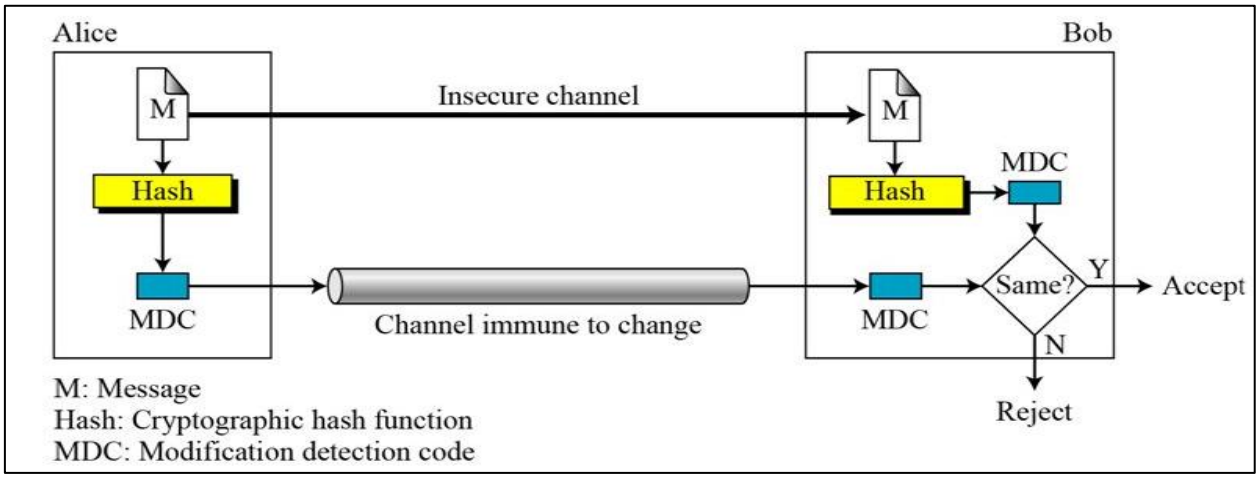
1. What is Message Authentication code? Explain its functions and basic uses.

Message Authentication:

- A message digest does not authenticate the sender of the message. To provide message authentication, Sender needs to provide proof that it is Sender sending the message and not an intruder. The digest created by a cryptographic hash function is normally called a modification detection code (MDC). What we need for message authentication is a message authentication code (MAC).

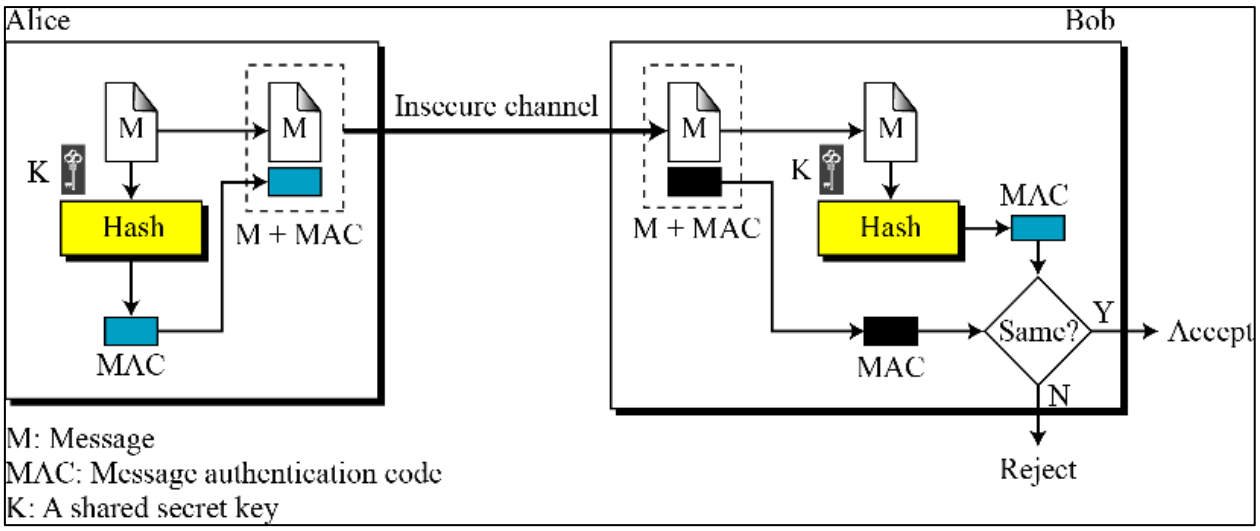
Modification Detection Code:

- A modification detection code (MDC) is a message digest that can prove the integrity of the message: that message has not been changed.
- If Alice needs to send a message to Bob and be sure that the message will not change during transmission, Alice can create a message digest MDC and send both the message and the MDC to Bob.
- Bob can create a new MDC from the message and compare the received MDC and the new MDC.
- If they are same, the message has not been changed. Figure 11.6 shows the idea.



Message Authentication Code (MAC):

- MAC algorithm is a symmetric key cryptographic technique to provide message authentication.
- For establishing MAC process, the sender and receiver share a symmetric key K. Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.
- The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail –

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Basic uses of MAC include:

- **Network Security:**
  MAC is widely used in network protocols like TLS (Transport Layer Security) and IPsec (Internet Protocol Security) to ensure secure communication over the internet.
- **Message Authentication:**
  MAC can be used to authenticate messages exchanged between two parties, ensuring that they are both genuine and unaltered.
- **Data Integrity Checking:**
  MAC is used to verify the integrity of data stored or transmitted across potentially insecure channels, such as in file transfers or database transactions.

2. Distinguish between Message Integrity and Message Authentication.

Aspect	Message Integrity	Message Authentication
Definition	Ensures that the message has not been altered or corrupted during transmission.	Verifies the authenticity of the message, ensuring it originated from a trusted source.
Objective	Detects unauthorized modifications or tampering with the message contents.	Verifies the identity of the sender and ensures the message's origin is genuine.
Mechanism	Typically achieved using cryptographic hash functions (e.g., SHA-256) or checksums.	Relies on cryptographic techniques such as MAC (Message Authentication Code) or digital signatures.

Key Component	Focuses on data integrity, ensuring that the message remains intact and unaltered.	Focuses on authentication, confirming the identity of the sender and the message's authenticity.
Example Application	Ensuring that a downloaded file matches the original, unaltered version provided by the server.	Verifying that an email was indeed sent by the claimed sender and has not been tampered with in transit.
Importance	Critical for maintaining the reliability and trustworthiness of data in transit or storage.	Essential for confirming the trustworthiness of the source and the integrity of the message content.
Verification Method	Comparing the computed hash or checksum of the received message with the expected value .	Validating the message's authenticity by verifying cryptographic signatures or MACs.
Goal	Detect and prevent unauthorized alterations to the message.	Confirm the identity of the sender and ensure the message's trustworthiness.

3. Explain about HMAC algorithm with a neat diagram.

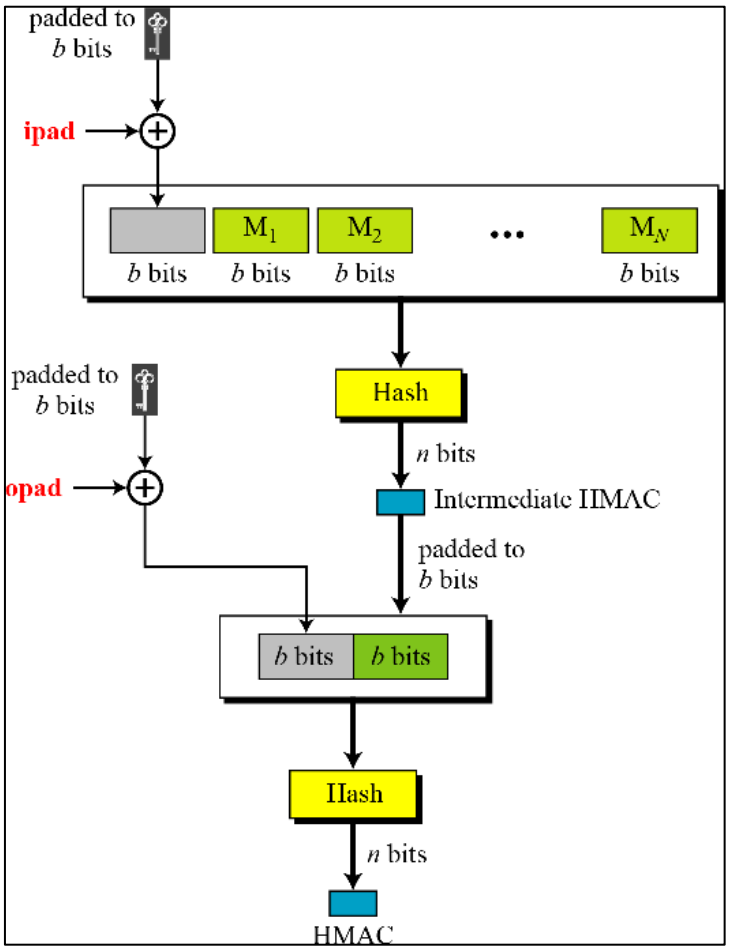
Hash-based Message Authentication Code:

- HMAC is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key.
- Like any of the MAC, it is used for both data integrity and authentication.
- Checking data integrity is necessary for the parties involved in communication.
- HTTPS, SFTP, FTPS, and other transfer protocols use HMAC.
- The cryptographic hash function may be MD-5, SHA-1, or SHA-256.
- Digital signatures are nearly similar to HMACs i.e. they both employ a hash function and a shared key.
- The difference lies in the keys i.e. HMACs use symmetric key (same copy) while Signatures use asymmetric (two different keys).

