MODULE 4

Authentication requirements- Authentication functions- Message authentication codes- Hash functions- SHA -1, MD5, Security of Hash functions and MACs- Authentication protocols-Digital signatures-Digital signature standards.

Authentication Requirements

1. **Disclosure:** Release of message contents to any person or process not pos sess- ing the appropriate cryptographic key.
2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connectionscould be determined. In either a connectionoriented or connect ionless environment, the number and length of messages between parties c ould be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent sou rce. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.
4. **Content modification:** Changes to the contents of a message, including inse rtion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages betw een parties, including insertion, deletion, and reordering.
6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some
previous valid session, or individual messages in the sequence could be del ayed or replayed. In a connectionless application, an individual message (e.g., data- gram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.

8. **Destination repudiation:** Denial of receipt of message by destination.

Message authentication

a. A procedure to verify that messages come from the alleged source and have not been altered
b. Message authentication may also verify sequencing and timeliness

Digital signature

c. *An authentication technique* that also includes measures to counter repudiation by either source or destination.

Authentication Functions

Three classes of functions that may be used to produce an authenticator

- Message encryption
  - Cipher text itself serves as authenticator
- Message authentication code (MAC)
  - A public function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- Hash function
  - A public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

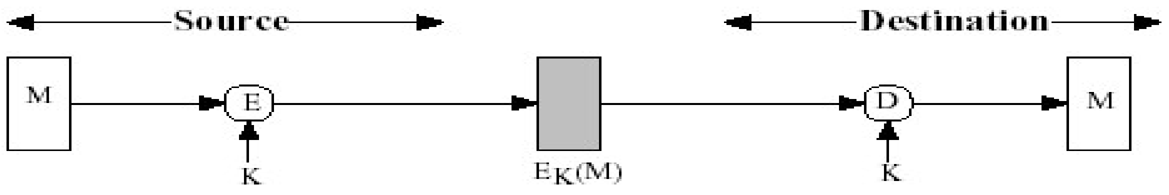
Message Encryption

- Conventional encryption can serve as authenticator
- Conventional encryption provides *authentication* as well as *confidentiality*

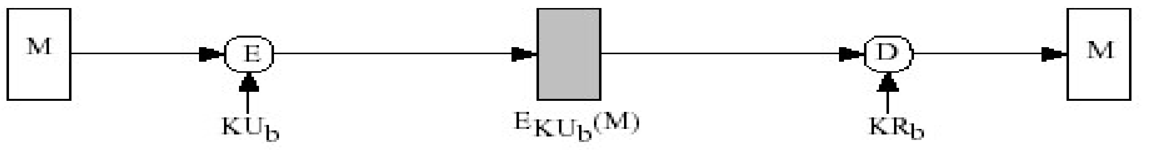
- Requires recognizable plaintext or other *structure* to distinguish between well-formed legitimate plaintext and meaningless random bits

  e.g., ASCII text, an appended checksum, or use of layered protocols

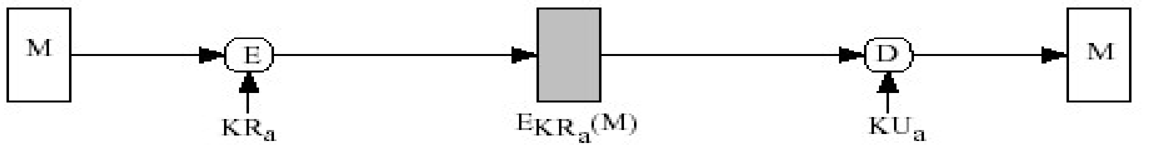
Basic Uses of Message Encryption



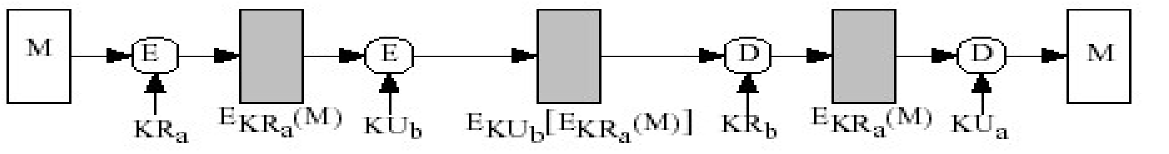
(a) Conventional encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality

