10.2 Construction of Hash Function

Construction principles of most hash functions:

\[ h_0 = 1V \quad \text{(Initial Value)} \]
\[ h_i = g(h_{i-1}, M_i) \quad i = 1, \ldots, N \]
\[ h_N = h(m) \quad \text{hash value of } m \]

Some hash functions of this type are:

- **MDS**
  - Revert (1942), 128 bit hash length
- **SHA-1**
  - Successor of SHA (Secure Hash Standard)
    - NIST, 1993, 160 bit length
- **SHA-256 / SHA-384 / SHA-512**
  - NIST, 2001, 256, 384, 512 bits of hash length
- **FIPS 180-2** (Federal Information Processing Standard)
  - Standard from Aug. 2002, contains the SHA-family, particularly SHA-3

**Description of SHA-1**

\[ M_i \text{: has length 512 bits} \]

1) Operation on words of 32 bits:

- \( A \& B, A \lor B, A \oplus B \)
  - bitwise and, or, xor
- \( \neg A \)
  - bitwise complement
- \( A + B \)
  - addition modulo \( 2^{32} \)
- \( \text{ROT L}^4(A) \)
  - cyclic shift to the left by 4 bits
- \( A || B \)
  - concatenation of A and B
b) Padding of message $m$ to a length $n \cdot t \leq 512$ bits

Note: $|m| \leq 2^{64} - 1$ is assumed ($1 m$ : length of $m$)

**SHA-1-PAD** ($m$):

<table>
<thead>
<tr>
<th>$m$</th>
<th>10...</th>
<th>0</th>
<th>$1 m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>448 bits</td>
<td>64 bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) Append a single 1 to $m$
2) Concatenate 0's of length $n$ to $m$ until a total length is computed as 448 mod 512
3) Concatenate length of $m$ with 64 bits, i.e., leading zeros are included

c) Functions and constants in SHA-1:

- $f_i (B, C, D) = \begin{cases} (B \land C) \lor (\neg B \land D) & 0 \leq i \leq 19 \\ B \oplus C \oplus D & 20 \leq i \leq 39 \\ (B \land C) \lor (B \land D) \lor (C \land D) & 40 \leq i \leq 59 \end{cases}$

- $k_i = \begin{cases} \text{5 A 8 2 F 9 9 9} & 0 \leq i \leq 19 \\ \text{6 E D 9 E B 4 1} & 20 \leq i \leq 39 \\ \text{8 F 7 B B C D C} & 40 \leq i \leq 59 \\ \text{C A 6 2 C 1 O 6} & 60 \leq i \leq 79 \end{cases}$

d) Algorithm SHA-1 (see lecture notes)

Severe problems with hash functions have been demonstrated, including recommendations by the NIST from 2005:

- Don't use MD4 or MD5 anymore
- Find alternatives for SHA-1 until 2010, don't use it afterwards

Shamir has suggested to develop a complete redesign of hash functions elsewhere AFS
Nov 2007 NIST put out a call for developing a new hash fed.
Oct 2012 end of competition, similar to AES
Winner "Keccak" published as NIST FIPS 202, contains
"SHA-3 standard"

Keccak developed by Daemen et al.
Finalists were
BLAKE (Rijmen et al.)
Grøstl (Knuutila et al.)
SHAKE (Honghun Win)
Keccak (Daemen et al.)
Skein (Schnorr et al.)

