CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 4 NOTES

UNIT IV MESSAGE AUTHENTICATION AND INTEGRITY

Authentication requirement – Authentication function – MAC – Hash function – Security of hash function and MAC – SHA –Digital signature and authentication protocols – DSS- Entity Authentication: Biometrics, Passwords, Challenge Response protocols- Authentication applications – Kerberos, X.509

4.1 AUTHENTICATION REQUIREMENT

Communication across the network, the following attacks can be identified.

Disclosure – release of message contents to any person or process not possessing the appropriate cryptographic key.

Traffic analysis – discovery of the pattern of traffic between parties.

- In a connection oriented application, the frequency and duration of connections could be determined.
- In either a connection oriented or connectionless environment, the number and length of messages between parties could be determined.

Masquerade – insertion of messages into the network from fraudulent source. This can be creation of message by the attacker using the authorized port.

Content modification – changes to the contents of a message, including insertion, deletion, transposition, and modification.

Sequence modification – any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

Timing modification – delay or replay of messages.

- In a connection oriented application, an entire session or sequence of messages could be replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.
- In a connectionless application, an individual message could be delayed or replayed.

Source repudiation – denial of transmission of message by source.

Destination repudiation – denial of receipt of message by destination.

4.2 AUTHENTICATION FUNCTION

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels.

At the lower level, there must be some sort of function that produces an authenticator, a value to be used to authenticate a message.

At the higher-level, low-level function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

The types of function that may be used to produce an authenticator are grouped into three classes.

1. Message Encryption – the ciphertext of the entire message serves as its authenticator.

2. Message Authentication Code (MAC) – a public function of the message and a secret key thatproduces a fixed length value that serves as the authenticator.

3. Hash Function – a public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator.

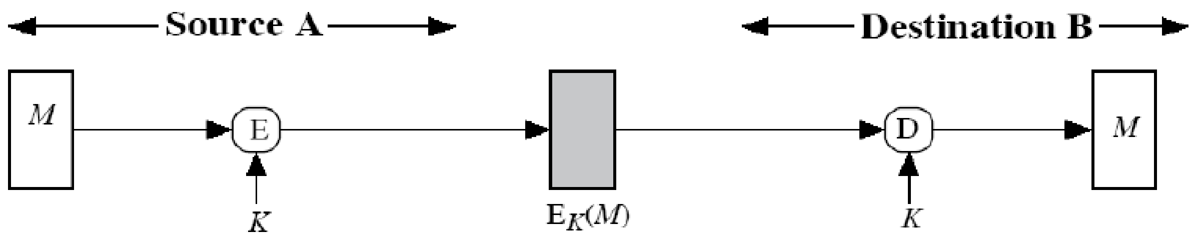
Message Encryption:

Message encryption Message encryption by itself can provide a measure of authentication.

The analysis differs from symmetric and public key encryption schemes.

(a) If symmetric encryption is used then:

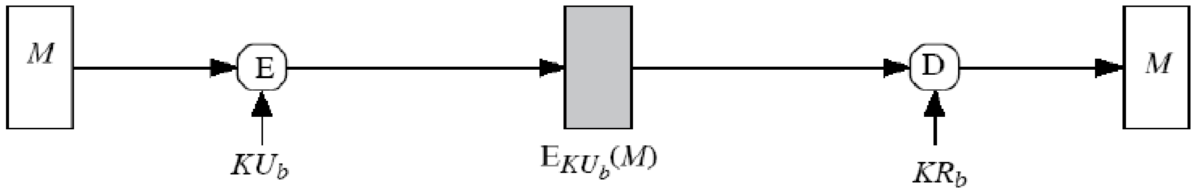
- A message m, transmitted from source A to destination B is encrypted using a secret key shared by A and B.
- Since only sender and receiver knows key used
- Receiver knows sender must have created it. Hence authentication is provided.
- Know content cannot have been altered. Hence confidentiality is also provided.
- If message has suitable structure, redundancy or a checksum to detect any changes
- Therefore Symmetric Encryption provides authentication and confidentiality.



Symmetric key encryption confidentiality, authentication and signature

(b) <u>If public-key encryption is used</u>:

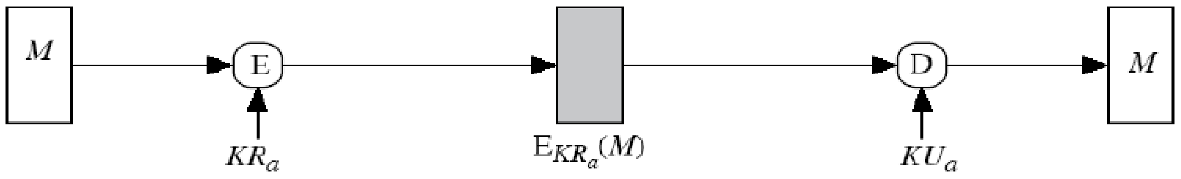
This method is the use of public key cryptography which provides confidentiality only. The sender A makes use of the public key of the receiver to encrypt the message. Here there is no authentication because any user can use B‟s public key to send a message and claim that only A has sent it.



Public key encryption confidentiality

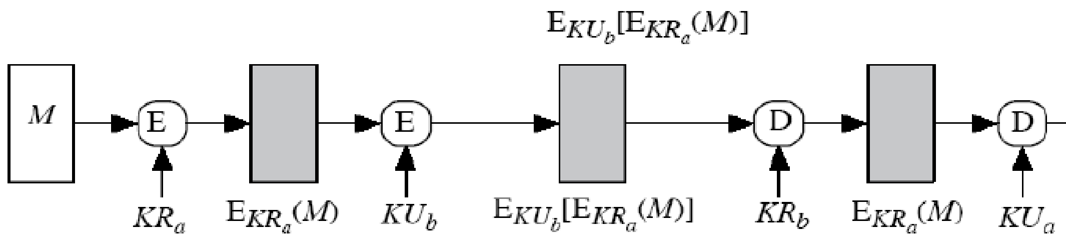
In this method to have only authentication, the message is encrypted with the sender‟s A‟s private key. The receiver B uses the sender‟s A‟s public key to decrypt the message. Now A cannot deny that it has not transmitted since it only knows its private key. This is called as authentication or Digital Signature. Hence the problem is the,

- Receiver cannot determine whether the packet decrypted contains some useful message or random bits.
- The problem is that anyone can decrypt the message when they know the public key of sender A.



Public key encryption authentication and signature

This method provides authentication, confidentiality and digital signature. But the problem with this method is the complex public key cryptography algorithm should be applied twice during encryption and twice during decryption.



Public key encryption confidentiality, authentication and signature

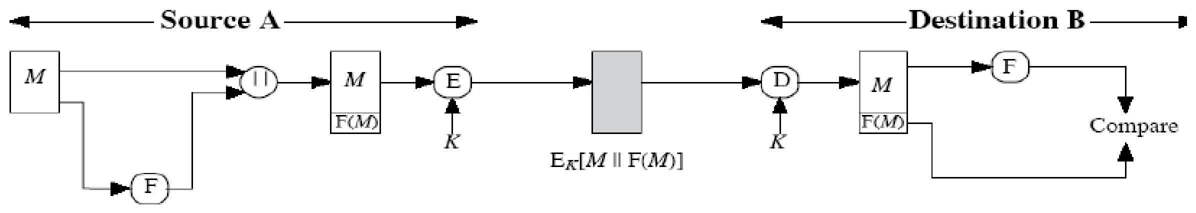
Suppose the message can be any arbitrary bit pattern, in that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message.

One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function.

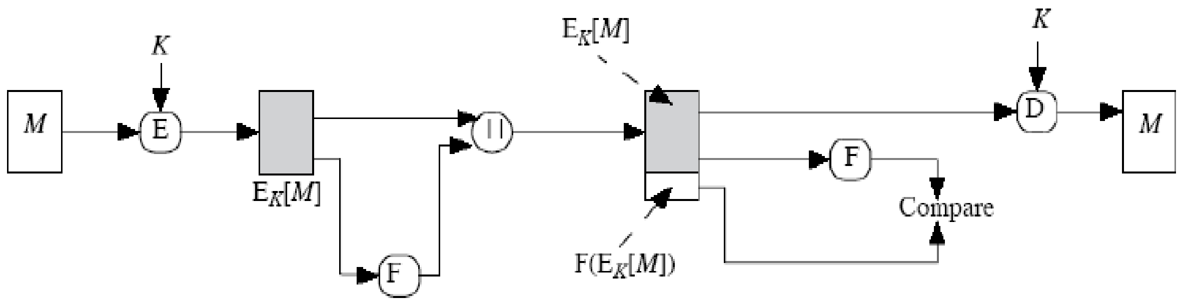
Append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption „A" prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted.

At the destination, B decrypts the incoming block and treats the result as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS.

If the calculated FCS is equal to the incoming FCS, then the message is considered authentic. In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext.



Internal error control



External error control

4.3 MAC

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message.

This technique assumes that two communication parties say A and B, share a common secret key „k". When A has to send a message to B, it calculates the MAC as a function of the message and the key.

MAC = C (K , M)

Where M – input message C – MAC function K – Shared secret key

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC.

The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic. A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function.

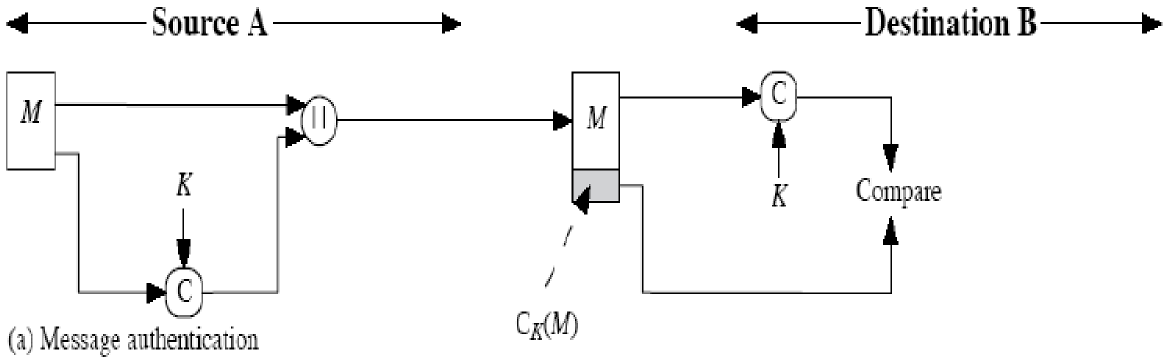


Fig 4.3a Message Authentication

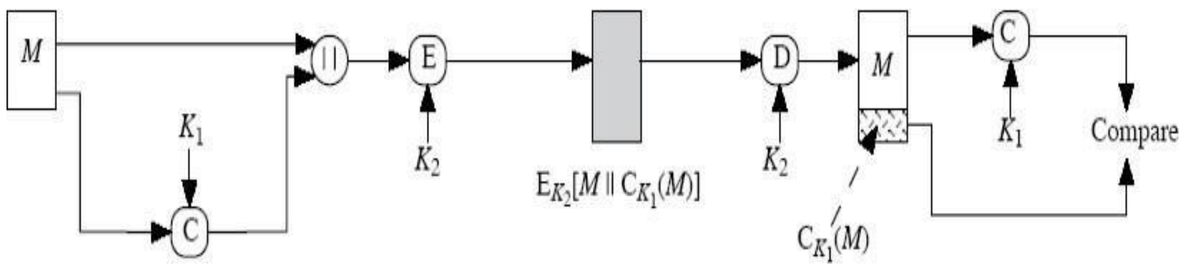


Fig 4.3b Message authentication and confidentiality, authentication tied to plain text

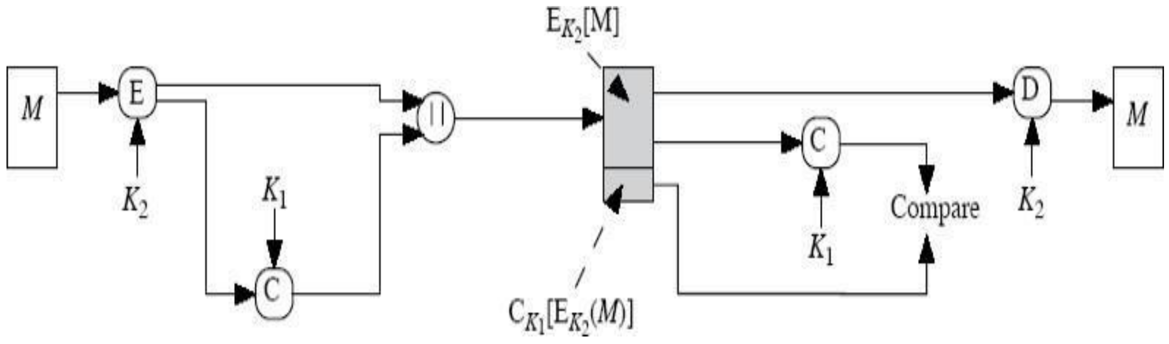


Fig 4.3c Message authentication and confidentiality, authentication tied to ciphertext

Requirements for MAC:

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key.

If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

3. If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

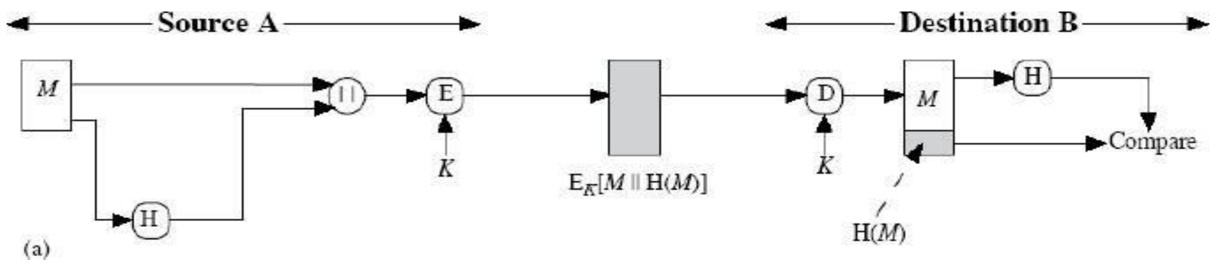
The process depicted in Figure 4.3a provides authentication but not confidentiality, because the message as a whole is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (Figure 4.3b) or before (Figure 4.3c) the MAC algorithm. In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver. In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted. In the second case, the message is encrypted first. Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block. Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure 12.4b is used.