HMAC Procedure:

1. The message is divided into N blocks, each of b-bits.
2. The Secrete Key is left-padded with 0's to create b-bits.
3. The result of step 2 is ex-ored with a constant called iPad (input pad) to create a b-bit block. The value of iPad is the b/8 repetition of the sequence 00110110(36 in Hexa-Decimal)
4. The resulting block is prepended to the N-block message. The result is N+1 blocks.
5. The result of step 4 is hashed to create an n-bit digest. We call the digest the intermediate HMAC.
6. The intermediate n-bit HMAC is left padded with 0's to make a b-bit block.

7. Step 2 and 3 are repeated by a different constant opad (output pad). The value of opad is the b/8 repetition of the sequence 01011100(5C in Hexa-Decimal)
8. The result of step 7 is prepended to the block of step 6.
9. The result of step 8 is hashed with the same hashing algorithm to create the final n-bit HMAC.



Advantages

- HMACs are ideal for high-performance systems like routers due to the use of hash functions which are calculated and verified quickly unlike the public key systems.
- Digital signatures are larger than HMACs, yet the HMACs provide comparably higher security.
- HMACs are used in administrations where public key systems are prohibited.

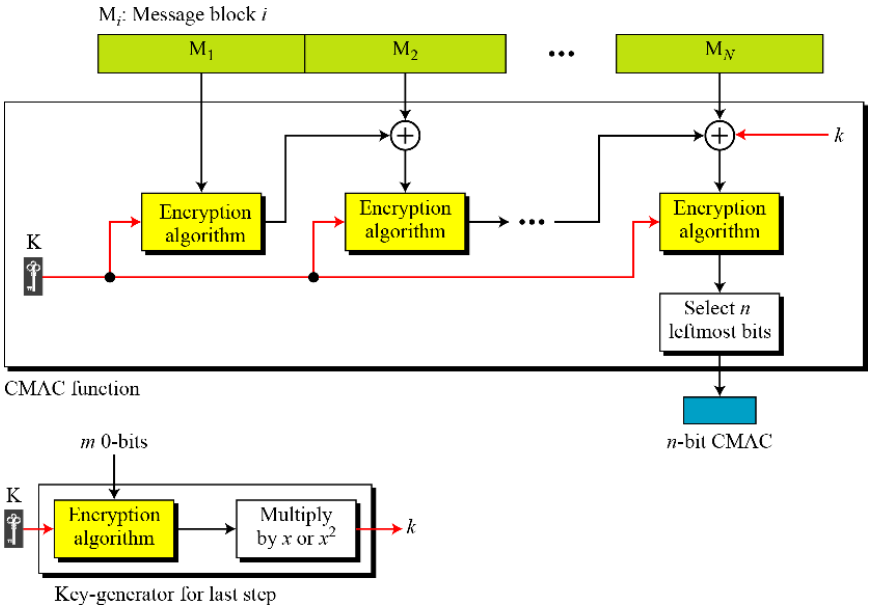
Disadvantages

- HMACs uses shared key which may lead to non-repudiation. If either sender or receiver's key is compromised then it will be easy for attackers to create unauthorized messages.

4. Explain about CMAC algorithm.

It is a block cipher-based message authentication code algorithm. It may be used to provide assurance of the authenticity and, hence, the integrity of binary data.

- CMAC codes are a tool for calculating message authentication codes using a block cipher coupled with a secret key. You can use an CMAC to verify both the integrity and authenticity of a message.
- The method is similar to CBC mode.
- However, the idea is to create one block of MAC from N blocks of plaintext using a symmetric-key cipher N times. Figure 11.10 shows the idea.



- The message is divided into N blocks, each m bits long.
- The size of CMAC is n bits.
- If the last block is not m bits, it is padded with a 1-bit followed by enough 0-bits to make it m bits.
- The first block of the message is encrypted with the symmetric key to create an m-bit block of encrypted data. =
- This block is XORed with the next block and the result is encrypted again to create a new mbit block. =
- The process continues until the last block of the message is encrypted. =
- The n leftmost bit from the last block is the CMAC.
- In addition to the symmetric key, K, CMAC also uses another key, k, which applied only at the last step.
- The result from the Encryption algorithm is multiplied by x if no padding is applied and is multiplied by x^2 if padding is applied.
- The multiplications are in GF(2^m) with irreducible polynomial of degree m selected by the particular protocol used.

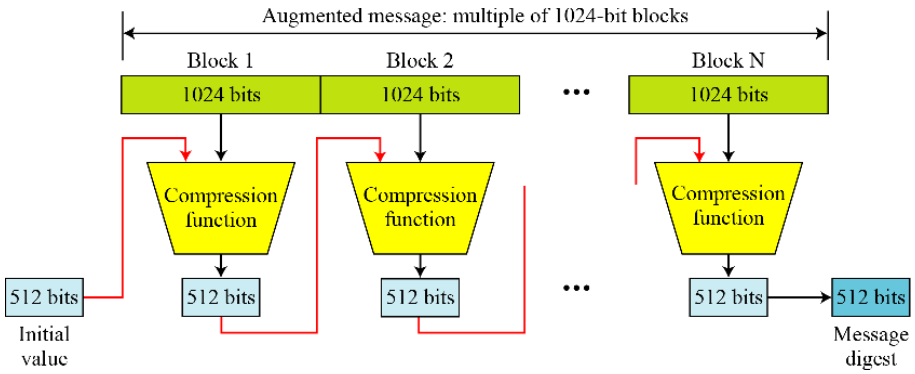
5. Discuss Secure Hash Algorithm in detail

- SHA-512 is the version of SHA with a 512-bit message digest. It is based on the Merkle Damgard scheme. SHA-512 is the latest version of SHA family with a more complex structure than others and the longest message digest.
- Input: a message with a maximum length of 2128 bit
- Output: a 512-bit message digest.

- The input is processed in 1024-bit blocks.

Characteristics	SHA-512
Maximum Message size	2128 - 1
Block size	1024
Message digest size	512
Number of rounds	80
Word size	64

Message Digest Creation in SHA-512:



Message Preparation:

- SHA-512 insists that the length of the original message be less than 2128 bits.
- Note: SHA-512 creates a 512-bit message digest out of a message less than 2128

Length Field and Padding:

- Before the message digest can be created, SHA-512 requires the addition of a 128-bit unsigned-integer length field to the message that defines the length of the message in bits.
- The field can define a number between 0 and 2^{128} -1.

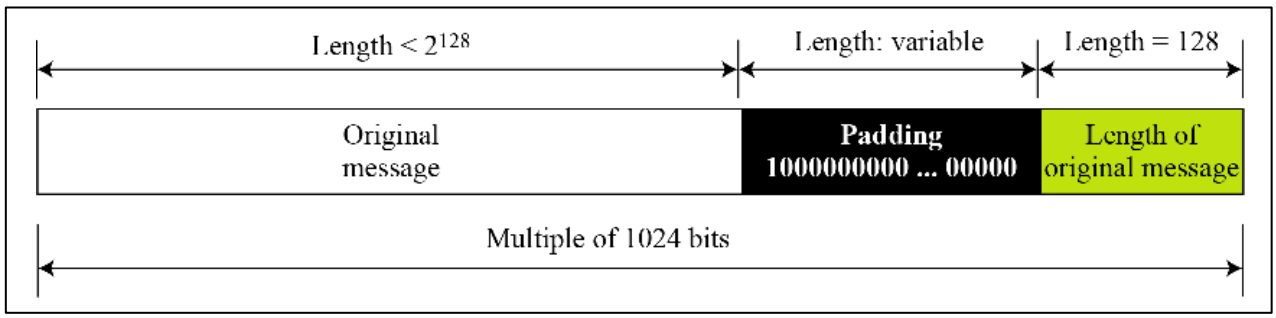


Figure: Padding and length field in SHA-512

Words:

- SHA-512 operates on words. Each word is defined as 64 bit. Each block of the message
- consists of sixteen 64-bit words.
- The message digest consists of only eight words.

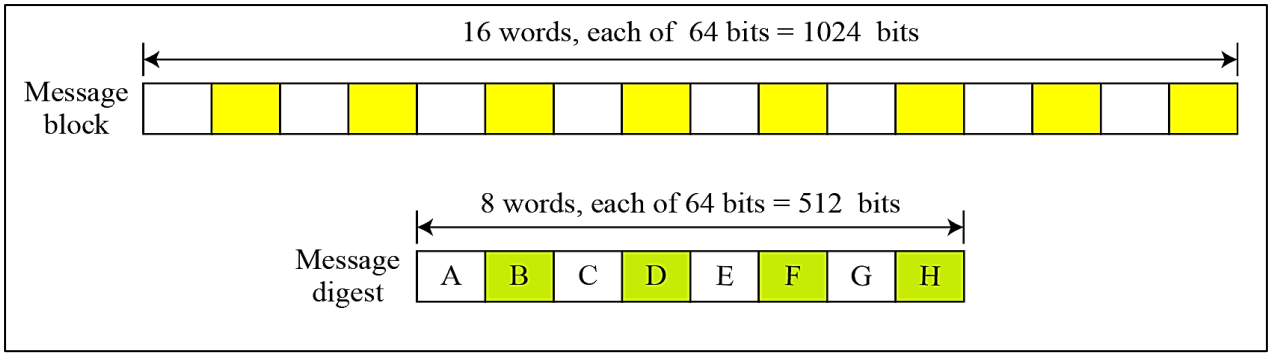


Figure 12.18 A message block and the digest as words

Word Expansion:

- Before processing, each message block (16 64-bit words) is expanded to 80 words.

