Hash Function Tutorial
Outline

- Some background knowledge
- Some hash functions
- Some attacks
A hash function (in the unrestricted sense) is a function $h$ which has, as a minimum, the following two properties:

- **Compression** - $h$ maps an input $x$ of arbitrary finite bitlength, to an output $h(x)$ of fixed bitlength $n$.
- **Ease of Computation** - given $h$ and an input $x$, $h(x)$ is easy to compute.
Basic properties and definitions

• Preimage resistance - for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage $x'$ such that $h(x') = y$ when given any $y$ for which a corresponding input is not known.

• 2nd-preimage resistance - it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given $x$, to find a 2nd-preimage $x' \neq x$ such that $h(x) = h(x')$.

• Collision resistance - it is computational infeasible to find any two distinct inputs $x, x'$ which hash to the same output, i.e., such that $h(x) = h(x')$. 
Relationships between properties

• Collision resistance implies 2nd-preimage resistance of hash functions

**Justification:** Suppose $h$ has collision resistance. Fix an input $x_j$. If $h$ does not have 2nd-preimage resistance, then it is feasible to find a distinct input $x_i$ such that $h(x_i) = h(x_j)$, in which case $(x_i, x_j)$ is a pair of distinct inputs hashing to the same output, contradiction collision resistance.
Relationships between properties

• Collision resistant does not guarantee preimage resistance

**Justification:** Let $g$ be a hash function which is collision resistant and maps arbitrary-length inputs to $n$-bit outputs. Consider the function $h$ defined as follows:

$$h(x) = 1||x, \text{ if } x \text{ has bitlength } n$$

$$0||g(x), \text{ otherwise}$$
Simplified classification

- Hash functions
  - Unkeyed
    - Modification detection (MDCs)
  - Keyed
    - Other applications
    - Message authentication (MACs)
- OWHF
  - Preimage resistant
  - 2nd preimage resistant
  - Collision resistant
- CRHF
Extending compression functions to hash functions

Any compression function $f$ which is collision resistant can be extended to a collision resistant hash function $h$.

- Merkle-Damgard construction (See Stinson’s book Page 241-244)
Merkle-Damgard Strengthening

Before hashing a message \( x = x_1x_2 \ldots x_t \) of bitlength \( b \), append a final length-block, \( x_{t+1} \), containing the right-justified binary representation of \( b \). (This presumes \( b < 2^r \)).

- The inclusion of the length-block effectively encodes all messages such that no encoded input is the tail end of any other encoded input.
Attacks on hash functions

- Birthday attacks (time complexity $2^{n/2}$)
- Pseudo-collision and compression function attacks
- Chaining attacks
- Attacks based on properties of underlying cipher
MD4

- Designed by Rivest (RFC 1320 April 1992)
- Input: a message of arbitrary length
- Output: 128-bit message digest
MD4 Padding

- The message is padded so that its length in bits is congruent to 448, modulo 512.
- A single “1” bit is appended to the message, and then “0” bits.
- A 64-bit representation of the length of the original message is appended to the result of the previous step.
- After padding, the message has a length that is an exact multiple of 512 bits.
Initialising MD4 Buffer

- A four-word buffer (A, B, C, D) is used
- A: 01 23 45 67
- B: 89 ab cd ef
- C: fe dc ba 98
- D: 76 54 32 10
Functions: F, G, and H

Each function takes three 32-bit words as input and produces one 32-bit word as output.

- \( F(X, Y, Z) = XY \lor \neg(X) \land Z \)
- \( G(X, Y, Z) = XY \lor XZ \lor YZ \)
- \( H(X, Y, Z) = X \oplus Y \oplus Z \)

\( \oplus = \) exclusive-or, \( \lor = \) inclusive-or
Processing Message in 16-Word Blocks

/* M[0,..., N-1], N = message length / 16 */
1. For i = 0 to N/16 - 1 do
   /* Copy block i into X */
   2. For j = 0 to 15 do
      3. Set X[j] = to M[i*16+j]
   end /* of loop on j */
/* Save A as AA, B as BB, C as CC, and D as DD. */
4. AA = A
5. BB = B
6. CC = C
7. DD = D
Processing Message Contd

