Hash functions & MACs

Digital Signature

Alice
- Message
- Signature
- Hash function
- Hash value
- Public key algorithm
- Alice's private key

Bob
- Message
- Signature
- Hash function
- Hash value
- Public key algorithm
- Alice's public key

Hash function

- arbitrary length
- $m$
- $h$
- $h(m)$
- hash function
- hash value
Vocabulary

<table>
<thead>
<tr>
<th>hash function</th>
<th>hash value</th>
</tr>
</thead>
<tbody>
<tr>
<td>message digest</td>
<td>message digest</td>
</tr>
<tr>
<td>hash total</td>
<td>hash total</td>
</tr>
<tr>
<td>fingerprint</td>
<td>fingerprint</td>
</tr>
<tr>
<td>imprint</td>
<td>imprint</td>
</tr>
<tr>
<td>cryptographic checksum</td>
<td>compressed encoding</td>
</tr>
<tr>
<td>MDC, Message Digest Code</td>
<td></td>
</tr>
</tbody>
</table>

Hash functions

Basic requirements

1. Public description, NO key
2. Compression
   arbitrary length input → fixed length output
3. Ease of computation

Security requirements

It is computationally infeasible

<table>
<thead>
<tr>
<th>Given</th>
<th>To Find</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>$x$, such that $h(x) = y$</td>
</tr>
<tr>
<td>$x$ and $y=h(x)$</td>
<td>$x' \neq x$, such that $h(x') = h(x) = y$</td>
</tr>
<tr>
<td>$x'$ $\neq x$, such that $h(x') = h(x)$</td>
<td></td>
</tr>
</tbody>
</table>
Hash functions

Dependence between requirements

2nd preimage resistant

Collision resistant

Hash functions (unkeyed)

OWHF

Collision-Resistant Hash Functions

CRHF

One-Way Hash Functions

preimage resistance

2nd preimage resistance

collision resistance

Brute force attack against One-Way Hash Function

Given $y$

$m_i'$

$i = 1..2^n$

$2^n$ messages with the contents required by the forger

$h(m_i') = y$

$n$ - bits
Creating multiple versions of the required message

I state thereby that I borrowed received

$10,000 from Mr. Kris Krzysztof

Gaj on November 17, 2009. This money sum of money

should be returned to Mr. Gaj

by the 8th day of December 2009.

---

Brute force attack against Collision Resistant Hash Function

Yuval

\[ r \text{ messages acceptable for the signer} \]
\[ m_i \]
\[ h \]
\[ \text{ } \]
\[ h(m_i) \]
\[ n - \text{ bits} \]

\[ r \text{ messages required by the forger} \]
\[ m'_j \]
\[ h \]
\[ \text{ } \]
\[ h(m'_j) \]
\[ n - \text{ bits} \]

\[ h(m_i) = h(m'_j) \]

Creating multiple versions of the required message

I state thereby that I borrowed received

$10,000 from Mr. Kris Krzysztof

Gaj on November 17, 2009. This money sum of money

should be returned to Mr. Gaj

by the 8th day of December 2009.

---
I state thereby that on December 1, 12/1/2009 I borrowed from Mr. Dr. Kris Krzysztof a paper manuscript on Benchmarking in software. This text item is required to be returned to Mr. Dr. Gaj by the 8th day of December 2009.

---

Birthday paradox

How many students must be in a class so that there is a greater than 50% chance that

1. one of the students shares the teacher’s birthday (up to the day and month)?
2. any two of the students share the same birthday (up to the day and month)?

~ \( \sqrt{366} \approx 19 \)

~ \( \frac{366}{2} = 188 \)
Brute force attack against Collision Resistant Hash Function

Probability \( p \) that two different messages have the same hash value:

\[
p = 1 - \exp \left( -\frac{r^2}{2n} \right)
\]

For \( r = 2^{n/2} \) \( p = 63\% \)

Brute force attack against Collision Resistant Hash Function

\textit{Storage requirements}

\textit{J.J. Quisquater}

\textit{collision search algorithm}

Number of operations: \( 2^{\sqrt{\pi/2} \cdot 2^{n/2}} \approx 2.5 \cdot 2^{n/2} \)