4.4 HASH FUNCTION

A variation on the message authentication code is the one way hash function. As with MAC, a hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M).

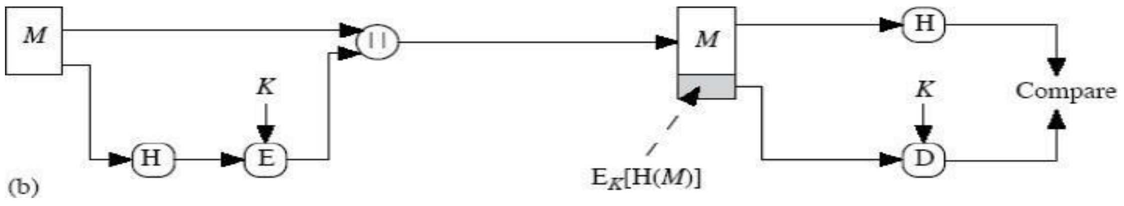
Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value. There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the hash code, confidentiality is also provided.

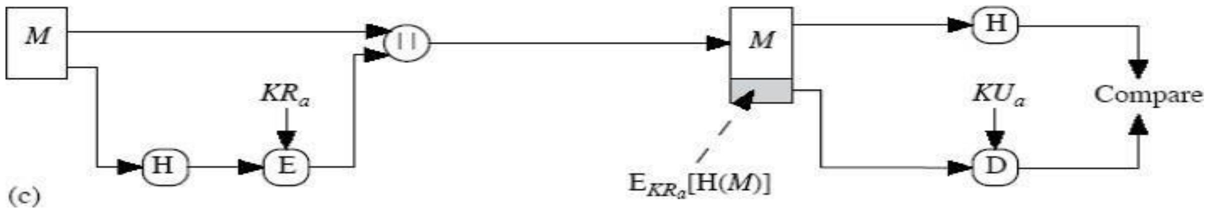


Hash Function

Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

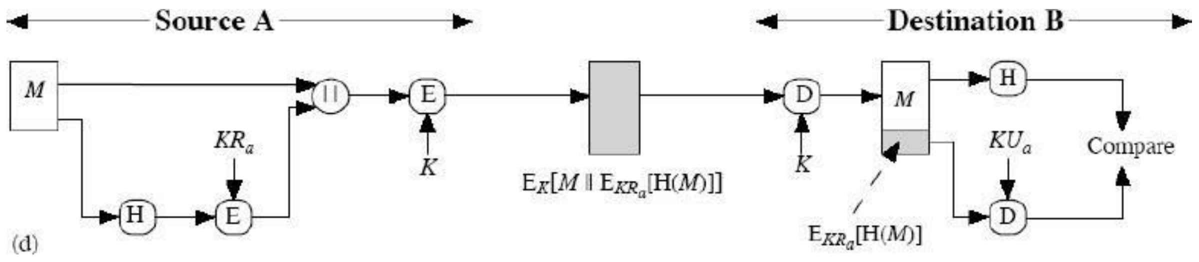


Only the hash code is encrypted, using the public key encryption and using the sender‟s private key. It provides authentication plus the digital signature.

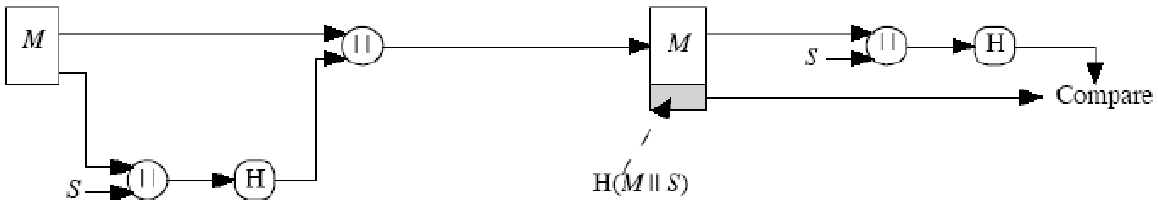


Basic use of Hash Function

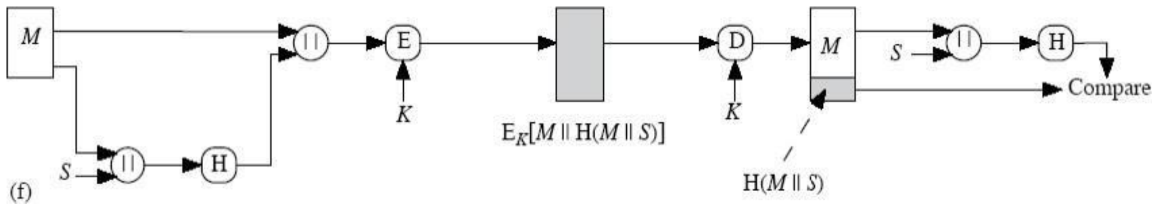
If confidentiality as well as digital signature is desired, then the message plus the public key encrypted hash code can be encrypted using a symmetric secret key.



This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value „S". The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.



Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.



Basic use of Hash Function

A hash value h is generated by a function H of the form

$$h = H(M)$$

where M is a variable-length message and H(M) is the fixed-length hash value.

The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value.

Requirements for a Hash Function

1. H can be applied to a block of data of any size.

2. H produces a fixed-length output.

3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.

5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as weak collision resistance.

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as strong collision resistance.
The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code.

The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used.

The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function. One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.

This can be expressed as follows: $C_i = b_{i1} + b_{i1} ... + b_{im}$

where

C_i = ith bit of the hash code, $1 \leq i$
$\leq n$ m = number of n-bit blocks in
the input b_{ij}= ith bit in jth block

Procedure:

1. Initially set the n-bit hash value to zero.

2. Process each successive n-bit block of data as follows:

   a. Rotate the current hash value to the left by one bit.
   b. XOR the block into the hash value.

Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted message M, then an opponent would need to find an M' such that H(M') = H(M) to substitute another message and fool the receiver.

On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message.

However, a different sort of attack is possible, based on the birthday paradox. The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key.

1. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. (Fraudulent message)

2. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.

3. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of 2^{32}

MEET-IN-THE-MIDDLE ATTACK.

Divide a message M into fixed-size blocks $M_1, M_2, \ldots, M_N$ and use a symmetric encryption system such as DES to compute the hash code G as follows:

H_o = initial value $H_i = E_{M_i}[H_{i-1}]$

$G = H_N$

This is similar to the CBC technique, but in this case there is no secret key. As with any hash code, this scheme is subject to the birthday attack and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.

Here is the scenario; we assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long:

1. Calculate the unencrypted hash code G.

2. Construct any desired message in the form $Q_1, Q_2, ..., Q_{N2}$.

3. Compute for $H_i = EQ_i [H_{i-1}]$for $1 \leq i \leq (N-2)$.

4. Generate $2^{m/2}$ random blocks; for each block X, compute $E_X[H_{N-2}.]$ Generate an additional $2^{m/2}$ random blocks; for each block Y, compute $D_Y[G]$, where D is the decryption function corresponding to E.

5. Based on the birthday paradox, with high probability there will be an X and Y such that $E_X [H_{N-2}] = D_Y[G]$.

6. Form the message $Q_1, Q_2, ..., Q_{N-2}$, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

4.5 SECURITY OF HASH FUNCTION AND MAC

Just as with symmetric and public-key encryption, we can group attacks on hash functions and MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

The nature of brute-force attacks differs somewhat for hash functions and MACs.

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm.

Requirements of Hash Function:

One-way: For any given code h, it is computationally infeasible to find x such that $H(x) = h$.

Weak collision resistance: For any given block x, it is computationally infeasible to find y x with $H(y) = H(x)$.

Strong collision resistance: It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

Message Authentication Codes

A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.. To attack a hash code, we can proceed in the following way. Given a fixed message x with n-bit hash code h = H(x), a brute-force method of finding a collision is to pick a random bit string y and check if H(y) = H(x). The attacker can do this repeatedly off line.

To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

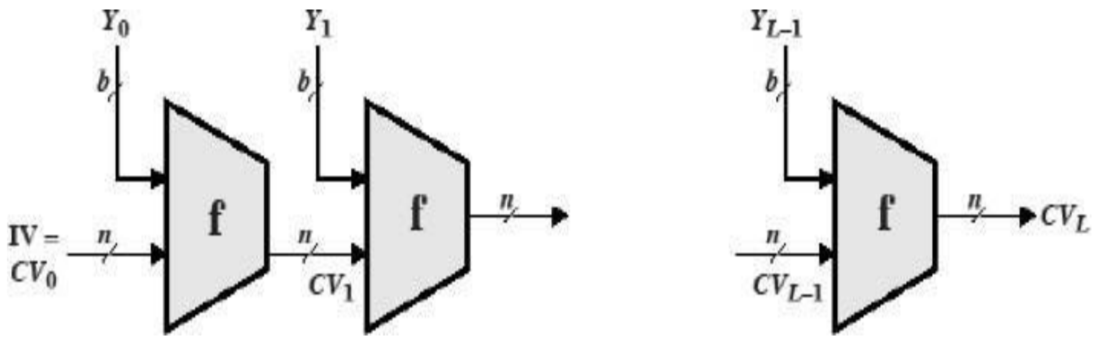
Cryptanalysis

As with encryption algorithms, cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Hash Functions

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits. The final block also includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult.

Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.



IV=Initial Value
Y_i = ith input block
n=Length of Hash code

CV=Changing Variable
L=number of input blocks
b=Length of input block

General structure of secure hash code

The hash algorithm involves repeated use of a compression function,f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output.

At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often b > n; hence the term compression.

The hash function can be summarized as follows:

CV0 = IV = initial n-bit value CVI = f(CVi-1, Yi-1) 1 ≤i ≤L H(M) = CVL

Where the input to the hash function is a message M consisting of the blocks Yo, Y1,..., YL-1. The structure can be used to produce a secure hash function to operate on a message of any length.

Message Authentication Codes :

There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs.

4.6 SHA

The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

Comparison of SHA Parameters

	SHA I	SHA 224	SHA 256	SHA 384	SHA 512
Message Digest Size	160	224	256	384	512
Message Size	<264	<264	<264	<2128	<2128
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

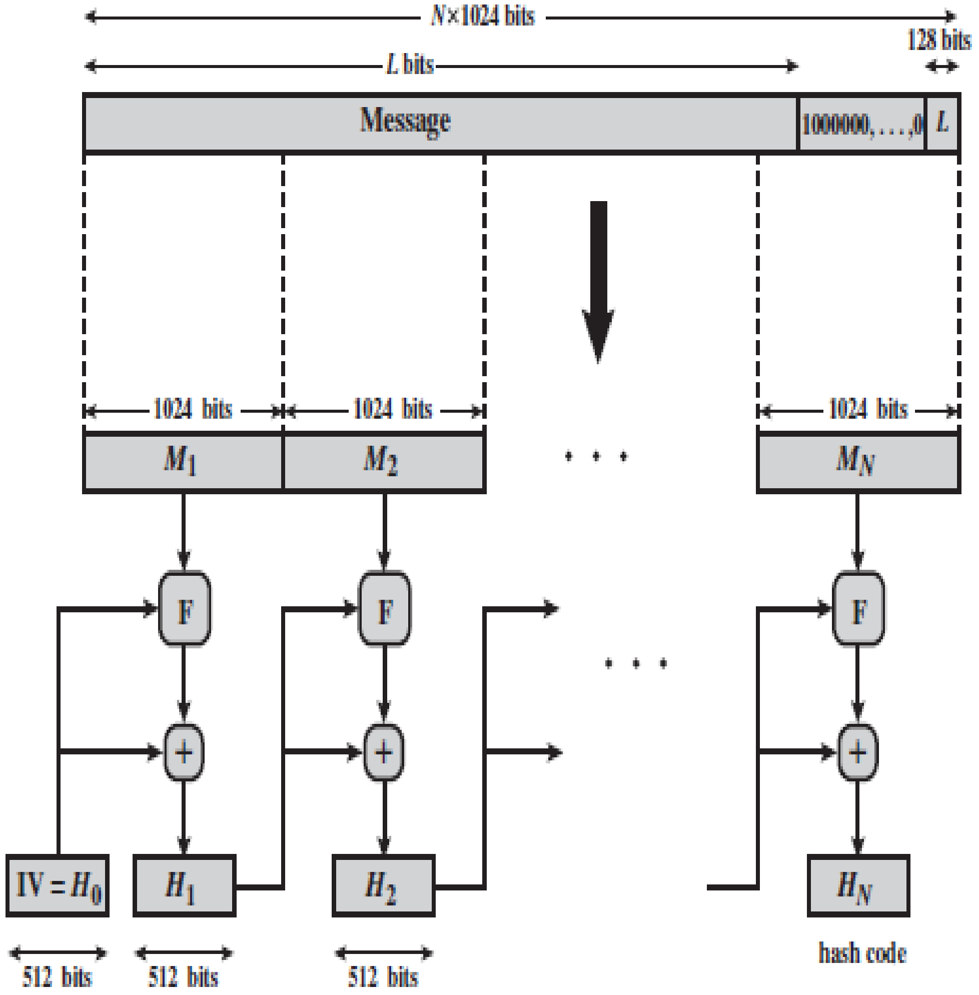
Step 1
Append padding bits.

The message is padded so that its length is congruent to 896 modulo 1024 [length K 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step 2
Append length.

A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.



