UNIT 4
MESSAGE AUTHENTICATION AND INTEGRITY

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.

Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require 2(k-1) attempts for a k-bit key.

If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the MAC size. Then, given a known M_1 and MAC_1, with $MAC_1 = C_K(M_1)$, the cryptanalyst can perform $MAC_i = CK_i$ (M1) for all possible key values Ki.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.

Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

Round 1

Given: M_1, $MAC_1 = C_K(M_1)$
Compute $MAC_i = C_{Ki}(M_1)$ for all 2^k keys
Number of matches $\approx 2^{(k-n)}$

Round 2

Given: M_2, $MAC_2 = C_K(M_2)$
Compute $MAC_i = C_{Ki}(M_2)$ for the $2^{(k-n)}$ keys resulting from Round
1 Number of matches $\approx 2^{(k-2xn)}$ and so on

Consider the following MAC algorithm. Let $M = (X_1||X_2||...||X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i. Then define

$$\Delta(M) = X_1 + X_2 \ldots X_m$$
$$C_k(M) = E_k(\Delta(M))$$

Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 2^{56} encryptions.

But the opponent can attack the system by replacing X1 through X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows:

$$Y_m = Y_1 + \ Y_2 \ldots\ldots Y_{m1} + \ \Delta(M)$$

The opponent can now concatenate the new message, which consists of Y_1 through Y_m, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 $X_{(m-1)}$ bits can be fraudulently inserted.

Then the MAC function should satisfy the following requirements: The MAC function should have the following properties:

- If an opponent observes M and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message M‟ such that $C_K(M‟) = C_K(M)$
- $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M‟, the probability that $C_K(M) = C_K(M‟)$ is 2^{-n} where n is the number of bits in the MAC.
- Let M‟ be equal to some known transformation on M. i.e., M‟ = f(M).

MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES.

The algorithm(Fig 2) can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: $D_1, D_2 \ldots D_n$. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code (DAC) is calculated as follows:

$$O_1 = E_K(D_1)$$
$$O_2 = E_K(D_2 + O_1)$$
$$O_3 = E_K(D_3 + O_2)$$
$$\cdots\cdots$$
$$O_N = E_K(D_N + O_{N-1})$$

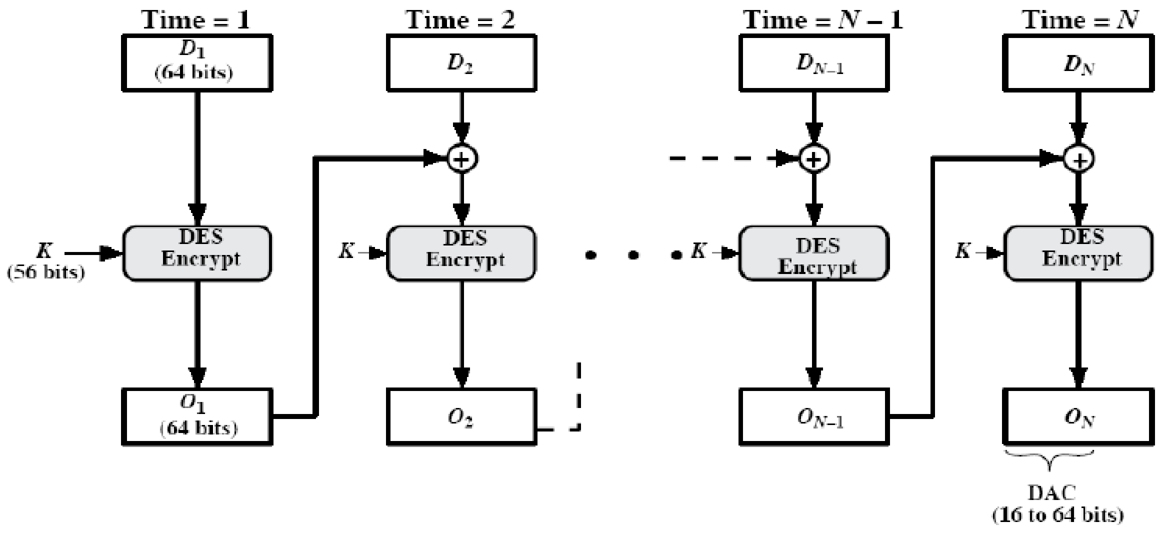


Figure.2 Data Authentication Algorithm

4.3.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:

In Fig (a) 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.

