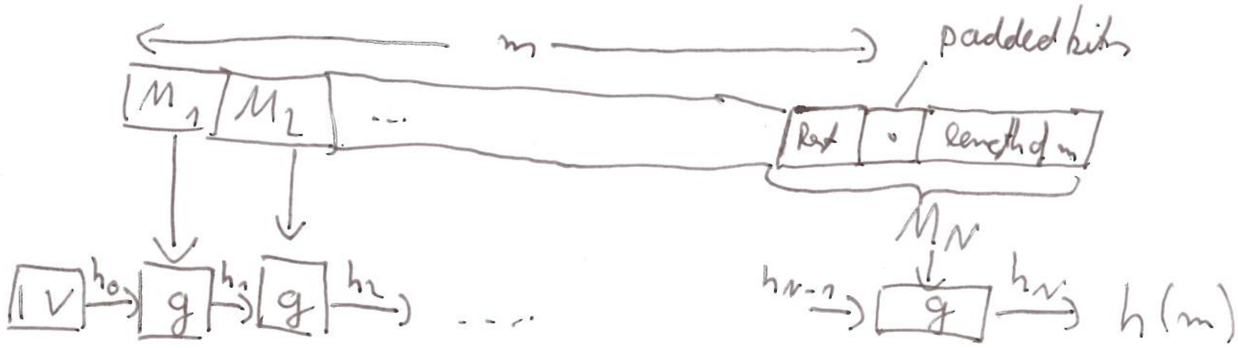
11.2 Construction of Hash Functions

Construction principle of most hash functions up to SHA-1



$h_0 = IV$ (initial value)

$h_i = g(h_{i-1}, M_i) \qquad i = 1, \dots, N$

$h_N = g(h_{N-1}, M_i) = h(m)$ (Hash value)

Some of hash functions of this type are

- MD5 — Rivest, 1992, 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 bit hash length
- FIPS 180-2 — Standard from Aug. 2002, contains the SHA family (Federal Information Processing Standards)

Description of SHA-1

M_i has length 512 bits

a) Operations on words of 32 bits

- $A \wedge B$, $A \vee B$, $A \oplus B$: bitwise and, or, xor
- $\neg A$: " complement
- $A + B$: addition modulo 2^{32}
- $ROTL^s(A)$: cyclic shift to the left by $0 \leq s \leq 31$ positions
- $A \,\|\, B$: concatenation of bits of A and B.

b). Padding of message m to a length l_1 s.t $512 \mid l_1$

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

SHA-1-PAD(m) :

$$\underbrace{\boxed{m}\boxed{1}\boxed{0}\;\cdots\;\boxed{0}}_{448}\underbrace{\boxed{\quad |m| \quad}}_{64 \text{ bits}}$$

i) append a single 1 to m

ii) Concatenate 0's s.t length is congr. 448 mod 512

iii) concatenate length of m with 64 bits, i.e., leading zeros are included

c) Functions and constants in SHA-1

$$f_i(B,C,D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \leq i \leq 19 \\ B \oplus C \oplus D & 20 \leq i \leq 39,\ 60 \leq i \leq 79 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \leq i \leq 59 \end{cases}$$

$$k_i = \begin{cases} 5A827999 & 0 \leq i \leq 19 \\ 6ED9EBA1 & 20 \leq i \leq 39 \\ 8F1BBCDC & 40 \leq i \leq 59 \\ CA62C1D6 & 60 \leq i \leq 79 \end{cases}$$

d) Algorithm SHA-1 (see lecture notes)

Severe problems with hash functions have been demonstrated

Recommendation of the NIST from 2005:

- Don't use MD4 and 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

Algorithm 4.8: SHA-1-PAD(x)

comment: $|x| \leq 2^{64} - 1$

$d \leftarrow (447 - |x|) \bmod 512$
$\ell \leftarrow$ the binary representation of $|x|$, where $|\ell| = 64$
$y \leftarrow x \,||\, 1 \,||\, 0^d \,||\, \ell$

Cryptosystem 4.1: *SHA-1*(x)

external SHA-1-PAD
global $K_0, \ldots, K_{79}$
$y \leftarrow$ SHA-1-PAD(x)
denote $y = M_1 \,||\, M_2 \,||\, \cdots \,||\, M_n$, where each M_i is a 512-bit block
$H_0 \leftarrow$ 67452301
$H_1 \leftarrow$ EFCDAB89
$H_2 \leftarrow$ 98BADCFE
$H_3 \leftarrow$ 10325476
$H_4 \leftarrow$ C3D2E1F0
for $i \leftarrow 1$ **to** n
do
- denote $M_i = W_0 \,||\, W_1 \,||\, \cdots \,||\, W_{15}$, where each W_i is a word
- **for** $t \leftarrow 16$ **to** 79
  - **do** $W_t \leftarrow \mathsf{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$
- $A \leftarrow H_0$
- $B \leftarrow H_1$
- $C \leftarrow H_2$
- $D \leftarrow H_3$
- $E \leftarrow H_4$
- **for** $t \leftarrow 0$ **to** 79
  - **do**
    - $temp \leftarrow \mathsf{ROTL}^5(A) + \mathsf{f}_t(B, C, D) + E + W_t + K_t$
    - $E \leftarrow D$
    - $D \leftarrow C$
    - $C \leftarrow \mathsf{ROTL}^{30}(B)$
    - $B \leftarrow A$
    - $A \leftarrow temp$
- $H_0 \leftarrow H_0 + A$
- $H_1 \leftarrow H_1 + B$
- $H_2 \leftarrow H_2 + C$
- $H_3 \leftarrow H_3 + D$
- $H_4 \leftarrow H_4 + E$