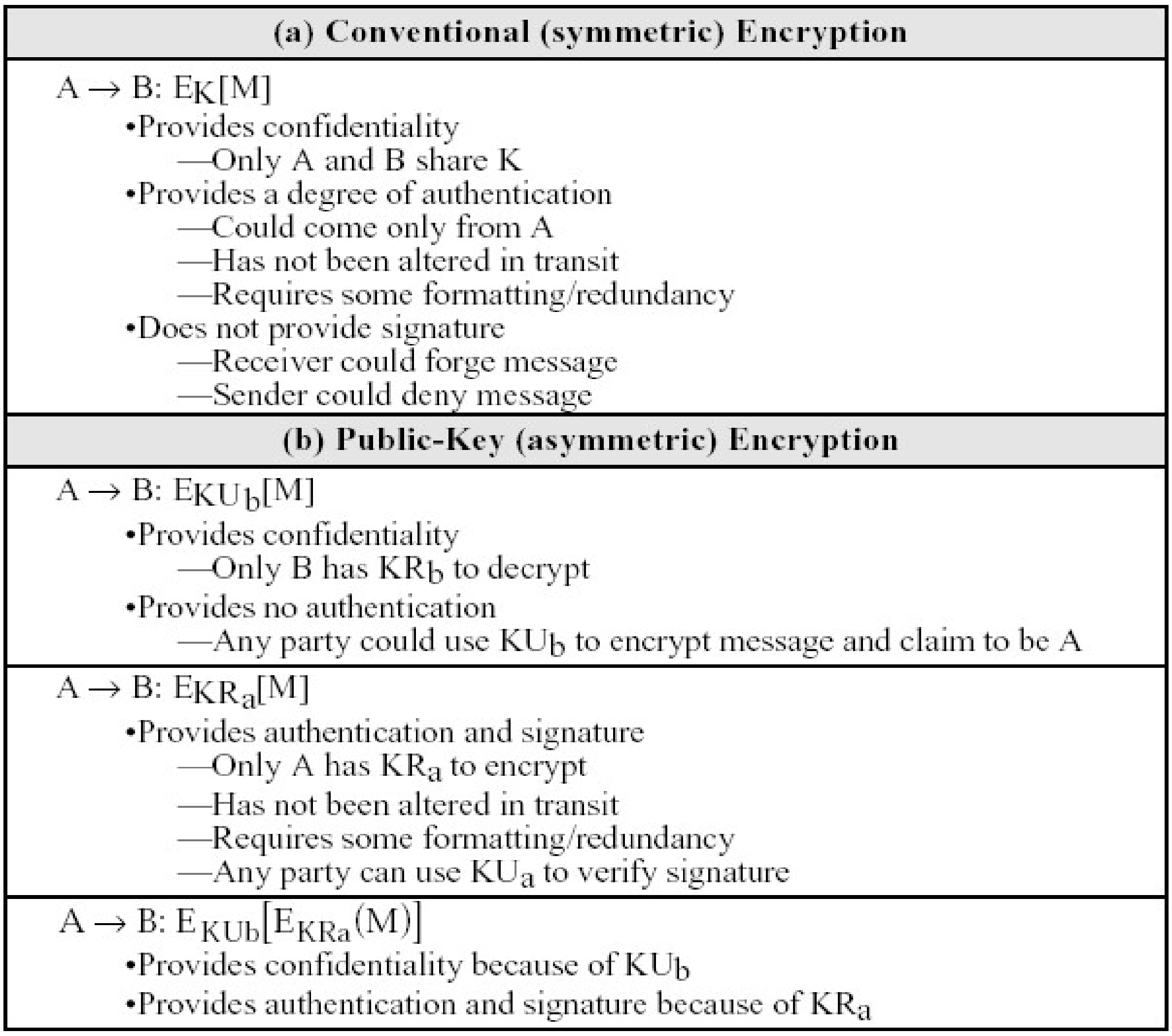


(c) Public-key encryption: authentication and signature



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

Confidentiality and Authentication Implications of Message Encryption



(a) Conventional (symmetric) Encryption

$A \rightarrow B: E_K[M]$

- Provides confidentiality
  - —Only A and B share K
- Provides a degree of authentication
  - —Could come only from A
  - —Has not been altered in transit
  - —Requires some formatting/redundancy
- Does not provide signature
  - —Receiver could forge message
  - —Sender could deny message

(b) Public-Key (asymmetric) Encryption

$A \rightarrow B: E_{KU_b}[M]$

- Provides confidentiality
  - —Only B has KR_b to decrypt
- Provides no authentication
  - —Any party could use KU_b to encrypt message and claim to be A

$A \rightarrow B: E_{KR_a}[M]$

- Provides authentication and signature
  - —Only A has KR_a to encrypt
  - —Has not been altered in transit
  - —Requires some formatting/redundancy
  - —Any party can use KU_a to verify signature

$A \rightarrow B: E_{KUb}[E_{KRa}(M)]$

- Provides confidentiality because of KU_b
- Provides authentication and signature because of KR_a

Message Authentication Code

- Uses a shared secret key to generate a fixed-size block of data (known as a cryptographic checksum or MAC) that is appended to the message

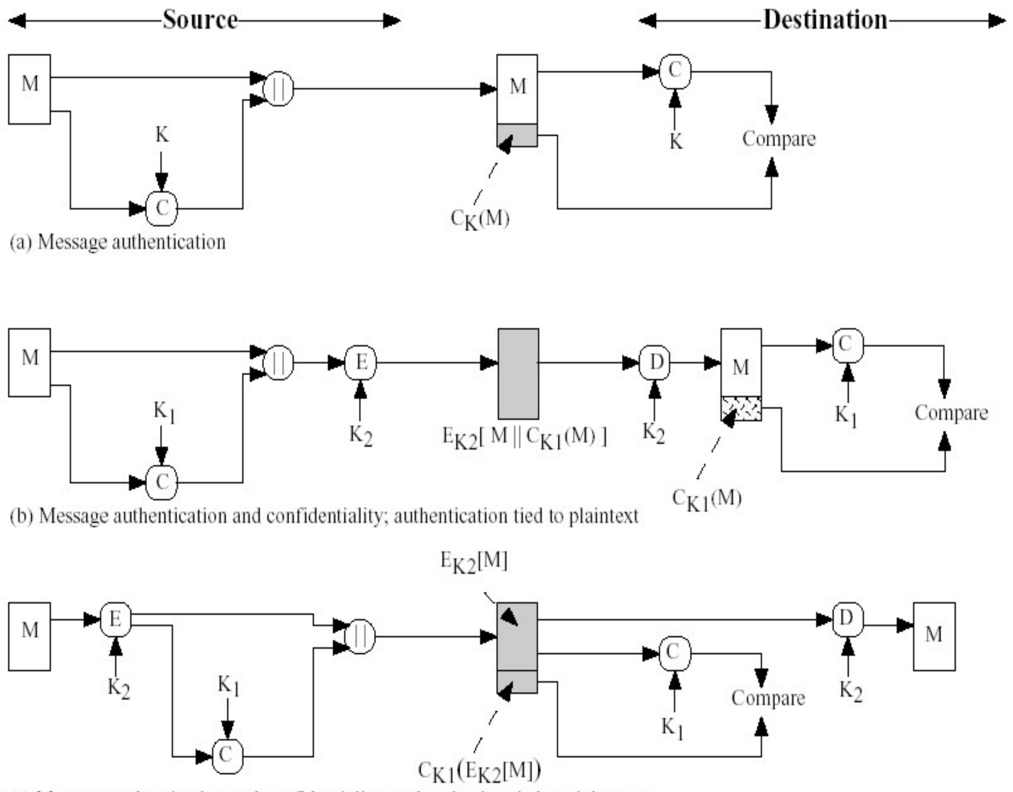
$$\textbf{MAC} = \mathbf{C_K(M)}$$

Assurances:

- Message has not been altered
- Message is from alleged sender

- Message sequence is unaltered (requires internal sequencing)

  Similar to encryption but MAC algorithm needs not be reversible.