Storage: Negligible

<table>
<thead>
<tr>
<th>Hash value size</th>
<th>One-Way</th>
<th>Collision-Resistant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Older algorithms:</td>
<td>( n \geq 64 )</td>
<td>( n \geq 128 )</td>
</tr>
<tr>
<td></td>
<td>8 bytes</td>
<td>16 bytes</td>
</tr>
<tr>
<td>Current algorithms:</td>
<td>( n \geq 80 )</td>
<td>( n \geq 160 )</td>
</tr>
<tr>
<td></td>
<td>10 bytes</td>
<td>20 bytes</td>
</tr>
<tr>
<td>Newly proposed algorithms:</td>
<td>( n = 128, 192, 256 )</td>
<td>( n = 256, 384, 512 )</td>
</tr>
<tr>
<td></td>
<td>16, 24, 32 bytes</td>
<td>32, 48, 64 bytes</td>
</tr>
</tbody>
</table>
Hash function algorithms

- **Hash functions algorithms**
  - Customized (dedicated)
    - MD2
      - Rivest, 1988
    - MD4
      - Rivest, 1990
    - MD5
      - Rivest, 1990
    - SHA-0
      - NSA, 1992
    - SHA-1
      - NSA, 1995
    - SHA-256, SHA-384, SHA-512
  - Based on block ciphers
    - MDC-2
      - Based on block ciphers
    - MDC-4
      - Based on block ciphers
    - MASH-1
      - IBM, Brachtl, Meyer, Schilling, 1988
  - Based on modular arithmetic
    - MASH-1
      - 1988-1996
    - RIPEMD
      - European RACE Integrity Primitives Evaluation Project, 1992
    - RIPEMD-160
      - NSA, 1995

Attacks against dedicated hash functions known by 2004

- MD2
  - partially broken
  - broken, H. Dobbertin, 1995
    - (one hour on PC, 20 free bytes at the start of the message)
- MD4
  - partially broken
- MD5
  - weakness discovered, 1995 NSA, 1998 France
  - reduced round version broken, Dobbertin 1995
  - partially broken, collisions for the compression function, Dobbertin, 1996
    - (10 hours on PC)
- SHA-0
  - NSA, 1992
- SHA-1
  - NSA, 1995
- SHA-256, SHA-384, SHA-512
  - NSA, 2000

What was discovered in 2004-2005?

- MD4
  - broken; Wang, Feng, Lai, Yu, Crypto 2004
    - (manually, without using a computer)
- MD5
  - broken; Wang, Feng, Lai, Yu, Crypto 2004
    - (1 hr on a PC)
  - SHA-0
    - attack with 2^{40} operations
      - Wang, Yin, Yu, Aug 2005
  - SHA-1
    - attack with 2^{63} operations
      - Wang, Yin, Yu, Aug 2005
    - broken; Wang, Feng, Lai, Yu, Crypto 2004
      - (manually, without using a computer)
- SHA-256, SHA-384, SHA-512
2^{63} \text{operations}
Schneier, 2005

In hardware:
Machine similar to the one used to break DES:
- Cost = $50,000-$70,000  Time: 18 days
- Cost = $0.9-$1.26M  Time: 24 hours

In software:
Computer network similar to distributed.net
used to break DES (~331,252 computers):
- Cost = ~ $0  Time: 7 months

Recommendations of NIST (1)
NIST Brief Comments on Recent Cryptanalytic Attacks on SHA-1
Feb 2005

The new attack is applicable primarily
to the use of hash functions in digital signatures.

In many cases applications of digital signatures
introduce additional context information,
which may make attacks impracticle.

Other applications of hash functions,
such as Message Authentication Codes (MACs),
are not threatened by the new attacks.

Recommendations of NIST (2)

NIST was already earlier planning to withdraw SHA-1
in favor of SHA-224, SHA-256, SHA-384 & SHA-512
do roku 2010

New implementations should use new hash functions.

NIST encourages government agencies to develop plans
for gradually moving towards new hash functions,
taking into account the sensitivity of the systems
when setting the timetables.
### SHA-3 Contest Timeline

**2007**
- Publication of requirements
  - 29.X. 2007: request for candidates

**2008**
- 31.X.2008: deadline for submitting candidates
  - 9.XII.2008: announcement of 51 candidates accepted for Round 1

**2009**
- 25-28.II.2009: 1st SHA-3 Candidate Conference, Leuven, Belgium
  - 24 VII.2009: 14 Round 2 candidates announced

**2010**
- 23-24.VIII.2010: 2nd SHA-3 Candidate Conference, Santa Barbara, CA
  - 3 Q: selection of finalists

**2012**
- 1 Q: last workshop
- 2 Q: selection of the winner
- 3 Q: draft version of the standard published
- 4 Q: final version of the standard published

