Hashes and Message Digests

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:
http://www.cse.wustl.edu/~jain/cse571-07/
Overview

- One-Way Functions
- Birthday Problem
- Probability of Hash Collisions
- Authentication and encryption using Hash
- Sample Hashes: MD2, MD4, MD5, SHA-1, SHA-2
- HMAC
One-Way Functions

- Hash = Message Digest
  = one way function
    - Computationally infeasible to find the input from the output
    - Computationally infeasible to find the two inputs for the same output
One-Way Functions (Cont)

- Easy to compute but hard to invert
- If you know both inputs it is easy to calculate the output
- It is unfeasible to calculate any of the inputs from the output
- It is unfeasible to calculate one input from the output and the other input
Examples of Hash Functions

- MD2 = Message Digest 2 [RFC 1319] - 8b operations
- Snefru = Fast hash named after Egyptian king
- MD4 = Message Digest 4 [RFC 1320] - 32b operations
- Snefru 2 = Designed after Snefru was broken
- MD5 = Message Digest 5 [RFC 1321] - 32b operations
- SHA = Secure hash algorithm [NIST]
- SHA-1 = Updated SHA
- SHA-2 = SHA-224, SHA-256, SHA-384, SHA-512
  SHA-512 uses 64-bit operations
- HMAC = Keyed-Hash Message Authentication Code
Birthday Problem

- What is the probability that two people have the same birthday (day and month)

<table>
<thead>
<tr>
<th>K</th>
<th>Total</th>
<th>Different</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$365^2$</td>
<td>$365 \times 364$</td>
</tr>
<tr>
<td>3</td>
<td>$365^3$</td>
<td>$365 \times 364 \times 363$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>k</td>
<td>$365^k$</td>
<td>$365 \times 364 \times 363 \times \cdots \times (365 - k + 1)$</td>
</tr>
</tbody>
</table>

$$P(\text{No common day}) = \frac{365 \times 364 \times 363 \times \cdots \times (365 - k + 1)}{365^k}$$

$$= \frac{365!}{365^k (365 - k)!}$$
Birthday Problem (Cont)

- With 22 people in a room, there is better than 50% chance that two people have a common birthday.
- With 40 people in a room there is almost 90% chance that two people have a common birthday.
- If there are k people, there are \( \frac{k(k-1)}{2} \) pairs.
- \[ P(1 \text{ pair having common birthday}) = \frac{k(k-1)}{2 \times 365} \]
  \[ k \geq \sqrt{365} \Rightarrow P > 0.5 \]
- In general, \( n \) possibilities \( \Rightarrow \sqrt{n} \) trials to find a collision.

<table>
<thead>
<tr>
<th>k</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>.01</td>
</tr>
<tr>
<td>3</td>
<td>.02</td>
</tr>
<tr>
<td>4</td>
<td>.03</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>.41</td>
</tr>
<tr>
<td>20</td>
<td>.44</td>
</tr>
<tr>
<td>21</td>
<td>.48</td>
</tr>
<tr>
<td>22</td>
<td>.51</td>
</tr>
<tr>
<td>23</td>
<td>.54</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>.88</td>
</tr>
<tr>
<td>39</td>
<td>.89</td>
</tr>
<tr>
<td>40</td>
<td>.90</td>
</tr>
</tbody>
</table>
Probability of Hash Collisions

- Arbitrary length message $\Rightarrow$ Fixed length hash
  $\Rightarrow$ Many messages will map to the same hash
- Given 1000 bit messages $\Rightarrow 2^{1000}$ messages
- 128 bit hash $\Rightarrow 2^{128}$ possible hashes
  $\Rightarrow \frac{2^{1000}}{2^{128}} = 2^{872}$ messages/hash value
- n-bit hash $\Rightarrow$ Need avg $2^{n/2}$ tries to find two messages with same hash
- 64 bit hash $\Rightarrow 2^{32}$ tries (feasible)
- 128 bit hash $\Rightarrow 2^{64}$ tries (not feasible)
Authentication using Hash

Anyone can compute $MD(m)$
$\Rightarrow$ Need to send shared secret $K_{AB}$

Message is split into blocks.
Digest of n-1 is used with block n

Issue: Anyone can append to the message

Solution:
- Put shared secret at the end
- Send only part of the MAC
- Put shared secret at both front and back $\Rightarrow$ Keyed Hash
Encryption Using Hash

