Message Digest 5 (MD5)

- Input: message of any length
- Output: 128-bit message digest
- Four main steps
  - append padding bits
  - append length
  - initialize MD buffer
  - process message in 512-bit blocks
**Message Digest 5 (MD5)**

Appending Padding Bits

- Message padded: length mod 512 = 448
  - 448 = 512 − 64 (used to represent length)
- Padding must be done at least once
  - if length mod 512 = 448, add 512 bits
- Padding: first 1-bit, remaining 0-bits
Append Length

- Length encoded as 64-bit number
  - length before padding
- If length > $2^{64}$, use length mod $2^{64}$
- Least significant byte first

Message Formatting

- Message + padding is now multiple of 512
- L blocks of 512: $Y_0, Y_1, \ldots, Y_{L-1}$
- Block = 512 bit = 16 × 32-bit words
- Let $M = [0, \ldots, N-1]$ denote words
  - each word = 32 bit = 8 hexadecimal digits
  - $N = L \times 16$ (32-bit words)
- Example
  - 5120 bits = 10 of 512-bit blocks ($L = 10$)
    = 160 of 32-bit words ($L \times 16 = 160$)
Initialize MD Buffer

- 128-bit buffer is used
- Holds intermediate and final hash result
- 4 32-bit registers (A, B, C, D) (little-endian)
  - A = 67452301 → 01 23 45 67
  - B = EFCDAB89 → 89 AB CD EF
  - C = 98BADCFE → FE DC BA 98
  - D = 10325476 → 76 54 32 10

Processing

- Heart of algorithm – $H_{MD5}$ function
- Table T: 64 elements
  - $T[i] = \text{floor}(2^{32} \times |\sin(i)|)$
- Four rounds
  - similar structure, different primitive function
Round 4 output is added to round 1 input $CV_q$ to produce $CV_{q+1}$ for next block $q+1$

For the first block, IV is used

Addition is mod $2^{32}$

Independent for each word A, B, C, D
MD5 Compression Function

- A, B, C, D processed in each round in turn
- Each round has 16 steps, each step
  \[ a \leftarrow b + ((a + g(b, c, d) + X[k] + T[i]) <<< s) \]
  - \(<<<s\) = circular shift if 32-bit word by \(s\) bits
  - \(X[k] = k^{th}\) word in the 512-bit block (\(k: 0-15\))
- Only 1 word is updated in each step
- Followed by word-level circular right shift
- After 16 steps, each word updated 4 times

Note: Addition (+) is mod \(2^{32}\)
## Functions F, G, H, I

<table>
<thead>
<tr>
<th>Round</th>
<th>Primitive function g</th>
<th>Function value</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F(B, C, D)</td>
<td>$(B \land C) \lor (\overline{B} \land D)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>G(B, C, D)</td>
<td>$(B \land D) \lor (C \land \overline{D})$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>H(B, C, D)</td>
<td>$B \oplus C \oplus D$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>I(B, C, D)</td>
<td>$C \oplus (B \lor \overline{D})$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Order of Words in Each Round

- Order of $X[i]$s varies depending on round
- Round 1: use $X[i]$s in order, $i:1-16$
- Round 2: use $X[\rho_2(i)]$, $\rho_2(i) = (1+5i) \mod 16$
- Round 3: use $X[\rho_3(i)]$, $\rho_3(i) = (5+3i) \mod 16$
- Round 4: use $X[\rho_4(i)]$, $\rho_4(i) = (7i) \mod 16$
Output

\[ CV_0 = IV \]

\[ CV_{q+1} = \sum_{32} \left( CV_q, RF_I \left[ Y_q, RF_H \left[ Y_q, RF_G \left[ Y_q, CV_q \right] \right] \right] \right) \]

\[ MD = CV_L \]

Secure Hash Algorithm

- Developed by NIST in 1993
- Revised in 1995 as SHA-1
- Similar structure to MD5, some differences
- Other versions: SHA-256, SHA-384, SHA-512

<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>Message digest size</td>
<td>160</td>
<td>256</td>
<td>384</td>
<td>512</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>
<tr>
<td>Block size</td>
<td>512</td>
<td>512</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Word size</td>
<td>32</td>
<td>32</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Number of steps</td>
<td>80</td>
<td>64</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Security</td>
<td>80</td>
<td>128</td>
<td>192</td>
<td>256</td>
</tr>
</tbody>
</table>