Message Digest Generation Using SHA-512

Step 3
Initialize hash buffer.

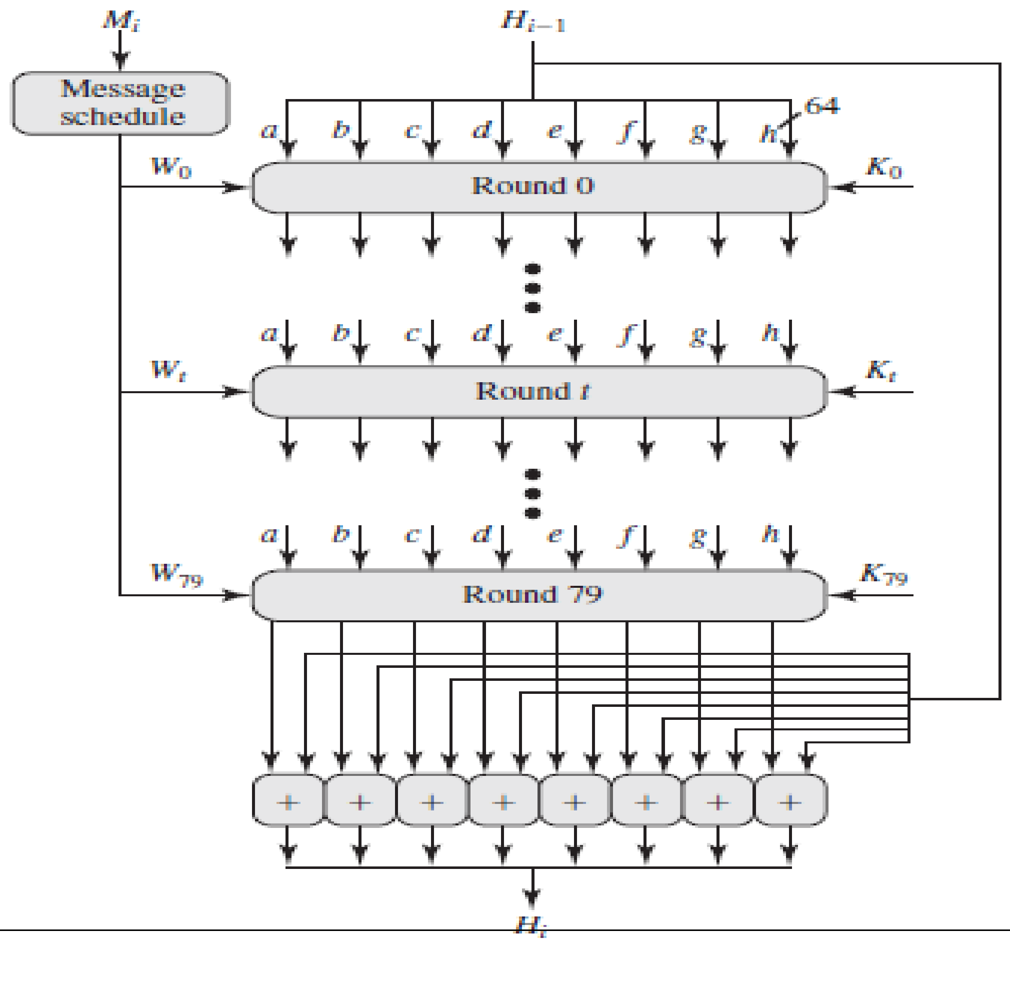
A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CD19137E2179

Step 4
Process message in 1024-bit (128-word) blocks.

The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F.
Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer.

At input to the first round, the buffer has the value of the intermediate hash value, Hi-1. Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi). These values are derived using a message schedule described subsequently.
Each round also makes use of an additive constant Kt, where 0 ≤ t ≤ 79 indicates one of the 80 rounds.
The output of the eightieth round is added to the input to the first round (Hi-1) to produce Hi. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in Hi-1, using addition modulo 264.

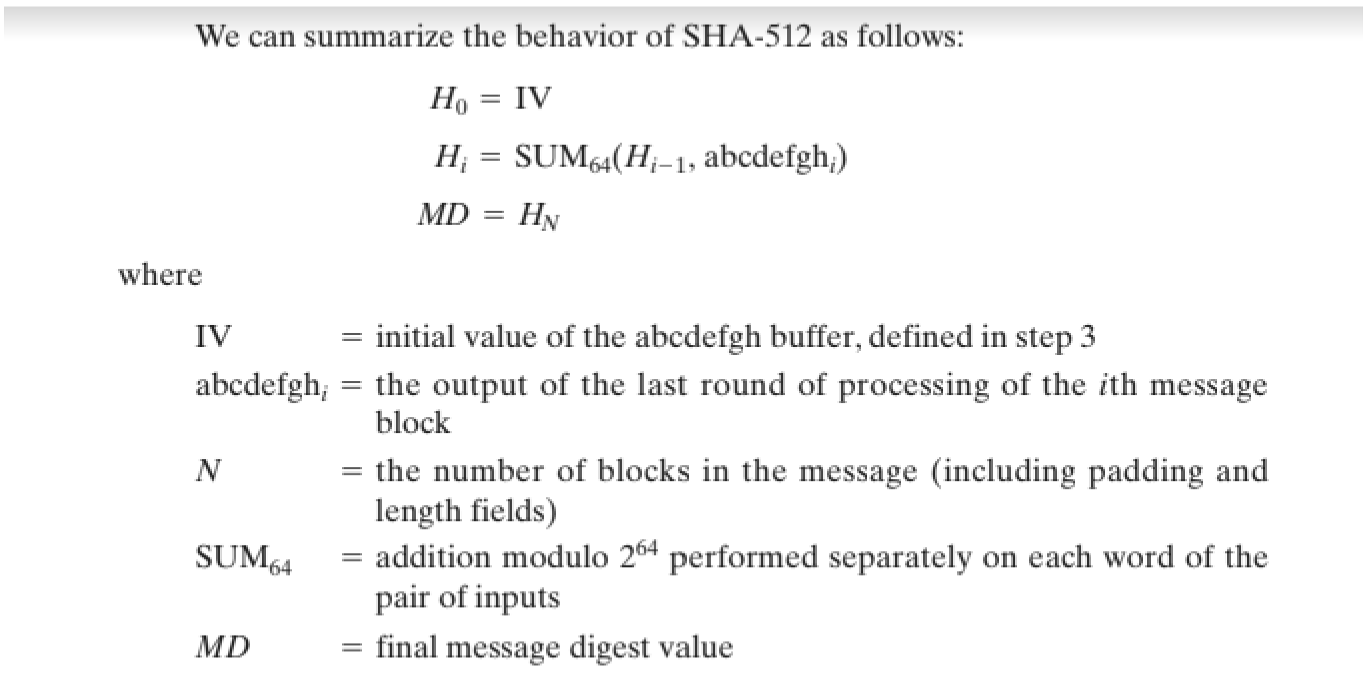


SHA-512 Processing of a Single 1024-Bit Block

Step 5 : Output

The following diagram illustrates how the 32bit word values wt are derived from the 1024 bit message.

After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.



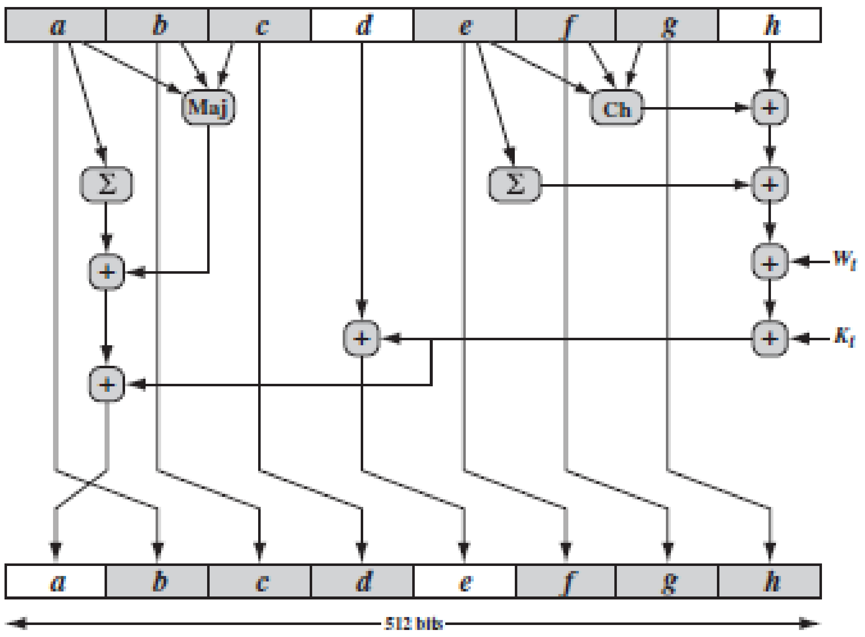
We can summarize the behavior of SHA-512 as follows:

$$H_0 = \mathrm{IV}$$
$$H_i = \mathrm{SUM}_{64}(H_{i-1}, \mathrm{abcdefgh}_i)$$
$$MD = H_N$$

where

IV	= initial value of the abcdefgh buffer, defined in step 3
$\mathrm{abcdefgh}_i$	= the output of the last round of processing of the ith message block
N	= the number of blocks in the message (including padding and length fields)
SUM_{64}	= addition modulo 2^{64} performed separately on each word of the pair of inputs
MD	= final message digest value

SHA-512 Round Function



Elementary SHA Operation (single step)

Let us look in more detail at the logic in each of the 80 steps of the processing of one 512-bit block. Each round is defined by the following set of
Equations:

.x

$$
\begin{aligned}
T_1 &= h + \mathrm{Ch}(e, f, g) + \left(\textstyle\sum_1^{512} e\right) + W_t + K_t \\
T_2 &= \left(\textstyle\sum_0^{512} a\right) + \mathrm{Maj}(a, b, c) \\
h &= g \\
g &= f \\
f &= e \\
e &= d + T_1 \\
d &= c \\
c &= b \\
b &= a \\
a &= T_1 + T_2
\end{aligned}
$$

where

t = step number; $0 \le t \le 79$

$\mathrm{Ch}(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$

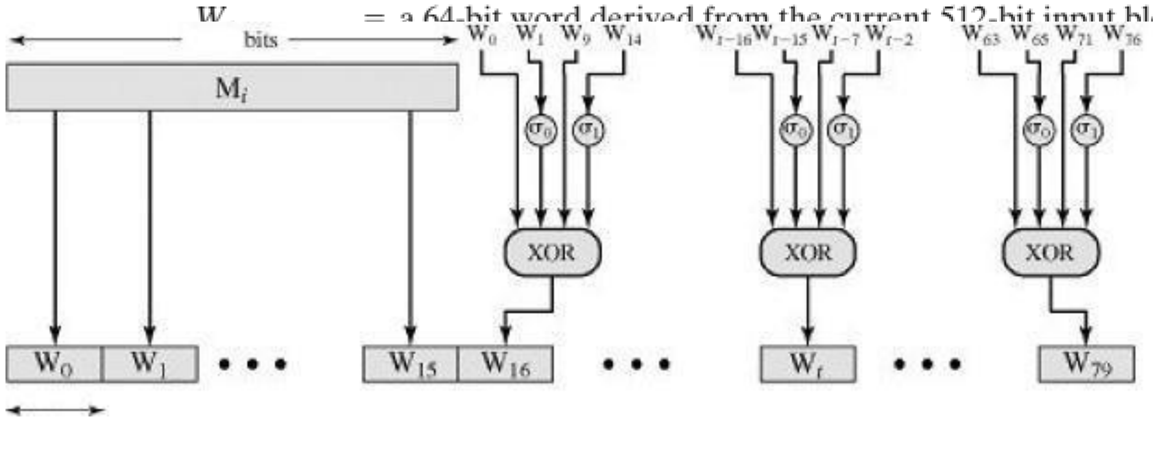
$\mathrm{Maj}(a, b, c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

The $\left(\sum_0^{512} a\right) = \mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right) = \mathrm{ROTR}^{14}(e) \oplus \mathrm{ROTR}^{18}(e) \oplus \mathrm{ROTR}^{41}(e)$

] $\mathrm{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

W_t = a 64-bit word derived from the current 512-bit input block



The first 16 values of are taken directly from the 16 words of the current block. The remaining values are defined as

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x)$$

$\mathrm{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$\mathrm{SHR}^n(x)$ = left shift of the 64-bit argument x by n bits with padding by zeros on the right

$+$ = addition modulo 2^{64}

Thus the SHA-512 algorithm has the property that every bit of the hash code is a function of every bit of the input.

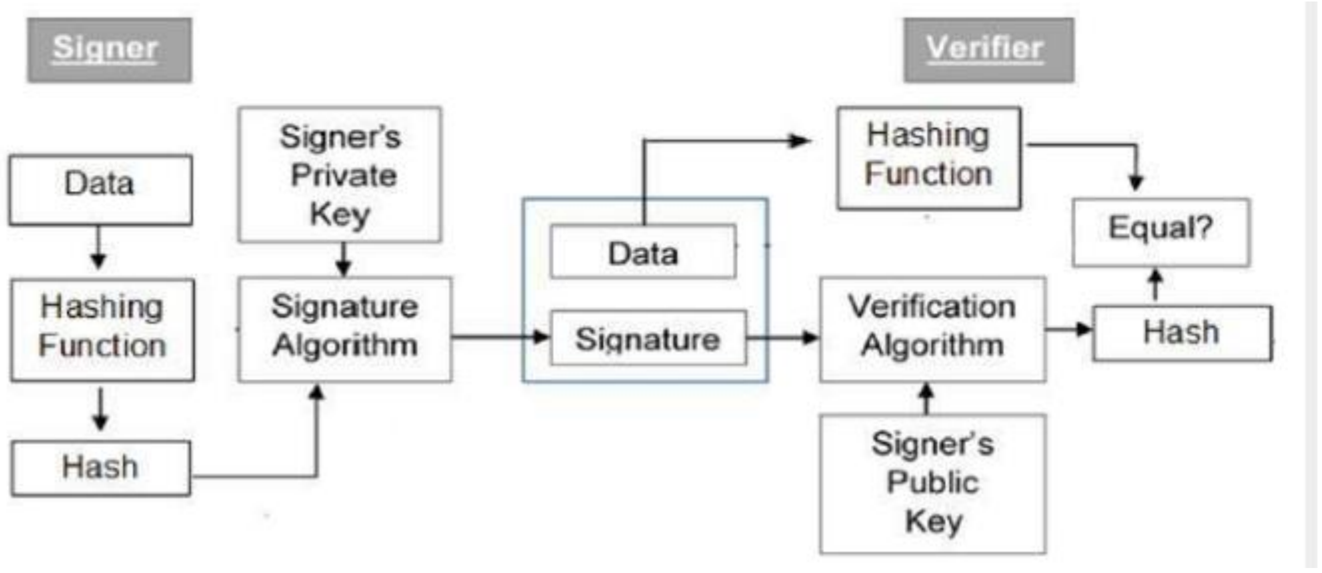
4.7 DIGITAL SIGNATURE AND AUTHENTICATION PROTOCOLS

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message.