- Use shared secret to generate hash
- Continually hash the hash to generate one-time pad
- XoR the pad to message
- Issue: If some one knows the plain text, they can compute the pad and use it to send another message
- Solution:
  - Use IV
  - Use cipher block chaining
Encryption Using Hash (Cont)
Hash Using Encryption

- Use the message as a key to encrypt a constant
- Unix Password Hash
  - ASCII 7-bits of 8 characters are used as 56bit DES key
- Issue: Can hash a large number of words and see if anyone matches from a set
- Solution: Use a different IV
  - Hash(IV|password).
  - IV is stored in clear.
  - IV = Salt
Hashing Large Messages

- Break the message into fixed size blocks

Diagram:

\[ \text{constant} \]

\[ m_1 \quad \text{key} \quad \text{encrypt} \]

\[ m_2 \quad \text{key} \quad \text{encrypt} \]

\[ \vdots \]

\[ \text{message digest} \]
Hashing Large Messages (Cont)

- Issue: DES produces 64-bit digest $\Rightarrow 2^{32}$ tries to find collision

- Solution:
  1. Xor with input in each round
  2. Get 128 bit using DES twice - forward, reverse
MD2 Hash

- Produces 128-bit hash using 128 bit blocks
- Designed by Ron Rivest in 1989
- Described in RFC 1319
- Used in certificates generated with MD2 and RSA
- Examples:
  - MD2("The quick brown fox jumps over the lazy dog")
    = 03d85a0d629d2c442e987525319fc471
  - MD2("The quick brown fox jumps over the lazy cog")
    = 6b890c9292668cddbfa00a4ebf31f05
MD2 Algorithm Steps

1. Padding: Message is padded to make it 16n octets.
2. Checksum: A 16 octet checksum is computed and appended.
3. Final Pass: 16(n+1) octets are hashed using 18 rounds.

- Padding: padded bytes contain length of pad
  Always pad (even if a multiple of 16).

<table>
<thead>
<tr>
<th>Original Message</th>
<th>R octets (1 ≤ r ≤ 16) each containing r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiple of 16 octets</td>
</tr>
</tbody>
</table>
Why Always Pad?

- Message 1 is not a multiple of 16 and so it is padded with 15 in 15 bytes
- Message 2 is a multiple of 16 but contains 15
- If message 2 is not padded, both these messages will have hash. \( \Rightarrow \) It is trivial to find collision.
MD2 Checksum

padded message

nth octet

π substitution

16-octet checksum

(n−1 mod 16)th octet

(n mod 16)th octet
MD2 $\pi$ Substitution Table

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 41 | 46 | 67 | 201 | 162 | 216 | 124 | 1 | 61 | 54 | 84 | 161 | 236 | 240 | 6 | 19 |
| 98 | 167 | 5 | 243 | 192 | 199 | 115 | 140 | 152 | 147 | 43 | 217 | 188 | 76 | 130 | 202 |
| 30 | 155 | 87 | 60 | 253 | 212 | 224 | 22 | 103 | 66 | 111 | 24 | 138 | 23 | 229 | 18 |
| 190 | 78 | 196 | 214 | 218 | 158 | 222 | 73 | 160 | 251 | 245 | 142 | 187 | 47 | 238 | 122 |
| 169 | 104 | 121 | 145 | 21 | 178 | 7 | 63 | 148 | 194 | 16 | 137 | 11 | 34 | 95 | 33 |
| 128 | 127 | 93 | 154 | 90 | 144 | 50 | 39 | 53 | 62 | 204 | 231 | 191 | 247 | 151 | 3 |
| 255 | 25 | 48 | 179 | 72 | 165 | 181 | 209 | 215 | 94 | 146 | 42 | 172 | 86 | 170 | 198 |
| 79 | 184 | 56 | 210 | 150 | 164 | 125 | 182 | 118 | 252 | 107 | 226 | 156 | 116 | 4 | 241 |
| 69 | 157 | 112 | 89 | 100 | 113 | 135 | 32 | 134 | 91 | 207 | 101 | 230 | 45 | 168 | 2 |
| 27 | 96 | 37 | 173 | 174 | 176 | 185 | 246 | 28 | 70 | 97 | 105 | 52 | 64 | 126 | 15 |
| 85 | 71 | 163 | 35 | 221 | 81 | 175 | 58 | 195 | 92 | 249 | 206 | 186 | 197 | 234 | 38 |
| 44 | 83 | 13 | 110 | 133 | 40 | 132 | 9 | 211 | 223 | 205 | 244 | 65 | 129 | 77 | 82 |