### Number of Submissions

- Number of submissions received by NIST: **64**
- Number of submissions publicly available: **56**
- Number of submissions qualified to the first round: **51**

### Basic Requirements for a new hash function

- Must support hash values of 224, 256, 384 and 512 bits
- Available worldwide without licensing fees
- Secure over tens of years
- Suitable for use in:
  - Digital signatures FIPS 186
  - Message authentication codes, HMAC, FIPS 198
  - Key agreement schemes, SP 800-56A
  - Random number generators, SP 800-90
- At least the same security level as SHA-2 with increased efficiency
1. Digital Signatures

**Advantages**

1. Shorter signature
2. Much faster computations
3. Larger resistance to manipulation (one block instead of several blocks of signature)
4. Resistance to the multiplicative attacks
5. Avoids problems with different sizes of the sender and the receiver moduli

2. Fingerprint of a program or a document (e.g., to detect a modification by a virus or an intruder)
Hash functions
Applications (3)

3. Storing passwords

Instead of:

ID, password

System stores:

ID, hash(password)

UNIX password scheme

"00000000"

hash(password, salt)

salt modifies the expansion function E of DES

Hash functions
Applications (4)

4. Fast encryption

PRNG

m_i

k_i

c_i

k_0 = hash(K_{AB} || IV )
k_1 = hash(K_{AB} || k_0)
.................
k_n = hash(K_{AB} || k_{n-1})

k_0 = hash(K_{AB} || IV )
k_1 = hash(K_{AB} || c_p)
.................
k_n = hash(K_{AB} || c_{p+1})
General scheme for constructing a secure hash function

Message m
Padding, appending bit length, M

M_1 \rightarrow f \rightarrow H_1
M_2 \rightarrow f \rightarrow H_2
\vdots
M_t \rightarrow f \rightarrow H_t

H_0 = IV
H_i = f(H_{i-1}, M_i)

h(m) = g(H_t)

In SHA-1
n=160
r=512

In MD5
n=128
r=512

Hash padding

<table>
<thead>
<tr>
<th>message</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000000000</td>
<td>64-bits</td>
</tr>
</tbody>
</table>

length of the entire message in bits

All zero padding: [X X X 0 0 0 0 0 X]
Correct padding: [X X X 0 0 1 0 0 X] [X X X 0 0 0 0 0] [X X X 0 0 0 0 0]
### Parameters of new hash functions

#### Features affecting security and functionality

<table>
<thead>
<tr>
<th></th>
<th>SHA-1</th>
<th>SHA-256</th>
<th>SHA-384</th>
<th>SHA-512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of hash value</td>
<td>160</td>
<td>256</td>
<td>384</td>
<td>512</td>
</tr>
<tr>
<td>Complexity of the best attack</td>
<td>$2^{80}$</td>
<td>$2^{128}$</td>
<td>$2^{192}$</td>
<td>$2^{256}$</td>
</tr>
<tr>
<td>Equivalently secure secret-key cipher</td>
<td>Skipjack</td>
<td>AES-128</td>
<td>AES-192</td>
<td>AES-256</td>
</tr>
<tr>
<td>Message size</td>
<td>$&lt; 2^{64}$</td>
<td>$&lt; 2^{64}$</td>
<td>$&lt; 2^{128}$</td>
<td>$&lt; 2^{128}$</td>
</tr>
</tbody>
</table>

#### Parameters of new hash functions

#### Features affecting implementation speed

<table>
<thead>
<tr>
<th></th>
<th>SHA-1</th>
<th>SHA-256</th>
<th>SHA-384</th>
<th>SHA-512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message block size</td>
<td>512</td>
<td>512</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Number of digest rounds</td>
<td>80</td>
<td>64</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

### Hardware implementations

#### Conceptual comparison

![Graph showing speed and area comparison for different hash functions](image)
**Hash functions**

**10 years ago**
- U.S. Government standards:
  - SHA-1
- Other popular hash functions:
  - MD5, RIPEMD
- Security status:
  - MD4 broken (1995)
  - SHA-1 replaced SHA-0 (1995)
  - MD6 partially broken (collisions in compression function, 1996)

**Present**
- U.S. Government standards:
  - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
- Other popular hash functions:
  - Whirlpool – winner of NESSIE
- Security status:
  - MD5 broken (1 hr on PC)
  - SHA-0 broken
  - RIPEMD broken (without a need for computer)
  - SHA-1 practically broken, best attack – $2^{63}$ operations – only 128 x more than breaking DES

**Hash functions**

**Timeline**

|---------------------------|------|------|------|------|------|------|------|------|------|------|------|