Model of Digital Signature

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration −



The following points explain the entire process in detail −

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by ‘private’ key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data**.

Importance of Digital Signature

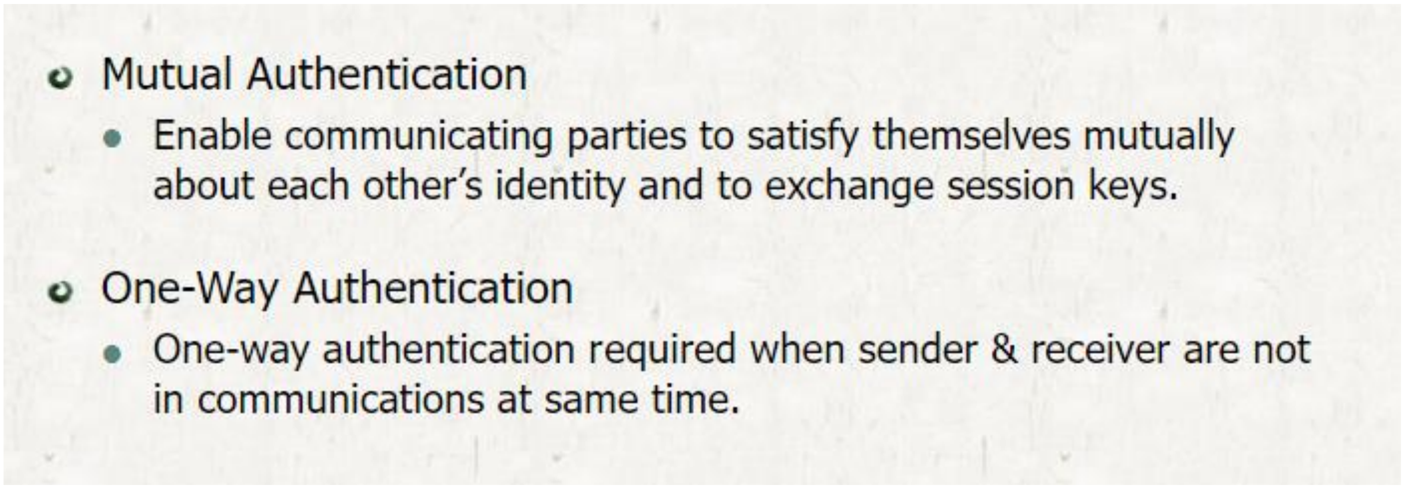
Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.

Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature –

- **Message authentication −** When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- **Data Integrity −** In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- **Non-repudiation −** Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely − Privacy, Authentication, Integrity, and Non-repudiation

Authentication Protocols



- Mutual Authentication
  - Enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.
- One-Way Authentication
  - One-way authentication required when sender & receiver are not in communications at same time.

4.8 DSS

The DSS Approach

The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

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.

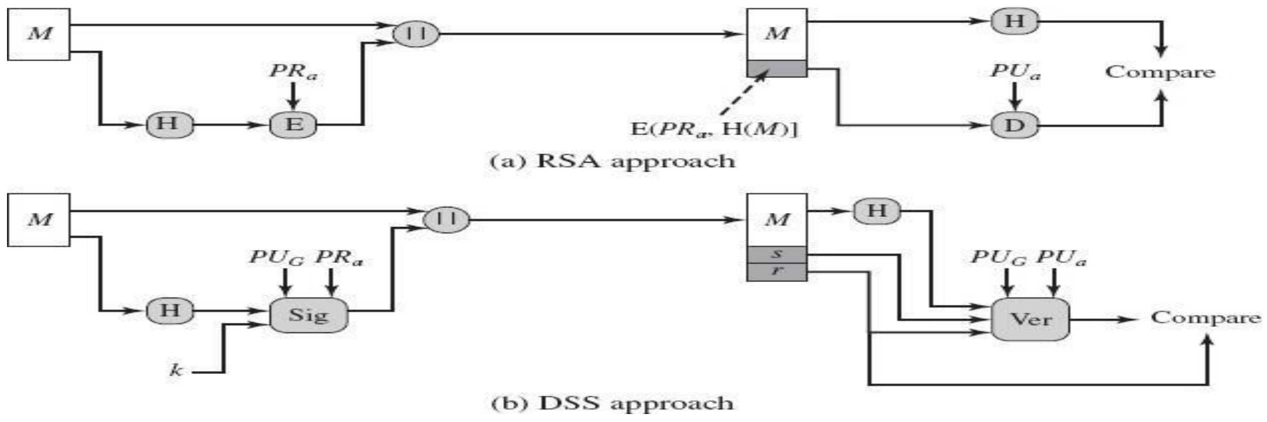
Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender‟s public key.

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.

DSS approach

The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number generated for this particular signature.

The signature function also depends on the sender‟s private key (*PRa*) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G). The result is a signature consisting of two components, labeled *s* and *r*.



Two Approaches to Digital Signatures

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, which is paired with the sender‟s private key.

The output of the verification function is a value that is equal to the signature component 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.

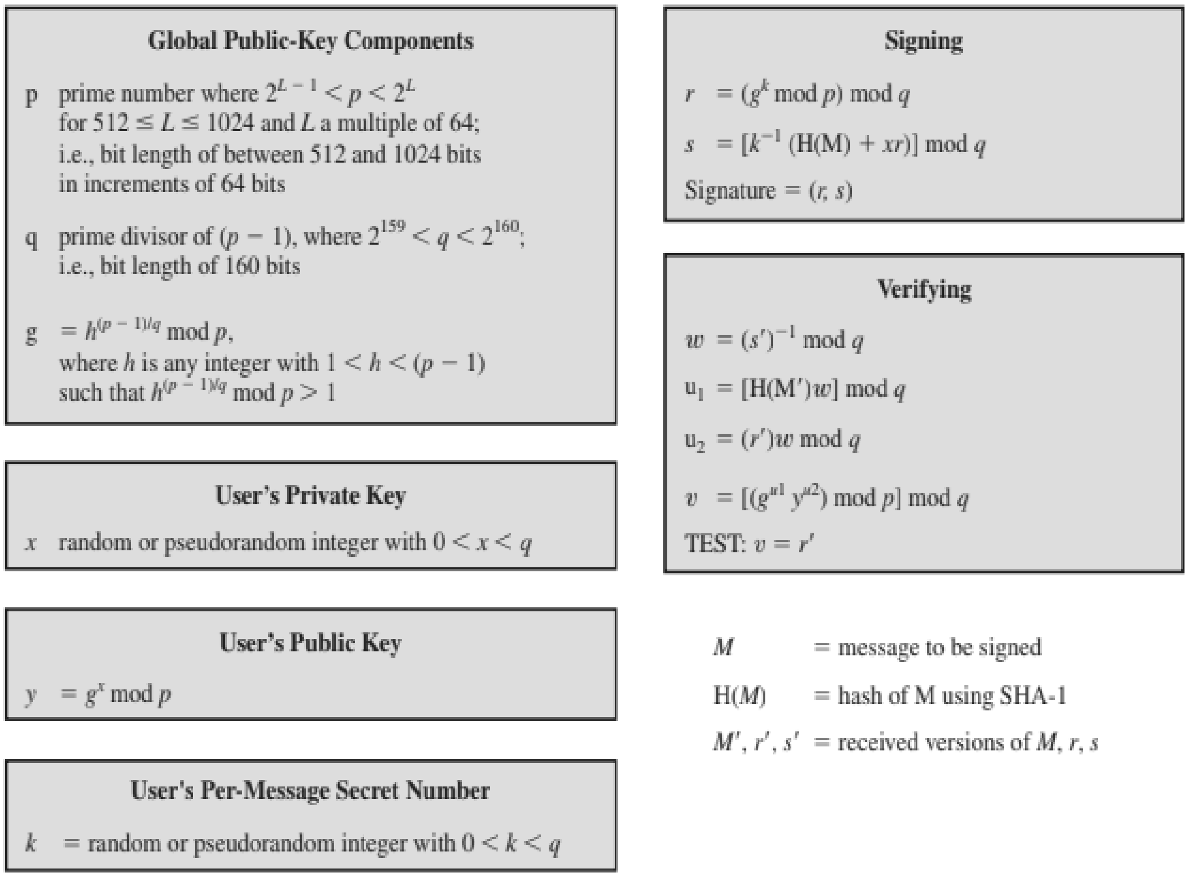
The Digital Signature Algorithm:

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 $h^{(p-1)/q}$mod *p*, where *h* is an integer between 1 and (p-1).

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. T

The public key is calculated from the private key as $y = g^x$ mod p . The calculation of given 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.



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 $2^{159} < q < 2^{160}$;
i.e., bit length of 160 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

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

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

Signature = (r, s)

Verifying

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

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

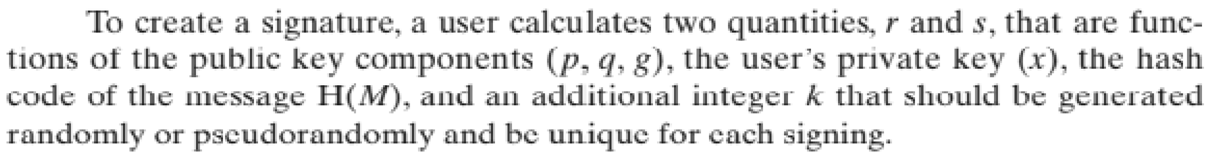
$u_2 = (r')w \bmod q$

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

TEST: $v = r'$

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

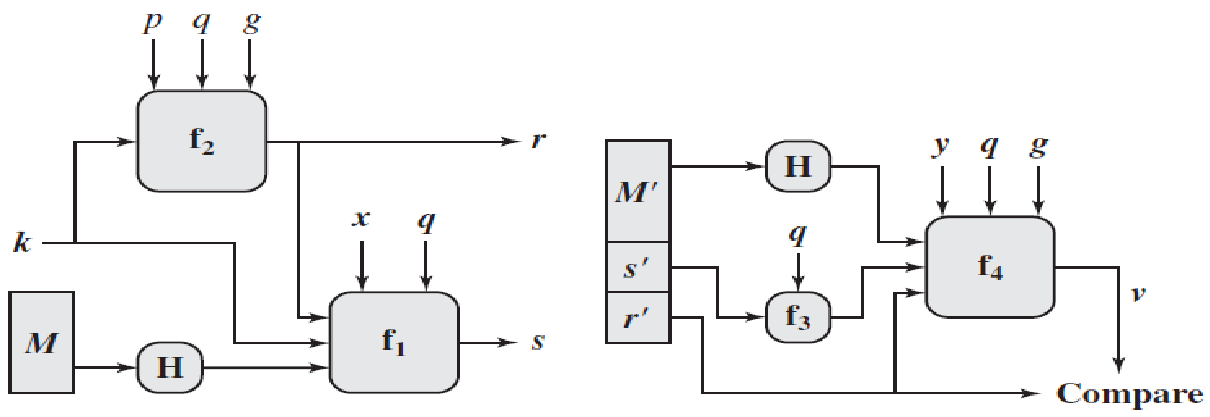
The Digital Signature Algorithm (DSA)



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 $\mathrm{H}(M)$, and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing.

At the receiving end, verification is performed using the formulas. 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 component of the signature, then the signature is validated.

DSS Signing and Verifying



The structure of this function is such that the receiver can recover using the incoming message and signature, the public key of the user, and the global public key. Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover k from r to recover x from s.

The only computationally demanding task in signature generation is the exponential calculation g^kmod p. Because this value does not depend on the message to be signed, it can be computed ahead of time.

4.9 ENTITY AUTHENTICATION

- Entity authentication is a technique designed to let one party prove the identity of another party.
- An entity can be a person, a process, a client, or a server.
- The entity whose identity needs to be proved is called the claimant;
- the party that tries to prove the identity of the claimant is called the verifier

Terminologies involved

- Verifier-The person who is in charge of checking that the correct entity is involved in communication is the Verifier. Verifier can also create some tokens by itself during communication which are used.
- Claimant-The person who wants to start communication by proving its identity is Claimant.
- Nonce–Time variant parameter which is served to distinguish one protocol instance from another like random number.
- Salt–Upon arrival of password we may add some bits upon initial entry. This t-bit random string is called “Salt”.

Entity Authentication Vs. Message Authentication

Message Authentication	Entity Authentication
This involves a meaningful message. This doesn‟t provide timeliness guaranteesas to when it was created etc.	This doesn‟t involve meaningful message; it involves some "claim" for proving its entity. Time is important, as in this protocol corroboration of a claimant‟s identity takes place.

Data-Origin Versus Entity Authentication

There are two differences between message authentication (data-origin authentication), and entity authentication.

1) Message authentication might not happen in real time; entity authentication does.

