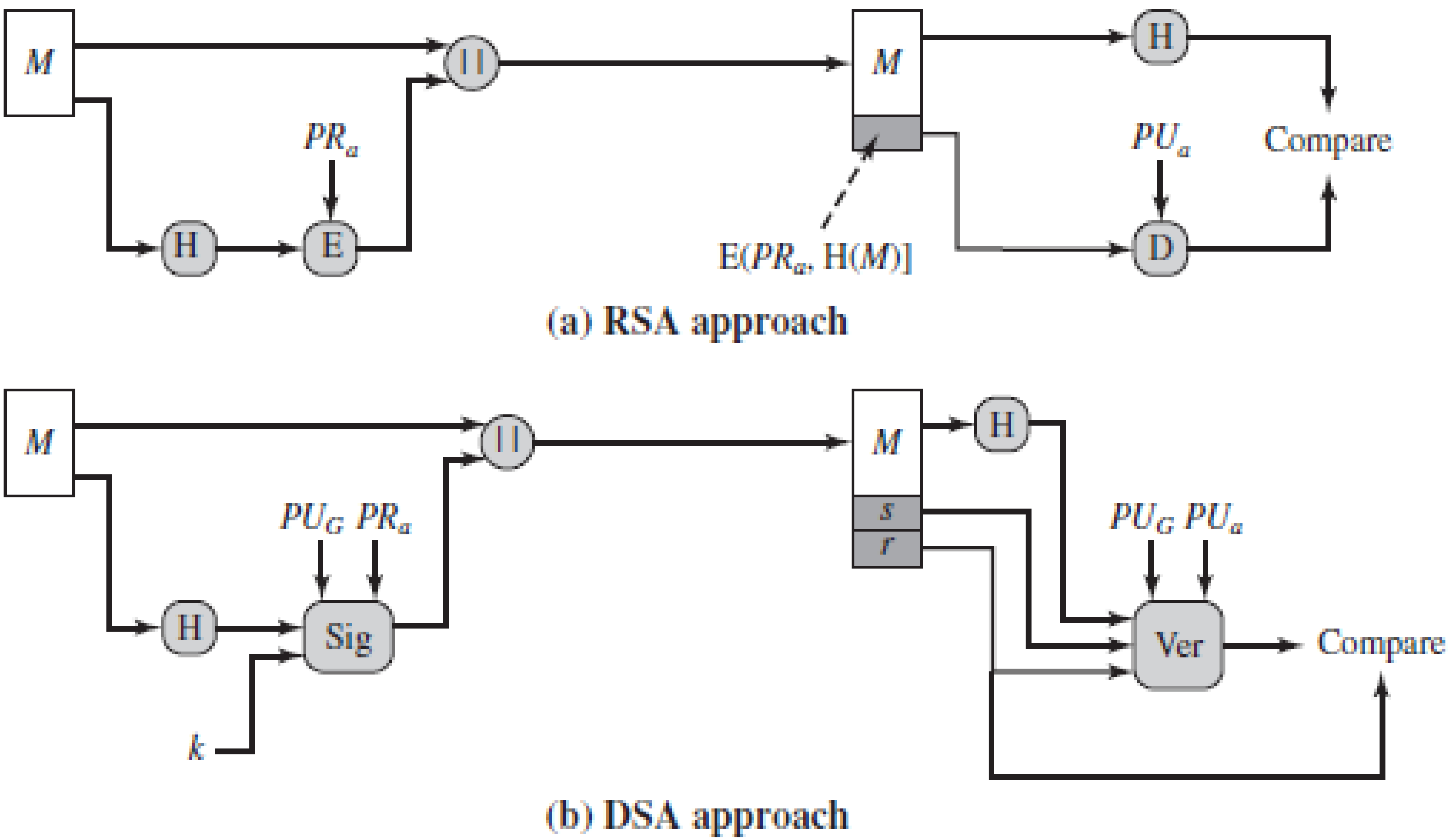


Figure 13.3 Two Approaches to Digital Signatures

Digital Signature Approaches

- In the RSA approach, 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.