/* Round 1. */
/* Let <abcd k s> denote the operation:
a = (a + F(b,c,d) + X[k]) <<< s. */
/* Do the following 16 operations. */

<ABCD 0 3> <DABC 1 7> <CDAB 2 11> <BCDA 3 19>
<ABCD 4 3> <DABC 5 7> <CDAB 6 11> <BCDA 7 19>
<ABCD 8 3> <DABC 9 7> <CDAB 10 11> <BCDA 11 19>
<ABCD 12 3> <DABC 13 7> <CDAB 14 11> <BCDA 15 19>
Processing Message Contd

/* Round 2. */
/* Let <abcd k s> denote the operation:
   a = (a + G(b,c,d) + X[k] + 5A827999) <<< s. */
/* Do the following 16 operations. */

<ABCD 0 3> <DABC 4 5> <CDAB 8 9> <BCDA 12 13>
<ABCD 1 3> <DABC 5 5> <CDAB 9 9> <BCDA 13 13>
<ABCD 2 3> <DABC 6 5> <CDAB 10 9> <BCDA 14 13>
<ABCD 3 3> <DABC 7 5> <CDAB 11 9> <BCDA 15 13>
/* Round 3. */
/* Let <abcd k s> denote the operation:
a = (a + H(b,c,d) + X[k] + 6ED9EBA1) <<< s. */
/* Do the following 16 operations. */

<ABCD 0 3> <DABC 8 9> <CDAB 4 11> <BCDA 12 15>
<ABCD 2 3> <DABC 10 9> <CDAB 6 11> <BCDA 14 15>
<ABCD 1 3> <DABC 9 9> <CDAB 5 11> <BCDA 13 15>
<ABCD 3 3> <DABC 11 9> <CDAB 7 11> <BCDA 15 15>
/* Then perform the following additions. */

A = A + AA
B = B + BB
C = C + CC
D = D + DD

end /* of loop on i */

Output: [A, B, C, D]
### Some hash functions based on MD4

<table>
<thead>
<tr>
<th>Name</th>
<th>Bitlength</th>
<th>Rounds × Steps per round</th>
<th>Relative speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD4</td>
<td>128</td>
<td>3 × 16</td>
<td>1.00</td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>4 × 16</td>
<td>0.68</td>
</tr>
<tr>
<td>RIPEMD-128</td>
<td>128</td>
<td>4 × 16 twice (in parallel)</td>
<td>0.39</td>
</tr>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>4 × 20</td>
<td>0.28</td>
</tr>
<tr>
<td>RIPEMD-160</td>
<td>160</td>
<td>5 × 16 twice (in parallel)</td>
<td>0.24</td>
</tr>
</tbody>
</table>
Cryptanalysis of MD4

MD4 is not collision-free [Dobbertin]
Cryptanalysis of MD4 Contd

Dobbertin proposed a method to find collisions:

- $X (= X_0, X_1, \ldots, X_{15})$ denotes a collection of 16 words
- $\tilde{X} (= \tilde{X}_0, \tilde{X}_1, \ldots, \tilde{X}_{15})$ is defined as:

$$\tilde{X}_i = X_i, \ i < 16 \text{ for } i \neq 12$$

$$\tilde{X}_{12} = X_{12} + 1.$$  

Finally,

$$compress(IV_0; X) = compress(IV_0; \tilde{X})$$
Cryptanalysis of MD4 Contd

- $X_{12}$ appears in each round exactly once, in steps 12, 19 and 35
- $X$ and $\tilde{X}$ give a collision if (and only if) $\Delta_{35} = 0$, because $X_{12}$ appears in step 35 the last time
- We require a well-chosen value for $\Delta_{19}$:

$$\Delta_{19} = (0, 1\ll 25, -1\ll 5, 0)$$

(Note: $\Delta_i = (A_i - \tilde{A}, B_i - \tilde{B}, C_i - \tilde{C}, D_i - \tilde{D})$)
Lemma 1. There is a practical algorithm, which allows us to compute an admissible inner almost-collision, i.e., an initial value \((A, B, C, D)\) and inputs \(X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8\) for \(compress_{12}^{19}\) such that we have

\[
\Delta_{19} = (0, 1^{<<25}, -1^{<<5}, 0)
\]

\[
G(B_{19}, C_{19}, D_{19}) = G(\tilde{B}_{19}, \tilde{C}_{19}, \tilde{D}_{19})
\]