2) Message authentication simply authenticates one message; the process needs to be repeated for each new message. Entity authentication authenticates the claimant for the entire duration of a session.

Verification Categories

- **Something known -** This is a secret known only by the claimant that can be checked by the verifier. Examples are a password, a PIN, a secret key, and a private key.
- **Something possessed -** This is something that can prove the claimant's identity. Examples are a passport, a driver's license, an identification card, a credit card, and a smart card
- **Something inherent -** This is an inherent characteristic of the claimant. Examples are conventional signatures, fingerprints, voice, facial characteristics, retinal pattern, and handwriting.

4.10 PASSWORDS (Weak Authentication)

The simplest and oldest method of entity authentication is the password-based authentication, where the password is something that the claimant knows. A password is used when a user needs to access a system to use the system's resources (login). Each user has a user identification that is public, and a password that is private. We can divide these authentication schemes into two groups: the fixed password and the one-time password.

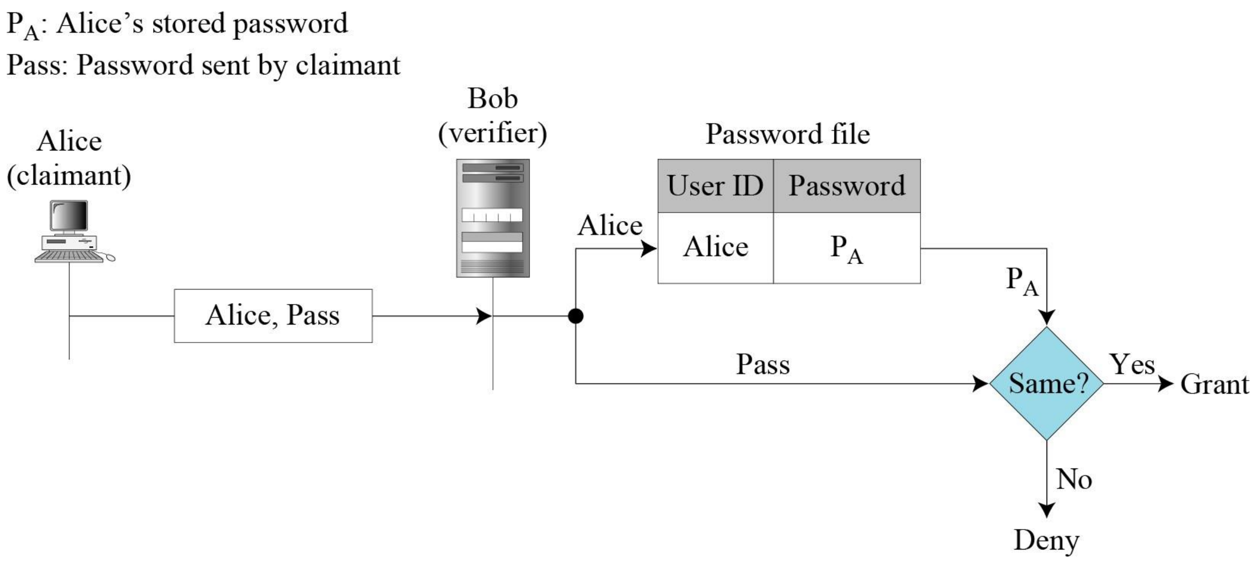
Fixed Password:

A fixed password is a password that is used over and over again for every access

First Approach User ID and password file

In the very rudimentary approach, the system keeps a table (a file) that is sorted by user identification. To access the system resources, the user sends her user identification and password, in plaintext, to the system. The system uses the identification to find the password in the table. If the password sent by the user matches the password in the table, access is granted; otherwise, it is denied.

Types of attack in first approach:
1. Eavesdropping
2. Stealing a password
3. Accessing a password file
4. Guessing



Second Approach Hashing the password

A more secure approach is to store the hash of the password (instead of the plaintext password) in the password file. Any user can read the contents of the file, but, because the hash function is a one-way function, it is almost impossible to guess the value of the password.

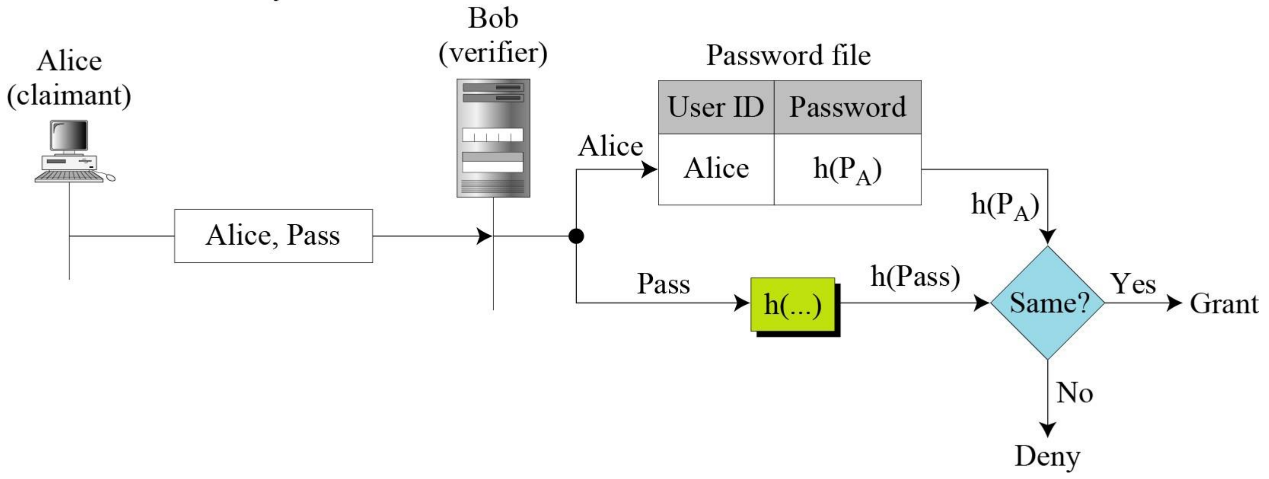
In the figure, when the password is created, the system hashes it and stores the hash in the password file. When the user sends the ID and the password, the system creates a hash of the password and then compares the hash value with the one stored in the file. If there is a match, the user is granted access; otherwise, access is denied. In this case, the file does not need to be read protected.

Types of attack in Second approach

1. Dictionary attack

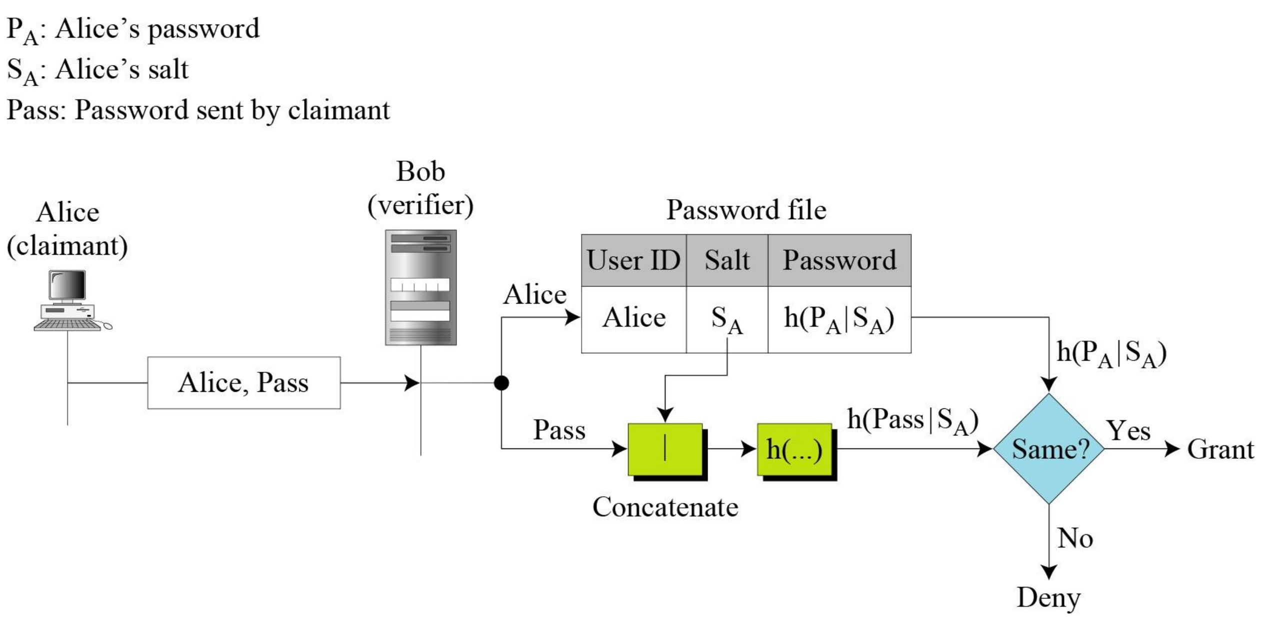
P_A: Alice’s stored password

Pass: Password sent by claimant



Third Approach Salting the password

The third approach is called salting the password. When the password string is created, a random string, called the salt, is concatenated to the password. The salted password is then hashed. The ID, the salt, and the hash are then stored in the file. Now, when a user asks for access, the system extracts the salt, concatenates it with the received password, makes a hash out of the result, and compares it with the hash stored in the file. If there is a match, access is granted; otherwise, it is denied. Salting makes dictinary attack more difficult.



Fourth Approach

In the fourth approach, two identification techniques are combined. A good example of this typeof authentication is the use of an ATM card with a PIN (personal identification number).

One Time Password

A one-time password is a password that is used only once. This kind of password makes eavesdropping and salting useless. Three approaches are discussed here

- First Approach In the first approach, the user and the system agree upon a list of passwords. Each password on the list can be used only once.
- In the second approach, the user and the system agree to sequentially update the password. The user and the system agree on an original password, P1, which is valid only for the first access. During the first access, the user generates a new password, P2, and encrypts this password with P1 as the key. P2 is the password for the second access. During the second access, the user generates a new password, P3, and encrypts it with P2; P3 is used for the third access. In other words, Pi is used to create Pi+1. Of course, if Eve can guess the first password (P1), she can find all of the subsequent ones.
- In the third approach, the user and the system create a sequentially updated password using a hash function

Advantages :

- It has better entropy than a short password.
- It is easier to remember than the usual passwords.

Disadvantage:

- This is really weak against attacks as intruder can hear over communication channel and impersonate it later.
- It is also very easy to replay the same message and use it later.
- Exhaustive search or password guessing i.e. dictionary attacks can also be used.
- One has to type extra.

4.11 CHALLENGE-RESPONSE (Strong Authentication)

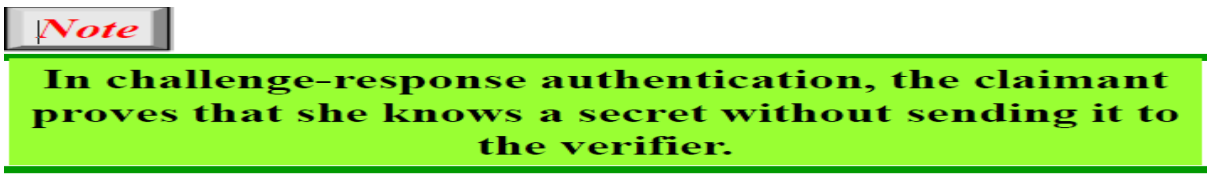
In password authentication, the claimant proves her identity by demonstrating that she knows a secret, the password.

In challenge-response authentication, the claimant proves that she knows a secret without sending it.

The central idea of challenge response is that claimant proves its identity to verifier by demonstrating knowledge of a secret known to be associated with entity without revealing the secret itself to the verifier during the protocol.

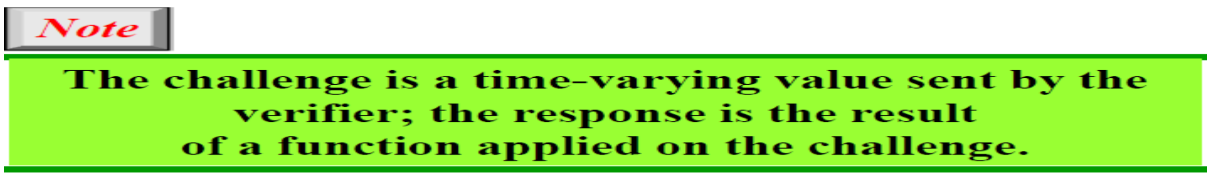
The challenge is usually time variant and is random number.

As every time the challenge is different, even if the adversary is monitoring the network it won‟t help as challenge changes every time.



Note

In challenge-response authentication, the claimant proves that she knows a secret without sending it to the verifier.