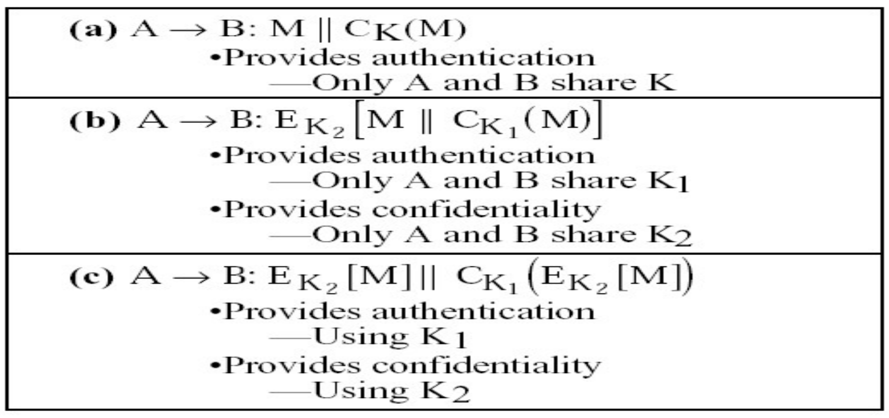
(a) Message authentication

(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

Basic Uses of MAC

Table 8.2 Basic Uses of Message Authentication Code C



(a) $A \rightarrow B$: $M \| C_K(M)$ •Provides authentication —Only A and B share K
(b) $A \rightarrow B$: $E_{K_2}[M \| C_{K_1}(M)]$ •Provides authentication —Only A and B share K_1 •Provides confidentiality —Only A and B share K_2
(c) $A \rightarrow B$: $E_{K_2}[M] \| C_{K_1}(E_{K_2}[M])$ •Provides authentication —Using K_1 •Provides confidentiality —Using K_2

Hash Function

- Converts a variable size message M into fixed size hash code H(M) (Sometimes called a *message digest*)
- Can be used with encryption for authentication
    - E(M || H)
    - M || E(H)
    - M || signed H
    - E(M || signed H) gives confidentiality
    - M || H(M || K)
    - E(M || H(M || K))

Basic Uses of Hash Function

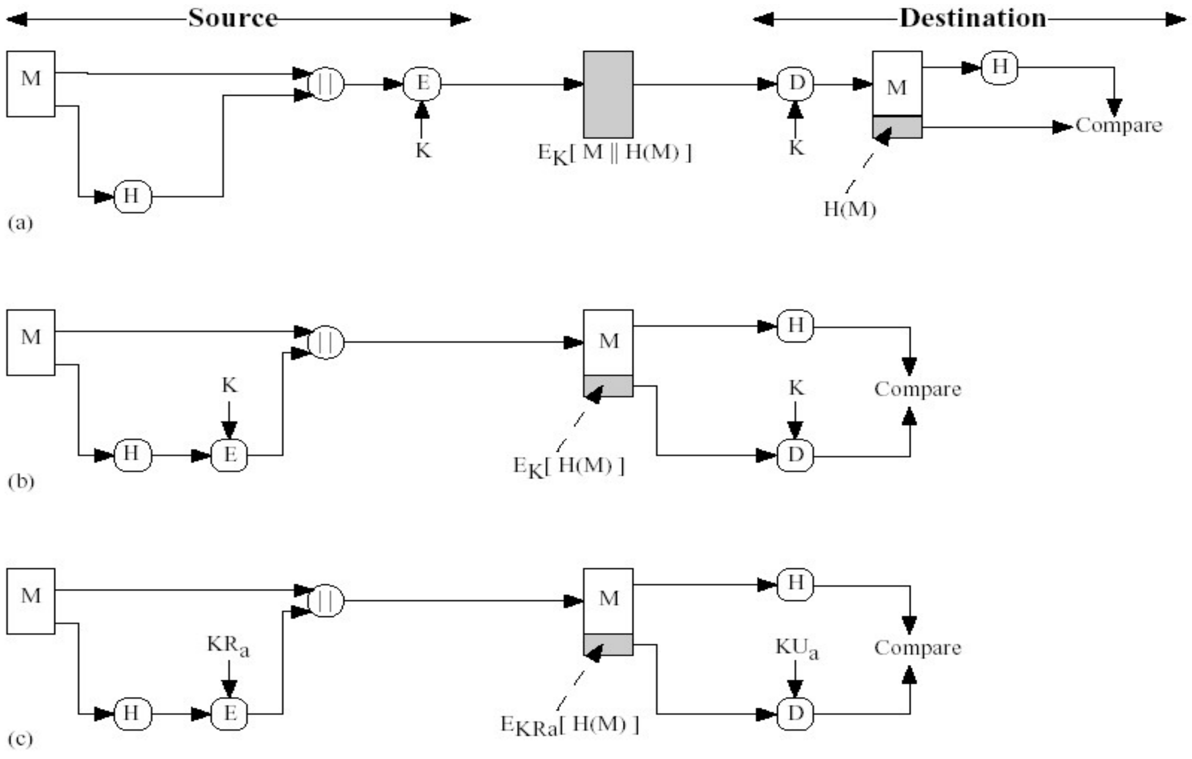


Figure 8.5 Basic Uses of Hash Function (page 1 of 2)

