Message Authentication and Hash Functions
Authentication Requirements

• following attacks can be identified:

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

2. Traffic analysis:
   – Discovery of the pattern of traffic between parties.

3. Masquerade:
   – Insertion of messages into the network from a fraudulent source.

4. Content modification:
   – Changes to the contents of a message.
5. **Sequence modification:**
   - Any modification to a sequence of messages between parties.

6. **Timing modification:**
   - Delay or replay of messages.

7. **Source repudiation:**
   - Denial of transmission of message by source.

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

• means to verify
  – received from alleged source
  – not been altered
  – verify sequencing and timeliness
Authentication Functions

• has two levels of functionality
  – at low level
    • some function that produces an authenticator
  – at higher level
    • enables a receiver to verify authenticity of a message
Types of authentication function

• **Message encryption**
  – The ciphertext of the entire message serves as its authenticator

• **Message authentication code (MAC):**
  – A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

• **Hash function:**
  – A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator
Symmetric Encryption

(a) Symmetric encryption: confidentiality and authentication
FCS (Frame check sequence)
FCS (Frame check sequence)
Public-key Encryption

(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature

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

• technique that use of a secret key to generate a small fixed-size block of data
• assumes that two communicating parties, say A and B, share a common secret key $K$.
• when A has a message to send to B
  – $\text{MAC} = C(K, M)$
    • $M =$ input message
    • $C =$ MAC function
    • $K =$ shared secret key
    • $\text{MAC} =$ message authentication code
(a) Message authentication

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

(c) Message authentication and confidentiality; authentication tied to ciphertext
only the receiver and the sender know the secret key, and if the received MAC matches the calculated MAC, then

– 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 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.

– If the message includes a sequence number, then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.
Uses of MAC

• same message is broadcast to a number of destinations
  – message must be broadcast in plaintext with an associated message authentication code
• one side has a heavy load and cannot afford the time to decrypt all incoming messages
• Authentication of a computer program in plaintext
Hash function

• accepts a variable-size message \( M \) as input and produces a fixed-size output
  – hash code \( H(M) \)
• does not use a key but is a function only of the input message
• A change to any bit or bits in the message results in a change to the hash code.
a. Because only A and B share the secret key, the message must have come from A and has not been altered

b. for those applications that do not require confidentiality.

c. Only the hash code is encrypted, using public-key encryption and using the sender's private key, provides digital signature
d. If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key

e. use a hash function but no encryption for message authentication

f. Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code
Requirements for MAC

• $k > n$; that is, suppose that the key size is greater than the MAC size
• total of $2^k$ MACs will be produced
  – but there are only $2^n$ different MAC values
• total of $2^k / 2^n = 2^{(k-n)}$ keys will produce a match
Requirements for MAC

- in assessing the security of a MAC function, we need to consider the types of attacks
- Assume that an opponent knows the MAC function \( C \) but does not know \( K \).
- MAC function should satisfy the three requirements:
• 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. That is, $M' = f(M)$. For example, f may involve inverting one or more specific bits. In that case, $Pr[C_K(M) = C_K(M')] = 2^{-n}$
Message Authentication Code Based on DES
Requirements for a Hash functions

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 referred to 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 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 referred to as strong collision resistance.
Simple Hash Functions

• The input 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.

• simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block

• $C_i = b_{i1} \oplus b_{i2} \oplus ... \oplus b_{im}$
**Simple Hash function using bitwise XOR**

<table>
<thead>
<tr>
<th>Block</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>...</th>
<th>Bit n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>$b_{11}$</td>
<td>$b_{21}$</td>
<td></td>
<td>$b_{n1}$</td>
</tr>
<tr>
<td>Block 2</td>
<td>$b_{12}$</td>
<td>$b_{22}$</td>
<td></td>
<td>$b_{n2}$</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>Block m</td>
<td>$b_{1m}$</td>
<td>$b_{2m}$</td>
<td></td>
<td>$b_{nm}$</td>
</tr>
<tr>
<td>Hash code</td>
<td>$C_1$</td>
<td>$C_2$</td>
<td></td>
<td>$C_n$</td>
</tr>
</tbody>
</table>
• perform a one-bit circular shift, or rotation, on the hash value after each block is processed

1. Initially set the \emph{n-bit hash value to zero}.
2. Process each successive \emph{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.
XOR with 1-bit rotation to the right

XOR of every 16-bit block
• simple XOR applied to 64-bit blocks of the message
• then an encryption of the entire message by using cipher block chaining (CBC) mode
• Given a message consisting of a sequence of 64-bit blocks $X_1, X_2, ..., X_N$
  
  $C = X_{N+1} = X_1 \oplus X_2 \oplus ... \oplus X_N$
• encrypt the entire message plus hash code, using CBC mode to produce the encrypted message $Y_1, Y_2, ..., Y_{N+1}$
Birthday attacks

• 64-bit hash code is used
• if an encrypted hash code $C$ is transmitted with corresponding unencrypted message $M$
  – opponent would need to find an $M'$ such that $H(M') = H(M)$
• opponent have to try about $2^{63}$ messages
1. 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.

2. Opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning.
   - Opponent prepares an equal number of messages, all of which are variations on the fraudulent message to be substituted for the real one.
3. two sets of messages are compared to find a pair of messages that produces the same hash code
   – The probability of success is greater than 0.5
4. 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.
• The generation of many variations that convey the same meaning is not difficult
• the opponent could simply reword the message but retain the meaning
Block Chaining techniques

• Divide a message M into fixed-size blocks M1, M2,..., MN and use a symmetric encryption system to compute the hash code $G$ as follows
  - $H_0$ = initial value
  - $H_i = E_{mi}(H_{i-1})$
  - $G = H_N$

• Other two versions:
  - $H_i = E_{Mi}(H_{i-1}) \oplus H_{i-1}$
  - $H_i = E_{Hi-1}(M_i) \oplus M_i$
Brute-force attack on Hash function

• For a hash code of length $n$, the level of effort required
  – One way $\rightarrow 2^n$
  – Weak collision resistance $\rightarrow 2^n$
  – Strong collision resistance $\rightarrow 2^{n/2}$
Brute-force attack on MAC

• it requires known message-MAC pairs
• security property of a MAC algorithm
  – Computation resistance: Given one or more text-MAC pairs \([x_i, C_K(x_i)]\), it is computationally infeasible to compute any text-MAC pair \([x, C_K(x)]\) for any new input \(x \neq x_i\).
Cryptanalysis of Hash function

• The hash function takes an input message and partitions it into $L$ fixed-sized blocks of $b$ bits each

• hash algorithm involves repeated use of a compression function, $f$
- $CV_o = IV = \text{initial n-bit value}$
- $CV_i = f(CV_{i-1}, Y_{i-1})$
- $H(M) = CV_L$
• if the compression function is collision resistant, then so is the resultant iterated hash function
• Cryptanalysis of hash functions focuses on the internal structure of f and is based on attempts to find efficient techniques for producing collisions for a single execution of f
• for any hash function there must exist collisions, required is that it is computationally infeasible to find collisions