Notes: 1. All sizes are measured in bits.
2. Security refers to the fact that a birthday attack on a message digest of size \( n \) produces a collision with a workfactor of approximately \( 2^{n/2} \).
SHA-1 Main Differences From MD5

- MD buffer is 160 bit (vs. 128)
- 5 registers: A, B, C, D, E (vs. 4)
- Stored as big-endian (vs. little-endian)
- Init. \( E = \text{C3D2E1F0} \rightarrow C3 \, D2 \, E1 \, F0 \)
- Processing is done in 20 steps (vs. 16)
- Constant \( K \) takes only 4 values in 80 steps (vs. T table with 64 values for 64 steps)
- Round functions \( f_1, f_2, f_3, f_4, f_5 \) (vs. F, G, H, I)

Round Functions

<table>
<thead>
<tr>
<th>Step Number ( 0 \leq t \leq 19 )</th>
<th>Hexadecimal ( K_t )</th>
<th>Integer Part of ( 2^{30} \times \sqrt{2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 20 \leq t \leq 39 )</td>
<td>( K_t = 6ED9EBA1 )</td>
<td>( 2^{30} \times \sqrt{3} )</td>
</tr>
<tr>
<td>( 40 \leq t \leq 59 )</td>
<td>( K_t = 8F1B8C7D )</td>
<td>( 2^{30} \times \sqrt{5} )</td>
</tr>
<tr>
<td>( 60 \leq t \leq 79 )</td>
<td>( K_t = CA62C1D6 )</td>
<td>( 2^{30} \times \sqrt{10} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Primitive Function ( f )</th>
<th>Function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 \leq t \leq 19 )</td>
<td>( f_1(B, C, D) )</td>
<td>( (B \land C) \lor (\overline{B} \land D) )</td>
</tr>
<tr>
<td>( 20 \leq t \leq 39 )</td>
<td>( f_2(B, C, D) )</td>
<td>( B \oplus C \oplus D )</td>
</tr>
<tr>
<td>( 40 \leq t \leq 59 )</td>
<td>( f_3(B, C, D) )</td>
<td>( (B \land C) \lor (B \land D) \lor (C \land D) )</td>
</tr>
<tr>
<td>( 60 \leq t \leq 79 )</td>
<td>( f_4(B, C, D) )</td>
<td>( B \oplus C \oplus D )</td>
</tr>
</tbody>
</table>
Derivation of $W_t$

- First 16 values from 512-bit block
  - $W_0 = X[0]$, $W_1 = X[1]$, ..., $W_{15} = X[15]$
- Remaining values
  - $W_t = S_1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$
  - $S_1 =$ left circular shift by 1 bit
- Notable difference from MD5
Derivation of $W_t$

SHA-1 Compression Function
Comparison of SHA-1 and MD5

- SHA-1 harder for brute force (160 vs 128 bit)
- SHA-1 is slower (80 vs 64 steps)
- SHA-1 uses big-endian (vs. little-endian)
- Both are simple and compact!
- MD5 is more vulnerable
- SHA-1 flaws discovered, currently impractical

HMAC

- MAC algorithm based on hash function
- Hash function can be replaced easily
- No significant performance overhead
  - only 3 more hash calc than message alone
HMAC

- $H = \text{hash function}$
- $M = \text{message input}$
- $Y_i = \text{ith block of } M$
- $L = \text{number of blocks}$
- $b = \text{block length in bits}$
- $n = \text{hash code length}$
HMAC

\[ \text{HMAC}(K, M) = H\left(\left(K^+ \oplus \text{opad}\right) \parallel H\left(\left(K^+ \oplus \text{ipad}\right) \parallel M\right)\right) \]

1. \( K^+ = K \) padded with 0s to equal block size
2. \( S_i = K^+ \oplus \) with ipad (0110110 repeated)
3. Append M to \( S_i \)
4. Apply H to result of (3) \hspace{1cm} (1 extra hash calc)
5. \( S_o = K^+ \oplus \text{opad} \) (01011100 repeated)
6. Append result of (4) to \( S_o \)
7. Apply H to result of (6) \hspace{1cm} (2 extra hash calc)