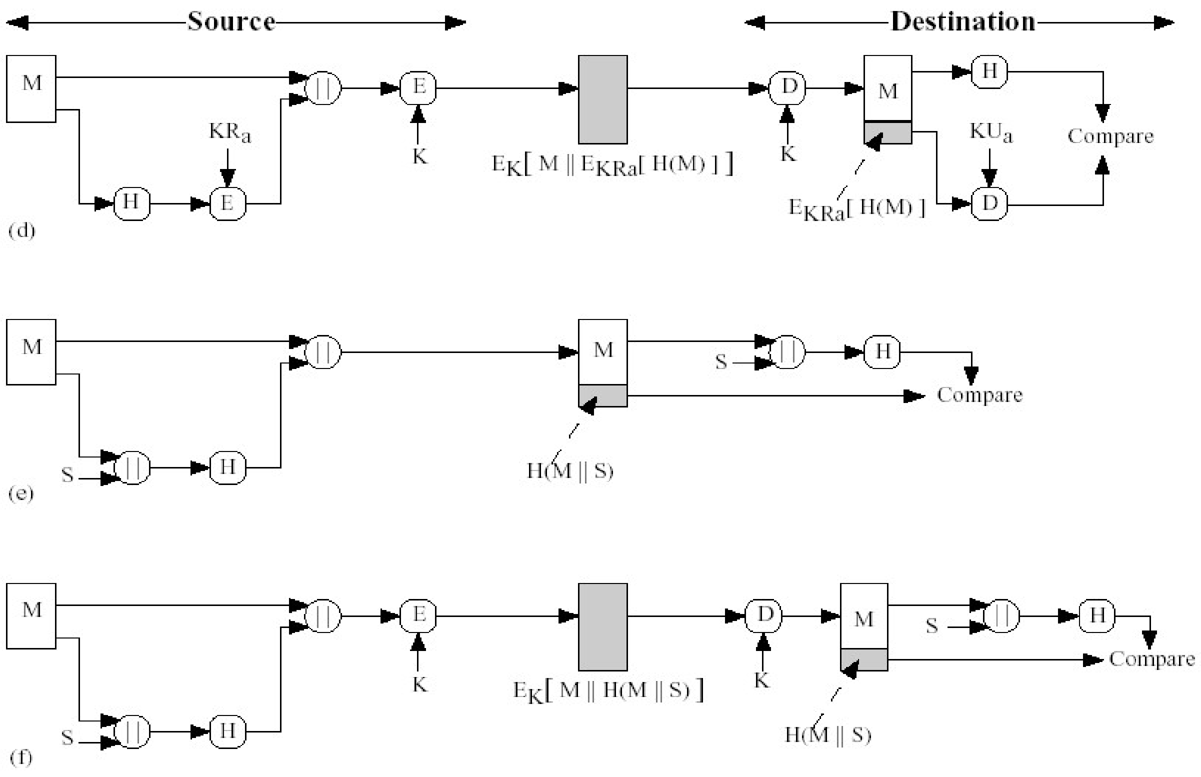


Figure 8.5 Basic Uses of Hash Function (page 2 of 2)

Basic Uses of Hash Function

Table 8.3 Basic Uses of Hash Function H

(a) $A \rightarrow B: E_K[M \parallel H(M)]$ •Provides confidentiality —Only A and B share K •Provides authentication —H(M) is cryptographically protected	**(d)** $A \rightarrow B: E_K[M \parallel E_{KRa}[H(M)]]$ •Provides authentication and digital signature •Provides confidentiality —Only A and B share K
(b) $A \rightarrow B: M \parallel E_K[H(M)]$ •Provides authentication —H(M) is cryptographically protected	**(e)** $A \rightarrow B: M \parallel H(M \parallel S)$ •Provides authentication —Only A and B share S
(c) $A \rightarrow B: M \parallel E_{KRa}[H(M)]$ •Provides authentication and digital signature —H(M) is cryptographically protected —Only A could create $E_{KR_a}[H(M)]$	**(f)** $A \rightarrow B: E_K[M \parallel H(M) \parallel S]$ •Provides authentication —Only A and B share S •Provides confidentiality —Only A and B share K

Secure Hash Algorithm-SHA-1

- ✓ based on design of MD5 with key differences
- ✓ produces 160-bit hash values
- ✓ *Padding:* Length of the message is 64 bits short of multiple of 512 after padding.
- ✓ *Append* a 64-bit *length* value of original message is taken.
- ✓ *Divide the input into 512-bit blocks*
- ✓ *Initialise CV*5-word (160-bit) buffer (A,B,C,D,E) to

  (*A*=01 23 45 67,

  B=89 AB CD EF,

  C=FE DC BA 98,

  D=76 54 32 10,

  E=C3 D2 E1 F0)

- ✓ ***ProcessBlocks***now the actual algorithm begins. message in 16-word (512-bit) chunks:
  - o Copy CV into single register for storing temporary intermediate as well as the final results.
  - o Divide the current 512-bit blocks into 16 sub-blocks, each consisting of 32 bits.
  - o Has No. Of Rounds=4,each round consisting of 20 ***bit /step iteration*** operations on message block & buffer
  - o expand 16 words into 80 words(20*4) by mixing & shifting.K[t] constant= ***Where t=0 to 79***
  - o Form new buffer value by adding output to input.
- ✓ output hash value is the final buffer value

