ECE 646 Lecture 11

Hash functions & MACs

Digital Signature

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

Bob
- Message
- Signature
- Public key algorithm
- Hash function
- Hash value 1
- Hash value 2

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>fingerprint</td>
</tr>
<tr>
<td>imprint</td>
<td>cryptographic checksum</td>
</tr>
<tr>
<td>compressed encoding</td>
<td>MDC, Message Digest Code</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>$x$, such that $h(x) = y$</td>
<td>$y$</td>
</tr>
<tr>
<td>$x' \neq x$, such that $h(x') = y$</td>
<td>$x'$</td>
</tr>
<tr>
<td>$x' \neq x$, such that $h(x') = h(x)$</td>
<td>$x'$</td>
</tr>
</tbody>
</table>
Hash functions

Dependence between requirements

- 2nd preimage resistant
- Collision resistant

One-Way Hash Functions
- Collision-Resistant Hash Functions

OWHF

- Preimage resistance
- 2nd preimage resistance
- Collision resistance

CRHF

Brute force attack against One-Way Hash Function

Given $y$

Given $y$

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

$h(m_i) = y$ for $i = 1, 2^n$

Creating multiple versions of the required message

- I state confirm thereby that I borrowed received
- $10,000$ from Mr. Dr. Kris Krzysztof
- Gaj on November 19, 2008
- This money sum of money
- I state confirm thereby that I borrowed received
- $10,000$ from Mr. Dr. Kris Krzysztof
- Gaj on November 19, 2008
- This money sum of money
- I state confirm thereby that I borrowed received
- $10,000$ from Mr. Dr. Kris Krzysztof
- Gaj on November 19, 2008
- This money sum of money

Message required by the forger

- I state confirm thereby that I borrowed received
- $10,000$ from Mr. Dr. Kris Krzysztof
- Gaj on November 19, 2008
- This money sum of money
- I state confirm thereby that I borrowed received
- $10,000$ from Mr. Dr. Kris Krzysztof
- Gaj on November 19, 2008
- This money sum of money
- I state confirm thereby that I borrowed received
- $10,000$ from Mr. Dr. Kris Krzysztof
- Gaj on November 19, 2008
- This money sum of money

Brute force attack against Collision Resistant Hash Function

- $r$ messages acceptable for the signer
- $m_i$ $i = 1, r$

- $r$ messages required by the forger
- $m_j^*$ $j = 1, r$

$h(m_i) = h(m_j^*)$ for $i = 1, r, j = 1, r$
I state and confirm thereby that on November 19, 2008, I borrowed from Dr. Krzysztof on November 19, 2008, a book implementing hash functions. This text item should be returned to Mr. Gaj by the 27th day of November 2008.

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 (day and month)?
   \[ \sqrt{366} \approx 19 \]
   \[ \sim \frac{366}{2} = 188 \]

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

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

One-Way Collision-Resistant

Older algorithms:
\[ n \geq 64 \quad n \geq 128 \]
8 bytes 16 bytes

Current algorithms:
\[ n \geq 80 \quad n \geq 160 \]
10 bytes 20 bytes

Newly proposed algorithms:
\[ n = 128, 192, 256 \quad n = 256, 384, 512 \]
16, 24, 32 bytes 32, 48, 64 bytes

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 (day and month)?

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

Brute force attack against Collision Resistant Hash Function

Storage requirements

J.J. Quisquater

collision search algorithm

Number of operations:
\[ 2^{n/2} \cdot 2^{n/2} \approx 2.5 \cdot 2^{n/2} \]

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>MD2, MD4</td>
<td>MDC-2, MDC-4</td>
<td>MASH-1</td>
</tr>
<tr>
<td>IBM, Bracht, Meyer, Schilling, 1988</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD5</td>
<td>SHA-0, SHA-1</td>
<td>RIPEMD</td>
</tr>
<tr>
<td>Rivest 1990</td>
<td>NSA, 1992</td>
<td>European RACE Integrity Primitives Evaluation Project, 1992</td>
</tr>
<tr>
<td>SHA-256, SHA-384, SHA-512</td>
<td>NSA, 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, Wang, Feng, Lai, Yu, Crypto 2004 (manually, without using a computer)
- MD5: partially broken, Dobbertin, 1996 (10 hours on PC) weakened, collisions for the compression function, Dobbertin, 1998 (10 hours on PC), NSA, 1995, France
- SHA-0: reduced round version broken, Dobbertin 1995
- SHA-1: weakness discovered, NSF, 1998 France
- 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: partial break, Wang, Feng, Lai, Yu, Crypto 2004 (1 hr on a PC)
- SHA-0: attack with 2^63 operations, Wang, Yin, Yu, Aug 2005 (manually, without using a computer)


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 impracticable.

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.

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

Schneier, 2005
SHA-3 Contest Timeline

2007
• publication of requirements
  • 29.X. 2007: request for candidates

2008
• 31.X. 2008: deadline for submitting candidates

2009
2 Q: first workshop devoted to the presentation of candidates

2010
2 Q: second workshop devoted to the analysis of candidates
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

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)

UNIX password scheme

Hash functions
Applications (4)

4. Fast encryption

PRNG

m_i \xrightarrow{salt} k_i \xrightarrow{c_i}

k_0 = \text{hash(K}_{AB} \| IV )
\begin{align*}
  k_0 &= \text{hash(K}_{AB} \| IV ) \\
  k_1 &= \text{hash(K}_{AB} \| k_0) \\
  &\vdots \\
  k_n &= \text{hash(K}_{AB} \| k_{n-1}) \\
  \end{align*}

k_0 = \text{hash(K}_{AB} \| IV )
\begin{align*}
  k_0 &= \text{hash(K}_{AB} \| IV ) \\
  k_1 &= \text{hash(K}_{AB} \| c_0) \\
  &\vdots \\
  k_n &= \text{hash(K}_{AB} \| c_{n-1}) \\
  \end{align*}

\begin{align*}
  hash(password) \\
  hash(password, salt) \\
  hash(password, salt) \\
  hash(password, salt)
\end{align*}
General scheme for constructing a secure hash function

