Network Security: Hashes

Henning Schulzrinne
Columbia University, New York
schulzrinne@cs.columbia.edu

Columbia University, Fall 2000

©1999-2000, Henning Schulzrinne
Last modified October 5, 2000
Hashes

- hash \(\equiv\) message digest
- one-way function: \(d = h(m)\), but no \(h'(d) = m\)
- cannot find message with given digest
- cannot find \(m_1, m_2\) where \(d_1 = d_2\)
- cannot find different \(m_2\) with same digest (given some \(m_1\))
- arbitrary-length message \(\rightarrow\) fixed-size hash
- different: checksums (CRC):
  - fast
  - easy to compute two messages
  - with same ’hash’, protect against noisy channels
Randomness

- for 1000 inputs, any bit in the outputs ’1’ half the time
- each output: 50% ’1’ bits
- similar inputs ➞ uncorrelated outputs (half the bits should differ)

note: efficiency **not** important
Random vs. Random

• random numbers for simulation (Knuth, Bratley/Fox/Schrage):
  – sequence
  – no short cycles
  – average = 0.5, variance, ...
  – completely predictable, repeatable
  – “white noise”: flat frequency spectrum
  – must be very efficient (multiply, add)

• random numbers for picking keys:
  – unpredictable by outside observer
  – two processes at the same time \(\Rightarrow\) different number
  – use audio, video
  – efficiency not very important
The Birthday Problem (Approximate)

two people with same birthday in room

- \( n \) inputs (humans)
- \( k = 365 \) possible outputs (birthdays)
- \( n \cdot (n - 1)/2 \) pairs of inputs
- check for each pair: \( 1/k \) of both humans having same birthday
- \( k/2 \) pairs for 50% matching probability
- \( n \approx \sqrt{k} \) for 50% chance
- warning: simplified, does not hold for large \( n \! \\
- or:

\[
p_{\text{same}} = \frac{n(n-1)}{2} \cdot \frac{1}{k}
\]

Example: \( n = 23 \Rightarrow p_{\text{alldifferent}} = 0.31 \)
Birthday Problem, Correct

compute probability of different birthdays \(\Rightarrow\) no sample twice

\[
\begin{align*}
\text{sampling without replacement} \\
\text{sampling with replacement}
\end{align*}
\]

- random sample of \(n\) people (birthdays) taken from \(k\) (365 days)
- \(k^n\) samples with replacement
- \((k)_n = k(k - 1) \cdots (k - n + 1)\) samples without replacement
Birthday Problem, Correct

probability of no repetition:

\[ p = \frac{(k)_n}{k^n} = \frac{k(k - 1) \cdots (k - n + 1)}{k^n} \approx 1 - \frac{n(n - 1)}{2k} \]

\[
p = \frac{(365)_n}{365^n} = \frac{365 \cdot 364 \cdots (365 - n + 1)}{365 \cdot 365 \cdots 365} \\
= 1 \cdot \frac{365 - 1}{365} \cdot \frac{365 - 2}{365} \cdots \frac{365 - n + 1}{365} \\
= \left(1 - \frac{1}{365}\right) \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{n + 1}{365}\right) \]

Example: \( n = 23 \implies p = 0.43 \)
How Many Bits for Hash?

- \( m \) bits, takes \( 2^{m/2} \) to find two with the same hash
- 64 bits \( \implies 2^{32} \) messages to search
- note: different from picking \( h \) and finding \( m \)

Example:

- Alice (secretary: Bob) wants to fire Fred
- Bob is Fred’s friend
- Bob composes two messages (same hash), one to be read, the other sent
- generate lots of similar messages: 2 choices of wording in 32 places \( \implies 2^{32} \) messages
# Hash Functions

RFC: [http://www.normos.org](http://www.normos.org)