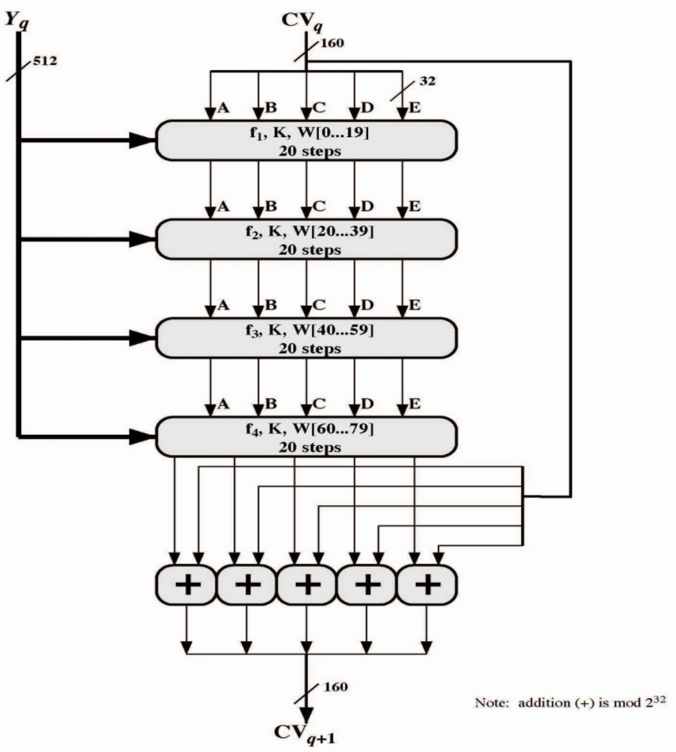
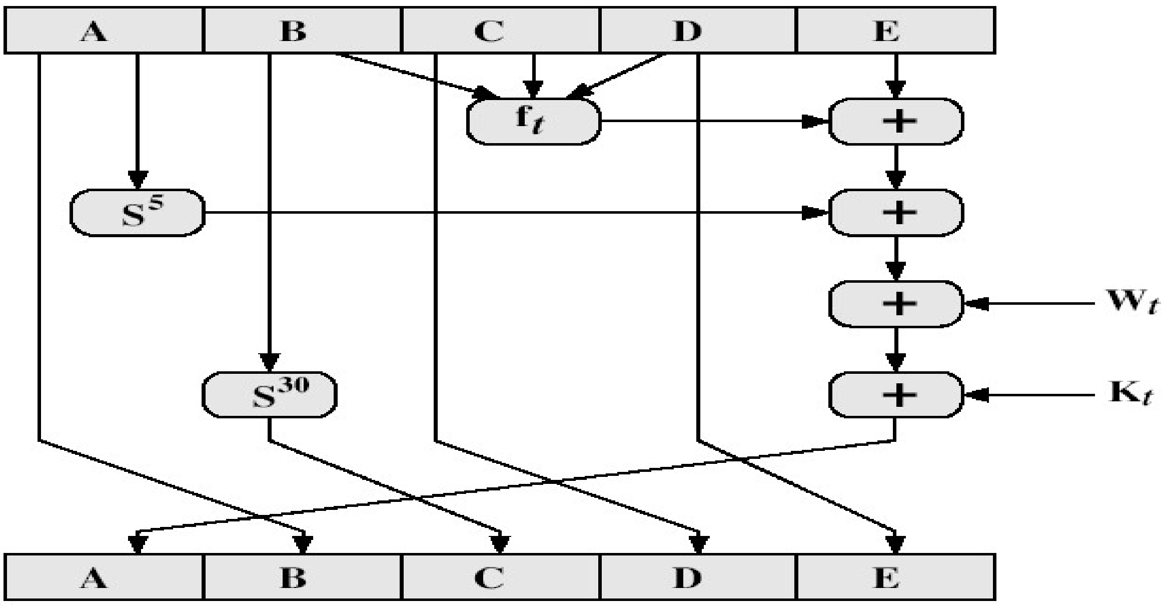


Figure 12.5 SHA-1 Processing of a Single 512-bit Block (SHA-1 Compression Function)

SHA-1 Compression Function

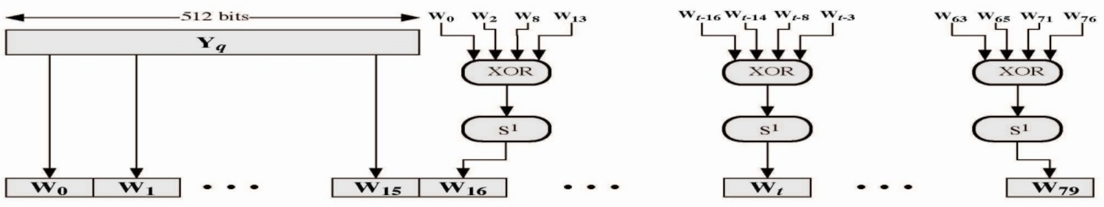


SHA-1 Compression Function terms

- ✓ each round has 20 steps which replaces the 5 buffer words thus:
  (A,B,C,D,E) <-(E+f(t,B,C,D)+(A<<5)+W_t+K_t),A,(B<<30),C,D)
- ✓ ABCDE refer to the 5 words of the buffer
- ✓ t is the step number
- ✓ f(t,B,C,D) is nonlinear function for round
- ✓ W_t is derived from the message block
- ✓ K_t is a constant value
- ✓ S^t circular left shift of 32 bit sub-block by t bits

Creation of 80-word input W_t

- ✓ Adds redundancy and interdependence among message blocks



512 bits
Y_q
W_0 W_2 W_8 W_{13}
W_{t-16} W_{t-14} W_{t-8} W_{t-3}
W_{63} W_{65} W_{71} W_{76}
XOR
S^1
W_0 W_1 W_{15} W_{16} W_t W_{79}

Figure 12.7 Creation of 80-word Input Sequence for SHA-1 Processing of Single Block

MD 5 hash algorithm

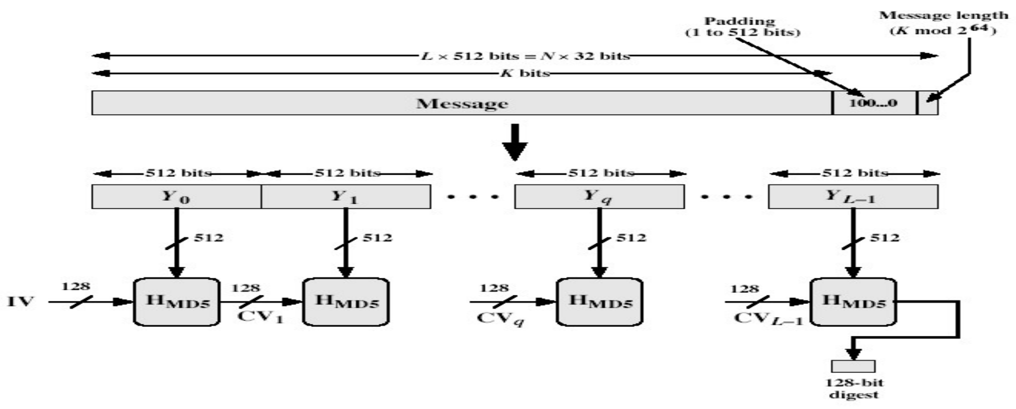
It was developed with the main motive of security as it takes an input of any size and produces an output if a 128-bit hash value. To be considered cryptographically secure MD5 should meet two requirements:

- It is impossible to generate two inputs that cannot produce the same hash function.
- It is impossible to generate a message having the same hash value.