return $(H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4)$

Aus: Stinson (02), p.134, 135

Nov, 2007 NIST put out a call for developing a new hash function
Oct, 2012 End of competition, similar to the AES

Winner "Keccak" published as NIST FIPS 202
"SHA-3 standard"

Keccak developed by Daemen et al
Finalists were BLAKE (Aumasson et al.)
Grøstl (Knudsen et al.)
JH (Hongjun Wu)
Keccak (Daemen et al)
Skein (Schneier et al.)

- Extension of construction principle:
  - Division in "rate" & "capacity" part of hash function
  - Distinction between
    * absorbing phase (message blocks are used)
    * squeezing phase (generate output)

11 Digital Signatures

Method of signing a message in electronic form

Requirements (same as on conventional signatures)

- verifiable (proof of ownership)
- forgery - proof
- firmly connected to the document

Problem for certain applications : repeated use of copies

Ex. : Signed digital messages 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 (O has signatures for a set of messages)
- Chosen " " (O obtains signatures for a set of chosen messages

Attacks may result in :

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

Known from Cryptography I : RSA signature scheme on hash value $h(m)$

Alice signs with public key (e,n), private key d

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

Verification $h(m) = s^e \mod n$

Presented ~~in~~ in Cryptography I : ElGamal signature scheme

11.1 ElGamal signature scheme

Parameters p: prime, a PE mod p, h: hash functions

Select random x, $y = a^x \mod p$

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

Signature generation:

Select random k s.t. $k^{-1} \mod p-1$

$$\left.\begin{aligned} r &= a^k \mod p \\ s &= k^{-1}(h(m) - x \cdot r) \mod p-1 \end{aligned}\right\} (*)$$

Signature for m: (r, s)

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

Verification:

verify $1 \leq r \leq p-1$

$v_1 = y^r r^s \mod p$

$v_2 = a^{h(m)} \mod p$

if $v_1 = v_2$ we accept signature

Verification works:

$(*)$: $k \cdot s \equiv h(m) - xr \pmod{p-1} \Leftrightarrow h(m) \equiv x \cdot r + k \cdot s \pmod{p-1}$

$\Leftrightarrow xr + k \cdot s = l(p-1) + h(m)$ for some $l \in \mathbb{Z}$

Hence $v_1 = y^r r^s \equiv a^{x \cdot r} a^{k \cdot s} \equiv a^{x \cdot r + ks} \equiv a^{l(p-1) + h(m)}$

$$\equiv \underbrace{(a^{p-1})^l}_{\equiv 1 \pmod p,\ \text{Fermat}} a^{h(m)} \equiv a^{h(m)} \equiv v_2 \pmod p \quad \checkmark$$

Security

a) Don't use the same k twice! Otherwise

$$s_1 = k^{-1}(h(m_1) - x \cdot r) \mod p-1 \quad (2)$$

$$s_2 = k^{-1}(h(m_2) - x \cdot r) \mod p-1 \quad (3)$$

$$\Rightarrow (s_1 - s_2) \cdot k \equiv h(m_1) - h(m_2) \mod p-1$$

$$\Rightarrow k \equiv (s_1 - s_2)^{-1}(h(m_1) - h(m_2)) \mod p-1$$

provided $(s_1 - s_2)^{-1} \mod p-1$ exists, but it exists with high prob.

Once k is known, x can be determined from (2) or (3)

b) Oscar can forge a signature on a hashed message $h(m)$ as follows

Select any pair (u, v) s.t $\gcd(v, p-1) = 1$

Compute $r = a^u y^v = a^{u + x \cdot v} \mod p$

$$\cancel{x} \quad s = -r \cdot v^{-1} \mod (p-1)$$

Then (r, s) is a valid signature for $h(m) = s \cdot u \mod (p-1)$

<u>Proof</u>: $v_1 = y^r r^s = a^{x \cdot r} a^{(u + xv)(-r \cdot v^{-1})}$ (mod p

$$= a^{x \cdot r - u \cdot r \cdot v^{-1} - x r \cdot \underbrace{v \cdot v^{-1}}_{\equiv 1}} \quad (\text{mod } p)$$

$$= a^{-u \cdot r \cdot v^{-1}} \quad (\text{mod } p)$$

$$v_2 = a^{h(m)} \mod p = a^{s \cdot u} \mod p = a^{-u \cdot r \cdot v^{-1}} \mod p$$

$$\Rightarrow v_1 = v_2 \quad \checkmark$$