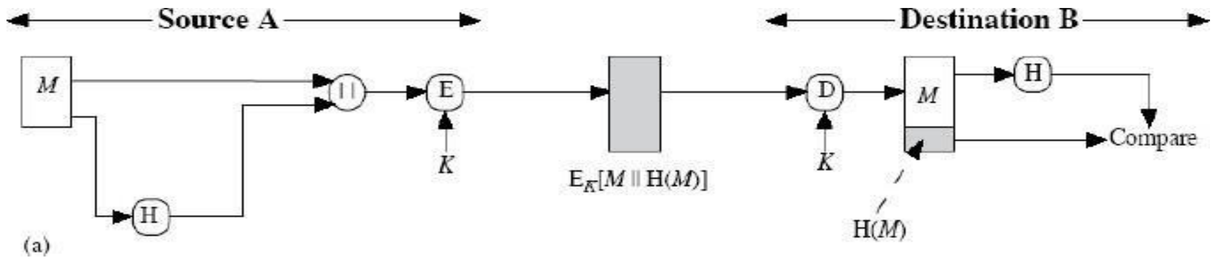
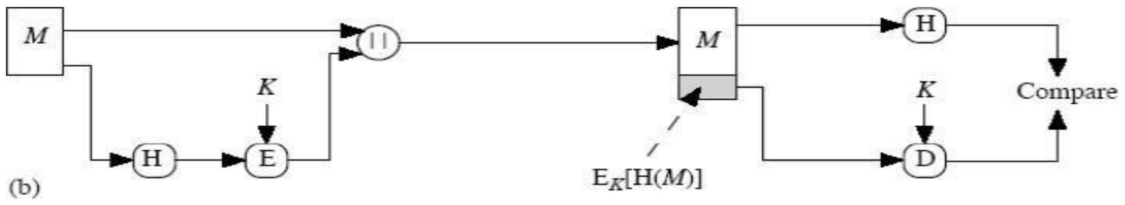


Figure (a) Hash Function

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



In Fig (c) 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.

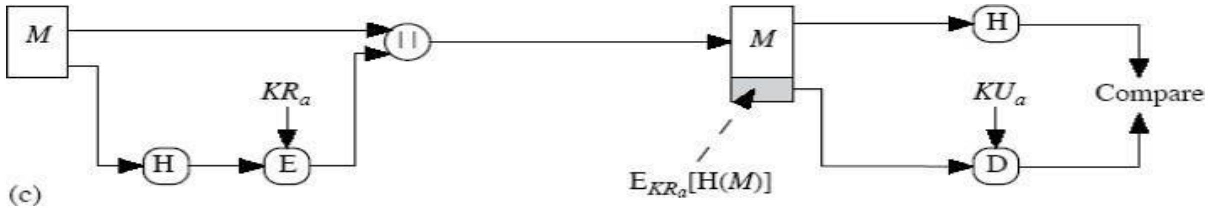
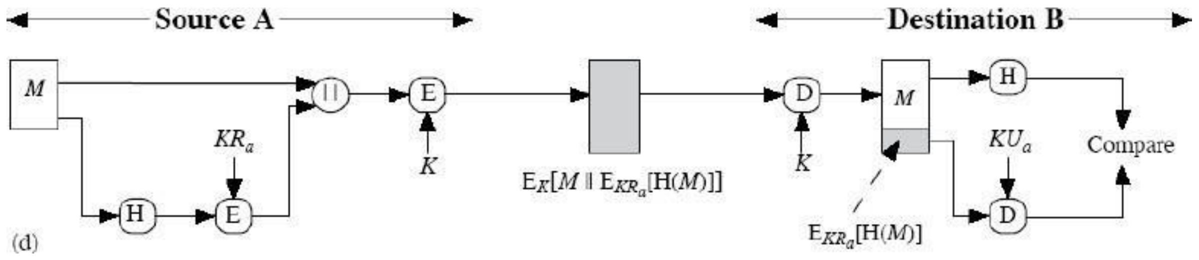
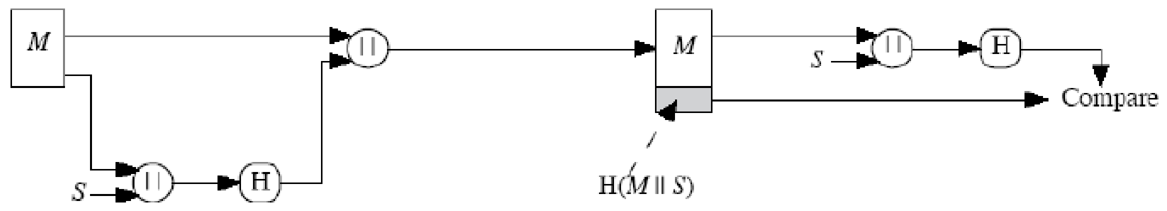


Figure (b & c) Basic use of Hash Function

In Fig (d) 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.



In Fig (e) 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.



In Fig(f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

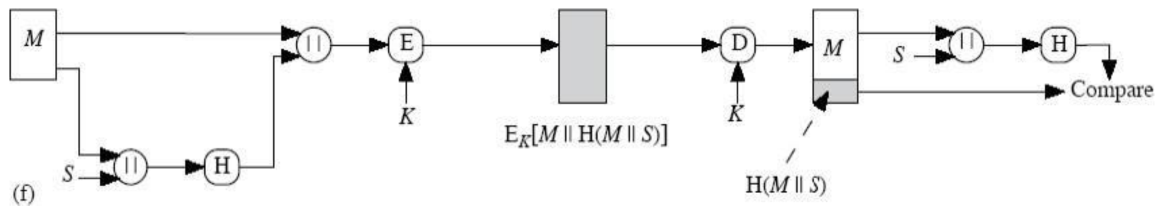


Figure (d,e & f) Basic use of Hash Function

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: $C_i = b_{i1} + b_{i1} \ldots + b_{im}$

where

C_i = i^{th} 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:

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

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.