Initially, MD5 was developed to store one way hash of a password and some file servers also provide pre-computed MD5 checksum of a file so that the user can compare the checksum of the downloaded file to it. Most Unix based Operating Systems include MD5 checksum utilities in their distribution packages.

- produces a 128-bit hash value

1. pad message so its length is 448 mod 512
2. append a 64-bit length value to message
3. initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
    a. using 4 rounds of 16 bit operations on message block & buffer
    b. add output to buffer input to form new buffer value
5. output hash value is the final buffer value

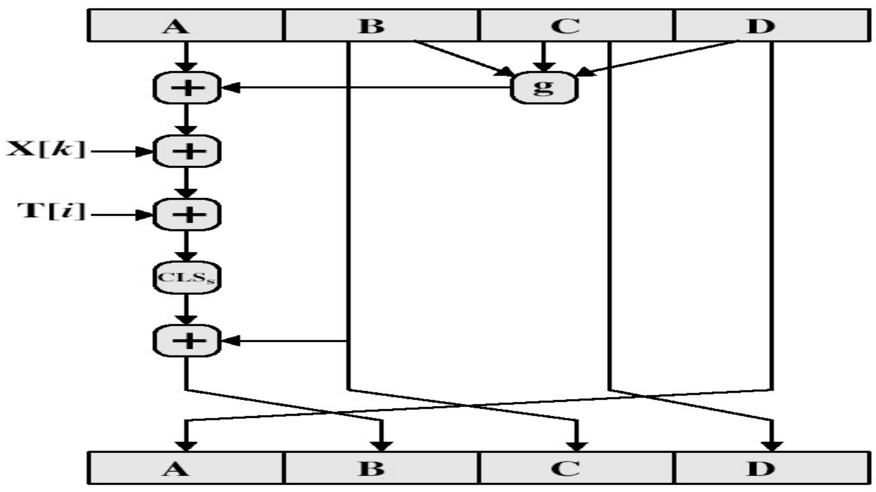


MD5 Compression Function

- each round has 16 steps of the form:

  a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)

- a,b,c,d refer to the 4 words of the buffer, but used in varying permutations
  - note this updates 1 word only of the buffer
  - after 16 steps each word is updated 4 times
- where g(b,c,d) is a different nonlinear function in each round (F,G,H,I)
- T[i] is a constant value derived from sin



SHA-1 verses MD5

- ✓ brute force attack is harder (160 vs 128 bits for MD5)
- ✓ not vulnerable to any known attacks (compared to MD4/5)
- ✓ a little slower than MD5 (80 vs 64 steps)
- ✓ both designed as simple and compact

- ✓ optimised for big endian CPU's (SUN) vs MD5 for little endian CPU's (PC)

Hash Functions & MAC Security

- like block ciphers have:
- brute-force attacks exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack key space (cf key search) or MAC
      - $\text{Min}(2^k, 2^n)$
    - at least 128-bit MAC and 128-bit key is needed for security
- cryptanalytic attacks exploit structure
  - like block ciphers want brute-force attacks to be the best alternative
- have a number of analytic attacks on iterated hash functions
  - $CV_i = f[CV_{i-1}, M_i]$; $H(M)=CV_N$
  - typically focus on collisions in function f
  - like block ciphers is often composed of rounds
  - attacks exploit properties of round functions

Authentication Protocol

1. Mutual Authentication

- A two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment. In general, this strategy involves the use of a trusted key distribution center (KDC). Each party in the network shares a secret key, known as a master key, with the KDC. The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for di stributing those keys using the master keys to protect the distribution. This approach is quite common.

  Needham and Schroeder Protocol

  Needham and Schroeder for secret key distribution using a KDC that, includes authentication features. The protocol can be summarized as follows.

  1. A → KDC: $ID_A \| ID_B \| N_1$
  2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
  3. A → B: $E(K_b, [K_s \| ID_A])$
  4. B → A: $E(K_s, N_2)$
  5. A → B: $E(K_s, f(N_2))$

Secret keys Ka and Kb are shared between A and the KDC and B and the KDC, respectively. The purpose of the protocol is to distribute securely a session key Ks to A and B.

A securely acquires a new session key in step 2.

The message in step 3 can be decrypted, and hence understood, only by B.

Step 4 reflects B's knowledge of Ks, and step 5 assures B of A's knowledge of Ks and assures B that this is a fresh Authentication Protocols message because of the use of the nonce N2.

The purpose of steps 4 and 5 is to prevent a certain type of replay attack.

In particular, if an opponent is able to capture the message in step 3 and replay it, this might in some fashion disrupt operations at B.

- Despite the handshake of steps 4 and 5, the protocol is still vulnerable to a form of replay attack.
- Xcan impersonate A and trick B into using the old key by simply replaying step 3. Unless B remembers indefinitely all previous session keys used with A, B will be unable to determine that this is a replay. If X can intercept the handshake message in step 4, then it can impersonate A's response in step 5. From this point on, X can send bogus messages to B that appear to B to come from A using an authenticated session key.

Denning Protocol

- Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 and 3.
- Her proposal assumes that the master keys, *Ka* and *Kb*, are secure, and it consists of the following steps.

