ECE 646 Lecture 11

Hash functions & MACs

Required Reading

W. Stallings, "Cryptography and Network-Security,"

Chapter 11 Cryptographic Hash Functions

Appendix 11A Mathematical Basis of Birthday Attack

Chapter 12 Message Authentication Codes

Recommended Reading

SHA-3 Competition 2007-2012
http://csrc.nist.gov/groups/ST/hash/sha-3
Digital Signature

Alice

Message → Signature

Hash function → Hash value

Public key algorithm → Alice’s private key

Bob

Message → Signature

Hash function → Hash value 1

yes → Hash value 2

no → Public key algorithm → Alice’s public key

Hash function

arbitrary length

message

h

hash function

h(m) hash value

fixed length
**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></td>
<td>hash total</td>
</tr>
<tr>
<td></td>
<td>fingerprint</td>
</tr>
<tr>
<td></td>
<td>imprint</td>
</tr>
<tr>
<td></td>
<td>cryptographic checksum</td>
</tr>
<tr>
<td></td>
<td>compressed encoding</td>
</tr>
<tr>
<td></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
## Hash functions

**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>
</tbody>
</table>

1. **Preimage resistance**

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 $h(x') = h(x)$

---

## Hash functions

**Dependence between requirements**

- 2nd preimage resistant
- Collision resistant
Hash functions (unkeyed)

One-Way Hash Functions
OWHF
- preimage resistance
- 2nd preimage resistance
- collision resistance

Collision-Resistant Hash Functions
CRHF

Brute force attack against One-Way Hash Function

Given $y$

\[ m_i' \]

\[ h \]

\[ h(m_i') = y \]

\[ i = 1 \ldots 2^n \]

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

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

I state confirm thereby that I borrowed received

$10,000 ten thousand dollars from Mr. Dr. Kris Krzysztof

Gaj on December 2, 12 / 02 / 2014. This money sum of money

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

by the 16th sixteenth day of December Dec. 2014.

Brute force attack against Collision Resistant Hash Function

Yuval

$r$ messages acceptable for the signer

\[ m_i \xrightarrow{h} h(m_i) \]

\[ n \text{- bits} \]

\[ h(m_i) = h(m_j') \]

$r$ messages required by the forger

\[ m_i' \xrightarrow{h} h(m_i') \]

\[ n \text{- bits} \]
Creating multiple versions of the required message

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

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

\text{Gaj on} \{\text{December 2,}\ \text{12/02/2014}\} \text{2014.} \text{This} \{\text{money}\ \text{sum of money}\}

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

\text{by the}\ \{\text{16th}\ \text{sixteenth}\} \text{day of} \{\text{December}\ \text{Dec.}\} \text{2014.}

Message acceptable for the signer

I \{\text{state}\ \text{confirm}\} \{\text{thereby}\} \text{that on} \{\text{December 2,}\ \text{12/02/2014}\}

I \{\text{borrowed}\ \text{received}\} \text{from} \{\text{Mr.}\ \text{Dr.}\ \text{Kris}\ \text{Krzysztof}\} \text{a} \{\text{paper manuscript}\}

\text{on} \{\text{security of biometric passports. security of text messaging.}\} \text{This} \{\text{item text}\}

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

\text{by the}\ \{\text{16th}\ \text{sixteenth}\} \text{day of} \{\text{December}\ \text{Dec.}\} \text{2014.}
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)?

$\sim \frac{366}{2} = 188$

$\sim \sqrt{366} \sim 19$
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 $r = 2^{n/2}$, $p = 63\%$

---

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
<table>
<thead>
<tr>
<th>Hash value size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-Way</strong></td>
</tr>
<tr>
<td>Older algorithms:</td>
</tr>
<tr>
<td>( n \geq 64 )</td>
</tr>
<tr>
<td>8 bytes</td>
</tr>
<tr>
<td>Old standards (e.g., SHA-1):</td>
</tr>
<tr>
<td>( n \geq 80 )</td>
</tr>
<tr>
<td>10 bytes</td>
</tr>
<tr>
<td>Current standards (e.g., SHA-2, SHA-3):</td>
</tr>
<tr>
<td>( n = 128, 192, 256 )</td>
</tr>
<tr>
<td>16, 24, 32 bytes</td>
</tr>
</tbody>
</table>

