Hash functions & MACs

- **Hash functions**
  - Basic requirements:
    1. Public description, NO key
    2. Compression
      - arbitrary length input → fixed length output
    3. Ease of computation

- **Hash functions**
  - Security requirements:
    - It is computationally infeasible
      - Given | To Find
        | 1. Preimage resistance | $y$
        | $x$, such that $h(x) = y$
        | 2. 2nd preimage resistance | $x$ and $y = h(x)$
        | $x' \neq x$, such that $h(x') = h(x) = y$
        | 3. Collision resistance | $x' \neq x$, such that $h(x') = h(x)$

- **Vocabulary**
  - hash function
  - hash value
    - message digest
    - hash total
    - fingerprint
    - imprint
    - cryptographic checksum
    - compressed encoding
    - MDC, Message Digest Code

- **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 functions
Dependence between requirements

- 2nd preimage resistant
- Collision resistant

Hash functions (unkeyed)

One-Way Hash Functions
OWHF
Collision-Resistant Hash Functions
CRHF

- Preimage resistance
- 2nd preimage resistance
- Collision resistance

Brute force attack against One-Way Hash Function

Given $y$

\[ m_i \]

\[ h \]

\[ h(m_i') = y \]

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

Creating multiple versions of the required message

I \{ state \ confirm \} \{ thereby \} that I \{ borrowed \} received

\{ $10,000 \} \{ ten thousand dollars \} \{ from \} \{ Mr. \} \{ Dr. \} \{ Krzysztof \}

Gaj on \{ November 22, \} \{ 11/22/2011 \}. This \{ money \} \{ sum of money \}

\{ should \} \{ is required to \} be \{ returned \} \{ given back \} to \{ Mr. \} \{ Dr. \} Gaj

by the \{ 8\text{th} \} \{ day of \} \{ December \} \{ Dec. \} \{ 2011 \}.

Brute force attack against Collision Resistant Hash Function

$r$ messages acceptable for the signer

$r$ messages required by the forger

\[ m_i \]

\[ h \]

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

Creating multiple versions of the required message

I \{ state \ confirm \} \{ thereby \} that I \{ borrowed \} received

\{ $10,000 \} \{ ten thousand dollars \} \{ from \} \{ Mr. \} \{ Dr. \} \{ Krzysztof \}

Gaj on \{ November 22, \} \{ 11/22/2011 \}. This \{ money \} \{ sum of money \}

\{ should \} \{ is required to \} be \{ returned \} \{ given back \} to \{ Mr. \} \{ Dr. \} Gaj

by the \{ 8\text{th} \} \{ day of \} \{ December \} \{ Dec. \} \{ 2011 \}. 
Message acceptable for the signer

I state thereby that on November 17, 2010, I borrowed from Mr. Kris Krzysztof a paper manuscript on security of voting machines, security of 3G and 4G telephony. This text item should be returned given back to Mr. Dr. Gaj by the 8th day of December 2011.

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)?

\[ \approx \frac{366}{2} = 188 \]

2. any two of the students share the same birthday (up to the day and month)?

\[ \approx \sqrt{366} \approx 19 \]

Brute force attack against Collision Resistant Hash Function

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

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

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

Hash value size

<table>
<thead>
<tr>
<th>One-Way</th>
<th>Collision-Resistant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Older algorithms:</td>
<td>Current algorithms:</td>
</tr>
<tr>
<td>( n \geq 64 ) 8 bytes</td>
<td>( n \geq 80 ) 10 bytes</td>
</tr>
<tr>
<td>( n \geq 128 ) 16 bytes</td>
<td>( n \geq 160 ) 20 bytes</td>
</tr>
<tr>
<td>8 bytes</td>
<td>16 bytes</td>
</tr>
<tr>
<td>16, 24, 32 bytes</td>
<td>32, 48, 64 bytes</td>
</tr>
</tbody>
</table>