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. $A \rightarrow B$: $E(K_b, [K_s \| ID_A \| T])$
4. $B \rightarrow A$: $E(K_s, N_1)$
5. $A \rightarrow B$: $E(K_s, f(N_1))$

- T is a timestamp that assures A and B that the session key has only just been
  generated. Thus, both A and B know that the key distribution is a fresh ex change. A and B can verify timeliness by checking that

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$

-

- Where $\Delta t1$ is the estimated normal discrepancy between the KDC’s clock an d the local
  clock (at A or B) and $\Delta t2$ is the expected network delay time. Each node c an set its clock against some standard reference source. Because the time stamp T is encrypted
  using the secure master keys, an opponent, even with knowledge of an old s ession key, cannot succeed because a replay of step 3 will be detected by B as untimely.
- A final point: Steps 4 and 5 were not included in the original presentation but were added later.
  These steps confirm the receipt of the session key at B.
- The Denning protocol seems to provide an increased degree of security compared to the Needham/Schroeder protocol.
- The Denning protocol requires reliance on clocks that are synchronized throughout the network
- A risk involved is based on the fact that the distributed clocks can become unsynchronized as a result of sabotage on or faults in the clocks or the synchronization mechanism
- The problem occurs when a sender’s clock is ahead of the intended recipient’s clock
  - An opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient’s site
  - Such attacks are referred to as ***suppress-replay attacks***

    One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock. The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonces.

This latter alternative is not vulnerable to a suppress-replay attack because the nonces the recipient will choose in the future are unpredictable to the sender. The Needham/Schroeder protocol relies on nonces only but, as we have seen, has other vulnerabilities.

1. AB:→IDA||Na

2. B K→DC: IDB||Nb||E(Kb, [IDA||Na||Tb])

3.KDCA:→E(Ka, [IDB||Na||Ks||Tb])||E(Kb,[IDA||Ks||Tb])||Nb

4. A B:→E(Kb, [IDA||Ks||Tb])||E(Ks, Nb)

One-Way Authentication

Using symmetric encryption, the decentralized key distribution scenario illus trated in Figure 14.5 is impractical. This scheme requires the sender to issue a request to the intended recipient, await a response that includes a session key, and only then send the message.

2. With some refinement, the KDC strategy illustrated in Figure 14.3 is a ca ndi- date for encrypted electronic mail. Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated. For a message with content *M*, the sequence is as follows:

   1. A → KDC: $ID_A \| ID_B \| N_1$
   2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
   3. A → B: $E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

3.

4. This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A. As specified, the protocol does not protect against replays.

Some measure of defense could be provided by including a timestamp with the message. However, because of the potential delays in the e-mail process, such timestamps may have limited usefulness.

Digital signatures

Digital signatures provide the ability to:

– verify author, date and time of signature

– authenticate message contents

– be verified by third parties to resolve disputes

• Hence include authentication function with additional capabilities

DSA Key Generation

- have shared global public key values (p,q,g):
  - choose 160-bit prime number q
  - choose a large prime p with $2^{L-1} < p < 2^{L}$
    - where L= 512 to 1024 bits and is a multiple of 64
    - such that q is a 160 bit prime divisor of (p-1)
  - choose $g = h^{(p-1)/q}$
    - where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- users choose private & compute public key:
  - choose random private key: $x<q$
  - compute public key: $y = g^{x} \bmod p$

DSA Signature Creation

- to sign a message M the sender:

- generates a random signature key k, $k<q$
- k must be random, be destroyed after use, and never be reused

➢ then computes signature pair: $r = (g^k \bmod p) \bmod q$ $s = [k^{-1}(H(M) + xr)] \bmod q$

➢ sends signature (r,s) with message M

DSA Signature Verification

➢ having received M &signature (r,s)

➢ to **verify** a signature, recipient computes:

