Message authentication and hash functions

Message authentication

- Message (or document) is **authentic** if
- It is genuine and
- came from its alleged source.

- Message authentication is a **procedure** which verifies that received messages are authentic

Aspects of message authentication

- We would like to ensure that
  - The content of the message has not been changed;
  - The source of the message is authentic;
  - The message has not been delayed and replayed;

Message authentication techniques

- **Using conventional message encryption:**
  - if we assume that only sender and receiver share a secret key then the fact that receiver can successfully decrypt the message means the message has been encrypted by the sender

- **Without message encryption**
  - The message is not encrypted, but special authentication tag is generated and appended to the message. Generation of a tag is a much more efficient procedure that encryption of the message.
Message Authentication Code
- Let A and B share a common secret key K
- If A would like to send a message M to B, she calculates a message authentication code MAC of M using the key K:
  \[ MAC = F(K,M) \]
- Then A appends MAC to M and sends all this to B;
- B applies the MAC algorithm to the received message and compares the result with the received MAC.

MAC algorithms
- The process of MAC generation is similar to the encryption;
- The difference is a MAC algorithm need not be reversible \( \rightarrow \) easier to implement and less vulnerable to being broken;
- Actually, standard encryption algorithms can be used for MAC generation:
  - For example, a message may be encrypted with DES and then last 16 or 32 bits of the encrypted text may be used as MAC.

Message authentication using MAC

One-way Hash functions
- An alternative method for the message authentication is to use one-way hash functions instead of MAC;
- The main difference is hash functions don’t use a secret key:
  \[ h = H(M) \]
- “One-way” in the name refers to the property of such functions: they are easy to compute, but their reverse functions are very difficult to compute.
Methods of authentication using hashes

Hash function requirements
- To be suitable for message authentication, the hash functions must have ideally the following properties:
  - $H$ can be applied to a block of data of any size;
  - $H$ produces a fixed-length output;
  - $H(x)$ is easy to compute for any given $x$;
  - For any value $h$ it is very difficult (infeasible) to compute $x$ such that $H(x) = h$ (one-way property);
  - For any given $x$, it is very difficult (infeasible) to find $y$ (not equal to $x$) such that $H(x) = H(y)$; (weak collision resistance);
  - It is very difficult (infeasible) to find any pair $(x, y)$ such that $H(x) = H(y)$; (strong collision resistance).

Simple hash function
- Let the input be a sequence of $n$-bit blocks
- Then simple hash function does bit-by-bit exclusive-OR (XOR) of every block

<table>
<thead>
<tr>
<th>block 1</th>
<th>block 2</th>
<th>...</th>
<th>block n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>$b_2$</td>
<td>...</td>
<td>$b_n$</td>
</tr>
<tr>
<td>$b_{1'}$</td>
<td>$b_{2'}$</td>
<td>...</td>
<td>$b_{n'}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Hash code

Simple hash function
- Simple hash function does not satisfy the weak (and strong) collision property;
- for any message $M$ it is very easy to generate a message $M_1$ such that $h(M) = h(M_1)$:
  - Take arbitrary message $M_2$, compute $h(M_2) = h_2$, then
  - Add additional block to $M_2$, such that for the resulting $M_3$ we have $h(M_3) = h(M_1)$. 
The SHA-1 Secure Hash Algorithm

- **SHA-1 algorithm (1993-1995):**
  - It has been used in the sample program illustrating password-based encryption (practical sessions);
  - Takes as input a message with a maximum length
  - less than 2 to power 64 bits and produces as output a 160-bit message digest;
  - The input is processed in 512-bit blocks;
  - Each bit of the output is computed using all bits of the input.

SHA-1 general scheme

SHA-1 processing a single block

- The compression function;
- Includes 4 rounds with 20 steps each;
- Each round takes the current 512-bits block and 160-bit buffer value and updates the content of the buffer.

Problems and Solutions

- In 2005 a possible mathematical weakness of SHA-1 has been established:
  - ~2000 time more efficient than brute force search attack was found by Xiaoyun Wang
- Further developments: SHA-2: (SHA-224,-256,-384,-512)
- New competition for the new standard of hash functions by NIST:
  - Deadline for submissions was 31.10.2008
  - New standard SHA-3 is announced a winner on 2nd October 2012; not a replacement, but alternative for SHA-2
Recent News

- **SHAmpening**, October 2015 “freestart” collision attack
  (by M. Stevens, P. Karpman, T. Peyrin)
  - ~ US $2000 of GPU time on EC2, est.

- **SHAtered**, February 2017, full collision attack
  - (by Google)
  - ~ 6,500 years of CPU time