<table>
<thead>
<tr>
<th>name</th>
<th>inventor</th>
<th>docs</th>
<th>entity</th>
<th>hash</th>
<th>sub-stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD2</td>
<td>Ron Rivest</td>
<td>RFC 1319</td>
<td>bytes</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>MD4</td>
<td>Ron Rivest</td>
<td>RFC 1320</td>
<td>32-bit</td>
<td>128</td>
<td>3</td>
</tr>
<tr>
<td>MD5</td>
<td>Ron Rivest</td>
<td>RFC 1321</td>
<td>32-bit</td>
<td>128</td>
<td>4</td>
</tr>
<tr>
<td>SHS</td>
<td>NIST</td>
<td>RFC 1319</td>
<td>32-bit</td>
<td>160</td>
<td></td>
</tr>
</tbody>
</table>
Authentication

\[
\begin{align*}
\text{Alice} & \quad \rightarrow \\
\ r_A & \quad \rightarrow \\
& \quad \leftarrow \quad \text{MD}(K_{AB}|r_A) \\
& \quad \leftarrow \quad r_B \\
\text{MD}(K_{AB}|r_B) & \quad \rightarrow \\
\end{align*}
\]

only need to compare result, not decrypt
Computing a MIC with a Hash

- can’t just compute $\text{MD}(m)$
- MIC: $\text{MD}(K_{AB}|m_1)$
- but: Trudy can get MIC of $m_2$, given $m_1$
  - 512-bit blocks, append (message length, pad)
  - allow to concatenate padding, plus additional message
  - put secret at end of message
  - or: use only half the bits of MD ➔ can’t continue message
Encryption: One-Time Pad

- \( b_1 = \text{MD}(K_{AB}|\text{IV}) \)
- \( b_2 = \text{MD}(K_{AB}|b_1) \)
- \( b_i = \text{MD}(K_{AB}|b_{i-1}) \)
- then: \( b \) used as one-time pad
- can’t use same OTP twice \( \iff \) use IV
- any OTP has privacy only: if plaintext known, can encrypt anything
- separate integrity function
Encryption: Mixing in the Plaintext

- similar to cipher feedback mode (CFB)
- break message into MD-length chunks $p_i$

$$
\begin{align*}
  b_1 &= \text{MD}(K_{AB}|IV) & c_1 &= p_1 \oplus b_1 \\
  b_2 &= \text{MD}(K_{AB}|c_1) & c_2 &= p_2 \oplus b_2 \\
  & \vdots \\
  b_i &= \text{MD}(K_{AB}|c_{i-1}) & c_i &= p_i \oplus b_i
\end{align*}
$$
Using Secret Key for a Hash

Unix password algorithm:

- compute hash of user password, compare to hash of password typed
- first 8 bytes of password → secret key
- encrypt 0 with DES-like algorithm
- *salt*: 12-bit random number determines bits to duplicate in mangler when expanding 32 to 48 bits
- salt stored with hashed result
Hashing Large Messages Using Encryption

- key length $k$, block length $b$
- divide message into $k$-bit chunks $m_1, m_2, \ldots$
- first, encrypt a constant with $m_1$ as key
- encrypt output with $m_2$ as key
- $\Rightarrow$ output $b(=64)$ too short
- generate 2nd 64-bit quantity by reversing chunk order
- or: use different initial constants
Hashing Large Messages
MD2

128-bit message digest

- arbitrary number of bytes
- pad to multiple of 16 bytes
- append MD2 checksum to end
- process whole message
MD2 Checksum

- $m_{nk}$: byte $nk$ of message
- $c_n := \pi(m_{nk} \oplus c_{n-1}) \oplus c_n$
- $\pi$: $0 \rightarrow 41, 1 \rightarrow 46, \ldots$
- *supposedly* based on $\pi$
MD2 Final Pass

- operate on 16-byte chunks
- 48-byte quantity $q$: (current digest | chunk | digest $\oplus$ chunk)
- 18 passes over $q$
- $c_n = c_n \oplus \pi(c_{n-1})$ for $n = 0, \ldots, 47$; $c_{-1} = 0$
- $c_{-1} = (c_{47} + \text{pass #}) \mod 256$
- after pass 17, use first 16 bytes as new digest
- minimal storage: checksum, 48 bytes
MD4

- any number of bits
- operates on 32-bit quantities
- pad to multiple of 512 bits (16 32-bit words):
  100... , message length in bits
- digest = $f(512$-bit blocks, previous digest or initial constant)
- 1 stage = three mangling passes
- 16 message words $m_0, m_1, \ldots, m_{15}$
- 4 message digest words $d_0, d_1, d_2, d_3$
- $d_0 = 01234567_{16}, d_1 = 89abcde_{16}, d_2 = feda98_{16}, \ldots$
MD4 Operations

- $|x|$ 
- $\sim$: bitwise complement 
- $x \land y$: bitwise and 
- $x \lor y$: bitwise or 
- $x + y$: addition mod $2^{32}$ 
- $x \leftarrow y$: rotate left $y$ bits
MD4 Pass 1: Selection