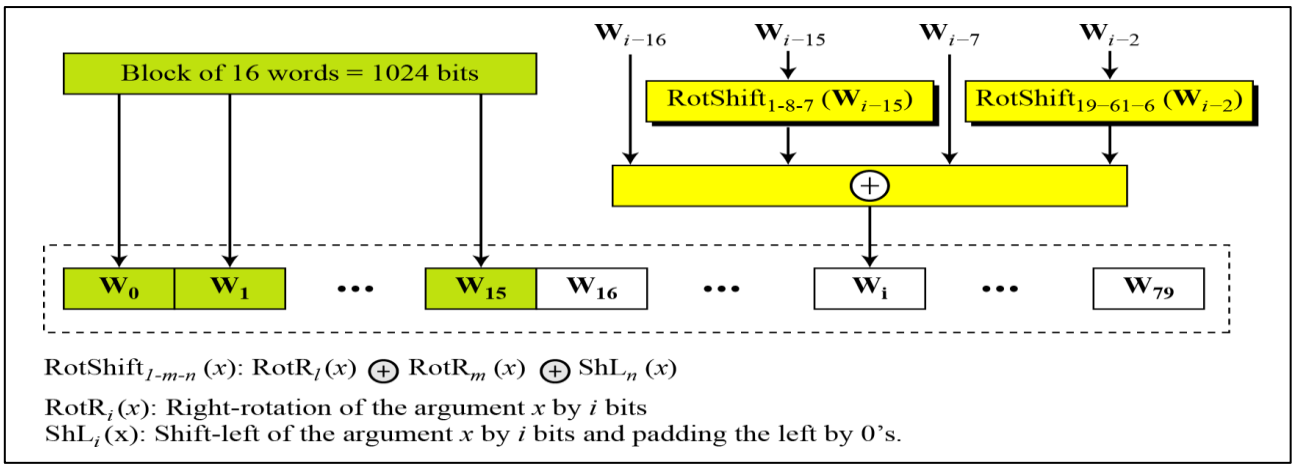
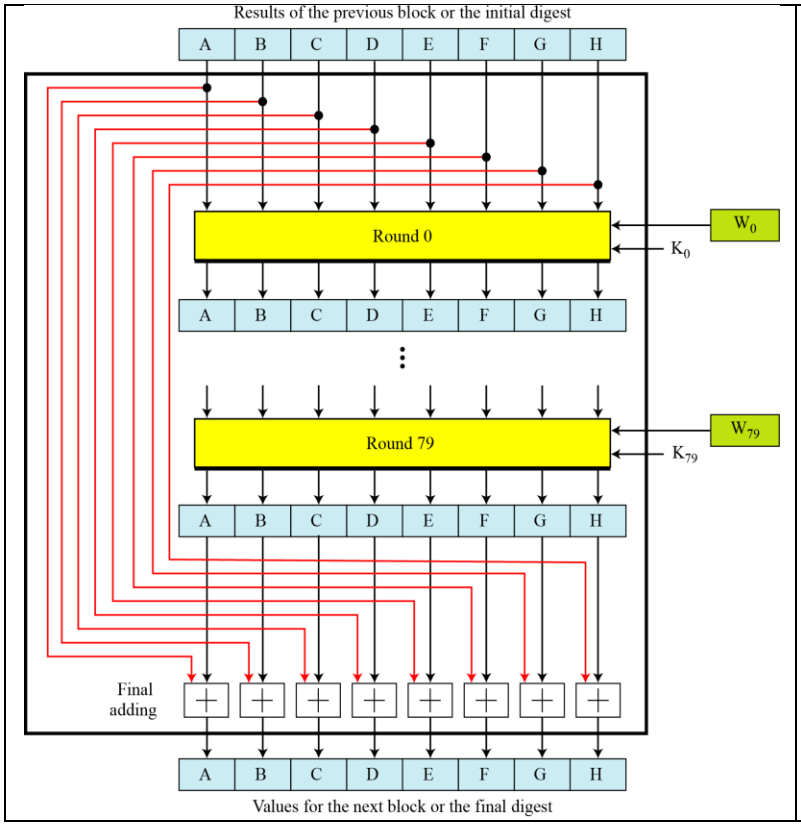


Figure 12.19 Word expansion in SHA-512

Message Digest Initialization: (ABCDEFGH)

- The constants are calculated from the first 8 primes (2, 3, 5, 7, 11, 17, and 19).
- Each value is the fraction part of square root of the corresponding prime number after converting to binary and keeping only the first 64 bits.

Compression Function:



- In ***each round,*** the contents of 8 previous buffers, one word (W_i) from the expanded block, and one 64-bit constant (K_i) are mixed together and then operated on to create a new set of 8 buffers.
- Final adding between the output of 80th round and input of first round.

Figure 12.20 Compression function in SHA-512

Structure of each round in SHA-512:

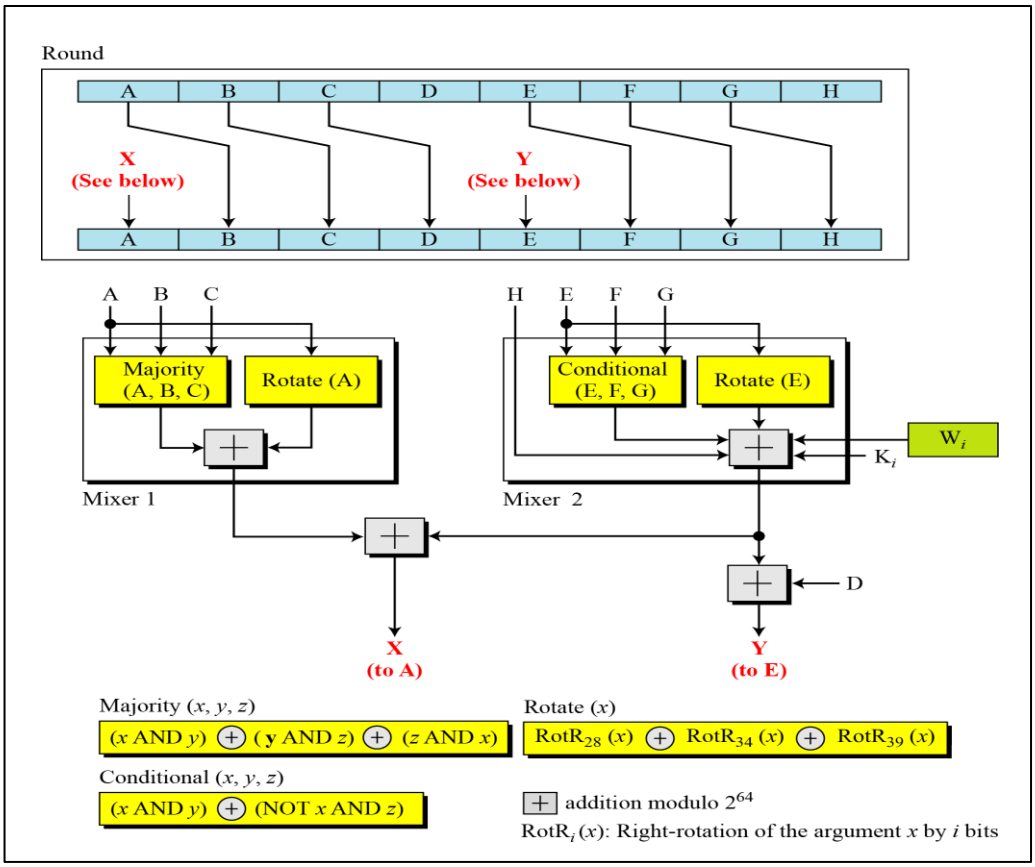


Figure 12.21 Structure of each round in SHA-512

Majority Function $(A_j \text{ AND } B_j) \oplus (B_j \text{ AND } C_j) \oplus (C_j \text{ AND } A_j)$

This is a bitwise function. If two or three bits are 1's, the resulting bit is 1; otherwise it is 0.

Conditional Function $(E_j \text{ AND } F_j) \oplus (\text{NOT } E_j \text{ AND } G_j)$

If E_j then F_j; else G_j.

Rotate function: $\text{Rotate (A): } \text{RotR}_{28}(A) \oplus \text{RotR}_{34}(A) \oplus \text{RotR}_{29}(A)$

$\text{Rotate (E): } \text{RotR}_{28}(E) \oplus \text{RotR}_{34}(E) \oplus \text{RotR}_{29}(E)$

Constants:

There are 80 constants, K_0 to K_{79}, each of 64 bits. Similar to the initial values for the eight digest buffers, these values are calculated from the first 80 prime numbers (2, 3, …, 409).

Each value is the fraction part of the *cubic root* of the corresponding prime number.