### 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 ( \text{Rivest 1988} )</td>
<td>MDC-2, MDC-4 ( \text{IBM, Bracht, Meyer, Schilling, 1988} )</td>
<td>MASH-1 ( \text{1988-1996} )</td>
</tr>
<tr>
<td>MD4 ( \text{Rivest 1990} )</td>
<td>( \text{SHA-0 NSA, 1992} )</td>
<td>( \text{RIPEMD NSA, 1992} )</td>
</tr>
<tr>
<td>MD5 ( \text{Rivest 1990} )</td>
<td>( \text{SHA-1 NSA, 1995} )</td>
<td>( \text{RIPEMD-160 NSA, 2000} )</td>
</tr>
<tr>
<td>( \text{SHA-256, SHA-384, SHA-512} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Attacks against dedicated hash functions known by 2004

- MD2: partially broken
- MD4: broken, H. Dobbertin, 1995 (one hour on PC, 20 free bytes at the start of the message)
- MD5: partially broken, collisions for the compression function, Dobbertin, 1996 (10 hours on PC)
- SHA-0: weakness discovered, 1995 NSA, 1998 France
- SHA-1: reduced round version broken, Dobbertin 1995
- 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)
- MD5: broken; Wang, Feng, Lai, Yu, Crypto 2004 (1 hr on a PC)
- SHA-0: attack with 2^{40} operations, Crypto 2004
- SHA-1: attack with 2^{63} operations, Wang, Yin, Yu, Aug 2005
- SHA-256, SHA-384, SHA-512

- RIPEMD: broken; Wang, Feng, Lai, Yu, Crypto 2004 (manually, without using a computer)
- RIPEMD-160

- SHA-256, SHA-384, SHA-512
### 2⁶³ 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 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 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.X.2012: selection of the winner

2013: draft 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)

program

\[ \text{hash} \]

fingerprint

safe place

? = original_fingerprint
3. Storing passwords

Instead of:
ID, password

System stores:
ID, hash(password)

UNIX password scheme

“00000000”

ID, salt,
hash(password, salt)

salt modifies the
expansion function E
of DES

hash(password, salt)
4. Fast encryption

**General scheme for constructing a secure hash function**

Message $m$

Padding, appending bit length, $M$

$M_1$, $M_2$, $\ldots$, $M_t$

$H_0$, $H_1$, $H_2$, $\ldots$, $H_t$

Output transformation

**Hash functions**

Applications (4)

$k_0 = \text{hash}(K_{AB} \ || \ IV)$

$k_1 = \text{hash}(K_{AB} \ || \ k_0)$

$\ldots$

$k_n = \text{hash}(K_{AB} \ || \ k_{n-1})$

or

$k_0 = \text{hash}(K_{AB} \ || \ IV)$

$k_1 = \text{hash}(K_{AB} \ || \ c_0)$

$\ldots$

$k_n = \text{hash}(K_{AB} \ || \ c_{n-1})$
Merkle-Damgard Scheme

\[ H_0 = IV \]
\[ H_i = f(H_{i-1}, M_i) \]
\[ h(m) = g(H_t) \]

Parameters of the Merkle-Damgard Scheme

**Compression function**

- In SHA-1:
  - \( n = 160 \)
  - \( r = 512 \)

- In SHA-256:
  - \( n = 256 \)
  - \( r = 512 \)

- In SHA-512:
  - \( n = 512 \)
  - \( r = 1024 \)
Sponge Scheme

Hash padding – SHA-1 & SHA-256

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

length of the entire message in bits

All zero padding:

- $\overline{X X X 0 0 0 0 0}$
- $\overline{X X X 0 0 0 0 0}$

Correct padding:

- $\overline{X X X 0 0 1 0 0}$
- $\overline{X X X 1 0 0 0 0}$
Hash padding – SHA-3 Candidates

<table>
<thead>
<tr>
<th>Hash Function</th>
<th>D</th>
<th>M</th>
<th>P</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE256</td>
<td>D</td>
<td>1000...0001</td>
<td>len64</td>
<td></td>
</tr>
<tr>
<td>Grøstl</td>
<td>D</td>
<td>1000...0000</td>
<td>#blocks</td>
<td></td>
</tr>
<tr>
<td>JH42</td>
<td>D</td>
<td>1000...0001</td>
<td>len128</td>
<td></td>
</tr>
<tr>
<td>Keccak</td>
<td>D</td>
<td>1000...0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skein</td>
<td>D</td>
<td>0000...0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA–2 (256)</td>
<td>D</td>
<td>1000...0000</td>
<td>len64</td>
<td></td>
</tr>
</tbody>
</table>