- 0 is replaced by 41. 1 is replaced by 46
- Based on digits of $\pi$
Final Pass

16 bytes at a time. 16 bytes are expanded to 48 bytes.
MD2 Insecurity

- 2004: Shown to have $2^{104}$ time complexity (rather than $2^{128}$)
MD4 Hash

- 128 bit hash using 512 bit blocks using 32-bit operations
- Invented by Ron Rivest in 1990
- Described in RFC 1320
- A variant of MD4 is used in eDonkey200/eMule P2P Networks in their ed2k URI scheme
  - Files with the same content get the same ID even if different names or location
  - ed2k://|file|The_Two_Towers-The_Purist_Edit-Trailer.avi|14997504|965c013e991ee246d63d45ea71954c4d|/
MD4 Algorithm

1. Padding

<table>
<thead>
<tr>
<th>original message</th>
<th>1000…000</th>
<th>original length in bits</th>
</tr>
</thead>
</table>

1–512 bits 64 bits

multiple of 512 bits
MD4 Overview

- f(512-bit message block + 128-bit Digest) → 128 bit Digest
- Three passes of 16 operations each over the message block
MD4 Overview

- Uses non-linear function, modular addition, and left rotation
- Different functions are used in each pass
  - 1. Selection: \( F(x,y,z) = (x^y)v(-x^z) \)
  - 2. Majority: \( G(x,y,z) = (x^y)v(x^z)v(y^z) \)
  - 3. XOR: \( H(x,y,z) = x \oplus y \oplus x \)
- Different rotations are used for each word
  - 3, 7, 11, 15 bit rotations in the first pass
  - 3, 5, 9, 13 bit rotations in the 2nd pass
  - 3, 9, 11, 15 bit rotations in the 3rd pass
- Constants are added in Pass 2 and 3
MD4 Insecurity

- 2004: MD4 collisions can be generated by hand or 5 seconds on a computer
MD5 Hash

- 128-bit hash using 512 bit blocks using 32-bit operations
- Invented by Ron Rivest in 1991
- Described in RFC 1321
- Commonly used to check the integrity of files (easy to fudge message and the checksum)
- Also used to store passwords
MD5 Algorithm

- 4 passes of 16 operations each over the message block
- Uses non-linear function, modular addition, and left rotation

[Source: Wikipedia]
MD5 Algorithm (Cont)

- Different functions are used in each pass
  
  \[ F(X, Y, Z) = (X \land Y) \lor (\neg X \land Z) \]
  
  \[ G(X, Y, Z) = (X \land Z) \lor (Y \land \neg Z) \]
  
  \[ H(X, Y, Z) = X \oplus Y \oplus Z \]
  
  \[ I(X, Y, Z) = Y \oplus (X \lor \neg Z) \]

  \( \land, \lor, \neg \) denote AND, OR, and complement

- Different rotations are used for each word
MD5 Insecurity

- 1993: Two different IV produce the same digest
- 1996: Collision of the compression function
- 2004: a distributed project was done to crack MD5 using birthday attack
- Aug 2004: collisions were found in 1 hour on IBM P690
- March 2005: collisions within a few hours on a single notebook
- March 2006: collisions within 1 minute on a single notebook
- "Rainbow Tables" are available on the Internet to crack MD5
Secure Hash Algorithm (SHA)

- Successor to and similar to MD5
- SHA-0: FIPS PUB 180, 1993. Withdrawn shortly after publ.
- SHA-1: FIPS PUB 180-1, 1995. 160 bit hash
- SHA-2: FIPS PUB 180-2, 2002
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- SHA-1 is used in TLS, SSL, PGP, SSH, S/MIME, and IPsec
  - Required by law in US Govt applications
  - Used in Digital Signature Standard
- Pseudo-codes for SHA algorithms are available.
- NIST certifies implementations.
SHA-1 Algorithm

- 160 bit hash using 512 bit blocks and 32 bit operations
- Five passes (4 in MD5 and 3 in MD4)
- Maximum message size is $2^{64}$ bit
- 512 bits are expanded to 5x512 bits:
  - $n^{th}$ word = xor of n-3, n-8, n-14, and n-16
- In SHA-1 these words are rotated left by one bit before xor
- Total 80 words: $W_0, ..., W_{79}$
SHA-1 Algorithm (Cont)

1. In revised version

16 words of message

generated data

complicated function

160-bit intermediate MD value

\[ A \quad B \quad C \quad D \quad E \]

\[ A \quad B \quad C \quad D \quad E \]

\[ \downarrow 30 \]
SHA Insecurity

- SHA-0:
  - 1998: Time complexity of SHA-0 was shown to be $2^{61}$ compared to $2^{80}$
  - 12 Aug 2004: Collision for SHA-0 with $2^{51}$ complexity
  - 17 Aug 2004: Collision for SHA-0 with $2^{40}$
  - Feb 2005: $2^{39}$