• selection function: \( F(x, y, z) = (x \land y) \lor (\sim x \land z) \)

• if bit \( x_n \) is 1: pick \( F_n = y_n \), otherwise \( z_n \)

\[
d_{(-i)\land3} = (d_{(-i)\land3} + F(d_{(1-i)\land3}, d_{(2-i)\land3}, d_{(3-i)\land3}) + m_i) \leftrightarrow S_1(i \land 3)
\]

where \( S_1(i) = 3 + 4i \), \(-1_{10} = 1111_2, -2_{10} = 1110_2, \ldots\)

\[
\begin{align*}
d_0 &= (d_0 + F(d_1, d_2, d_3) + m_0) \leftrightarrow 3 \\
d_1 &= (d_3 + F(d_0, d_1, d_2) + m_1) \leftrightarrow 7 \\
d_2 &= (d_2 + F(d_3, d_0, d_1) + m_2) \leftrightarrow 11 \\
d_3 &= (d_1 + F(d_2, d_3, d_0) + m_3) \leftrightarrow 15 \\
d_0 &= (d_0 + F(d_1, d_2, d_3) + m_4) \leftrightarrow 3 
\end{align*}
\]
MD4 Pass 2: Majority

- $G(x, y, z) = (x \land y) \lor (x \land z) \lor (y \land z)$
- $G_n = 1$ iff 2 out of 3 bits $x_n, y_n, z_n$ are 1
- constant $\lfloor 2^{30} \sqrt{2} \rfloor = 5a827999_{16}$

$$d_{(-i)\land3} = (d_{(-i)\land3} + G(d_{(1-i)\land3}, d_{(2-i)\land3}, d_{(3-i)\land3}) + m_X(i) + 5a827999_{16}) \leftarrow S_2(i \land 3)$$

where $X(i)$ eXchange bits (2,3) with (0,1); $S_2 = 3, 5, 9, 13, 3, \ldots$

$$d_0 = (d_0 + G(d_1, d_2, d_3) + m_0 + 5a827999_{16}) \leftarrow 3$$
$$d_1 = (d_3 + G(d_0, d_1, d_2) + m_4 + 5a827999_{16}) \leftarrow 5$$
$$d_2 = (d_2 + G(d_3, d_0, d_1) + m_8 + 5a827999_{16}) \leftarrow 9$$
$$d_3 = (d_1 + G(d_2, d_3, d_0) + m_{12} + 5a827999_{16}) \leftarrow 13$$
$$d_0 = (d_0 + G(d_1, d_2, d_3) + m_1 + 5a827999_{16}) \leftarrow 3$$
MD4 Pass 3: Ex-Or

- \( H(x, y, z) = x \oplus y \oplus z \)
- constant \( 2^{30} \sqrt{3} = 6ed9eba1_{16} \)

\[
d_{(-i) \wedge 3} = (d_{(-i) \wedge 3} + H(d_{(1-i) \wedge 3}, d_{(2-i) \wedge 3}, d_{(3-i) \wedge 3}) + m_{R(i)} + 6ed9eba1_{16}) \leftrightarrow S_3(i \wedge 3)
\]

where \( R(i) \) Reverses bits in \( i \); \( S_3 = 3, 9, 11, 15 \)

\[
\begin{align*}
    d_0 & = (d_0 + H(d_1, d_2, d_3) + m_0 + 6ed9eba1_{16}) \leftrightarrow 3 \\
    d_1 & = (d_3 + H(d_0, d_1, d_2) + m_4 + 6ed9eba1_{16}) \leftrightarrow 9 \\
    d_2 & = (d_2 + H(d_3, d_0, d_1) + m_8 + 6ed9eba1_{16}) \leftrightarrow 11 \\
    d_3 & = (d_1 + H(d_2, d_3, d_0) + m_{12} + 6ed9eba1_{16}) \leftrightarrow 15 \\
    d_0 & = (d_0 + H(d_1, d_2, d_3) + m_2 + 6ed9eba1_{16}) \leftrightarrow 3
\end{align*}
\]
MD5