Analysis:

With a message digest of 512 bits, SHA-512 is expected to be resistant to all attacks, including collision attacks. It has been claimed that this version's improved design makes it more efficient and more secure than the previous versions. However, more research and testing are needed to confirm this claim.

6. What is KDC? Explain with neat diagrams.

7. What are the methods used to distribute the symmetric key? Explain.

Secret key is distributed using two techniques

1. Symmetric encryption

1. Centralized key distribution
2. Automatic key distribution
3. Decentralized key distribution

2. Asymmetric encryption

1. Simple secret key distribution
2. secret key distribution with confidentiality and authentication

1. Secret Key distribution using Symmetric encryption

- There are number of ways to distribute the secret key to both the parties A and B
  1. A can select key and physically deliver to B
  2. Third party can select & deliver key to A & B
  3. If A & B have communicated previously can use previous key to encrypt a new key
  4. If A & B have secure communications with a third party C, C can relay key between A & B
- Option 1 & 2 delivers a key manually. These methods are not suitable for large systems.
- In option 3, if an attacker succeeds in gaining access to one key, then all subsequent keys will be revealed.

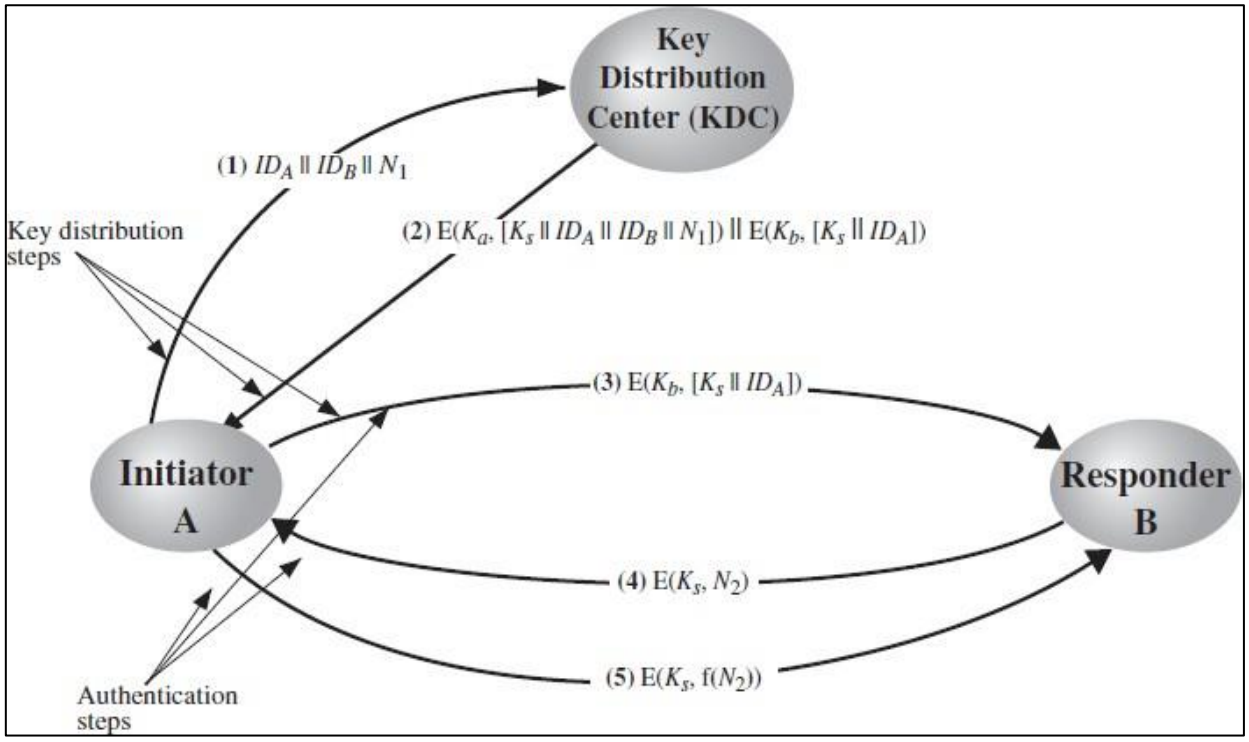
- Option 4 has been widely adopted. In this scheme, a key distribution centre is responsible for distributing keys to pairs of users. Each user must share a unique key with the key distribution centre for purposes of key distribution.

1. Centralized key distribution

- In this, each user shares a unique master key with the Key Distribution Centre (KDC).
- Let A shares a master key K_a, with the KDC; B shares the master key K_b with the KDC.

If user A wishes to establish a logical connection with B, performs the following steps:

1. "A" issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N1.
2. The KDC responds with a message encrypted using K_a. The message contains 2 items for A, session key Ks, and original request. And also, it contains 2 items for B, K_b and ID_a encrypted with K_b.
3. Thus, A reads the message, and sends information to B originated by KDC.
4. B sends N2 to A by encrypting it with Ks.
5. Using Ks, A responds with f(N2). 'f' is some function performed on N2.
6. Alice sends a message to the KDC that includes her nonce, RA, her identity and Bob's identity.
7. 2. The KDC sends an encrypted message to Alice that includes Alice's nonce, Bob's identity, the session key, and an encrypted ticket for Bob. The whole message is encrypted with Alice's key.
8. 3. Alice sends Bob's ticket to him.
9. 4. Bob sends his challenge to Alice (RB), encrypted with the session key.
10. 5. Alice responds to Bob's challenge. Note that the response carries RB-1 instead of RB.



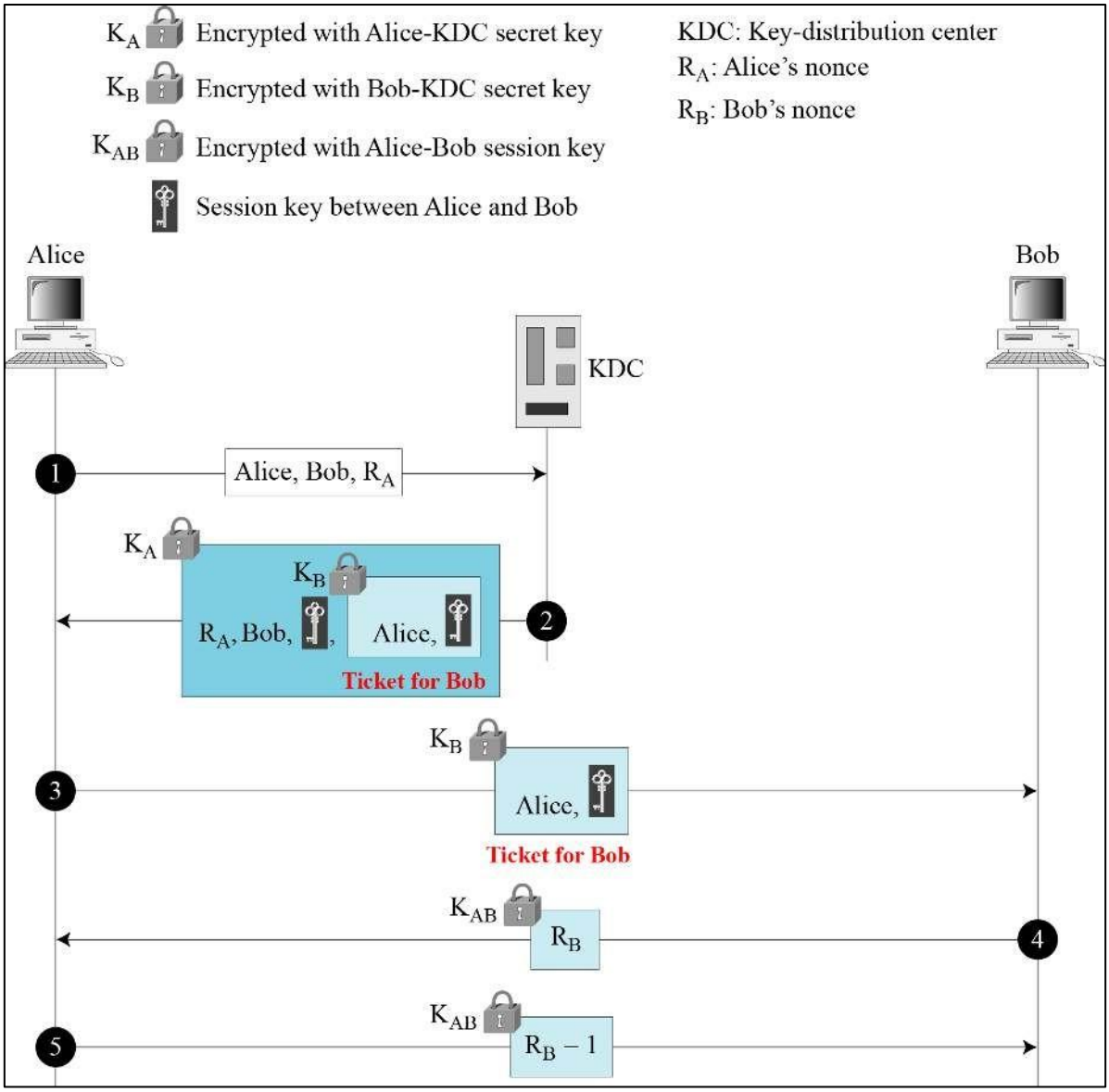


Figure 15.5 Needham-Schroeder protocol

2. Automatic key distribution

- The steps involved in establishing a connection are shown in figure.
- When one host wants to setup a connection to another host, it transmits a connection request packet. (step 1)
- The SSM (Session Security Module) saves that packet and applies to the KDC foe permission to establish the connection (step 2).
- The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.
- If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM (step 3).
- The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4).
- All user data exchanged between the two end systems are encrypted by their respective SSMs using the onetime session key.