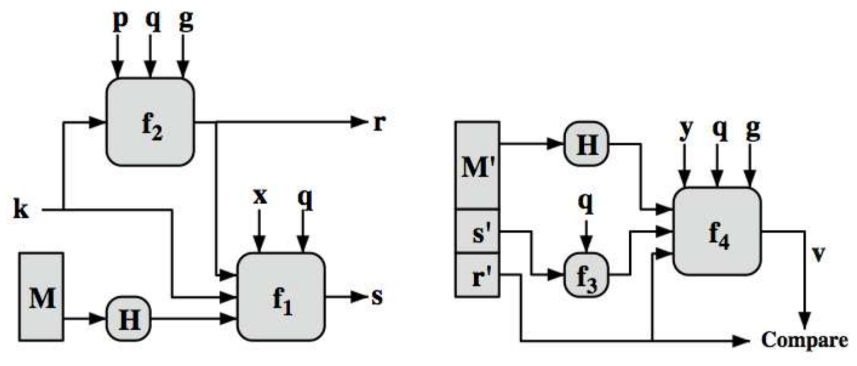
$w = s^{-1} \bmod q$

$u1= [H(M)w] \bmod q$

$u2= (rw) \bmod q$

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

➢ if v=r then signature is verified



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

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

(b) Verifying

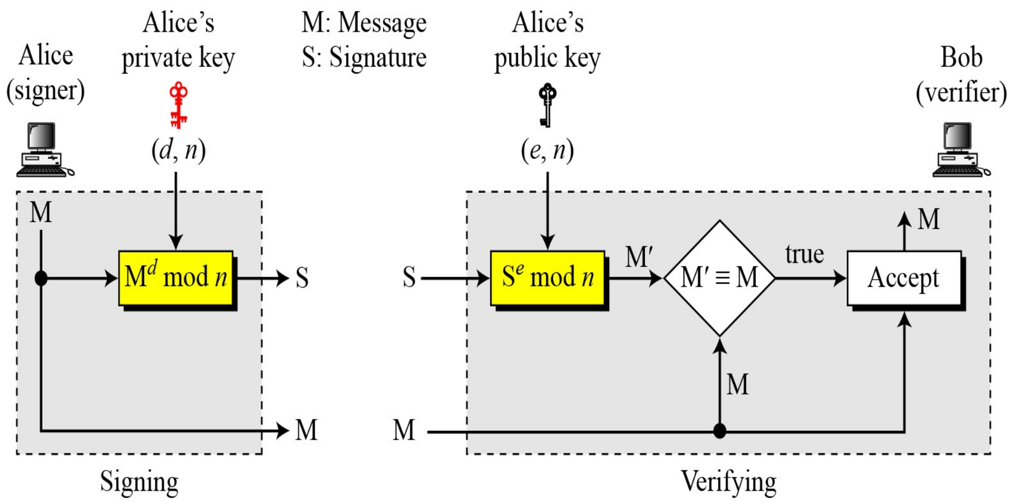
RSA Digital Signature

Key Generation

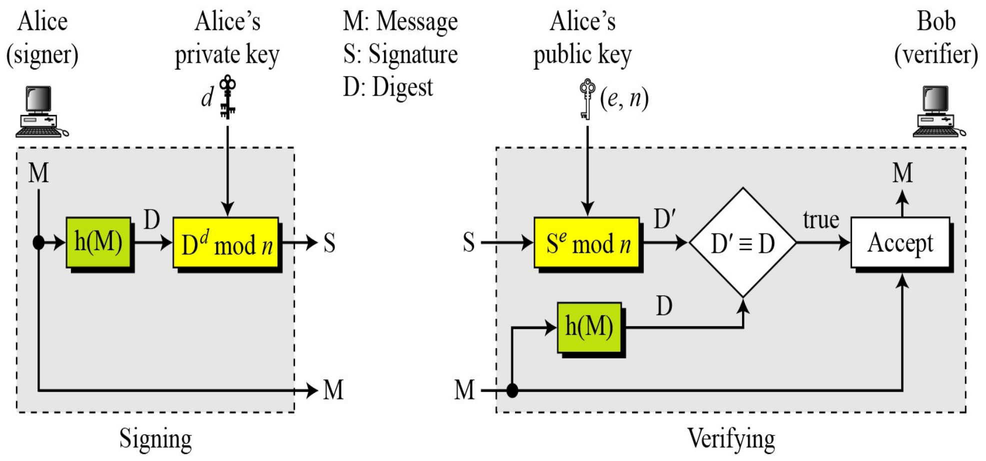
Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA

In the RSA digital signature scheme, *d* is private; *e* and *n* are public.

Signing and Verifying



RSA Signature on the Message Digest

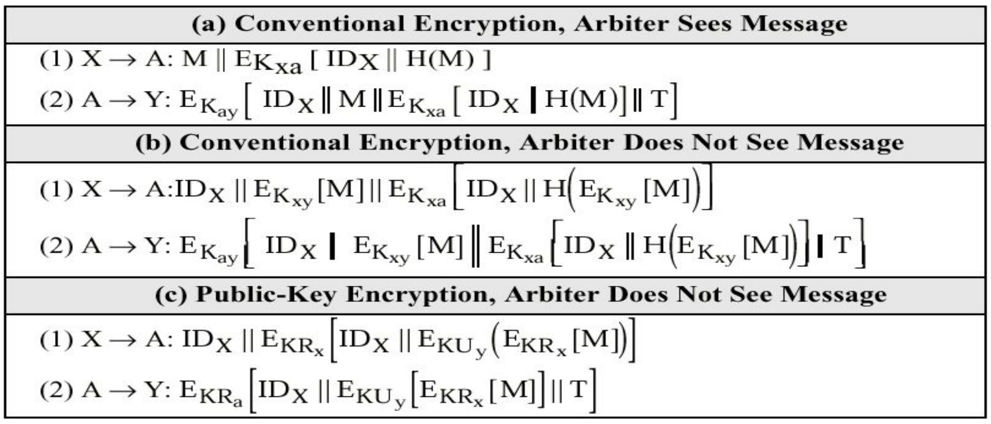


When the digest is signed instead of the message itself, the susceptibility of the RSA digital signature scheme depends on the strength of the hash algorithm**.**

Arbitrated Digital Signature Techniques

- 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

Table 10.1 Arbitrated Digital Signature Techniques



(a) Conventional Encryption, Arbiter Sees Message
(1) $X \rightarrow A: M \| E_{K_{xa}}[ID_X \| H(M)]$
(2) $A \rightarrow Y: E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)] \| T]$
(b) Conventional Encryption, Arbiter Does Not See Message
(1) $X \rightarrow A: ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])]$
(2) $A \rightarrow Y: E_{K_{ay}}[ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])] \| T]$
(c) Public-Key Encryption, Arbiter Does Not See Message
(1) $X \rightarrow A: ID_X \| E_{KR_x}[ID_X \| E_{KU_y}(E_{KR_x}[M])]$
(2) $A \rightarrow Y: E_{KR_a}[ID_X \| E_{KU_y}[E_{KR_x}[M]] \| T]$

Notation:

X	=	sender
Y	=	recipient
A	=	Arbiter
M	=	Message

The arbiter uses Kay to recover IDX, M, and the signature, and then uses Kxa to decrypt the signature and verify the hash code. In this scheme, Y cannot directly check X's signature; the signature is there solely to settle disputes. Y considers the message from X authentic because it comes through A. In this scenario, both sides must have a high degree of trust in A:

● X must trust A not to reveal Kxa and not to generate false signatures of the form E(Kxa, [IDX||H(M)]).

● Y must trust A to send E(Kay, [IDX||M||E(Kxa, [IDX||H(M)])||T]) only if the hash value is correct and the signature was generated by X.

● Both sides must trust A to resolve disputes fairly.

If the arbiter does live up to this trust, then X is assured that no one can forge his signature and Y is assured that X cannot disavow his signature.

The preceding scenario also implies that A is able to read messages from X to Y and, indeed, that any eavesdropper is able to do so. Table 10.1b shows a scenario that provides the arbitration as before but also assures confidentiality. In this case it is assumed that X and Y share the secret key Kxy. Now, X transmits an identifier, a copy of the message encrypted with Kxy, and a signature to A. The signature consists of the identifier plus the hash value of the encrypted message, all encrypted using Kxa. As before, 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 Kay, to Y.

Although unable to read the message, the arbiter is still in a position to prevent fraud on the part of either X or Y. A remaining problem, one shared with the first scenario, is that the arbiter could form an alliance with the sender to deny a signed message, or with the receiver to forge the sender's signature.

All the problems just discussed can be resolved by going to a public-key scheme, one version of which is shown in Table 10.1c. 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 is a signed, secret version of the message. 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. This 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. Digital Signatures the content of the message from X to Y is secret from A and anyone else. However, this final scheme involves encryption of the message twice with a public-key algorithm.