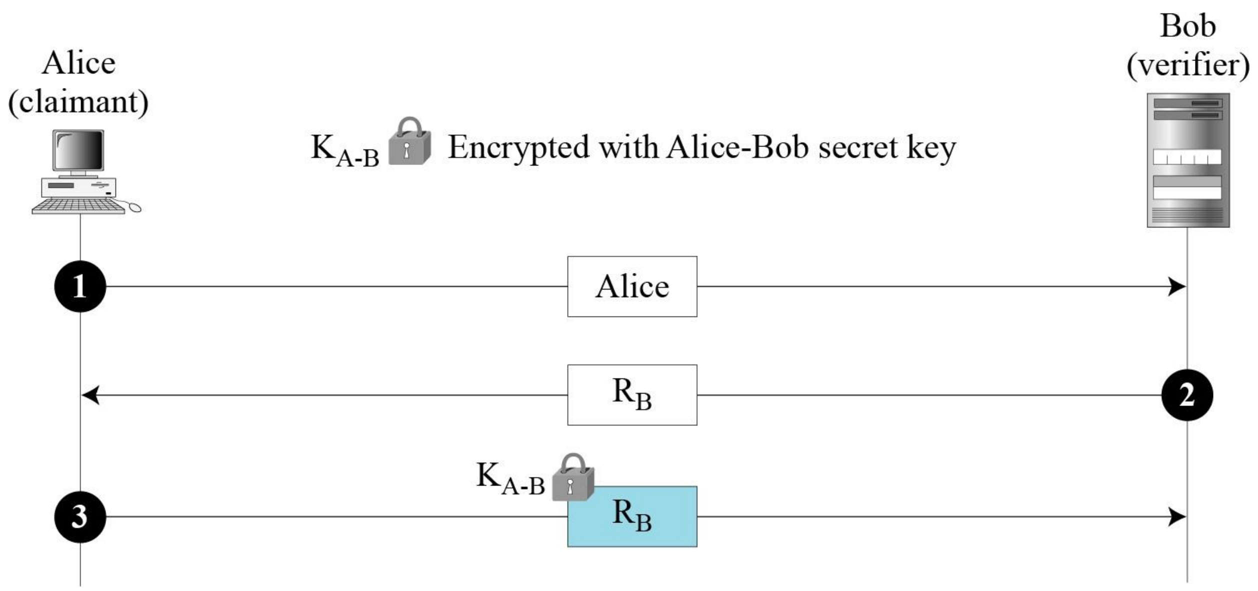
Note

The challenge is a time-varying value sent by the verifier; the response is the result of a function applied on the challenge.

1. Using a Symmetric-Key Cipher

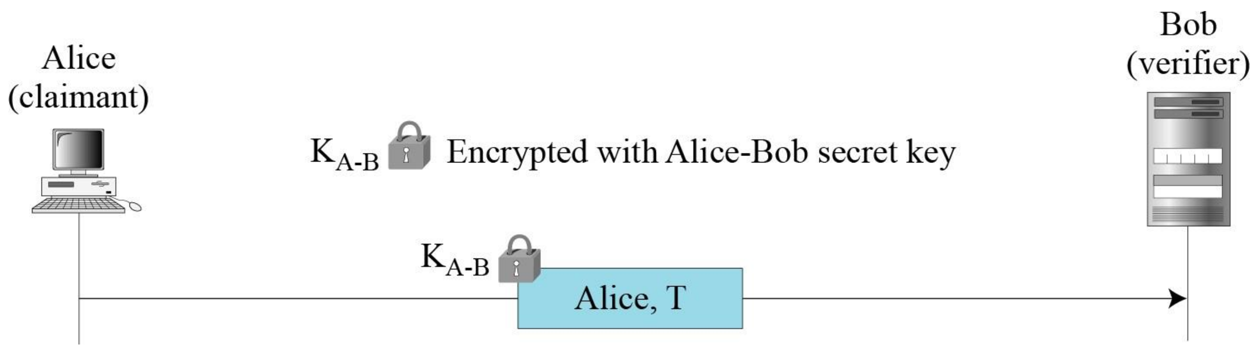
Several approaches to challenge-response authentication use symmetric-key encryption. The secret here is the shared secret key, known by both the claimant and the verifier. The function is the encrypting algorithm applied on the challenge.

First Approach



In the first approach, the verifier sends a nonce, a random number used only once, to challenge the claimant. A nonce must be time-varying; every time it is created, it is different. The claimant responds to the challenge using the secret key shared between the claimant and the

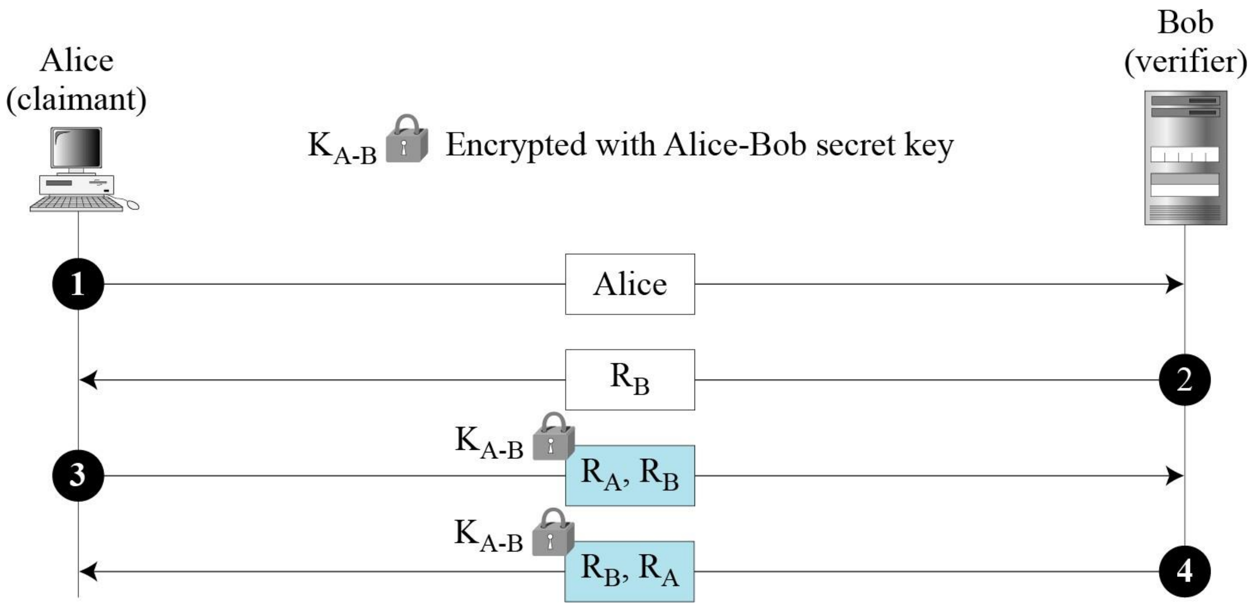
Second Approach



In the second approach, the time-varying value is a timestamp, which obviously changes with time. In this approach the challenge message is the current time sent from the verifier to the claimant. However, this supposes that the client and the server clocks are synchronized; the claimant knows the current time. This means that there is no need for the challenge message. The first and third messages can be combined. The result is that authentication can be done using one message, the response to an implicit challenge, the current time.

Third Approach

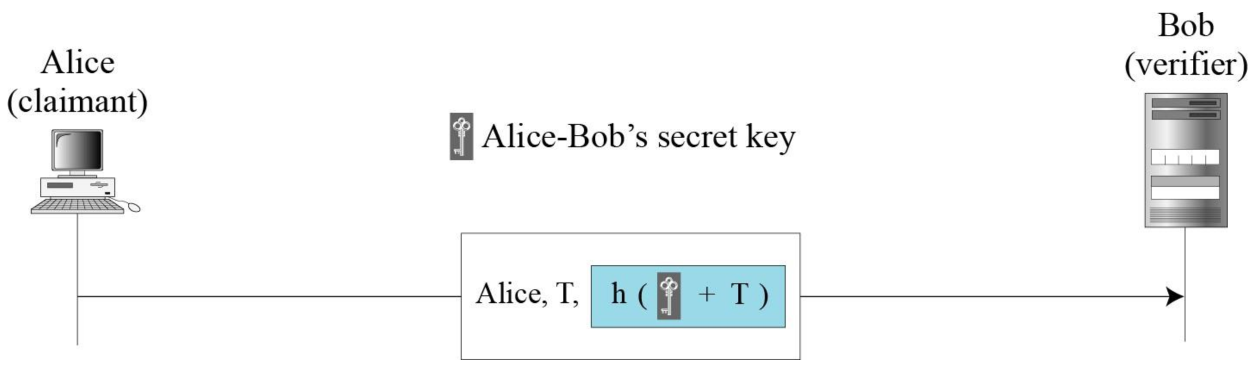
The first and second approaches are for unidirectional authentication. Alice is authenticated to Bob, but not the other way around. If Alice also needs to be sure about Bob's identity, we need bidirectional authentication.
The second message RB is the challenge from Bob to Alice. In the third message, Alice responds to Bob's challenge and at the same time, sends her challenge RA to Bob. The third message is Bob's response. Note that in the fourth message the order of RA and RB are switched to prevent a replay attack of the third message by an adversary.



2. Using Keyed-Hash Functions

Instead of using encryption/decryption for entity authentication, we can also use a keyed-hash function (MAC). One advantage to the scheme is that it preserves the integrity of challenge and response messages and at the same time uses a secret, the key.

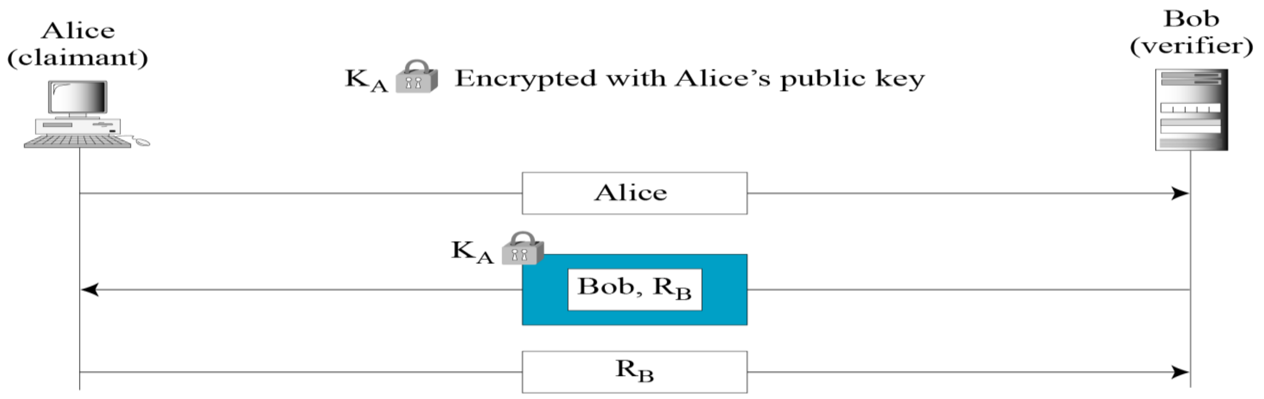
Keyed-hash function



3. Using an Asymmetric-Key Cipher

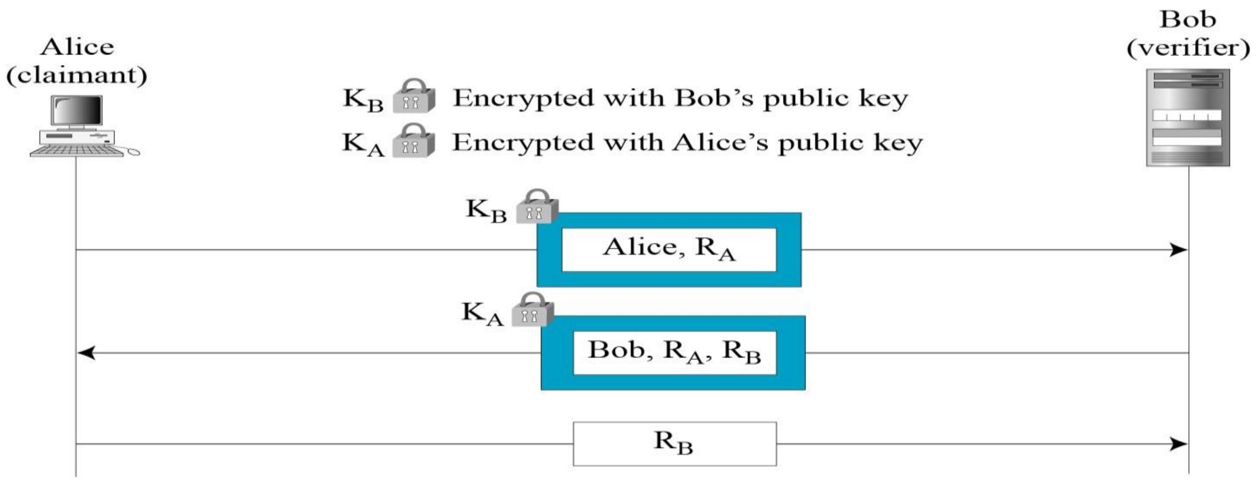
- Verifier encrypts the challenge with the Public key of the claimant.
- Claimant decrypts the challenge with her private key.
- Sends the challenge back to the verifier.

First Approach Unidirectional, asymmetric-key authentication



In the first approach, Bob encrypts the challenge using Alice's public key. Alice decrypts the message with her private key and sends the nonce to Bob.

Second Approach Bidirectional, asymmetric-key


Alice
(claimant)
Bob
(verifier)
K_B Encrypted with Bob's public key
K_A Encrypted with Alice's public key
K_B
Alice, R_A
K_A
Bob, R_A, R_B
R_B

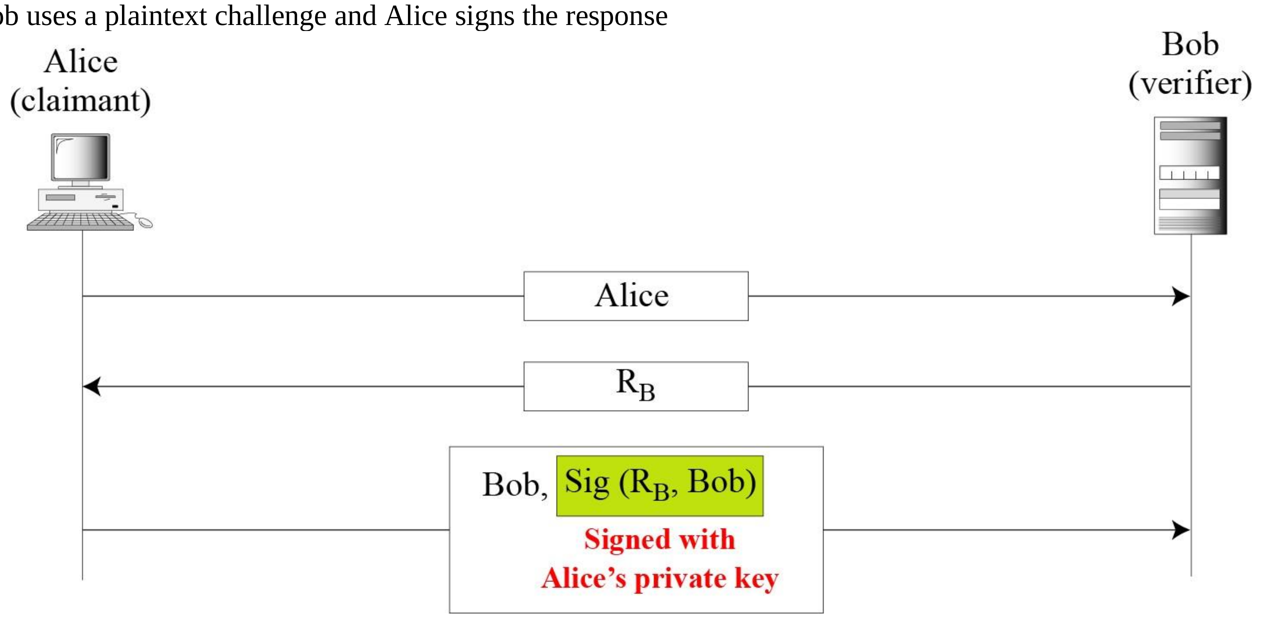
In the second approach, two public keys are used, one in each direction. Alice sends her identity and nonce encrypted with Bob's public key. Bob responds with his nonce encrypted with Alice's public key. Finally, Alice, responds with Bob's decrypted nonce.