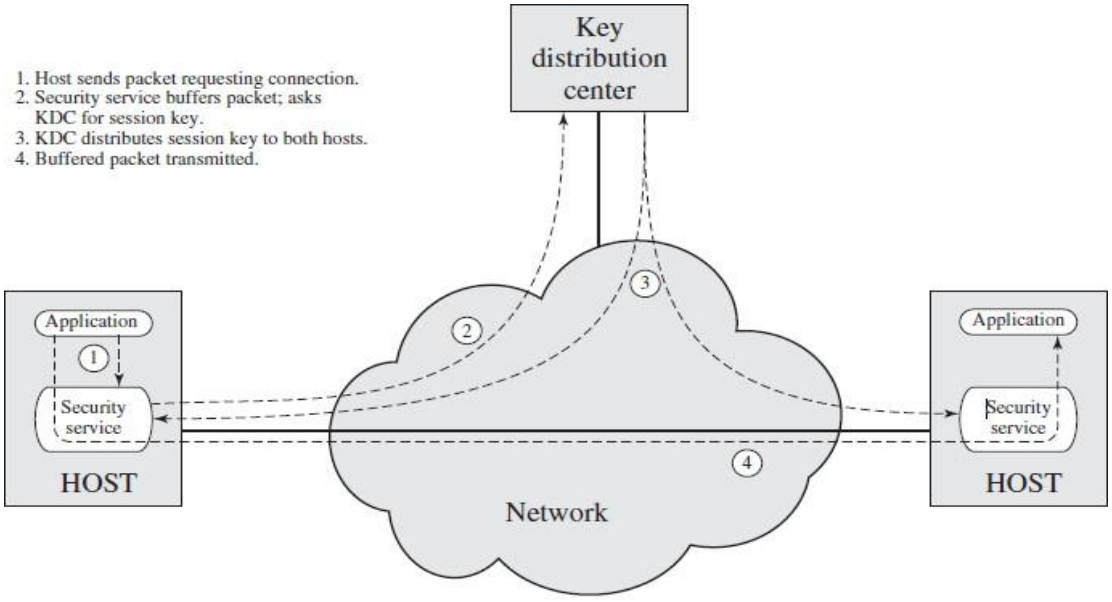


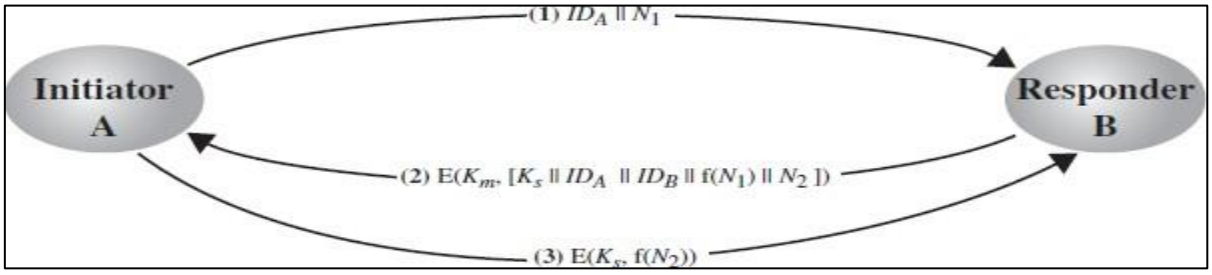
Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

3. Decentralized key distribution

- In this there is no key distribution centre. Whenever a user wants to communicate with other user, sends request for the session key to that user.

A session key is distributed with the following sequence of steps:

1. **A** issues a request to **B** for a session key and includes a nonce, N_1.
2. **B** responds with a message that is encrypted using the shared master key. The response includes the session key selected by **B**, an identifier of B, the value f(N_1), and another nonce, N_2.
3. Using the new session key, **A** returns f(N2) to **B**.



2. Secret key is distributed using Asymmetric encryption

1. Simple Secret Key Distribution

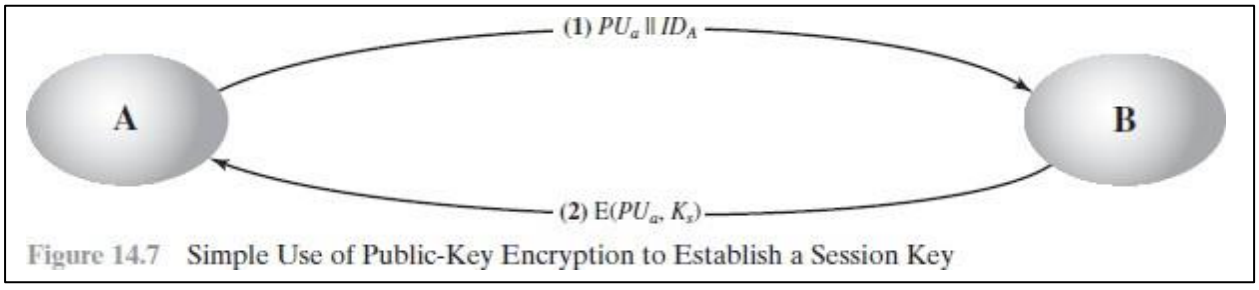


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PU_a and an identifier of A, IDA.
2. B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.

3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
4. A discards PU_a and PR_a and B discards PU_a.

- A and B can now securely communicate using conventional encryption and the session key Ks. At the completion of the exchange, both A and B discard Ks.
- No keys exist before the start of the communication and after the completion of communication.
- **Adv:** Secure from eavesdropping.
- **Problem:** man-in-the-middle attack

Steps involved in man-in-the-middle attack:

1. A generates a public/private key pair { PU_a, PR_a } and transmits a message intended for B consisting of PU_a and an identifier of A, IDA.
2. E intercepts the message, creates its own public/private key pair { PU_e, PR_e } and transmits PU_e || ID_A to B.
3. B generates a secret key, K_S, and transmits $E(PU_e, K_S)$.
4. E intercepts the message and learns Ks by computing $D(PRe, E(PU_e, K_S))$.
5. E transmits $E(PU_a, K_S)$ to A.

2. Secret Key Distribution with Confidentiality and Authentication

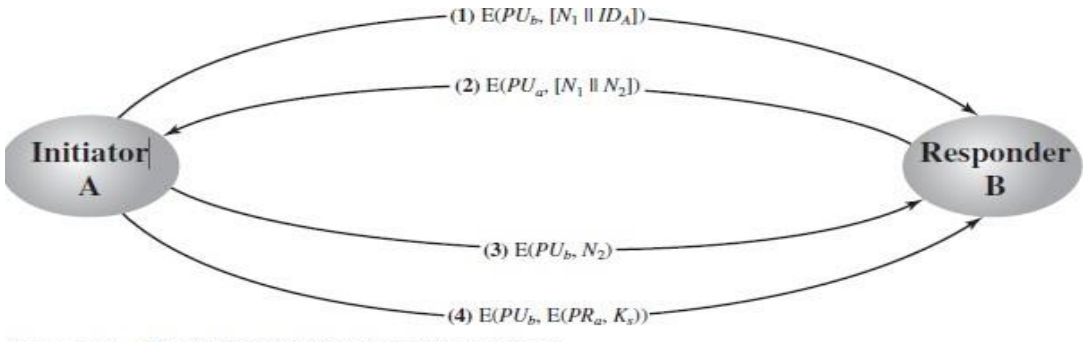


Figure 14.8 Public-Key Distribution of Secret Keys

Procedure to distribute public key:

1. A uses B's public key to encrypt a message to B containing an identifier IDA and a nonce N_1, which is used to identify this transaction uniquely
2. B sends a message to A encrypted PUa with and containing A's nonce (N_1) and new nonce (N_2) generated by B.
3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key Ks and sends $M=E(PU_b, E(PR_a, K_S))$to B.
5. B computes D(PUa, $E(PR_b, Ks)$) to recover the secret key.

This scheme ensures that confidentiality and authentication in the exchange of secret key

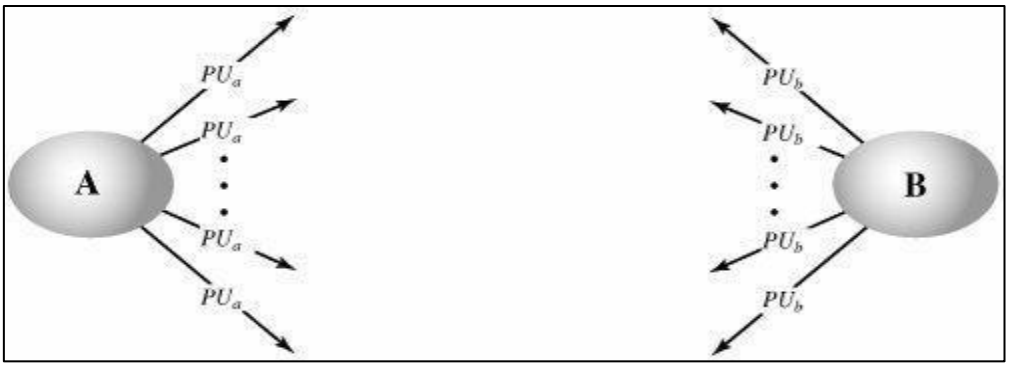
8. Discuss how public key is distributed in Asymmetric key Cryptography.