Message m
Padding, appending bit length, M

Compression function

Entire hash

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

h(m) = g(H_t)

Parameters of dedicated hash functions

<table>
<thead>
<tr>
<th>name</th>
<th># bits of hash value</th>
<th># bits of message block</th>
<th>no. of rounds (steps)</th>
<th>speed relative to MD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD4</td>
<td>128</td>
<td>512</td>
<td>3 x 16</td>
<td>1.00</td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>512</td>
<td>4 x 16</td>
<td>0.68</td>
</tr>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>512</td>
<td>4 x 20</td>
<td>0.28</td>
</tr>
<tr>
<td>RIPEMD</td>
<td>128</td>
<td>512</td>
<td>4 x 16</td>
<td>0.39</td>
</tr>
<tr>
<td>RIPEMD-160</td>
<td>160</td>
<td>512</td>
<td>5 x 16</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Parameters of new hash functions

Features affecting security and functionality

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>Complexity of the best attack</td>
<td>2^{80}</td>
<td>2^{128}</td>
<td>2^{192}</td>
</tr>
<tr>
<td>Equivalently secure secret-key cipher</td>
<td>Skipjack</td>
<td>AES-128</td>
<td>AES-192</td>
</tr>
<tr>
<td>Message size</td>
<td>&lt; 2^{64}</td>
<td>&lt; 2^{64}</td>
<td>&lt; 2^{128}</td>
</tr>
</tbody>
</table>

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>

Hash padding

<table>
<thead>
<tr>
<th>message</th>
<th>100000000000</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>length of the entire message in bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All zero padding: X X X 0 0 0
Correct padding: X X X 0 0 0 0

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>
### Hash functions

**10 years ago**
- **U.S. Government standards:** SHA-1
- **Other popular hash functions:** MD5, RIPEMD
- **Security status:**
  - MD5 broken (1995)
  - SHA-1 replaced SHA-0 (1995)
  - SHA-0 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 $128 \times$ more than breaking DES

### Hash functions Timeline

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>Contest I. 2000</td>
</tr>
<tr>
<td>1997</td>
<td>Contest XII. 2002</td>
</tr>
<tr>
<td>1998</td>
<td>Contest VIII. 1998</td>
</tr>
<tr>
<td>1999</td>
<td>Contest VIII. 2000</td>
</tr>
<tr>
<td>2000</td>
<td>Contest II. 2001</td>
</tr>
<tr>
<td>2001</td>
<td>Contest II. 2003</td>
</tr>
<tr>
<td>2002</td>
<td>Contest II. 2003</td>
</tr>
<tr>
<td>2003</td>
<td>Contest II. 2003</td>
</tr>
<tr>
<td>2004</td>
<td>Contest II. 2004</td>
</tr>
<tr>
<td>2005</td>
<td>Contest II. 2005</td>
</tr>
<tr>
<td>2006</td>
<td>Contest II. 2005</td>
</tr>
</tbody>
</table>

### Message Authentication Codes (keyed hash functions)

**Alice**

- Secret key of Alice and Bob

**Bob**

- Secret key of Alice and Bob

**MAC**

- Message Authentication Codes

**MAC - Message Authentication Codes**

- **Arbitrary length**
  - secret key $K$
  - Message $m$
  - MAC function
  - MAC

- **Fixed length**
  - secret key $K$
  - Message $m$
  - MAC function
  - MAC
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)

\[
H_0 = IV = 0 \\
H_i = \text{DES}_K(m_i \oplus H_{i-1}) \quad i = 1..t \\
\text{MAC}(m) = H_t[1..32] \\
\text{or} \\
\text{MAC}(m) = E_K(E_K^{-1}(H_t))[1..32]
\]

MAC functions

Based on block ciphers

- CBC-MAC
- CFB-MAC
- RIPE-MAC

Based on hash functions

- HMAC
- MD5-MAC
- CRC-MAC

Based on dedicated ciphers

- MAA

Based on stream ciphers

- FIPS-113
**CMAC**

- **RIPE-MAC**

\[
H_0 = IV = 0 \\
H_i = DES_k(m_i \oplus H_{i-1}) \oplus m_i \quad i = 1..t \\
MAC(m) = E_k(E_k^{-1}(H_t))[0..31] \\
K' = K \oplus 0xf0f0...f0
\]

**HMAC**

_Bellare, Canetti, Krawczyk, 1996_

_used in SSL and IPSec_

HMAC(m) = h(K \oplus \text{ipad} \parallel h(K \oplus \text{opad} \parallel 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

**Message Authentication Codes - MACs**

- **10 years ago**
  - U.S. Government standards:
    - MAC (DAC) based on DES (since 1985)
  - Number of certified implementations:
    - Other MACs in use:
      - RIPE-MAC3, CRC-MAC, MAA

- **Present**
  - U.S. Government standards:
    - MAC (DAC) based on DES
    - 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:
      - UMAC, TTMAC, EMAC – winners of the NESSIE contest

**NESSIE: Winners of the contest: 2002**

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

- Name
  - 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)
  - V. 2004
- HMAC
  - 1996
- CMAC
  - 2002
  - V. 2005

Contests:
- NNESSIE
  - Contest winners:
    - UMAC, TTMAC, EMAC

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