6.857 Lecture 4: Hash Functions

Emily Shen

Most slides courtesy of Ron Rivest (Crypto 2008)
Outline

- Review hash function basics
- Revisit indistinguishability from RO
- MD5
- MD6
Review: Hash function basics (I)

Hash function $h: \{0,1\}^* \rightarrow \{0,1\}^d$

maps arbitrary-length strings of data to fixed-length output ("digest")

in deterministic, public, "random" manner
Review: Hash function basics (II)

- Hash function typically consists of:
  - Compression function
    \[ f : \{0,1\}^c \times \{0,1\}^b \rightarrow \{0,1\}^c \]
    maps fixed-length input to fixed-length output
  - Mode of operation \( h^f \)
    how to apply \( f \) repeatedly to arbitrary-length input to get fixed-length output (of length \( d \))
Review: Desirable properties (I)

- One-wayness (preimage resistance)
  - Infeasible, given $y \leftarrow_{R} \{0,1\}^d$, to find any $x$ s.t. $h(x) = y$

- Collision resistance
  - Infeasible to find $x, x'$ s.t. $x \neq x'$ and $h(x) = h(x')$

- Weak collision resistance ($2^{nd}$ preimage resistance)
  - Infeasible, given $x$, to find $x' \neq x$ s.t. $h(x) = h(x')$
Review: Desirable properties (II)

- **Pseudorandomness**
  - Infeasible to distinguish behavior from random oracle (RO)

- **Non-malleability**
  - Infeasible, given \( h(x) \), to produce \( h(x') \), where \( x \) and \( x' \) are “related”
Formal definitions

- Family of functions
  \[ H: \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^d \]
- For each \( K \in \{0,1\}^k \), we have
  \[ h_K: \{0,1\}^* \rightarrow \{0,1\}^d \]
- Security properties defined in terms of game played w/ adversary
Collision resistance

- Security game:
  - Adversary A gets $K \leftarrow_R \{0,1\}^k$
  - A outputs $x, x'$
  - A wins if $x \neq x'$ and $h(x) = h(x')$

- Advantage of A = probability that A wins

- H is collision resistant if no efficient adversary has more than negligible advantage
A makes hash queries, i.e. outputs \( x \), gets back \( h_K(x) \) or \( RO(x) \) (depending on which world \( A \) is in).

At end of game, \( A \) outputs 0 or 1.

Advantage of \( A \) = \(|\Pr[A^{hK} = 1] - \Pr[A^{RO} = 1]|\)

\( H \) is indistinguishable from \( RO \) if no efficient adversary has more than negligible advantage.

\[ K \leftarrow_R \{0,1\}^k \]

\[ h_K \quad ? \quad RO \]

\[ A \]

\[ ? \quad or \quad ? \]
Indistinguishability from RO

- But $h_K$ and $f$ are fixed, public functions…
- No randomness in $h_K$, so it will be distinguishable from RO
- Adversary should have access to comp. fn $f$
- Need a new notion: “indifferentiability” from RO
Indifferentiability (Maurer et al. ‘04)

- Variant notion of indistinguishability appropriate when distinguisher has access to inner component (e.g. mode of operation \( h^f \) / \( \text{comp. fn f} \)).

\[ h^\text{RO} \rightarrow \text{FIL RO} \rightarrow \text{VIL RO} \rightarrow S \]

- FIL = fixed input length, VIL = variable input length
Indifferentiability from RO

- Indifferentiability: \( \exists \) simulator S s.t. no adversary can distinguish left from right with more than negligible advantage

\[
h^{RO} \rightarrow \text{FIL RO} \rightarrow \text{VIL RO} \rightarrow S
\]

\( ? \) or ?
MD5 compression function

- Chaining variable and output = 128 bits
- IV = fixed value
- 64 steps (4 rounds of 16 steps)
- 512-bit message block considered as 16 32-bit words
**MD5 compression function**

- $M_i =$ 32-bit message word
- $K_i =$ 32-bit constant, differs in each step
- $<<<_s =$ left bit rotation by $s$ bits; $s$ differs in each step
- $\oplus =$ addition mod $2^{32}$

$$F(x,y,z) = \begin{cases} 
(x \land y) \lor (\neg x \land z) \\
(x \land z) \lor (y \land \neg z) \\
x \oplus y \oplus z \\
y \oplus (x \land \neg z)
\end{cases}$$

depending on round

Wang et al. break MD5 (2004)

- Differential cryptanalysis (re)discovered by Biham and Shamir (1990). Considers step-by-step `difference` (XOR) between two computations…
- Applied first to block ciphers (DES)…
- Used by Wang et al. to break collision-resistance of MD5
- Many other hash functions broken similarly; others may be vulnerable…
NIST SHA-3 competition!