Digital Signature Approaches

- In DSS (Digital Signature Standard) approach, 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 (*PRa)* and a global public key (PU_G).
- The result is a signature consisting of two components, labeled *s* and *r*.

Digital Signature Approaches

- At the receiving end, the hash code of the incoming message is generated.

- This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (*PUa*).

- 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.

NIST Digital Signature Algorithm

- The DSA is based on the difficulty of computing discrete logarithms

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

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

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

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: $v = r'$

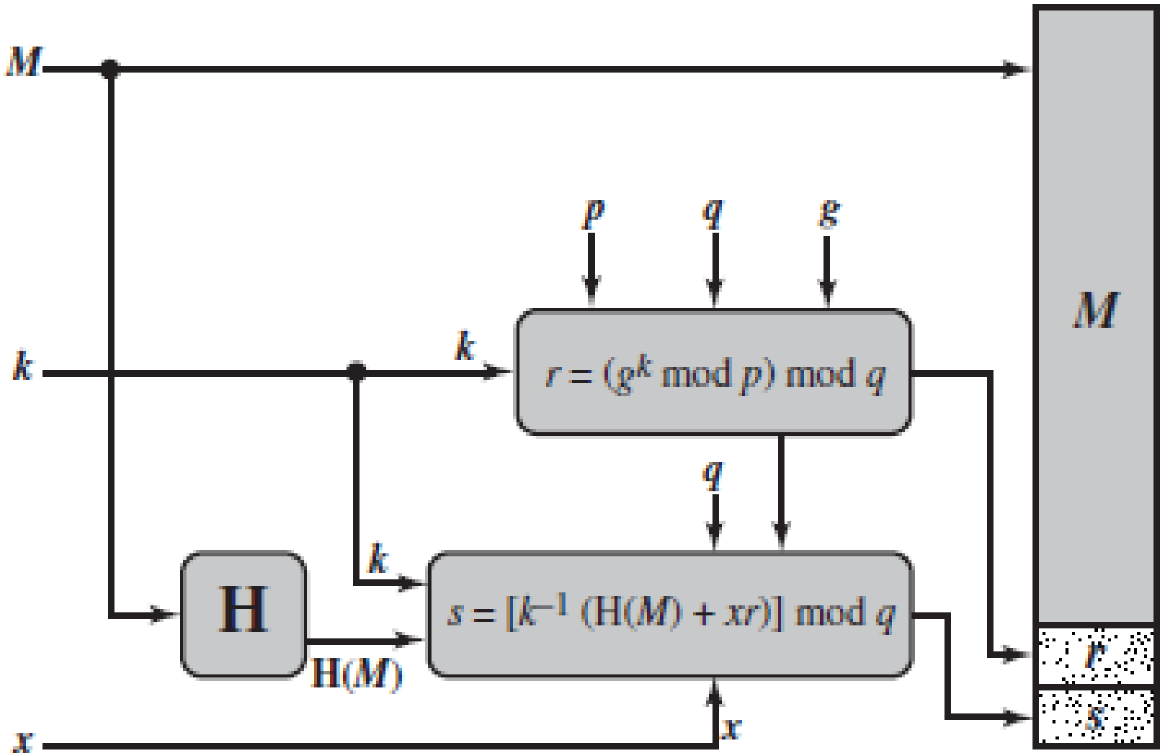
M = message to be signed

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

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

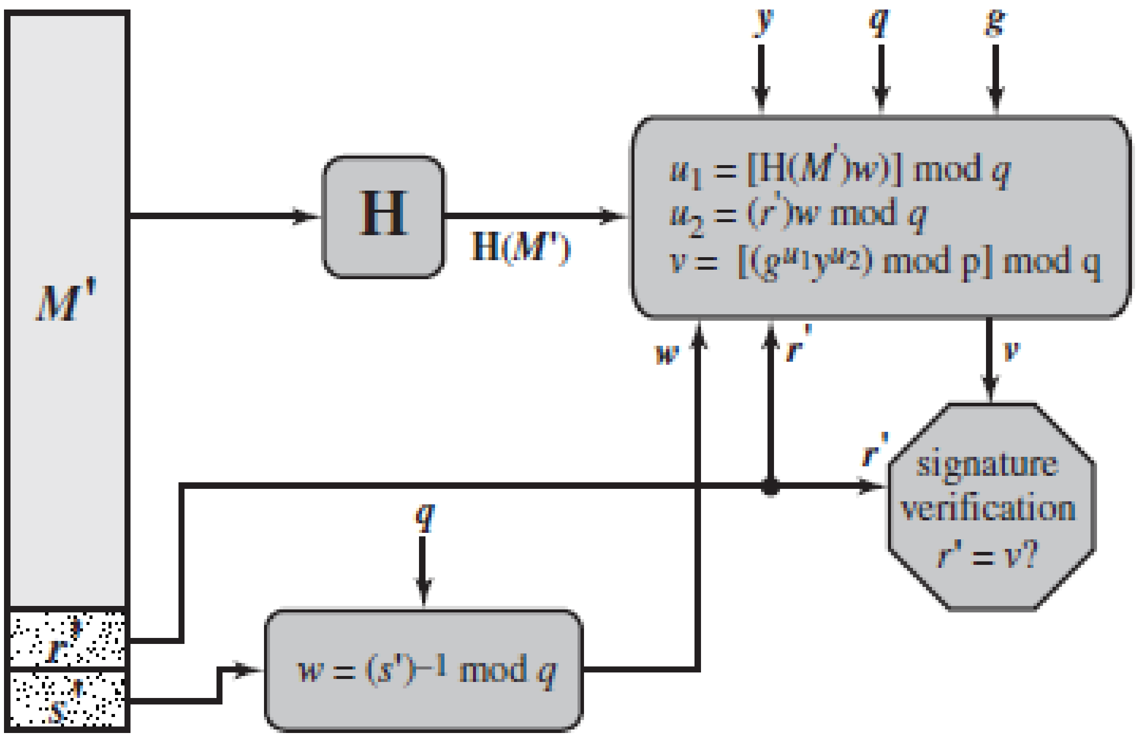
Figure 13.4 The Digital Signature Algorithm (DSA)

Digital Signature Algorithm



(a) Signing

Digital Signature Algorithm



(b) Verifying

Digital Signature Algorithm

- There are three parameters that are public and can be common to a group of users.
- A 160-bit prime number q is chosen.
- Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides $(p-1)$.
- Finally, g is chosen to be of the form $h^{(p-1)/q} \bmod p$ where h is an integer between 1 and $(p-1)$ with the restriction that g must be greater than 1.

Digital Signature Algorithm

- With these numbers in hand, each user selects a private key and generates a public key.
- The private key *x* must be a number from 1 to (*q*-1) and should be chosen randomly or pseudo randomly.
- The public key is calculated from the private key as $y = g^x \bmod p$.
- The calculation of *y* given *x* is relatively straightforward.
- However, given the public key *y*, it is believed to be computationally infeasible to determine *x*, which is the discrete logarithm of *y* to the base *g, mod p*.

Digital Signature Algorithm

- To create a signature, a user calculates two quantities, r and s, that are functions of
  - the public key components (*p, q, g*),
  - the user's private key (*x*),
  - the hash code of the message, *H(M)*, and
  - an additional integer *k* that should be generated randomly or pseudo randomly and be unique for each signing.

Digital Signature Algorithm

- At the receiving end, verification is performed using the formulas shown in Figure.

- The receiver generates a quantity *v* that is a function of
  - the public key components,
  - the sender's public key, and
  - the hash code of the incoming message

- If this quantity matches the *r* component of the signature, then the signature is validated.