4. Using Digital Signature

Entity authentication can also be achieved using a digital signature. When a digital signature is used for entity authentication, the claimant uses her private key for signing. Two approaches are shown here,

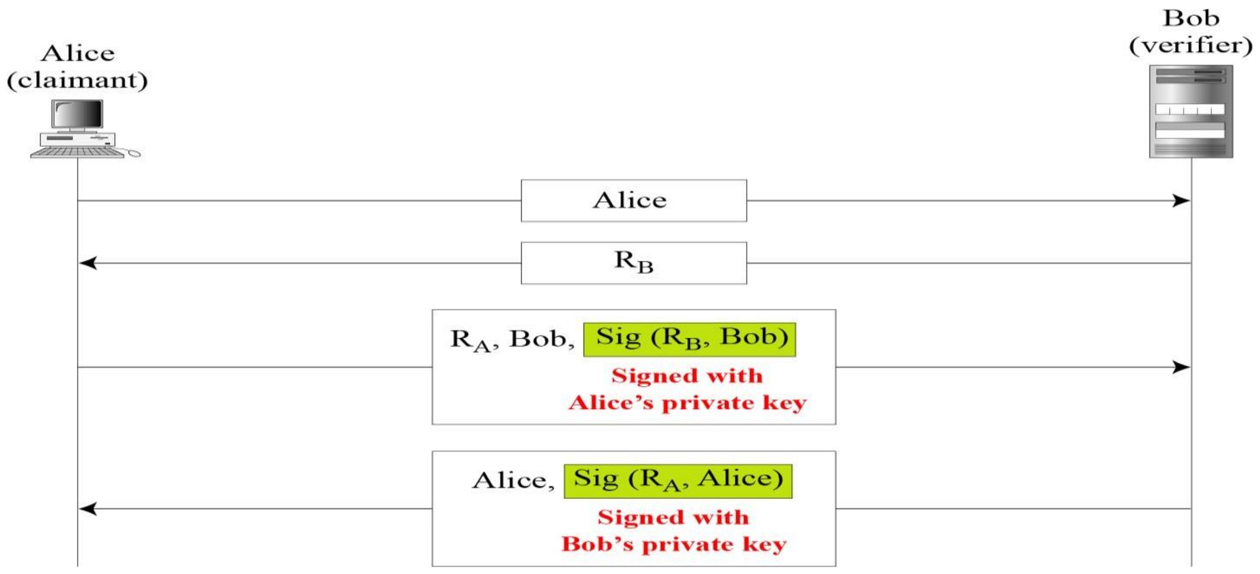
First Approach Digital signature, Unidirectional

Bob uses a plaintext challenge and Alice signs the response



Second Approach Digital signature, bidirectional authentication

Alice and Bob authenticate each other.



Advantages:

It is a time variant challenge where the response depends on entity‟ssecret and challenge.

- Even if the communication line is monitored, the response from one execution of identification protocol will not provide an adversary with useful information.

Disadvantages::

- This protocol still would reveal some partial information about the claimant‟s secret, an adversarial verifier might still be able to strategically select challenges to obtain responses providing information.

4.12 BIOMETRICS

- Biometrics is the measurement of physiological or behavioral features that identify a person (authentication by something inherent).
- Biometrics measures features that cannot be guessed, stolen, or shared.

Components

Several components are needed for biometrics, including capturing devices, processors, and storage devices.

Enrollment

Before using any biometric techniques for authentication, the corresponding feature of each person in the community should be available in the database. This is referred to as enrollment.

Authentication

Authentication done by two ways:

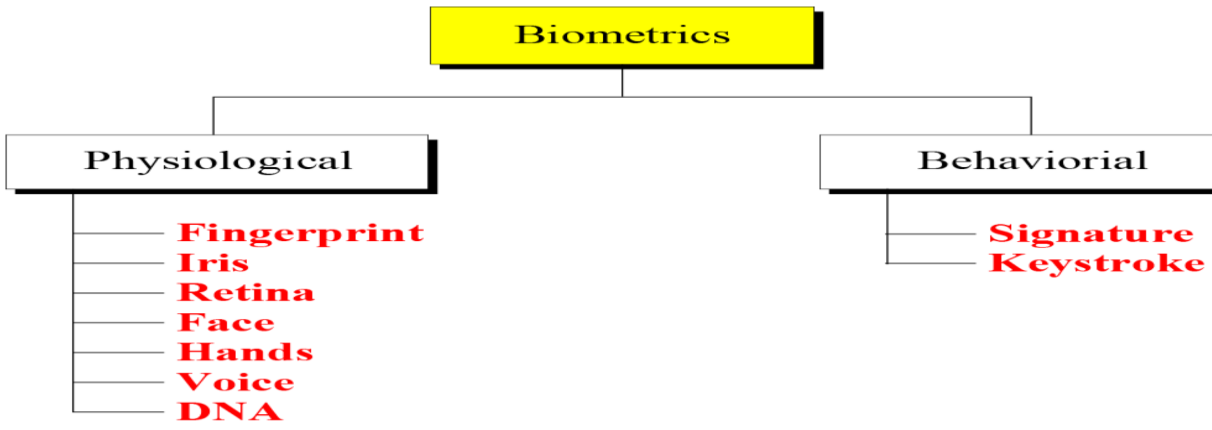
- Verification

  In verification, a person's feature is matched against a single record in the database (one-to-one matching) to find if she is who she is claiming to be. This is useful, for example, when a bank needs to verify a customer's signature on a check.

- Identification

  In identification, a person's feature is matched against all records in the database (oneto-many matching) to find if she has a record in the database. This is useful, for example, when a company needs to allow access to the building only to employees.

Techniques



Physiological Techniques

Physiological techniques measure the physical traits of the human body for verification and identification.

- Fingerprint

  Although there are several methods for measuring characteristics associated with fingerprints, the two most common are minutiae-based and image-based. In the minutiae-based technique, the system creates a graph based on where individual ridges start/stop or branch. In the image-based technique, the system creates an image of the fingertip and finds similarities to the image in the database. Fingerprints have been used for a long time. They show a high level of accuracy and support verification and identification. However, fingerprints can be altered by aging, injury, or diseases.

- Iris

  This technique measures the pattern within the iris that is unique for each person. It normally requires a laser beam (infrared). They are very accurate and stable over a person's life. They also support verification and identification. However, some eye diseases, such as cataracts, can alter the iris pattern.

- Retina

  The devices for this purpose examine the blood vessels in the back of the eyes. However, these devices are expensive and not common yet.

- Face

This technique analyzes the geometry of the face based on the distance between facial features such as the nose, mouth, and eyes. Some technologies combine geometric features with skin texture. Standard video cameras and this technique support both verification and identification. However, accuracy can be affected by eyeglasses, growing facial hair, and aging.

- Voice
  Voice recognition measures pitch, cadence, and tone in the voice. It can be used locally (microphone) or remotely (audio channel). This method is mostly used for verification. However, accuracy can be diminished by background noise, illness, or age.
- DNA

DNA is the chemical found in the nucleus of all cells of humans and most other organsims. The pattern is persistent throughout life and even after death. It is extremely accurate. It can be used for both verification and identification. The only problem is that identical twins may share the same DNA.

Behavioral Techniques

Behavioral techniques measure some human behavior traits

- Signature
  Biometric approaches use signature tablets and special pens to identify the person. These devices not only compare the final product, the signature, they also measure some other behavioral traits, such as the timing needed to write the signature. Signatures are mostly used for verification.
- Keystroke
  The keystrokes (typing rhythm) technique measures the behavior of a person related to working with a keyboard. It can measure the duration of key depression, the time between keystrokes, number and frequency of errors, the pressure on the keys, and so on. It is inexpensive because it does not require new equipment. However, it is not very accurate because the trait can change with time (people become faster or slower typists). It is also text dependent.

Applications

Several applications of biometrics are already in use. In commercial environments, these include access to facilities, access to information systems, transaction at point-ofsales, and employee timekeeping. In the law enforcement system, they include investigations (using fingerprints or DNA) and forensic analysis. Border control and immigration control also use some biometric techniques.

- In commercial environments, these include access to facilities, access to information systems, transaction at point-of-sales, and employee timekeeping.
- In the law enforcement system, they include investigations (using fingerprints or DNA) and forensic analysis.
- Border control and immigration control also use some biometric techniques. Applications

4.13 Authentication Application

Kerberos

Kerberos is a computer network authentication protocol which works on the basis of “tickets” to allow nodes communication over a non secure network.

(Or)

It is a secure method (service) for authenticate a request for a service in a computer network.
It was developed in “Athena Project” at MIT.
It provide authentication by using Secret-Key cryptography.

Use of Kerberos:

- Kerberos is used for decreasing the burden for server, means; Kerberos will take responsibility of authentication.
- It is designed for providing for strong authentication for client/server applications by using secret-key.

Versions:

Kerberos Version4
Kerberos Version5

Characteristics of KERBEROS:

1. It is secure: it never sends a password unless it is encrypted.
2. Only a single login is required per session. Credentials defined at login are then passed between resources without the need for additional logins.
3. The concept depends on a trusted third party – a Key Distribution Center (KDC). The KDC is aware of all systems in the network and is trusted by all of them.
4. It performs mutual authentication, where a client proves its identity to a server and a server proves its identity to the client.

Requirements Kerberos:

1. Secure: Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
2. Reliable: Kerberos should be highly reliable and should employ distributed server architecture with one system able to back up another.
3. Transparent: Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
4. Scalable: The system should be capable of supporting large numbers of clients and servers

Kerberos Version 4

Version 4 of Kerberos makes use of DES

A SIMPLE AUTHENTICATION DIALOGUE

- An authentication server (AS) is used in the simple authentication
- Authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner.

(1) $C \rightarrow AS: ID_C \| P_C \| ID_V$
(2) $A \rightarrow C$: Ticket
(3) $C \rightarrow V: ID_C \|$ Ticket
Ticket = $E(K_v, [ID_C \| AD_C \mid |ID_V])$

where
C = client
AS = authentication server
V = server
ID_C = identifier of user on C
ID_V = identifier of V
P_C = password of user on C
AD_C = network address of C
K_v = secret encryption key shared by AS and V

A More Secure Authentication Dialogue

- The main problem in A SIMPLE AUTHENTICATION DIALOGUE, the user must enter password for every individual service.
- Kerberos overcome this by using a new server, known as Ticket granting server (TGS).
- Now in Kerberos we have two servers; AS and TGS.

Once per user logon session:
(1) $C \rightarrow AS: ID_C \| IDt_{gs}$
(2) $AS \rightarrow C: E(K_c, Ticket_{tgs})$

Once per type of service:
(3) $C \rightarrow TGS: ID_C \| ID_V \| Ticket_{tgs}$
(4) $TGS \rightarrow C: Ticket_v$

Once per service session:
(5) $C \rightarrow V: ID_C \| Ticket_v$
$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime1])$
$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime2])$

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket (*$Ticket_{tgs}$*) from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived
from the user's password (K_c), which is already stored at the AS. When this
response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS (K_{tgs}) and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user’s ID and the servicegranting ticket. The server authenticates by using the contents of the ticket.