- Input: 0 to $2^{64}-1$ bits, size not known in advance
- Output sizes 224, 256, 384, 512 bits
- Collision-resistance, preimage resistance, second preimage resistance, pseudorandomness, …
- Simplicity, flexibility, efficiency, …
- Due Halloween ‘08
MD5 was designed in 1991…

- Same year WWW announced…
- Clock rates were 33MHz…
- Requirements:
  - $\{0,1\}^*$ $\rightarrow$ $\{0,1\}^d$ for digest size $d$
  - Collision-resistance
  - Preimage resistance
  - Pseudorandomness
- What’s happened since then?
- Lots… 😊 😞
- What should a hash function --- MD6 --- look like today?
Design Considerations / Responses
Memory is now ``plentiful”… 😊

- Memory capacities have increased 60% \textit{per year} since 1991
- Chips have 1000 times as much memory as they did in 1991
- Even ``embedded processors” typically have at least 1KB of RAM
So... MD6 has...

- *Large* input message block size: 512 bytes (not 512 bits)
- This has many advantages...
Parallelism has arrived 😊😊😊😊😊

- Uniprocessors have “hit the wall”
  - Clock rates have *plateaued*, since power usage is quadratic or cubic with clock rate:
    \[ P = VI = V^2/R = O(\text{freq}^2) \] (roughly)

- Instead, *number of cores* will double with each generation: tens, hundreds (thousands!) of cores coming soon

4 16 64 256 ⋯
So… MD6 has…

- Bottom-up tree-based mode of operation (like Merkle-tree)
- 4-to-1 compression ratio at each node
Which works very well in parallel

- Height is $\log_4(\text{number of nodes})$
But... most CPU’s are small... 😞

- Storage proportional to tree height may be too much for some CPU’s...
So... MD6 has...

- Alternative sequential mode

(Fits in 1KB RAM)
Actually, MD6 has...

- a smooth sequence of alternative modes: from purely sequential to purely hierarchical... $L$ parallel layers followed by a sequential layer, $0 \leq L \leq 64$

- Example: $L=1$:
Hash functions often ``keyed” 😊 😞

- Salt for password, key for MAC, variability for key derivation, theoretical soundness, etc…
- Current modes are “post-hoc”
So… MD6 has… 😊

- Key input $K$ of up to 512 bits
- $K$ is input to every compression function
Generate-and-paste attacks

- Kelsey and Schneier (2004), Joux (2004), ...
- Generate sub-hash and fit it in somewhere
- Has advantage proportional to size of initial computation…
So… MD6 has… 😊

- 1024-bit intermediate (chaining) values
- root truncated to desired final length
- Location (level,index) input to each node
Extension attacks…

- Hash of one message useful to compute hash of another message (especially if keyed):

\[ H( K \parallel A \parallel B ) = H( H( K \parallel A ) \parallel B ) \]
So... MD6 has... 😊

- ``Root bit” (aka “z-bit”) input to each compression function:
  
  z = 1
Putting it all together…

Chop to $d$ bits

$z = 1$

partially filled

empty
Side-channel attacks 😞

- Timing attacks, cache attacks…
- Operations with data-dependent timing or data-dependent resource usage can produce vulnerabilities.
- This includes data-dependent rotations, *table lookups* (S-boxes), some complex operations (e.g. multiplications), …
So... MD6 uses... 😊

- Operations on 64-bit words
- The following operations *only*:
  - XOR ⊕
  - AND ∧
  - SHIFT by fixed amounts:
    - $x \gg r$
    - $x \ll l$
Security needs vary… 😊

- Already recognized by having different digest lengths $d$ (for MD6: $1 \leq d \leq 512$)
- But it is useful to have reduced-strength versions for analysis, simple applications, or different points on speed/security curve.
So… MD6 has … 😊