- SHA-1:
  - Will be phased out by 2010 by SHA-2
  - Feb 2005: $2^{69}$ operations in stead of $2^{80}$
  - 17 Aug 2005: $2^{63}$ for finding a collision
  - $2^{35}$ compression fn evaluations for 64-round SHA-1
SHA-2

- SHA-256 uses 32-bit operations
- SHA-512 uses 64-bit operations
- Use different shift amounts and additive constants
- SHA-224 and SHA-384 are simply truncated versions of SHA-256 and SHA-512 using different initial values.
- SHA-224 matches the key length of two-key triple-DES

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Output size (bits)</th>
<th>Internal state size (bits)</th>
<th>Block size (bits)</th>
<th>Max message size (bits)</th>
<th>Word size (bits)</th>
<th>Rounds</th>
<th>Operations</th>
<th>Collision</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-0</td>
<td>160</td>
<td>160</td>
<td>512</td>
<td>2^{64} - 1</td>
<td>32</td>
<td>80</td>
<td>+, and, or, xor, rotl</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>160</td>
<td>512</td>
<td>2^{64} - 1</td>
<td>32</td>
<td>80</td>
<td>+, and, or, xor, rotl</td>
<td>2^63 attack</td>
</tr>
<tr>
<td>SHA-256/224</td>
<td>256/224</td>
<td>256</td>
<td>512</td>
<td>2^{64} - 1</td>
<td>32</td>
<td>64</td>
<td>+, and, or, xor, shr, rotr</td>
<td>None yet</td>
</tr>
<tr>
<td>SHA-512/384</td>
<td>512/384</td>
<td>512</td>
<td>1024</td>
<td>2^{128} - 1</td>
<td>64</td>
<td>80</td>
<td>+, and, or, xor, shr, rotr</td>
<td>None yet</td>
</tr>
</tbody>
</table>

[Source: Wikipedia]
HMAC

- Keyed-Hash Message Authentication Code
- Guarantees both data integrity and authenticity
- Can use any crypto-graphic hash function such as MD5 or SHA-1
- Described in RFC 2104
- FIPS PUB 198 generalizes and standardizes HMACs
- HMACS-MD5 and HMAC-SHA-1 are used in IPsec and TLS
- \( \text{HMAC}_k(m) = h((k \oplus \text{opad}) || h((k \oplus \text{ipad}) || m)) \)
- Here ipad and opad are constants
- Designed to be secure provided the main compression function is secure
HMAC (Cont)

- Secure:
  1. Collision Resistance:
     Can't find 2 inputs with same output
  2. If you don't know k, Cannot compute digest(K,x) even if you know digest(K,y) for many arbitrary y's.
- The secret key is prepended to the message and then again to the digest
HMAC (Cont)

key
0

Constant 2

⊕

Constant 1

Message

digest

digest

HMAC(key, message)
Hashes can be used for authentication, message integrity

Birthday attack: N-bit hash requires $2^{n/2}$ tries to find a collision

MD4, MD5, SHA-1 consist of padding followed by multiple rounds of compression using rotation, substitution, xor, mangling functions, and constants.

SHA-1 is currently the most secure hash. SHA-2 is coming.

HMAC provides both authentication and integrity
References

- Chapter 5 of text book
- Wikipedia:
Homework 7

- Read chapter 5 of the book
- Submit answer to Exercise 5.14
- **Exercise 5.14**: Find minimal sufficient conditions for x, y, and z that would make the following functions random:
  - \(-x\)
  - \(x \text{ XOR } y\)
  - \(x \text{ or } y\)
  - \(x \text{ and } y\)
  - \((x \text{ and } y) \text{ or } (-x \text{ and } z)\)
  - \((x \text{ and } y) \text{ or } (x \text{ and } z) \text{ or } (y \text{ and } z)\)
  - \(\text{XOR } (x, y, z)\)
  - \(\text{XOR } (y, (x \text{ or } -z))\)

One-way function: \( d = h(m) \) but no \( h(d) = m \). Cannot find the message given a digest. Cannot find \( m_1, m_2 \), where \( d_1 = d_2 \).

A cryptographic hash function takes an arbitrary block of data and calculates a fixed-size bit string (a digest), such that different data results (with a high probability) in different digests.