Brute force attack against Collision Resistant Hash Function

Storage requirements

J.J. Quisquater

Collision search algorithm

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

Storage: Negligible
### Hash function algorithms

<table>
<thead>
<tr>
<th>Customized (dedicated)</th>
<th>Based on block ciphers</th>
<th>Based on modular arithmetic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD2</td>
<td>MDC-2</td>
<td>MASH-1</td>
</tr>
<tr>
<td>MD4</td>
<td>MDC-4</td>
<td></td>
</tr>
<tr>
<td>(Rivest 1990)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD5</td>
<td>SHA-0</td>
<td>RIPMD</td>
</tr>
<tr>
<td>(Rivest 1990)</td>
<td>NSA, 1992</td>
<td></td>
</tr>
<tr>
<td>SHA-1</td>
<td>NSA, 1995</td>
<td>RIPEMD-160</td>
</tr>
<tr>
<td>SHA-256, SHA-384, SHA-512</td>
<td>RSA, 2000</td>
<td></td>
</tr>
</tbody>
</table>

### Attacks against dedicated hash functions known by 2004

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

### What was discovered in 2004-2005?

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

### 2\(^{63}\) operations

**Schneier, 2005**

**In hardware:**

- Machine similar to the one used to break DES:
  - Cost = $50,000-$70,000
  - Time: 18 days
  - or
  - 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 impractical.

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 by 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
- 9.XII.2010: 5 Round 3 candidates announced

2012
- 22-23.III.2012: 3rd SHA-3 Candidate Conference, Washington, D.C.

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

Hash functions
Applications (1)
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

Hash functions
Applications (2)
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)

program
\[
\text{hash} \rightarrow \text{fingerprint} \rightarrow ? = \text{original_fingerprint}
\]
### UNIX password scheme

```
password  DES     salt
password  DES     salt
password  DES     salt
    hash(password, salt)
```

"00000000"

ID, salt, 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} || c_{n-1})
```

### General scheme for constructing a secure hash function

```
Message m
Padding, appending bit length, M

M_0  M_1  ...  M_t

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

h(m) = g(H_t)
```

*Compress function*

*Output transformation*

### Merkle-Damgard Scheme

```
M_0  M_1  M_2

IV  h_0  h_1  ...  h_{n-1}

h = h_{n-1}
```

### Parameters of the Merkle-Damgard Scheme

**Compression function**

```
M_i  f
H_{i+1}  H_i
```

- In SHA-1:
  - n = 160
  - r = 512
- In SHA-256:
  - n = 256
  - r = 512
- In SHA-512:
  - n = 512
  - r = 1024

**Entire hash**

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

