Cryptographic Hash Functions
Message Authentication
Digital Signatures
Abstract

We will discuss

• Cryptographic hash functions
• Message authentication codes
  – HMAC and CBC-MAC
• Digital signatures
Encryption/Decryption

- Provides *message confidentiality*.

- Does it provide *message authentication*?
Bob receives a message $m$ from Alice, he wants to know

- (Data origin authentication) whether the message was really sent by Alice;
- (Data integrity) whether the message has been modified.

**Solutions:**

- Alice attaches a message authentication code (MAC) to the message.
- Or she attaches a digital signature to the message.
Hash function

- A hash function maps from a domain to a smaller range, typically many-to-one.
- Properties required of a hash function depend on its applications.
- Applications:
  - Fast lookup (hash tables)
  - Error detection/correction
  - Cryptography: cryptographic hash functions
  - Others
Cryptographic hash function

- **Hash functions:** \( h : X \rightarrow Y, \ |X| > |Y|. \)
- For example, \( h : \{0,1\}^* \rightarrow \{0,1\}^n \)
  \[ h : \{0,1\}^* \rightarrow Z_n \]
  \[ h : \{0,1\}^k \rightarrow \{0,1\}^l, \ k > l. \]
- If \( X \) is finite, \( h \) is also called a compression function.
- A classical application: users/clients passwords are stored in a file
  not as (username, password),
  but as (username, \( h(\text{password}) \)) using some cryptographic hash function \( h \).
Security requirements

- Pre-image: if \( h(m) = y \), \( m \) is a pre-image of \( y \).
- Each hash value typically has multiple pre-images.
- Collision: a pair of \((m, m')\), \( m \neq m' \), s.t. \( h(m) = h(m') \).

A hash function is said to be:

- **Pre-image resistant** if it is computationally infeasible to find a pre-image of a hash value.
- **Collision resistant** if it is computationally infeasible to find a collision.
- A hash function is a **cryptographic hash function** if it is collision resistant.
• Collision-resistant hash functions can be built from collision-resistant compression functions using Merkle-Damgard construction.
Merkle-Damgard construction

- Construct a cryptographic hash function \( h : \{0,1\}^* \to \{0,1\}^n \)
  from a compression function \( f : \{0,1\}^{n+b} \to \{0,1\}^n \).

1. For \( m \in \{0,1\}^* \), add padding to \( m \) so that \(|m'|\) is a multiple of \( b \).
   Let padded \( m' = m_1 m_2 \ldots m_k \), each \( m_i \) of length \( b \).
   (padding = 10...0 \( |m| \), where \( |m| \) is the length of \( m \))

3. Let \( v_0 = IV \) and \( v_i = f(v_{i-1} \| m_i) \) for \( 1 \leq i \leq k \).

4. The hash value \( h(m) = v_k \).

Theorem. If \( f \) is collision-resistant, then \( h \) is collision-resistant.
Merkle-Damgard Construction

Compression function $f : \{0,1\}^{n+b} \rightarrow \{0,1\}^n$
The Secure Hash Algorithm (SHA-1)

- an NIST standard.
- using Merkle-Damgard construction.
- input message $m$ is divided into blocks with padding.
- $\text{padding} = 10...0\ell$, where $\ell \in \{0,1\}^{64}$ indicates $|m|$ in binary.
- thus, message length limited to $|m| \leq 2^{64} - 1$.
- block = 512 bits = 16 words = $W_0 \| \ldots \| W_{15}$.
- IV = a constant of 160 bits = 5 words = $H_0 \| \ldots \| H_4$.
- resulting hash value: 160 bits.
- underlying compression function $f : \{0,1\}^{160+512} \rightarrow \{0,1\}^{160}$, a series (80 rounds) of $\land$, $\lor$, $\oplus$, $\neg$, $+$, and Rotate on words $W_i$'s & $H_i$'s.
Is SHA-1 secure?

- An attack is to produce a collision.
- Birthday attack: randomly generate a set of messages \( \{m_1, m_2, \ldots, m_k\} \), hoping to produce a collision.
- \( n = 160 \) is big enough to resist birthday attacks for now.
- There is no mathematical proof for its collision resistancy.
- In 2004, a collision for a "58 rounds" SHA-1 was produced. (The compression function of SHA-1 has 80 rounds.)
- Newer SHA's have been included in the standard: SHA-256, SHA-384, SHA-512.
• **Birthday problem:** In a group of $k$ people, what is the probability that at least two people have the same birthday?
  • Having the same birthday is a collision?