D = Data, M = Minimum Padding, P = Padding, C = Counter

Parameters of new hash functions
Features affecting security and functionality

<table>
<thead>
<tr>
<th>Feature</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 birthday 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><strong>Message block size</strong></td>
<td>512</td>
<td>512</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td><strong>Number of digest rounds</strong></td>
<td>80</td>
<td>64</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

Hardware implementations
Conceptual comparison

![Diagram showing speed and area comparison]

- SHA-1
- SHA-256
- SHA-512, SHA-384
Results of the prototype FPGA implementation

GMU, 2002

<table>
<thead>
<tr>
<th>Complexity of the best attack</th>
<th>Speed in hardware [Mbit/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>462</td>
</tr>
<tr>
<td>SHA-512</td>
<td>616</td>
</tr>
<tr>
<td>the same as</td>
<td></td>
</tr>
<tr>
<td>$2^{80}$</td>
<td>$2^{256}$</td>
</tr>
<tr>
<td><strong>Skipjack</strong></td>
<td><strong>AES-256</strong></td>
</tr>
</tbody>
</table>

Hash functions

15 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)
- MD5 partially broken (collisions in compression function, 1996)

Present

U.S. Government standards:
- SHA-1,
- SHA-224, SHA-256, SHA-384, SHA-512
- SHA-3 (draft FIPS)

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

**U.S. Government standards:**

- SHA-1: FIPS 180
- SHA-256, 384, 512: FIPS 180-2
- SHA-224: FIPS 180-2

**Contests:**

- I. 2000: SHA-0

**Attacks:**

- VIII. 1998: SHA-0 – attack with $2^{61}$ operations
- II-VIII. 2005: attack on SHA-1 with $2^{63}$ operations
- VIII. 2004: broken: MD4, MD5, SHA-0, RIPEMD

**Authentication**

**Alice**

- Message
- MAC
- Secret key algorithm
- $K_{AB}$

**Bob**

- Message
- MAC
- Secret key algorithm
- $K_{AB}$

**Secret key of Alice and Bob**

**MAC**

**MAC’**

**yes**

**no**
**MAC - Message Authentication Codes**  
*(keyed hash 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

\[ m_i, MAC_K(m_i) \quad i = 1..k \]

it is computationally impossible to find any new pair

\[ m', MAC_K(m') \]

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)

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

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
  - MD5-MAC
- Dedicated
  - MAA
- Based on stream ciphers
  - CRC-MAC

CMAC

(a) Message length is integer multiple of block size
(b) Message length is not integer multiple of block size

Figure 12.12 Cipher-Based Message Authentication Code (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

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

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
**HMAC**

\[
\begin{align*}
\text{KEY} & \oplus \text{opad} \\
\text{=} & \\
\text{KEY'} & \text{message m} \\
\text{KEY} & \oplus \text{ipad} \\
\text{=} & \\
\text{KEY’’} & \\
\h & \\
\text{HMAC} &
\end{align*}
\]

- American standard FIPS 198
- Arbitrary hash function and key size

---

**Message Authentication Codes - MACs**

**15 years ago**

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

**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)

**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) FIPS 113 (based on DES)
- HMAC FIPS 198 (based on hash functions)
- CMAC SP 800-38C
- V. 2004

Contests:

NESSIE Contest winners: UMAC, TTMAC, EMAC

Attacks:

2002: RMAC – practical attack against MAC proposed by NIST and based on Triple DES
Confidentiality & Authentication
Authenticated Ciphers

Bob

\[ K_{AB} \rightarrow \text{Authenticated Cipher} \]

\[ \text{IV} \quad \text{Message} \rightarrow \text{Ciphertext} \quad \text{Tag} \]

Alice

\[ K_{AB} \rightarrow \text{Authenticated Cipher} \]

\[ \text{IV} \quad \text{Ciphertext} \quad \text{Tag} \rightarrow \text{valid} \]

\[ K_{AB} - \text{Secret key of Alice and Bob} \]
\[ \text{IV} - \text{Initialization Vector} \]
Examples of Most Commonly Used Authenticated Ciphers

- AES-GCM
- AES-OCB3
- AES-OCB
- AES-CCM
- AES-EAX