**Contests:**
- I. 2000
- Xll. 2003
- NNESSIE: SHA-256, SHA-384, SHA-512, Whirlpool

**Attacks:**
- MD5 – collisions for compression function, 10 hrs on PC
- VIII. 1998
- SHA-0 – attack with $2^{61}$ operations
- VIII. 2004
- broken: MD4, MD5, SHA-0, RIPEMD
- VIII. 2005
- attack on SHA-1
- $2^{61} = 2^{63}$ operations
Authentication

Alice

Message → MAC

Secret key algorithm

K_{AB}

Secret key of Alice and Bob

Bob

Message → MAC

Secret key algorithm

K_{AB}

Secret key of Alice and Bob

MAC

yes

MAC'

no

MAC - Message Authentication Codes (keyed hash functions)

arbitrary length

m

message

secret key

K

MAC function

MAC

fixed length

MAC functions

Basic requirements

1. Public description, SECRET key parameter

2. Compression
   arbitrary length input → fixed length output

3. Ease of computation
MAC functions

Security requirements

Given zero or more pairs
\[ m_i, \text{MAC}_K(m_i) \quad i = 1..k \]

it is computationally impossible to find any new pair
\[ m', \text{MAC}_K(m') \]

Such that
\[ m' \neq m_i \quad i = 1..k \]

MAC functions

Security requirements

Resistance against

1. Known-text attack
2. Chosen-text attack
3. Adaptive chosen-text attack

CBC-MAC (1)
CBC-MAC (1)

\[ H_0 = IV = 0 \]
\[ H_i = DES_K(m_i \oplus H_{i-1}) \quad i = 1..t \]

MAC(m) = \( H_t[1..32] \)
or
MAC(m) = \( E_K(E_K^{-1}(H_t))[1..32] \)

MAC functions

Based on block ciphers
- CBC-MAC
- CFB-MAC
- RIPE-MAC

Based on hash functions
- HMAC
- MDS-MAC

Dedicated
- MAA

Based on stream ciphers
- CRC-MAC

CMAC

Figure 12.12 Cipher-Based Message Authentication Code (CMAC)
**RIPE-MAC**

\[
\begin{align*}
H_0 &= IV = 0 \\
H_i &= DES_K(m_i \oplus H_{i-1}) \oplus m_i \quad i = 1..t \\
MAC(m) &= E_K(E_{K'}^{-1}(H_t))[0..31] \\
K' &= K \oplus 0xf0f0\ldots f0
\end{align*}
\]

**HMAC**

*Bellare, Canetti, Krawczyk, 1996*

*Used in SSL and IPSec*

\[HMAC(m) = h(K \oplus ipad \ || \ h(K \oplus opad \ || \ m))\]

ipad, opad - constant padding strings of the length of the message block size in the hash function h

ipad = repetitions of 0x36 = 00110110

opad = repetitions of 0x5A = 01011010

**HMAC**

- American standard
  - FIPS 198
- Arbitrary hash function and key size
Message Authentication Codes - MACs

10 years ago
U.S. Government standards:
MAC (DAC) based on DES (since 1985)

Number of certified implementations:
RIPE-MAC3, CRC-MAC, MAA

Other MACs in use:
RIPE-MAC3, CRC-MAC, MAA

Present
U.S. Government standards:
MAC (DAC) based on DES

Number of certified implementations:

Other MACs in use:
UMAC, TTMAC, EMAC
– winners of the NESSIE contest

NESSIE: Winners of the contest: 2002
Message Authentication Codes, MACs

<table>
<thead>
<tr>
<th>Security level</th>
<th>Key size</th>
<th>Output width</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>≥ 256</td>
<td>32-k</td>
</tr>
<tr>
<td>normal</td>
<td>≥ 128</td>
<td>32-k</td>
</tr>
</tbody>
</table>

Name     | Origin
---|---
1. UMAC       | UC Davis
2. TTMAC      | K.U. Leuven
3. EMAC       | U. of Toronto
4. HMAC       | NIST & NSA

Message Authentication Codes
Timeline

U.S. standards:
MAC (DAC) based on DES: 1985
HMAC based on hash functions: 2005
CMAC – block cipher mode (AES, Triple DES, Skipjack): 2005

Contests:
NESSIE: Contest winners: UMAC, TTMAC, EMAC

Attacks:
2002 RMAC – practical attack against MAC proposed by NIST and based on Triple DES

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S. standards:</td>
<td>V. 2004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC (DAC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>