The computation requires less than 1 second on a PC.
Cryptanalysis of MD4 Contd

Lemma 2. Suppose that an admissible inner almost-collision, i.e., an initial value \((A, B, C, D)\) for step 12 and variables \(X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8\) are given according to Lemma 1. Choose the remaining \(X_i\)'s randomly and determine the corresponding initial value by computing \(compress_{11}^0\) backward starting with 
\[
(A_{11}, B_{11}, C_{11}, D_{11}) = (A, B, C, D).
\]
Then the probability that \(X\) and \(\tilde{X}\) form a collision for the compression function of MD4 (i.e., \(\Delta_{35} = 0\)) is about \(2^{-22}\).
The MD4 Collision Search Algorithm

1. Compute $A$, $B$, $C$, $D$ and $X_0$, $X_4$, $X_8$, $X_{12}$, $X_{13}$, $X_{14}$, $X_{15}$, which give an inner almost-collision (from steps 12 to 19). The technical details of a suitable algorithm is provided in Section 3. It also fixed values for $A_{19}$, $B_{19}$, $C_{19}$, $D_{19}$ and $\tilde{A}$, $\tilde{B}_{19}$, $\tilde{C}_{19}$, $\tilde{D}_{19}$. 
Collision Search Algorithm Contd

2. According to Sections 4 and 5 choose $X_1$, $X_2$, $X_3$, $X_5$ randomly and compute

\[
(A_5, B_5, C_5, D_5) = \text{compress}_5^0(IV_0; X_0, \ldots, X_5),
\]

\[
t = A^{<<29} - A_5 - X_8,
\]

\[
X_6 = t^{<<21} - C_5 - F(D_5, A_5, B_5),
\]

\[
X_7 = -1 - B_5 - F(t, D_5, A_5),
\]

\[
X_9 = D^{<<25} - D_5 - F(A, -1, t),
\]

\[
X_{10} = C^{<<21} - t - F(D, A, -1),
\]

\[
X_{11} = B^{<<13} - t - F(C, D, A),
\]

\[
(A_{35}, B_{35}, C_{35}, D_{35}) = \text{compress}_{35}^{20}(A_{19}, B_{19}, C_{19}, D_{19}; X),
\]

\[
(\tilde{A}_{35}, \tilde{B}_{35}, \tilde{C}_{35}, \tilde{D}_{35}) = \text{compress}_{35}^{20}(\tilde{A}_{19}, \tilde{B}_{19}, \tilde{C}_{19}, \tilde{D}_{19}; \tilde{X}),
\]

\[
\Delta_{35} = (A_{35}, B_{35}, C_{35}, D_{35}) - (\tilde{A}_{35}, \tilde{B}_{35}, \tilde{C}_{35}, \tilde{D}_{35})
\]
Collision Search Algorithm Contd

3. If $\Delta_{35} = 0$, then we have found a collision. Otherwise make a new trial by going to 2.
Two Colliding Messages

Message 1:
************************************************
CONTRACT
At the price of $176,495 Alf Blowfish sells his house to Ann Bonidea....