Elgamal Digital Signature Scheme

- The Elgamal signature scheme involves the use of the private key for encryption and the public key for decryption

- For a prime number *q,* if α is a primitive root of *q,* then

$$\alpha, \alpha^2, \ldots, \alpha^{q-1}$$

  are distinct (mod *q).*

- It can be shown that, if α is a primitive root of q, then
  - 1. For any integer m, $\alpha^m \equiv 1 \pmod{q}$ if and only if $m \equiv 0 \pmod{q-1}$.
  - 2. For any integers, i, j, $\alpha^i \equiv \alpha^j \pmod{q}$ if and only if $i \equiv j \pmod{q-1}$.

Elgamal Digital Signature Scheme

- The global elements of **Elgamal digital signature** are a prime number q and α, 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 $Y_A = \alpha^{X_A} \bmod q$.
  - 3. A's private key is X_A; *A's pubic key is* $\{q, \alpha, Y_A\}$.

Elgamal Digital Signature Scheme

- To sign a message *M*, user *A* first computes the hash $m = H(M)$, such that *m* is an integer in the range $0 <= m <= q - 1$.

- *A* then forms a digital signature as follows.
  - 1. Choose a random integer *K* such that $1 <= K <= q - 1$ and $gcd(K, q - 1) = 1$. That is, *K* is relatively prime to $q - 1$.
  - 2. Compute $S_1 = \alpha^K \bmod q$.
  - 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).

Elgamal Digital Signature Scheme

- Any user B can verify the signature as follows.
  - 1. Compute $V_1 = \alpha^m \bmod q$.
  - 2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.

- The signature is valid if $V_1 = V_2$.

Public/ Private Keys Generation

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

Digital Signature

1. Choose a random integer K *such that* $1 <= K <= q - 1$ *and* $gcd(K, q - 1) = 1$.
2. Compute $S_1 = \alpha^K \bmod q$.
3. Compute $K^{-1} \bmod (q - 1)$.
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).

Verification

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

Elgamal Digital Signature Scheme

- Assume that the equality is true. Then we have

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

Elgamal Digital Signature Scheme

- For example, let us start with the prime field $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}. We choose $\alpha = 10$.

- 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. $S_1 = \alpha^K \bmod q = 10^5 \bmod 19 = 3$
  - 3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
  - 4. $S_2 = K^{-1} (m - X_A S_1) \bmod (q - 1) = 11 (14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Elgamal Digital Signature Scheme

- Bob can verify the signature as follows.
  - 1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16.$
  - 2. $V_2 = (Y_A)^{S1}(S_1)^{S2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16.$

- Thus, the signature is valid.

Schnorr Digital Signature Scheme

- The Schnorr signature scheme is based on discrete logarithms .
- The main work for signature generation does not depend on the message and can be done during the idle time of the processor.
- The scheme is based on using a prime modulus *p,* with *p - 1* having a prime factor *q* of appropriate size. Typically, we use $p \approx 2^{1024}$ and $q \approx 2^{160}$.
- Thus, *p* is a 1024-bit number, and *q* is a 160-bit number, which is also the length of the SHA-1 hash value.

Schnorr Digital Signature Scheme

- 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.*

Schnorr Digital Signature Scheme

- 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 pre processing 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).

Schnorr Digital Signature Scheme

- 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'
  - $\equiv a^y v^e$
  - $\equiv a^y a^{-se}$ $\quad$ (as $v = a^{-s} \bmod p$)
  - $\equiv a^{y-se}$
  - $\equiv a^r$ $\quad$ (as $y = (r + se) \bmod q$)
  - $\equiv x \pmod p$

- Hence, $H(M \,||\, x') = H(M \,||\, x)$.

Public/ Private Keys Generation

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.*

Digital Signature

1. Choose a random integer r *with* $0 < r < q$ *and* *compute* $x = a^r \bmod p$.

2. Compute the value e: $e = H(M \,||\, x)$

3. Compute $y = (r + se) \bmod q$.

The signature consists of the pair (e, y).

Verification

1. Compute $x' = a^y v^e \bmod p$.

2. Verify that $e = H(M \,||\, x')$

END