• Techniques for the distribution of public keys are grouped into:

1. public announcement
2. publicly available directory
3. public-key authority
4. public-key certificates

1. Public Announcement

- Users distribute public keys to recipients or broadcast to all the users in the network
- E.g., Append PGP (pretty good privacy) keys to **email messages** or post to **news groups** or **email list**



Major weakness is forgery

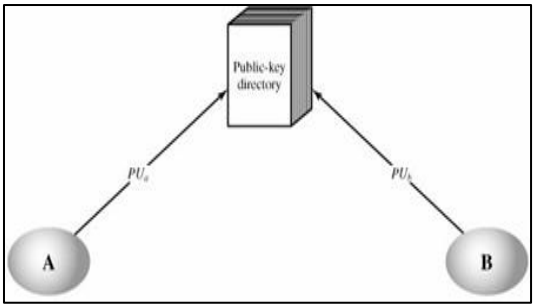
- Anyone can create a key claiming to be someone else and broadcast it
- Until forgery is discovered an attacker can masquerade as claimed user

2. Publicly Available Directory

- A public directory of public keys is used. When user A requires a public key of user B, he gets that key from the directory. Some organization or trusted one is responsible for maintenance and distribution of the public directory.

Directory is having following properties:

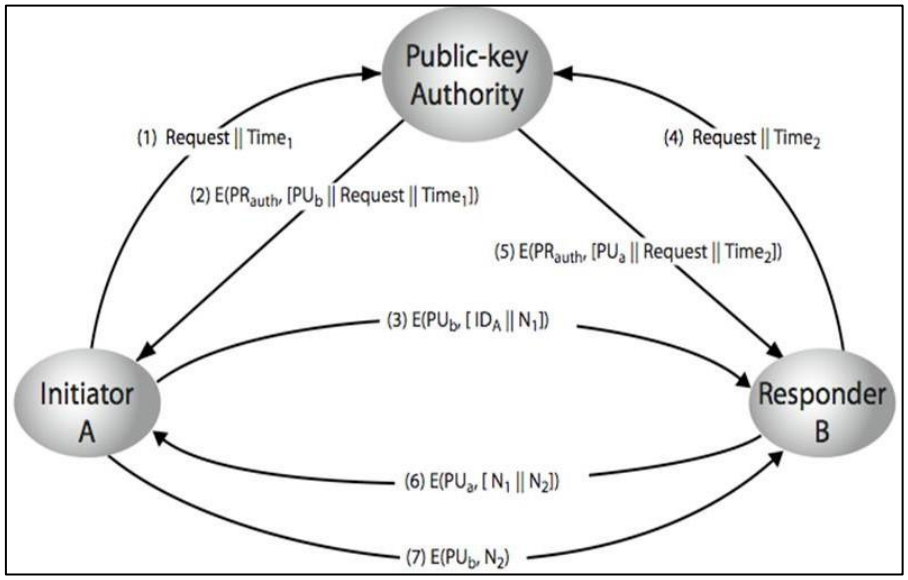
1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time.
4. Directory is periodically published
5. Participants could also access the directory electronically.



- Still vulnerable to tampering or forgery

3. Public-Key Authority

- In this public key authority maintains a directory of public keys. Required uses can request and get the public key from the public key authority. The user must know the public key of the authority PUauth.



Drawbacks:

1. There is only one authority, so performance may be decreased.
2. If authority is hacked, the entire database in the hand of hacker.

4. Public key certificates

- There exists a trusted Certification Authority (CA), which issues public key certificates. These certificates are used for exchanging the keys without interaction of public-key authority.
- The certificate authority can create and update certificates
- Any participant can get the certificates by supplying public key.
- A certificate binds identity of the user to public key – Contains other info such as period of validity, issuer's info, etc
- All contents signed by a trusted Certification Authority (CA)
- Any participant can read a certificate to determine the name and public key of the certificate's owner.
- Certificate can be verified by anyone who knows the CA public-key.

Example:

Certificate of A

$CA = E(PR_{auth}, [T||IDA || PUa])$

Where,

PR_{auth} – private key of Certificate Authority

T- Timestamp

Any participants can read and verify the certificate of A as…….

$D(PU_{auth}, CA) = D(PU_{auth}, E(PR_{auth}, [T \| IDA \| PUa])) = (T \| IDA \| PUa)$

9.Explain about Digital Signature algorithm in detail.

A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature, where the prerequisites are satisfied, gives a recipient very strong reason to believe that the message was created by a known sender(authentication), and that the message was not altered in transit (integrity).

A digital signature scheme typically consists of 3 algorithms:

- A key generation algorithm that selects a private key uniformly at random from a set ofpossible private keys. The algorithm outputs the private key and a corresponding publickey.
- A signing algorithm that, given a message and a private key, produces a signature.
- A signature verifying algorithm that, given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key.

Secondly, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key.

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

The differences between conventional signatures and digital signatures:

Inclusion:

- A conventional signature is included in the document; it is part of the document. We sign a document digitally, we send the signature as a separate document.

Verification Method:

- For a conventional signature, when the recipient receives a document, she compares the signature on the document with the signature on file.
- For a digital signature, the recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

Relationship:

- For a conventional signature, there is normally a one-to-many relationship between a signature and documents. For a digital signature, there is a one-to-one relationship between a signature and a message.

Duplicity:

- In conventional signature, a copy of the signed document can be distinguished from the original one on file.
- In digital signature, there is no such distinction unless there is a factor of time on the document

PROCESS OF CREATING DIGITAL SIGNATURE

The sender uses a signing algorithm to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the verifying algorithm to the combination. If the result is true, the message is accepted; otherwise, it is rejected.

Digital signature process

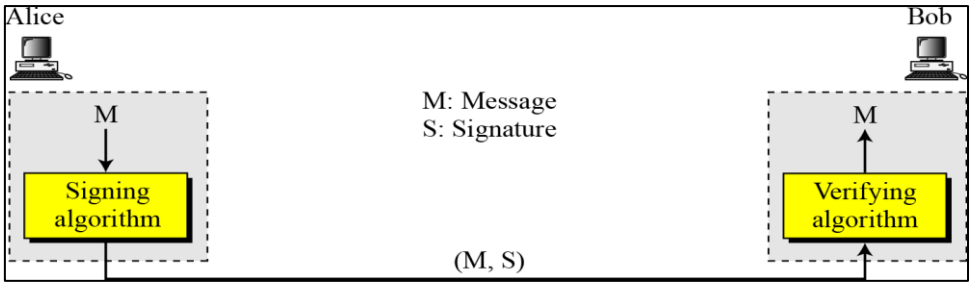


Figure 13.1 Digital signature process

- Figure 13.1 shows the digital signature process. The sender uses a signing algorithm to sign the message.
- The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the verifying algorithm to the combination.
- If the result is true, the message is accepted; otherwise, it is rejected.

Need for Keys:

A conventional signature is like a private "key", belonging to the signer of the document.

- In a digital signature, the signer uses her private key, applied to a signing algorithm to sign the document.
- The verifier, on the other hand, uses the public key of the signer, applied to the verifying algorithm, to verify the document. See Figure 13.2.

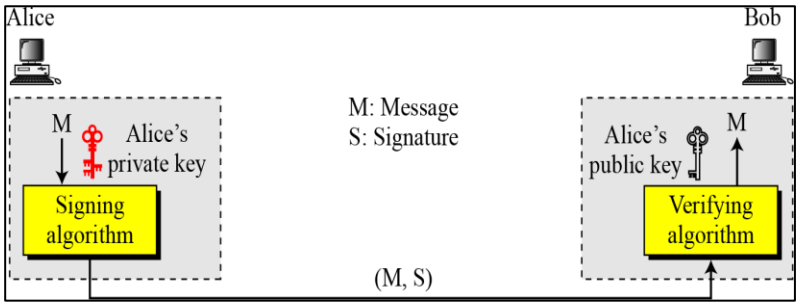


Figure 13.2 Adding key to the digital signature process

- A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.

- A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.

Signing the digest

Figure 13.3 shows signing a digest in a digital signature system.

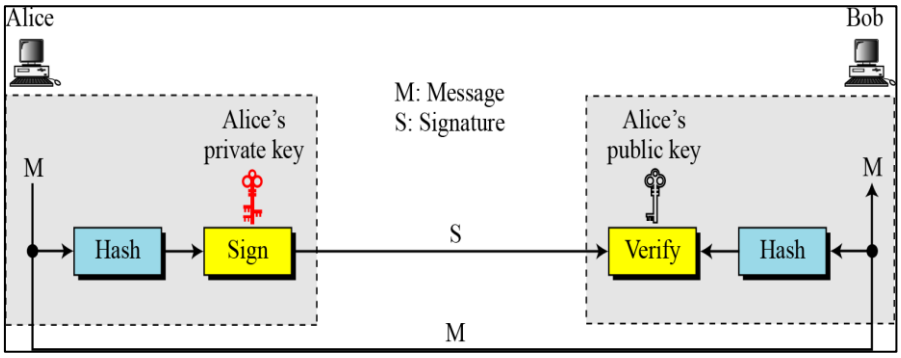


Figure 13.3 Signing the digest

- A digest is made out of the message at Alice's site. The digest then goes through the signing process using Alice's private key. Alice then sends the message and the signature to Bob.
- At Bob's site, using the same public hash function, a digest is first created out of the received message. Calculations are done on the signature and the digest. The verifying process also applies criteria on the result of the calculation to determine the authenticity of the signature. If authentic, the message is accepted; otherwise, it is rejected.