• **Birthday paradox:** $p \geq 1/2$ with $k$ as small as 23.

• Consider a hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$.
• If we randomly generate $k$ messages, the probability of having a collision depends on $n$.
• To resist birthday attack, we choose $n$ to be sufficiently large that it will take an infeasibly large $k$ to have a non-negligible probability of collision.
Applications of cryptographic hash functions

- Storing passwords
- Used to produce modification detection codes (MDC)
  - $h(m)$, called an MDC, is stored in a secure place;
  - if $m$ is modified, we can detect it;
  - protecting the integrity of $m$.
- We will see some other applications.
Message Authentication

• Bob receives a message $m$ from Alice, he wants to know
  • (Data origin authentication) whether the message was really sent by Alice;
  • (Data integrity) whether the message has been modified.

• Solutions:
  • Alice attaches a message authentication code (MAC) to the message.
  • Or she attaches a digital signature to the message.
MAC

- Message authentication protocol:
  1. Alice and Bob share a secret key $k$.
  2. Alice sends $m \ || \ MAC_k(m)$ to Bob.
  3. Bob authenticates the received $m' \ || \ MAC'$ by checking if $MAC' = MAC_k(m')$?

- $MAC_k(m)$ is called a message authentication code.
- Security requirement: infeasible to produce a valid pair $(x, MAC_k(x))$ without knowing the key $k$. 
Constructing MAC from a hash

• A common way to construct a MAC is to incorporate a secret key $k$ into a fixed hash function $h$ (e.g. SHA-1).

• $MAC_k(m) = h_k(m) = h(m)$ with IV = $k$

• $MAC_k(m) = h_k(m) = h(k \parallel m)$
• Insecure: $MAC_k(m) = h(m)$ with IV = $k$.
  (For simplicity, without padding)

\[ m = m_1 \quad m_2 \quad m_3 \quad \cdots \quad m_s \]

\[ k \xrightarrow{X} f \xrightarrow{f} f \xrightarrow{f} \cdots \xrightarrow{f} h(m) \xrightarrow{X} h_k(m) \]

• Easy to forge:
  
  \((m', h_k(m'))\),
  where \(m' = m \parallel m_{s+1}\)
HMAC (Hash-based MAC)

- A FIPS standard for constructing MAC from a hash function $h$. Conceptually,
  \[
  \text{HMAC}_k(m) = h(k_2 \Vert h(k_1 \Vert m))
  \]
  where $k_1$ and $k_2$ are two keys generated from $k$.
- Various hash functions (e.g., SHA-1, MD5) may be used for $h$.
- If we use SHA-1, then HMAC is as follows:
  \[
  \text{HMAC}_k(m) = \text{SHA-1}(k \oplus \text{ipad} \Vert \text{SHA-1}(k \oplus \text{opad} \Vert m))
  \]
  where
  - $k$ is padded with 0's to 512 bits
  - $\text{ipad} = 3636\cdots36$ (x036 repeated 64 times)
  - $\text{opad} = 5c5c\cdots5c$ (x05c repeated 64 times)
CBC-MAC

- A FIPS and ISO standard.
- One of the most popular MACs in use.
- Use a block cipher in CBC mode with a fixed, public IV.
- Called DES CBC-MAC if the block cipher is DES.
- Let $E : \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher.
- CBC-MAC($m, k$)
  
  $m = m_1 \parallel m_2 \parallel \ldots \parallel m_l$, where $|m_i| = n$.
  
  $c_0 \leftarrow$ IV (typically $0^n$)

  for $i \leftarrow 1$ to $l$ do
  
  $c_i \leftarrow E_k(c_{i-1} \oplus m_i)$

  return($c_l$)
Cipher Block Chaining (CBC)
CMAC (Cipher-based MAC)

- A refined version of CBC-MAC.
- Adopted by NIST for use with AES and 3DES.
- Use two keys: $k, k'$ (assuming $|m|$ is a multiple of $n$).
- Let $E : \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher.

CMAC($m, k$)

$$m = m_1 \| m_2 \| \ldots \| m_l, \text{ where } |m_i| = n.$$  
$c_0 \leftarrow$ IV (typically $0^n$)

for $i \leftarrow 1$ to $l - 1$ do

$$c_i \leftarrow E_k(c_{i-1} \oplus m_i)$$

$$c_l \leftarrow E_k(c_{l-1} \oplus m_l)$$

return($c_l$)
Digital Signatures

- RSA can be used for digital signatures.
- A digital signature is the same as a MAC except that the tag (signature) is produced using a public-key cryptosystem.
- Digital signatures are used to provide message authentication and non-repudiation.