- A variable number \( r \) of rounds. (Each round is 16 steps.)
- Default \( r \) depends on digest size \( d \):
  \[
  r = 40 + \frac{d}{4}
  \]

<table>
<thead>
<tr>
<th>( d )</th>
<th>160</th>
<th>224</th>
<th>256</th>
<th>384</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>80</td>
<td>96</td>
<td>104</td>
<td>136</td>
<td>168</td>
</tr>
</tbody>
</table>

- But \( r \) is also an (optional) input.
MD6 Compression function
Compression function inputs

- 64 word (512 byte) data block
  - message, or chaining values
- 8 word (512 bit) key K
- 1 word \( U = (\text{level}, \text{index}) \)
- 1 word \( V = \text{parameters:} \)
  - Data padding amount
  - Key length \( (0 \leq \text{keylen} \leq 64 \text{ bytes}) \)
  - z-bit (aka ``root bit'')
  - \( L \) (mode of operation height-limit)
  - digest size \( d \) (in bits)
  - Number \( r \) of rounds
- 74 words total
Prepend Constant + Map + Chop

const  key+UV  data

15  8+2  64

89 words

1-1 map π

89 words

Map

89 words

16 words

Chop

Prepend

Chop
Simple compression function:

**Input:** \( A[0 .. 88] \) of \( A[0 .. 16r + 88] \)

**for** \( i = 89 \) **to** \( 16r + 88 : \)

\[
x = S_i \oplus A[i-17] \oplus A[i-89] \\
\quad \oplus ( A[i-18] \land A[i-21] ) \\
\quad \oplus ( A[i-31] \land A[i-67] ) \\
x = x \oplus ( x \gg r_i ) \\
A[i] = x \oplus ( x \ll l_i )
\]

**return** \( A[16r + 73 .. 16r + 88] \)
Constants

- Taps 17, 18, 21, 31, 67 optimize diffusion
- Constants $S_i$ defined by simple recurrence; change at end of each 16-step round
- Shift amounts repeat each round (best diffusion of 1,000,000 such tables):

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i$</td>
<td>10</td>
<td>5</td>
<td>13</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>15</td>
<td>7</td>
<td>13</td>
<td>11</td>
<td>7</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>$\ell_i$</td>
<td>11</td>
<td>24</td>
<td>9</td>
<td>16</td>
<td>15</td>
<td>9</td>
<td>27</td>
<td>15</td>
<td>6</td>
<td>2</td>
<td>29</td>
<td>8</td>
<td>15</td>
<td>5</td>
<td>31</td>
<td>9</td>
</tr>
</tbody>
</table>
Large Memory (sliding window)

- Array of $16r + 89$ 64-bit words.
- Each word computed as function of preceding 89 words.
- Last 16 words computed are output.
Small memory (shift register)

- Shift-register of 89 words (712 bytes)
- Data moves right to left
Security Analysis
Generate-and-paste attacks (again)

Because compression functions are “location-aware”, attacks that do speculative computation hoping to “cut and paste it in somewhere” don’t work.
Analyzing mode of operation

General approach:

If compression function $f$ is “secure”,
then mode of operation $\text{MD6}^f$ is “secure”

e.g.,

- $f$ collision-resistant $\Rightarrow$ $\text{MD6}^f$ collision-resistant
- $f$ preimage-resistant $\Rightarrow$ $\text{MD6}^f$ preimage-resistant
- $f$ PRF $\Rightarrow$ $\text{MD6}^f$ PRF
Property preservations

- **Theorem.** If \( f \) is collision-resistant, then \( \text{MD6}^f \) is collision-resistant.
- **Theorem.** If \( f \) is preimage-resistant, then \( \text{MD6}^f \) is preimage-resistant.
- **Theorem.** If \( f \) is a FIL-PRF, then \( \text{MD6}^f \) is a VIL-PRF.
- **Theorem.** If \( f \) is a FIL-MAC and root node effectively uses distinct random key (due to z-bit), then \( \text{MD6}^f \) is a VIL-MAC.