DIGITAL SIGNATURE SERVICES

Security services include message confidentiality, message authentication,

message integrity, and non-repudiation. A digital signature can directly provide the last three; for

message confidentiality we still need encryption/decryption.

1. Message Authentication:

A secure digital signature scheme, like a secure conventional signature can provide message authentication. A digital signature provides message authentication.

2. Message Integrity:

The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.

A digital signature provides message integrity.

3. Non-repudiation:

Using a trusted center for nonrepudiation

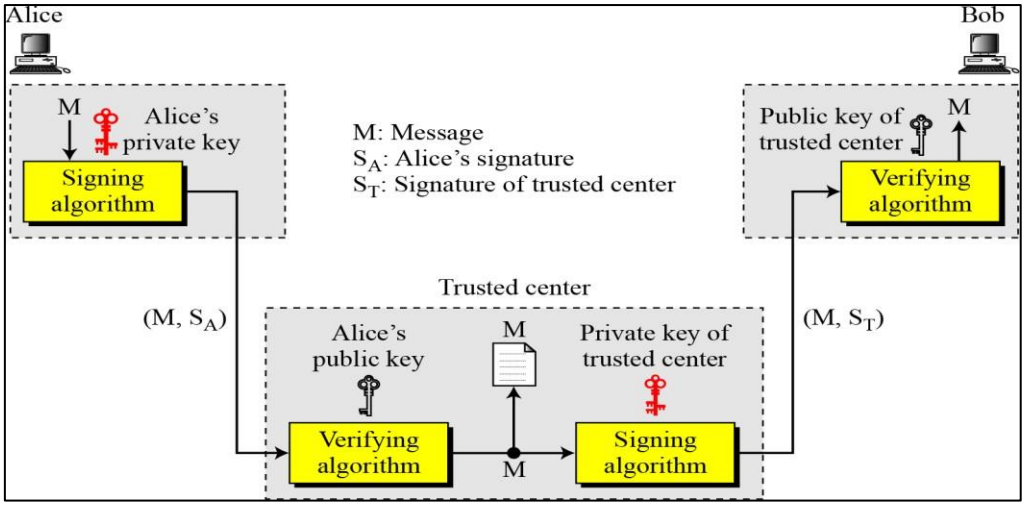


Figure 13.4 Using a trusted center for nonrepudiation

Nonrepudiation can be provided using a trusted party.

4. Confidentiality:

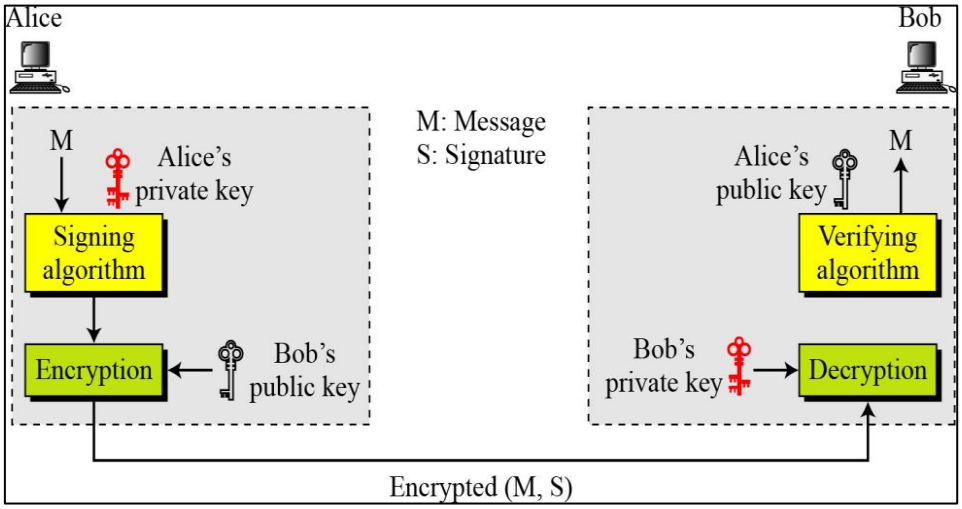


Figure 13.5 Adding confidentiality to a digital signature scheme

A digital signature does not provide privacy.

If there is a need for privacy, another layer of encryption/decryption must be applied.

10. What is Kerberos protocol and explain in detail.

- Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular.

- Several systems, including Windows 2000, use Kerberos.
- It is named after the three-headed dog in Greek mythology that guards the gates of Hades.

Servers

- Three servers are involved in the Kerberos protocol: an authentication server (AS), ticketgranting server (TGS), and a real (data) server that provides services to others.
- In our examples and figures, Bob is the real server and Alice is the user requesting service.
- Figure 15.7 shows the relationship between these three servers.

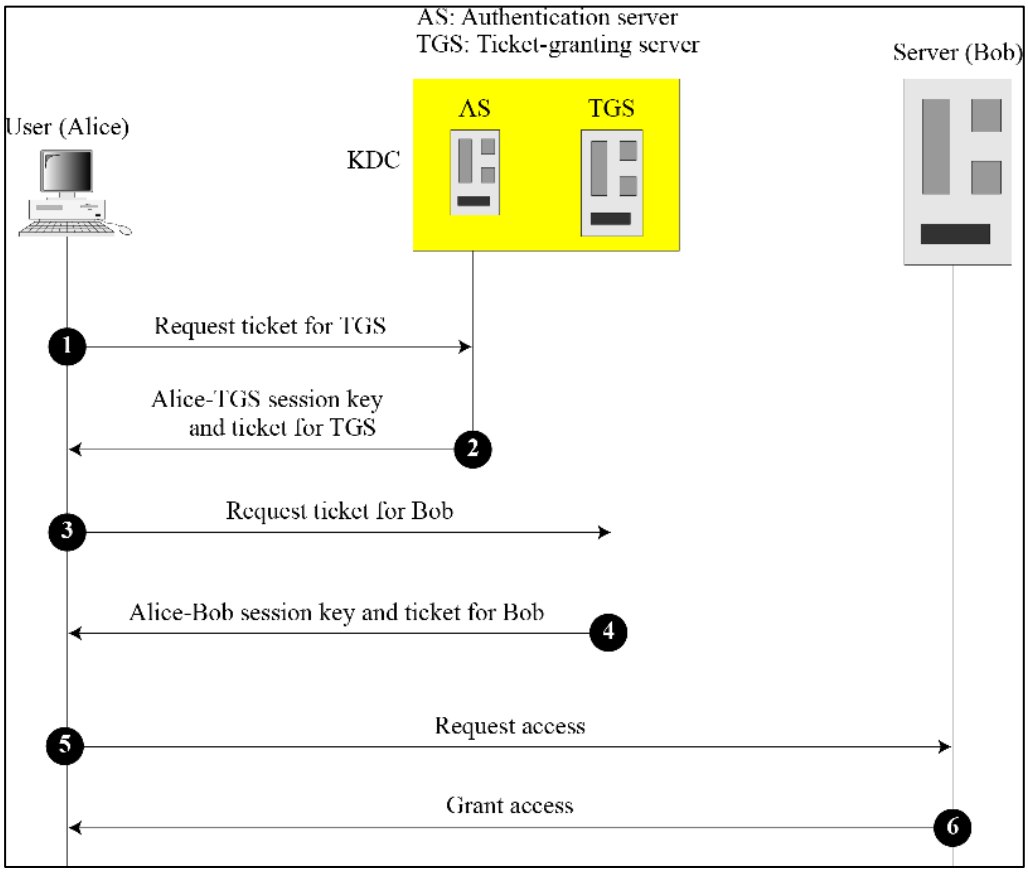


Figure 15.7 Kerberos servers

Authentication Server:

- The authentication server (AS) is the KDC in the Kerberos protocol. Each user registers with the AS and is granted a user identity and a password.
- The AS has a database with these identities and the corresponding passwords.
- The AS verifies the user, issues a session key to be used between Alice and the TGS, and sends a ticket for the TGS.

Ticket-Granting Server:

- TS issues a ticket for the real server (Bob).
- It also provides the session key (KAB) between Alice and Bob.
- Kerberos has separated user verification from the issuing of tickets.
- In this way, though Alice verifies her ID just once with the AS, she can contact the TGS multiple times to obtain tickets for different real servers.

Real server

- The real server (Bob) provides services for the user (Alice).

- Kerberos is designed for a client –server program, such as FTP, in which a user uses the client process to access the server process.
- Kerberos is not used for person-to –person authentication.

Operation

A client process can access a process running on the real server (Bob) in six steps, as shown in figure 15.8.