```
h(m) = g(H_t)
```

### Sponge Scheme

```
p_0  p_1  p_2

absorbing  squeezing
```

```
c  f  ...  f
```

```
r  0  1  2
```

```
f  f  f  f  ...
```
Strengthened JH Scheme

Hash padding – SHA-1 & SHA-256

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

length of the entire message in bits

All zero padding: Correct padding:

<table>
<thead>
<tr>
<th>X X X 0 0 0 0 0</th>
<th>X X X 0 0 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X X X 0 0 0 0 0</td>
<td>X X X 1 0 0 0 0</td>
</tr>
</tbody>
</table>

Hash padding – SHA-3 Candidates

<table>
<thead>
<tr>
<th>Hash</th>
<th>D</th>
<th>1000 ... 0001</th>
<th>len64</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE256</td>
<td>D</td>
<td>1000 ... 0001</td>
<td>len64</td>
</tr>
<tr>
<td>Grostl</td>
<td>D</td>
<td>1000 ... 0000</td>
<td>#blocks</td>
</tr>
<tr>
<td>JH42</td>
<td>D</td>
<td>1000 ... 0001</td>
<td>len128</td>
</tr>
<tr>
<td>Keccak</td>
<td>D</td>
<td>1000 ... 0001</td>
<td></td>
</tr>
<tr>
<td>Skein</td>
<td>D</td>
<td>0000 ... 0000</td>
<td></td>
</tr>
<tr>
<td>SHA-2 (256)</td>
<td>D</td>
<td>1000 ... 0000</td>
<td>len64</td>
</tr>
</tbody>
</table>

Parameters of new hash functions

<table>
<thead>
<tr>
<th>Features affecting security and functionality</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>$2^{160}$</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>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>
</tr>
<tr>
<td>Number of digest rounds</td>
<td>80</td>
<td>64</td>
<td>80</td>
</tr>
</tbody>
</table>

Hardware implementations

Conceptual comparison

Speed

SHA-1

SHA-256

SHA-512, SHA-384
**Results of the prototype FPGA implementation**

**Hash functions**

<table>
<thead>
<tr>
<th>Complexity of the best attack</th>
<th>U.S. Government standards:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>SHA-1, SHA-224, SHA-256, SHA-384, SHA-512</td>
</tr>
<tr>
<td>SHA-512</td>
<td>SHA-1 replaced SHA-0 (1995)</td>
</tr>
<tr>
<td></td>
<td>(collisions in compression function, 1996)</td>
</tr>
</tbody>
</table>

**U.S. Government standards:**
- SHA-1
- SHA-224, SHA-256, SHA-384, SHA-512

**Other popular hash functions:**
- MD5, RIPEMD

**Security status:**
- MD4 broken (1995)
- SHA-0 broken
- RIPEMD broken (without a need for computer)
- SHA-1 practically broken, best attack – $2^{63}$ operations – only $128 \times$ more than breaking DES

**U.S. Governemnt 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 $128 \times$ more than breaking DES

**Timeline**

<table>
<thead>
<tr>
<th>U.S. Government standards:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1 FIPS 180 I. 2003</td>
</tr>
<tr>
<td>SHA-224 FIPS 180 I. 2004</td>
</tr>
<tr>
<td>SHA-256, 384, 512 FIPS 180-2 II. 2003</td>
</tr>
<tr>
<td>SHA-384, 512 FIPS 180-2 II. 2004</td>
</tr>
</tbody>
</table>

**Contests:**
- SHA-1 FIPS 180 I. 2003
- SHA-224 FIPS 180-2 II. 2004
- SHA-256, 384, 512 FIPS 180-2 II. 2004

**Attacks:**
- MD5 – collisions for compression function, 10 hrs on PC
- SHA-0 – attack with $2^{64}$ operations
- SHA-0 replaced SHA-0 (1995)
- SHA-1 practically broken, best attack – $2^{63}$ operations – only $128 \times$ more than breaking DES

**MAC - Message Authentication Codes**

**Basic requirements**

1. Public description, SECRET key parameter
2. Compression
   - arbitrary length input $\rightarrow$ fixed length output
3. Ease of computation

**Authentication**

- Alice
  - Message
  - MAC
  - Secret key of Alice and Bob
  - $K_{AB}$

- Bob
  - Message
  - MAC
  - Secret key of Alice and Bob
  - $K_{AB}$

- $\text{MAC'\ does not match MAC}$
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

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

Resistance against

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

MAC functions

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

Based on hash functions
- HMAC
- MD5-MAC
- MAA

Dedicated
- CMAC

Based on stream ciphers
- CRC-MAC

CMAC
RIPE-MAC

\[ H_0 = IV = 0 \]
\[ H_i = \text{DES}_K(m_i \oplus H_{i-1}) \oplus m_i \quad i = 1..t \]
\[ \text{MAC}(m) = E_K(E_{K'}^{-1}(H_t))[0..31] \]
\[ K' = K \oplus 0xf0f0\ldots f0 \]

HMAC

Bellare, Canetti, Krawczyk, 1996

Used in SSL and IPSec

\[ \text{HMAC}(m) = h(K \oplus \text{ipad} \ || \ h(K \oplus \text{opad} \ || \ m)) \]
\[ \text{ipad} = \text{repetitions of } 0x36 = 00110110 \]
\[ \text{opad} = \text{repetitions of } 0x5A = 01011010 \]

Message Authentication Codes - MACs

10 years ago

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

HMAC – based on hash functions used in SSL and IPSec
CMAC – block cipher mode (AES, Triple DES, Skipjack)

Number of certified implementations:
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>

<table>
<thead>
<tr>
<th>Name</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMAC</td>
<td>UC Davis</td>
</tr>
<tr>
<td>TTMAC</td>
<td>K.U. Leuven</td>
</tr>
<tr>
<td>EMAC</td>
<td>U. of Toronto</td>
</tr>
<tr>
<td>HMAC</td>
<td>NIST &amp; NSA</td>
</tr>
</tbody>
</table>

Message Authentication Codes Timeline

- U.S. standards: FIPS 113 (based on DES)
- HMAC: FIPS 198 (based on hash functions)
- CMAC: SP 800-38C

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

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