(See thesis by Chris Crutchfield.)
Indifferentiability (Maurer et al. ‘04)

Variant notion of indistinguishability appropriate when distinguisher has access to inner component (e.g. mode of operation $\text{MD6}^f$ / comp. fn $f$).
Indifferentiability (I)

- **Theorem.** The MD6 mode of operation is indifferentiable from a random oracle (viewing compression function as RO)

- **Proof:** Construct simulator for compression function that makes it consistent with any VIL RO and MD6 mode of operation...

- **Advantage:** $\epsilon \leq 2 \frac{q^2}{2^{1024}}$ where $q =$ number of calls (measured in terms of compression function calls).
Indifferentiability (II)

- **Theorem.** MD6 compression function $f^\pi$ is indifferentiable from a FIL random oracle (with respect to random permutation $\pi$).
- **Proof:** Construct simulator $S$ for $\pi$ and $\pi^{-1}$ that makes it consistent with FIL RO and comp. fn. construction.
- **Advantage:** $\epsilon \leq q / 2^{1024} + 2q^2 / 2^{4672}$
Differential attacks don’t work

- **Theorem.** *Any* standard differential attack has less chance of finding collision than standard birthday attack.
- *Proven only for MD6 with large number of rounds.*
Summary

- MD6 is:
  - Arguably secure against known attacks
  - Relatively simple
  - Highly parallelizable
  - Reasonably efficient
MD6 Team

- Dan Bailey
- Sarah Cheng
- Christopher Crutchfield
- Yevgeniy Dodis
- Elliot Fleming
- Asif Khan
- Jayant Krishnamurthy
- Yuncheng Lin
- Leo Reyzin
- Emily Shen
- Jim Sukha
- Eran Tromer
- Yiqun Lisa Yin

- Juniper Networks
- Cilk Arts
- NSF
THE END

IT'S THE GREAT PUMPKIN, CHARLIE BROWN!

MD6

03744327e1e959fbdcdfe7331e959cb2c28101166
Round constants $S_i$

- Since they only change every 16 steps, let $S'_j$ be the round constant for round $j$.
- $S'_0 = 0x0123456789abcdef$
- $S'_{j+1} = (S'_j \ll 1) \oplus (S'_j \land \text{mask})$
- $\text{mask} = 0x7311c2812425cfa0$
Software Implementations
Software implementations

◆ Simplicity of MD6:
  – *Same* implementation for all digest sizes.
  – *Same* implementation for SHA-3 Reference or SHA-3 Optimized Versions.
  – Only optimization is *loop-unrolling* (16 steps within one round).
## NIST SHA-3 Reference Platforms

<table>
<thead>
<tr>
<th></th>
<th>32-bit</th>
<th>64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MD6-160</strong></td>
<td>44 MB/sec</td>
<td>97 MB/sec</td>
</tr>
<tr>
<td><strong>MD6-224</strong></td>
<td>38 MB/sec</td>
<td>82 MB/sec</td>
</tr>
<tr>
<td><strong>MD6-256</strong></td>
<td>35 MB/sec</td>
<td>77 MB/sec</td>
</tr>
<tr>
<td><strong>MD6-384</strong></td>
<td>27 MB/sec</td>
<td>59 MB/sec</td>
</tr>
<tr>
<td><strong>MD6-512</strong></td>
<td>22 MB/sec</td>
<td>49 MB/sec</td>
</tr>
<tr>
<td><strong>SHA-512</strong></td>
<td>38 MB/sec</td>
<td>202 MB/sec</td>
</tr>
</tbody>
</table>
Multicore efficiency

Over 1 GB/sec for 15 cores

SHA-256

MD6-256

Cilk!

data courtesy Cilk Arts
Efficiency on a GPU

- Standard $100 NVidia GPU
- 375 MB/sec on one card

1.5 GB/sec for 4 cards
8-bit processor (Atmel)

- With \( L=0 \) (sequential mode), uses less than 1KB RAM.
- 20 MHz clock
- 110 msec/comp. fn for MD6-224 (gcc actual)
- 44 msec/comp. fn for MD6-224 (assembler est.)
Hardware Implementations
FPGA Implementation (MD6-512)

- Xilinx XUP FPGA (14K logic slices)
- 5.3K slices for round-at-a-time
- 7.9K slices for two-rounds-at-a-time
- 100MHz clock
- 240 MB/sec (two-rounds-at-a-time) (Independent of digest size due to memory bottleneck)