- Extension of construction principle
- Division in "rate" and "capacity" part of hash function
- Distinction between absorbing phase (message blocks are used)
  and squeezing phase (generates output)
11 Digital Signatures

Method of signing a message in electronic form
required (same as an conventional signature)

- forge - proof
- verifiable (proof of ownership)
- firmly connected to document

Problem for certain applications: repeated use of copies
Ex: Signed digital message for money transfer
Countermeasure against repeated use: time stamps

Attacks on signature schemes:

- Key only attack (Oscar knows the public key only)
- Known message attack (Oscar knows signatures for a set of messages)
- Chosen message attack (Oscar obtains signatures for a set of chosen messages)

Attacks may result in:

- Total break (O can sign any message)
- Selective forgery (O can sign a particular class of messages)
- Existential forgery (O can sign at least one message)

Known from Cryptography I: RSA signature scheme

Oscar signs with public key \((e, n)\)
private key \(d\)

\[ c = [h(m)]^d \mod n \]

Verification: \( h(m) = c^e \mod n \)

Presented: Cryptography I: El Gamal signature scheme
11.1 ElGamal signature scheme

Parameters: \( p, p \), a prime, \( a \), a \( \mathbb{F} \) mod \( p \), \( h \), hash function

Select random \( x \), \( 1 < y < \) mod \( p \)

Public key: \( (p, a, y) \)
Private key: \( x \)

Signature generation:
Select random \( k \)
\[ r = a^k \mod p \]
\[ c = k^{-1}(h(m) - x \cdot r) \mod p - 1 \]

Signature for \( m \): \( (r, c) \)

Remark: \( k^{-1} \cdot r, x \cdot r \) can be computed in advance

Verification: Verify \( 1 \leq r \leq p - 1 \)

\[ V_1 = y^r \cdot r^c \mod p \]
\[ V_2 = a^{h(m)} \mod p \]

If \( V_1 = V_2 \) we accept signature

Verification works:
Hence \( V_1 \equiv y^r \cdot r^c \equiv a^{k \cdot x \cdot r + h(m)} \equiv a^{k \cdot (p-1) + h(m)} \equiv V_2 \) (mod \( p \))

(\#): from (\#): \( k \cdot c \equiv h(m) - x \cdot r \) \( \equiv h(m) \equiv k \cdot x + x \cdot r \) (mod \( p \))

(\#) \( x \cdot r + x \cdot k \equiv k \cdot (p-1) + h(m) \) for some \( k \in \mathbb{Z} \)
Security,
a) Don't use the same key twice! Otherwise
\[ \begin{align*}
\sigma_1 &= b^{-1} (h(m_1) - x \cdot r) \mod p-1 \\
\sigma_2 &= b^{-1} (h(m_2) - x \cdot r) \mod p-1
\end{align*} \]

\[ = (\sigma_1 - \sigma_2) \cdot b \equiv h(m_1) - h(m_2) \pmod{p-1} \]

\[ = b \equiv (\sigma_1 - \sigma_2)^{-1} (h(m_1) - h(m_2)) \pmod{p-1} \]

provided \((\sigma_1 - \sigma_2)^{-1} \pmod{p-1}\) exists, but it exists with high probability.

Once \( b \) is known it can be determined from (2) or (3), if \( r \) is invertible, which is the case with high probability.
b) Oscar can forge a signature on a hashed message as follows:

Select any pair \((x, v) \) s.t \( g \cdot c \cdot d \cdot (v, p-1) \equiv 1 \)

Compute \( r = a^x \cdot v \equiv a^x + v \cdot v \pmod{p} \)

\[ \sigma = -r \cdot v^{-1} \pmod{p-1} \]

Then \( (\sigma, v) \) is a valid signature for \( h(m) \equiv c \cdot u \pmod{p-1} \)

**Proof:**
\[ V_1 = g^r \cdot c^x \equiv a^{x+r} \cdot (a+x \cdot v) \pmod{p} \]
\[ \equiv a^{x+r} - a^x \cdot v \cdot v^{-1} - a^x \cdot v \cdot v^{-1} \pmod{p} \]
\[ \equiv a^x \cdot v \cdot v^{-1} \pmod{p} \]

\[ V_2 = a \cdot h(m) \equiv a^x \cdot v \equiv a^x \cdot v \cdot v^{-1} \cdot u \equiv v_1 \pmod{p} \]

\[ \Rightarrow V_1 = V_2 \]