Authentication Service Exchange to obtain ticket-granting ticket

$$
\begin{array}{lll}
\textbf{(1)} & \mathbf{C \rightarrow AS} & ID_c \| ID_{tgs} \| TS_1 \\
\textbf{(2)} & \mathbf{AS \rightarrow C} & \mathrm{E}(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]) \\
 & & Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,tgs} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_{tgs} \| \mathrm{TS}_2 \| \mathrm{Lifetime}_2])
\end{array}
$$

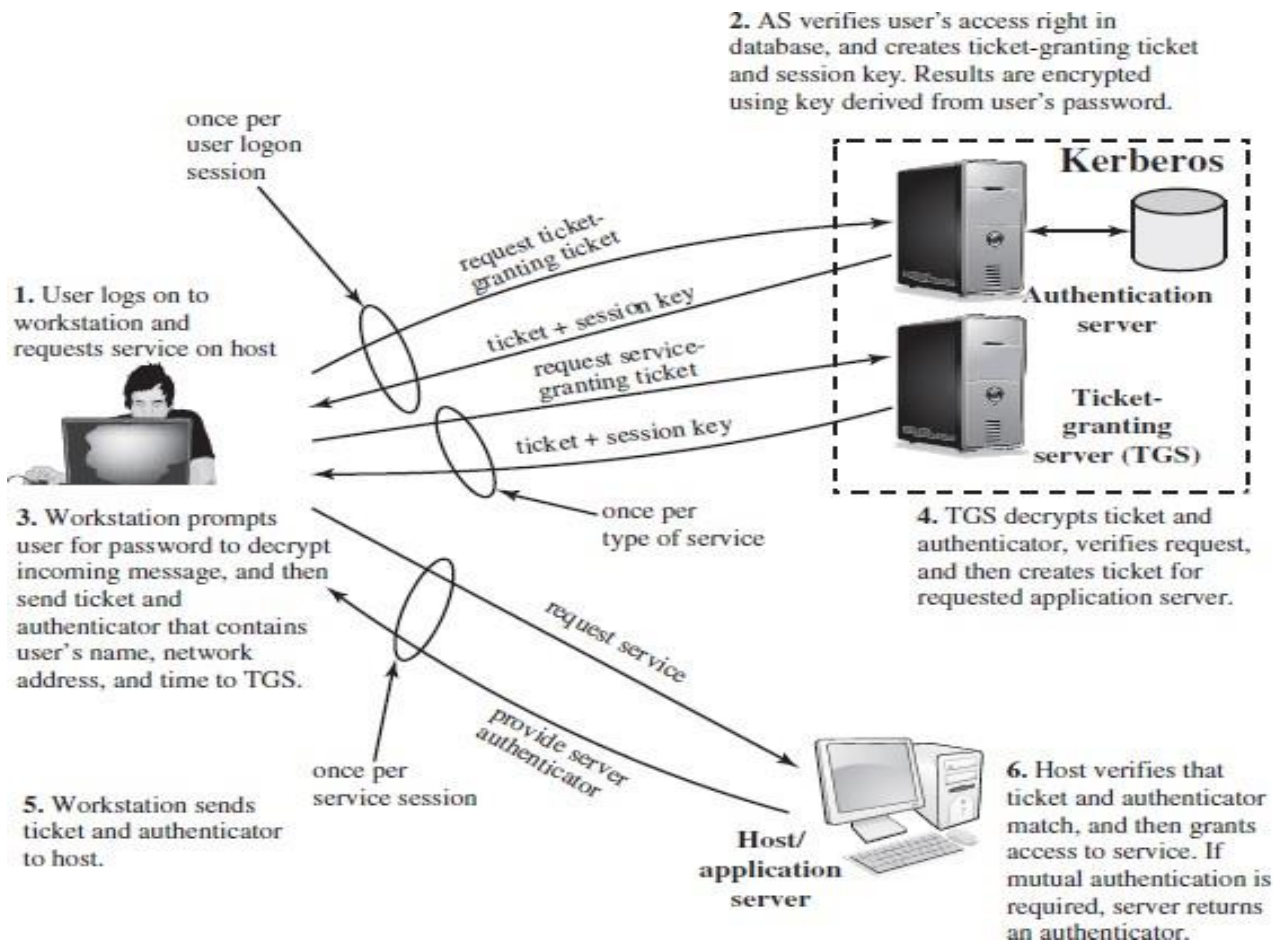
Ticket-Granting Service Exchange to obtain service-granting ticket

$$
\begin{array}{lll}
\textbf{(3)} & \mathbf{C \rightarrow TGS} & ID_v \| Ticket_{tgs} \| Authenticator_c \\
\textbf{(4)} & \mathbf{TGS \rightarrow C} & \mathrm{E}(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v]) \\
 & & Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,tgs} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_{tgs} \| \mathrm{TS}_2 \| \mathrm{Lifetime}_2]) \\
 & & Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_4 \| \mathrm{Lifetime}_4]) \\
 & & Authenticator_c = \mathrm{E}(\mathrm{K}_{c,tgs}, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{TS}_3])
\end{array}
$$

\

Client/Server Authentication Exchange to obtain service

$$
\begin{array}{lll}
\textbf{(5)} & \mathbf{C \rightarrow V} & Ticket_v \| Authenticator_c \\
\textbf{(6)} & \mathbf{V \rightarrow C} & \mathrm{E}(K_{c,v}, [TS_5 + 1]) \text{ (for mutual authentication)} \\
 & & Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,v} \| \mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_4 \| \mathrm{Lifetime}_4]) \\
 & & Authenticator_c = \mathrm{E}\,(\mathrm{K}_{c,v}, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{TS}_5])
\end{array}
$$

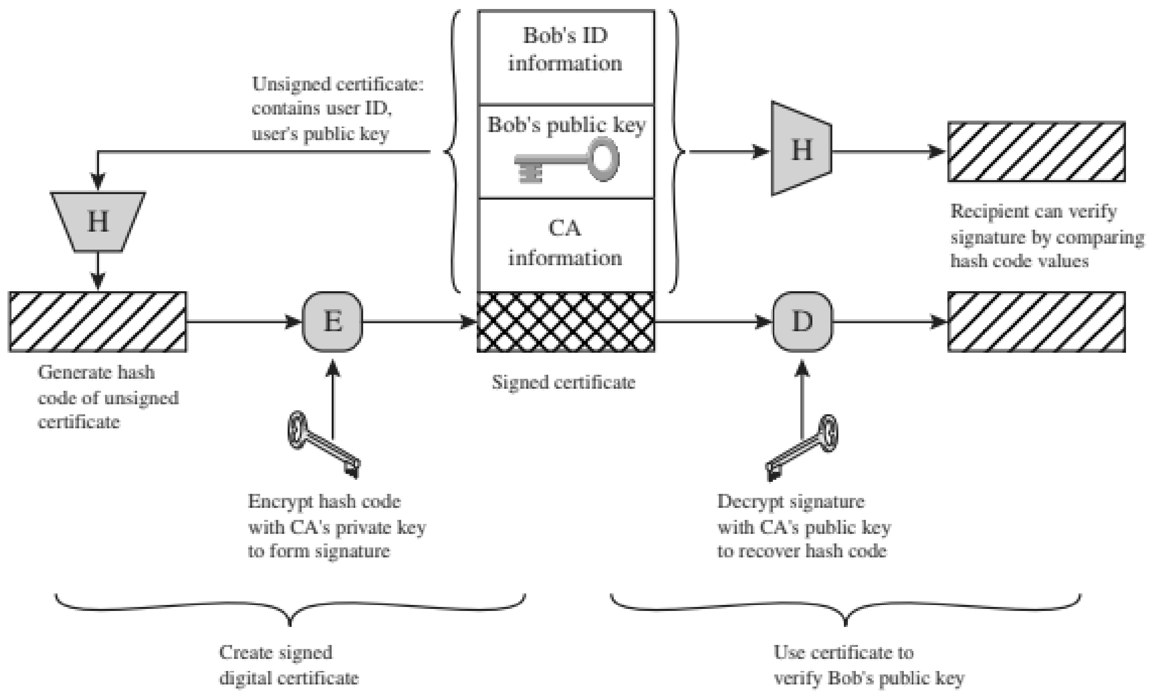

2. AS verifies user's access right in database, and creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
once per user logon session
Kerberos
request ticket-granting ticket
ticket + session key
Authentication server
1. User logs on to workstation and requests service on host
request service-granting ticket
ticket + session key
Ticket-granting server (TGS)
3. Workstation prompts user for password to decrypt incoming message, and then send ticket and authenticator that contains user's name, network address, and time to TGS.
once per type of service
4. TGS decrypts ticket and authenticator, verifies request, and then creates ticket for requested application server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to host.
Host/ application server
6. Host verifies that ticket and authenticator match, and then grants access to service. If mutual authentication is required, server returns an authenticator.

4.14 X.509 certificate

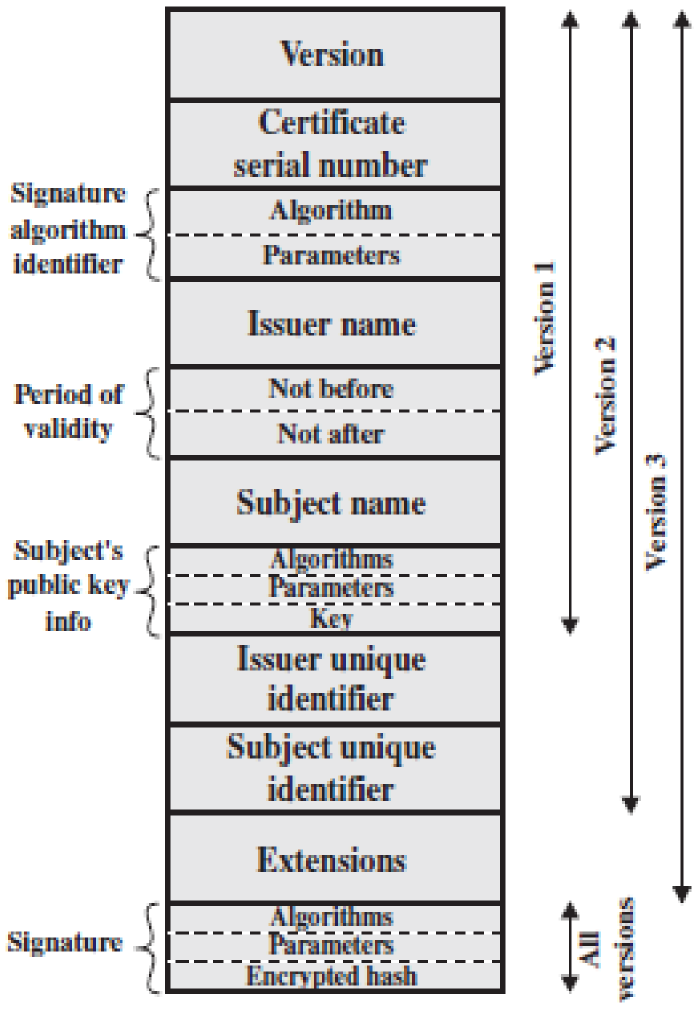
X.509 is based on the use of public-key cryptography and digital signatures. The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

1. Certificates

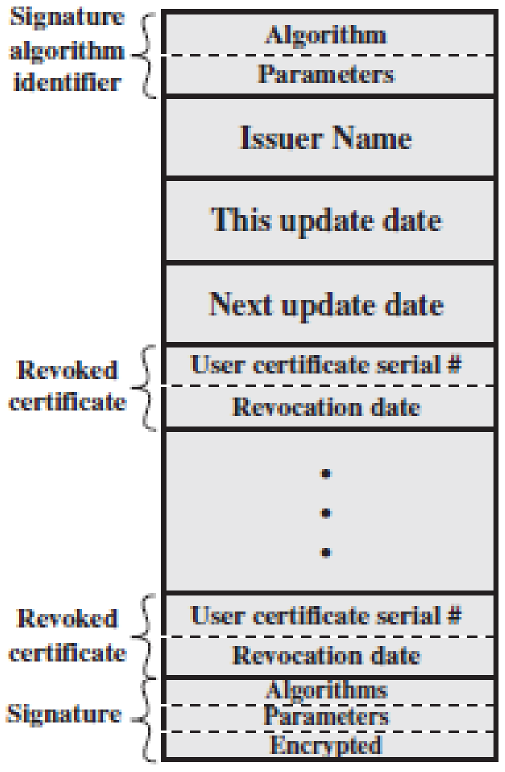
Public Key -Certificate Use


Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
H
Recipient can verify signature by comparing hash code values
E
D
Generate hash code of unsigned certificate
Signed certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

General format of Certificates , which include the following details:



(a) X.509 certificate



(b) Certificate revocation list

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the issuer unique identifier or subject unique identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
- **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- **Signature algorithm identifier:** The algorithm used to sign the certificate
Together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- **Issuer name:** X.500 name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

• **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
• **Subject unique identifier:** An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
• **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.

• **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

The standard use the following notations to define a certificate:

$$\mathbf{CA \ll A \gg = CA\ \{V, SN, AI, CA, UCA, A, UA, Ap, T^{A}\}}$$

where
Y <<X>> = the certificate of user X issued by certification authority Y
Y {I} = the signing of I by Y. It consists of I with an encrypted hash code appended
V = version of the certificate
SN = serial number of the certificate
AI = identifier of the algorithm used to sign the certificate
CA = name of certificate authority
UCA = optional unique identifier of the CA
A = name of user A
UA = optional unique identifier of the user A
Ap = public key of user A
T^{A} = period of validity of the certificate

2. Obtaining a user's certificate

User certificates generated by a CA have the following characteristics:
• Any user with access to the public key of the CA can verify the user public key that was certified.
• No party other than the certification authority can modify the certificate without this being detected.

A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2. If A does not securely know the public
key of X_2, then B's certificate, issued by X_2, is useless to A. A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.

Step 1 A obtains from the directory the certificate of X_2 signed by X_1. Because A securely knows X_1's public key, A can obtain X_2's public key from its certificate and verify it by means of X_1's signature on the certificate.
Step 2 A then goes back to the directory and obtains the certificate of B signed by X_2. Because A now has a trusted copy of X_2's public key, A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

$$X1 << X2 >> X2 << B >>$$

In the same fashion, B can obtain A's public key with the reverse chain:

$$X2 << X1 >> X1 << A >>$$

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with *N* elements would be expressedas

$$X_1 << X_2 >> X_2 << X_3 >> .....X_N << B >>$$

In this case, each pair of CAs in the chain (X*i*, X*i*+1) must have created certificates for each other.

All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's publickey certificate.
X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.

• **Forward certificates:** Certificates of X generated by other CAs
• **Reverse certificates:** Certificates generated by X that are the certificates of other CAs

3. Revocation of Certificates:

In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons:

1. The user's private key is assumed to be compromised.

2. The user is no longer certified by this CA. Reasons for this include that the subject's name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.

3. The CA's certificate is assumed to be compromised.

Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs.

These lists should also be posted on the directory.

Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate.

Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.

When a user receives a certificate in a message, the user must determine whether the certificate has been revoked. The user could check the directory each time a certificate is received. To avoid the delays (and possible costs) associated with directory searches, it is likely that the user would maintain a local cache of certificates and lists of revoked certificates.