<table>
<thead>
<tr>
<th>Message m</th>
<th>MAC_k(m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message m</td>
<td>Sig_pr(m)</td>
</tr>
</tbody>
</table>
• Digital signature protocol:
  1. Bob has a key pair \((pr, pu)\).
  2. Bob sends \(m \parallel \text{Sig}_{pr}(m)\) to Alice.
  3. Alice verifies the received \(m' \parallel s'\) by checking if \(s' = \text{Verify}_{pu}(m')\).

• \(\text{Sig}_{pr}(m)\) is called a signature for \(m\).

• Security requirement: infeasible to forge a valid pair \((m, \text{Sig}_{pr}(m))\) without knowing \(pr\).
Encryption (using RSA):

Digital signature (using RSA^{-1}):
RSA Signature

- **Keys** are generated as for RSA encryption:
  
  Public key: $PU = (n,e)$. Private key: $PR = (n,d)$.
  
- **Signing** a message $m \in Z_n^*$: $\sigma = D_{PR}(m) = m^d \mod n$.
  
  That is, $\sigma = RSA^{-1}(m)$.
  
- **Verifying** a signature $(m,\sigma)$:
  
  check if $m = E_{PU}(\sigma) = \sigma^e \mod n$, or $m = RSA(\sigma)$.
  
- Only the key's owner can sign, but anybody can verify.
Security of RSA Signature

• Existential forgeries:

1. Every message \( m \in \mathbb{Z}_n^* \) is a valid signature for its ciphertext \( c := RSA(m) \).

   Encryption (using Bob's public key): \( m \xrightarrow{\text{RSA}} c \)

   Sign (if using Bob's private key): \( m \xleftarrow{\text{RSA}^{-1}} c \)

2. If Bob signed \( m_1 \) and \( m_2 \), then the signature for \( m_1m_2 \) can be easily forged: \( \sigma(m_1m_2) = \sigma(m_1)\sigma(m_2) \).

• Countermeasure: hash and sign: \( \sigma = \text{Sign}_{PR}(h(m)) \), using some collision resistant hash function \( h \).
Question:
Does hash-then-sign make RSA signature secure against all chosen-message attacks?

Answer:
Yes, if $h$ is a full-domain random oracle, i.e.,
- $h$ is a random oracle mapping $\{0, 1\}^* \rightarrow Z_n$
- ($Z_n$ is the full domain of RSA)
Problem with full-domain hash:
In practice, $h$ is not full-domain.
For instance, the range of SHA-1 is $\{0,1\}^{160}$, while $Z_n = \{0,1,...,2^n - 1\}$, with $n \geq 1024$.

Desired: a secure signature scheme that does not require a full-domain hash.
Probabilistic signature scheme

- Hash function \( h : \{0,1\}^* \rightarrow \{0,1\}^l \subseteq Z_N \) (not full domain).
  \[ l < n = |N|. \] (E.g., SHA-1, \( l = 160 \); RSA, \( n = 1024 \).)

- Idea: \( m \xrightarrow{\text{pad}} m \| r \in \{0,1\}^* \)
  \[ m \xrightarrow{\text{hash}} w = h(m \| r) \in \{0,1\}^l \]
  \[ m \xrightarrow{\text{expand}} y = w \| (r \| 0^{n-1-l-k}) \oplus G(w) \in \{0,1\}^{n-1} \]
  \[ m \xrightarrow{\text{sign}} \sigma = \text{RSA}^{-1}(y) \in Z_N \]

where \( r \in \{0,1\}^k \)

\( G : \{0,1\}^l \rightarrow \{0,1\}^{n-1-l} \) (pseudorandom generator)
• **Signing** a message $m \in \{0,1\}^*$:

1. choose a random $r \in \{0,1\}^k$; compute $w = h(m \parallel r)$;
2. compute $y = w \parallel r \oplus G_1(w) \parallel G_2(w)$;  // $G = G_1 \parallel G_2$ //
3. The signature is $\sigma = RSA^{-1}(y)$. 
Remarks

- PSS is secure against chosen-message attacks in the random oracle model (i.e., if $h$ and $G$ are random oracles).
- PSS is adopted in PKCS #1 v.2.1.
- Hash functions such as SHA-1 are used for $h$ and $G$.
- For instance,
  let $n = 1024$, and $l = k = 160$
  let $h = \text{SHA-1}$
  \[(G_1, G_2)(w) = G(w) = h(w \Vert 0) \parallel h(w \Vert 1) \parallel h(w \Vert 2), \ldots\]