Message Authentication and Hash function

Concept and Example
Approaches for Message Authentication

- Encryption protects message against passive attack, while Message Authentication protects against active attack.

- Message is considered authenticated if
  1. Its genuine and comes from alleged source.
  2. Verify that message is not altered.
  3. It comes from its original source.

- Approaches for message authentication
  1. Authentication using conventional encryption.
Authentication using conventional encryption

• Authentication done using symmetric encryption.
• Assume sender and receiver share a key.
• Message will be successfully authenticated if
  1. The message includes error detection code.
  2. The message includes a sequence number.
  3. The message includes a timestamp.
• (1) will protect against data alteration, (2) will protect against reordering, (3) will protect against latency.
• However, symmetric encryption is not considered as a suitable tool for data authentication.
  – Block reordering is a threat because sequence number is not always provided.
Message Authentication without message encryption

• Message is not always encrypted, but an authentication tag is generated and appended to the message.

• Authentication and Confidentiality services are combined in a single algorithm by encrypting the message plus the authentication tag.
Message Authentication Code (MAC)

- An authentication technique involves that involve a use of a secret key to generate a small block of data known as message authentication code, that is appended to the message.
- MAC scenario
  1. Both Alice (A) and Bob (B) know a secret key $K_{AB}$
  2. A calculates $MAC = F(K_{AB}, M)$.
  3. A sends $M + MAC$.
  4. B computes $MAC = F(K_{AB}, M$ as received$)$.
  5. If MAC from (2) equal MAC from (4) then message is authenticated.
Message Authentication Code (MAC)

Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)
Message Authentication Code (MAC)

- If MAC from (2) equal MAC from (4) then message is authenticated, and this means
  1. Message is not altered.
  2. Message is coming from alleged user.
  3. If message include a sequence number then ordering is guaranteed.

- A number of algorithm is used to generate the code.
  - Its recommended to use DES to generate an encrypted version of the message and the last number of bits in the cipher text is user as the code (16 or 32 bits).
  - Authentication algorithm need not to be reversible.
One-way hash function

• Alternative to MAC.
• Accept a variable-sized message \( M \) as input.
• Produced fixed-sized message digest \( H(M) \).
• Unlike MAC: a hash function doesn’t take a secret key as input.
• To authenticate a message: the message digest is sent with the message.
One-way hash function | authentication approaches

Figure 3.2 Message Authentication Using a One-Way Hash Function
One-way hash function | authentication approaches

1. Conventional Encryption :
   - Assume sender and receiver share the key.

2. Public key Encryption :
   - Advantages :
     1. It provides digital signatures as well as message authentication.
     2. It doesn't require key distribution.

3. Using Secret Value:
   - No encryption used.
   - Assume sender and receiver share the secret key $S_{AB}$. 
Hash Function
Requirements

1. apply on any size of data.
2. Produced fixed length output.
3. $H(x)$ is relatively easy to compute for any given $z$.
4. One way resistant: for any given code $h$, its computationally infeasible to find $x$ such that $H(x) = h$. 
Hash Function
Requirements

5. Weak collision resistant: for any given block $x$, its computationally infeasible to find $y$ where $H(x) = H(y)$.

- Attacker must find collision for a specific $x$. (By contrast, to break collision resistance, enough to find any collision.)

- Brute-force attack takes too long
6. Strong Collision resistant: its computationally infeasible to find a peer (x,y) such that H(x) = H(y).

- If 1st five requirements are fulfilled, then we will have a weak hash function.
- If all requirements are fulfilled then we have a strong hash function.
Simple hash function

- The input is viewed as a sequence of n-bit block.
- The input is processed one by one in an iterative fashion to produce an n-bit hash function.
- Simplest hash function: is a bit-by-bit XOR of every block.

<table>
<thead>
<tr>
<th>bit 1</th>
<th>bit 2</th>
<th>...</th>
<th>...</th>
<th>bit n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{11}$</td>
<td>$b_{21}$</td>
<td></td>
<td></td>
<td>$b_{n1}$</td>
</tr>
<tr>
<td>$b_{12}$</td>
<td>$b_{22}$</td>
<td></td>
<td></td>
<td>$b_{n2}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>$b_{1m}$</td>
<td>$b_{2m}$</td>
<td></td>
<td></td>
<td>$b_{nm}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hash code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
</tr>
<tr>
<td>$C_2$</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$C_n$</td>
</tr>
</tbody>
</table>

Figure 3.3 Simple Hash Function Using Bitwise XOR
Simple hash function

• A simple way to improve this hash function as follows
  1. Initially set the n-bit hash value to zero.
  2. Process each successive b-bit block of data:
     A. Rotate the current hash value to left by one bit.
     B. XOR the block into the hash value.

• Cons: its easy to produce a new message that yields the same hash code.

• Pros: simple and easy to implement.