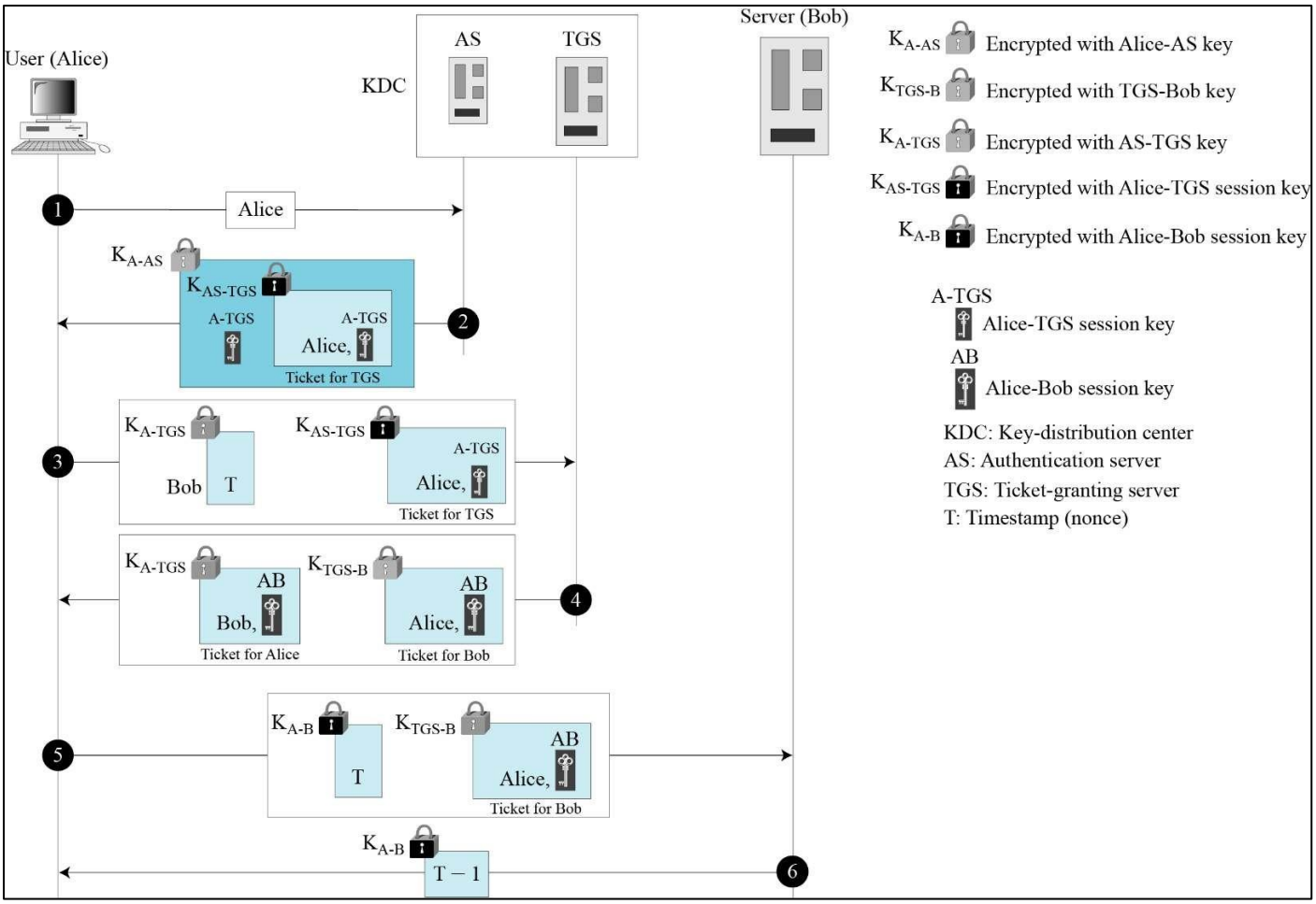


Figure 15.8 Kerberos example

1. Alice sends her request to the AS in plaintext using her register identity.
2. The AS sends a message encrypted with Alice's permanent symmetric key, KA-AS. The message contains two items: a session key, KA-TGS that is used by Alice to contact the TGS, and a ticket for the TGS that is encrypted with the TGS symmetric key, KA-TGS.
3. Alice now sends three items to the TGS. The first is the ticket received from the AS. The second is the name of the real server (Bob), the third is a timestamp that is encrypted by KA-TGS and the ticket are extracted.
4. Now, the TGS sends two tickets, each containing the session key between Alice and Bob KA-B. The ticket for Alice is encrypted with KA-TGS; the ticket for Bob is encrypted with Bob's key, KTGS-B.
5. Alice sends Bob's ticket with the timestamp encrypted by KA-B.

6. Bob confirms the receipt by adding 1 to the timestamp. The message is encrypted with KA-B and sent to Alice.

11. Describe Certificate Authority and X.509 Certificate.

- Although the use of a CA has solved the problem of public key fraud, it has created a sideeffect. Each certificate may have a different format.
- If, Alice wants to use a program to automatically download different certificates and digests belonging to different people, the program may not be able to do this.
- One certificate may have the public key in one format and another in a different format.
- The public key may be on the first line in one certificate, and on the third line in another anything that needs to be used universally must have a universal format.
- The ITU has designed X.509.

Certificate

Figure 15.17 shows the format of a certificate.

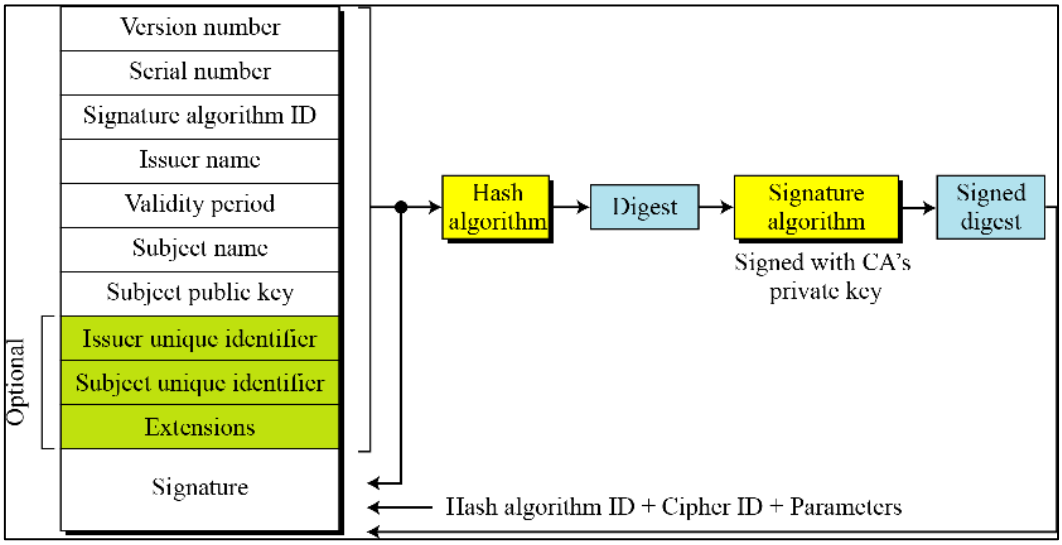


Figure 15.17 X.509 certificate format

A certificate has the following fields:

• Version number:

This field defines the version of X.509 of the certificate. The version number started at 0; the current version is 2.

• Serial number:

This field defines a number assigned to each certificate. The value is unique for each certificate issuer.

• **Signature algorithm ID:**

This field identifies the algorithm used to sign the certificate. Any parameter that is needed for the signature is also defined in this field.

• **Issuer name:**

This field identifies the certification authority that issued the certificate. The name is normally a hierarchy of strings that defines a country, a state, organization, department, and so on.

• **Validity Period:**

This field defines the earliest time and the latest time the certificate is valid.

• **Subject name:**

This field defines the entity to which the public key belongs. It is also a hierarchy of strings. Part of the field defines what is called the common name, which is the actual name of the beholder of the key.

• **Subject public key:**

This field defines the Owner's public key, the heart of the certificate. This field also defines the corresponding public-key algorithm and its parameters.

• **Issuer unique identifier:**

This optional field allows two issuers to have the same issuer field value, if the issuer unique identifiers are different.

• **Subject unique identifier:**

This optional field allows two different subjects to have the same subject field value, if the subject unique identifiers are different.

• **Extensions:**

This optional field allows issuers to add more private information to the certificate.

• **Signature:**

This field is made of three sections. The first section contains all other fields in the certificate. The second section contains the digest of the first section encrypted with the CA's public key. The third section contains the algorithm identifier used to create the second section.

Certificate Renewal

- Each certificate has a period of validity. If there is no problem with the certificate, the CA issues a new certificate before the old one expires.
- The process is like the renewal of credit cards by a credit card company; the credit card holder normally receives a renewed credit card before the one expires.

Certificate Revocation:

In some cases, a certificate must be revoked before its expiration. Here are some examples:

a) The user's private key might have been comprised.
b) The CA is no longer willing to certify the user. For example, the user's certificate relates to an organization that she no longer works for.
c) The CA's private key, which can verify certificates, may have been compromised. In this case, the CA needs to revoke all unexpired certificates.

The revocation is done by periodically issuing a certificate revocation list (CRL). Figure 15.18 shows the certificate revocation list.

Figure 15.17 Certificate revocation format

A certificate revocation list has the following fields:

• **Signature algorithm ID:** This field is the same as the one in the certificate.

• **Issuer name:** this field is the same as the one in the certificate.

• **This update date:** This field defines when the list is released.

• **Next update date:** This field defines the next date when the new list will be released.

• **Revoked certificate:** This is a repeated list of all unexpired certificates that have been revoked. Each list contains two sections: user certificate serial number and revocation date.

• **Signature:** This field is the same as the one in the certificate list.