Message 2:
************************************************
CONTRACT
At the price of $276,495 Alf Blowfish sells his house to Ann Bonidea....
MD4 Collisions by Wang et al

- can find MD4 collisions with hand calculation.

\[
M' = M + \Delta C, \\
\Delta C = (0, 2^{31}, -2^{28} + 2^{31}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \\
\text{and} \\
MD4(M) = MD4(M')
\]
MD5

Differences between MD4 and MD5:

1. A fourth round has been added.
2. Each step now has a unique additive constant.
3. The function \( g \) in round 2 was changed from \( (XY \lor XZ \lor YZ) \) to \( (XZ \lor Y \neg Z) \) to make \( g \) less symmetric.
4. Each step now adds in the result of the previous step. This promotes a faster “avalanche effect”.
5. The order in which input words are accessed in rounds 2 and 3 changed, to make these patterns less like each other.
6. The shift amounts in each round have been approximately optimized, to yeild a faster “avalanche effect”. The shift in different rounds are distinct.
Collisions for MD5

• Bert den Boer and Antoon Bosselaers found a pseudo-collision for MD5 which is made of the same message with two different sets of initial values.

• Dobbertin found a free-start collision which contains of two different 512-bit message with a chosen initial value $IV'_0$.

$$IV'_0: A'_0 = 0x12AC2375, B'_0 = 0x3B341042, C'_0 = 0x5F62B97C, D'_0 = 0x4BA763ED$$
Collisions for MD5 Contd

The real MD5 collisions found by Wang et al.
Their attack can find many collisions which are composed of two 1024-bit messages with the original initial value $IV_0$. 
Collisions for MD5 Contd

\[ IV_0 : A_0 = 0x67452301, B_0 = 0xefcdab89, \]
\[ C_0 = 0x98badcfe, D_0 = 0x10325476 \]

\[ M' = M + \Delta C_1, \]
\[ \Delta C_1 = (0, 0, 0, 0, 2^{31}, \ldots, 2^{15}, \ldots, 2^{31}, 0) \]

\[ N'_i = N_i + \Delta C_2, \]
\[ \Delta C_2 = (0, 0, 0, 0, 2^{31}, \ldots, -2^{15}, \ldots, 2^{31}, 0) \]

(non-zeros at position 4, 11 and 14)

such that

\[ MD5(M, N_i) = MD5(M', N'_i) \]
One Pair of Colliding Messages

file1.dat
00000000 d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c
00000010 2f ca b5 87 12 46 7e ab 40 04 58 3e b8 fb 7f 89
00000020 55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 71 41 5a
00000030 08 51 25 e8 f7 cd c9 9f d9 1d bd f2 80 37 3c 5b
00000040 96 0b 1d d1 dc 41 7b 9c e4 d8 97 f4 5a 65 55 d5
00000050 35 73 9a c7 f0 eb fd 0c 30 29 f1 66 d1 09 b1 8f
00000060 75 27 7f 79 30 d5 5c eb 22 e8 ad ba 79 cc 15 5c
00000070 ed 74 cb dd 5f c5 d3 6d b1 9b 0a d8 35 cc a7 e3

file2.dat
00000000 d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c
00000010 2f ca b5 07 12 46 7e ab 40 04 58 3e b8 fb 7f 89
00000020 55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 f1 41 5a
00000030 08 51 25 e8 f7 cd c9 9f d9 1d bd 72 80 37 3c 5b
00000040 96 0b 1d d1 dc 41 7b 9c e4 d8 97 f4 5a 65 55 d5
00000050 35 73 9a 47 f0 eb fd 0c 30 29 f1 66 d1 09 b1 8f
00000060 75 27 7f 79 30 d5 5c eb 22 e8 ad ba 79 4c 15 5c
00000070 ed 74 cb dd 5f c5 d3 6d b1 9b 0a 58 35 cc a7 e3
Checking

Under Unix:
$ md5sum file1.dat
MD5 Sum = a4c0d35c95a63a805915367dcfe6b751
$ md5sum file2.dat
MD5 Sum = a4c0d35c95a63a805915367dcfe6b751
SHA-0

- Near-Collisions of SHA-0 (Eli Biham and Rafi Chen Crypto 2004)
- New results on SHA-0 ans SHA-1 (Eli Biham and Rafi Chen Crypto 2004 Rump Session)
- Collisions in SHA-0 (Antoine Joux Crypto 2004 Rump Session)
So far, there are only attacks on “reduced round” versions of SHA-1, which do not extend to the full version.
Other hash functions

Please see a website called The Hash Function Lounge