- MD4, MD5: same message padding, blocking, initial $d_i$
- MD4: 3 passes $\implies$ MD5: 4 passes
- different functions, different shifts
- MD4: two constants for all $m_i$ $\implies$ MD5: different

$$T_i = \lfloor 2^{32} | \sin i | \rfloor$$
MD5 Pass 1: Selection

\[
d_{(-i)\land 3} = (d_{(-i)\land 3} + F(d_{(1-i)\land 3}, d_{(2-i)\land 3}, d_{(3-i)\land 3}) + m_i + T_{i+1})
\leftarrow S_1(i \land 3)
\]

where \( S_1 = 7 + 5i \):

\[
\begin{align*}
    d_0 &= (d_0 + F(d_1, d_2, d_3) + m_0 + T_1) \leftarrow 7 \\
    d_1 &= (d_3 + F(d_0, d_1, d_2) + m_1 + T_2) \leftarrow 12 \\
    d_2 &= (d_2 + F(d_3, d_0, d_1) + m_2 + T_3) \leftarrow 17 \\
    d_3 &= (d_1 + F(d_2, d_3, d_0) + m_3 + T_4) \leftarrow 22 \\
    d_0 &= (d_0 + F(d_1, d_2, d_3) + m_4 + T_5) \leftarrow 7
\end{align*}
\]
MD5 Pass 2: Selection’

Selection function:

$$G(x, y, z) = (x \land z) \lor (y \land \sim z)$$

$$d_{(-i)^{\land3}} = (d_{(-i)^{\land3}} + F(d_{(1-i)^{\land3}}, d_{(2-i)^{\land3}}, d_{(3-i)^{\land3}}) + m_{(5i+1)^{\land15} + T_{i+17}}) \leftrightarrow S_2(i \land 3)$$

where $$S_2(i) = i(i + 7)/2 + 5$$:

$$d_0 = (d_0 + G(d_1, d_2, d_3) + m_1 + T_{17}) \leftrightarrow 5$$
$$d_1 = (d_3 + G(d_0, d_1, d_2) + m_6 + T_{18}) \leftrightarrow 9$$
$$d_2 = (d_2 + G(d_3, d_0, d_1) + m_{11} + T_{19}) \leftrightarrow 14$$
$$d_3 = (d_1 + G(d_2, d_3, d_0) + m_0 + T_{20}) \leftrightarrow 20$$
$$d_0 = (d_0 + G(d_1, d_2, d_3) + m_5 + T_{21}) \leftrightarrow 5$$
MD5 Pass 3: Ex-Or

\[ H(x, y, z) = x \oplus y \oplus z \text{ as in MD4} \]

\[
d_{(-i)\wedge 3} = (d_{(-i)\wedge 3} + H(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(3i+5)\wedge 15} + T_{i+33}) \leftrightarrow S_3(i \wedge 3)
\]

where \( S_3(i) = 4, 11, 16, 23, 4, \ldots \)

\[
d_0 = (d_0 + H(d_1, d_2, d_3) + m_5 + T_{33}) \leftrightarrow 4
\]
\[
d_1 = (d_3 + H(d_0, d_1, d_2) + m_8 + T_{34}) \leftrightarrow 11
\]
\[
d_2 = (d_2 + H(d_3, d_0, d_1) + m_{11} + T_{35}) \leftrightarrow 16
\]
\[
d_3 = (d_1 + H(d_2, d_3, d_0) + m_{14} + T_{36}) \leftrightarrow 23
\]
\[
d_0 = (d_0 + H(d_1, d_2, d_3) + m_1 + T_{37}) \leftrightarrow 4
\]
MD5 Pass 4: Ex-or’

\[ I(x, y, z) = y \oplus (x \lor \sim z) \]

\[ d_{(-i) \land 3} = (d_{(-i) \land 3} + I(d_{(1-i) \land 3}, d_{(2-i) \land 3}, d_{(3-i) \land 3}) + m_{(7i) \land 15} + T_{i+49}) \leftrightarrow S_4(i \land 3) \]

where \( S_4(i) = 6, 10, 15, 21, 6, \ldots \)

\[ d_0 = (d_0 + I(d_1, d_2, d_3) + m_0 + T_{49}) \leftrightarrow 6 \]
\[ d_1 = (d_3 + I(d_0, d_1, d_2) + m_7 + T_{50}) \leftrightarrow 10 \]
\[ d_2 = (d_2 + I(d_3, d_0, d_1) + m_{14} + T_{51}) \leftrightarrow 15 \]
\[ d_3 = (d_1 + I(d_2, d_3, d_0) + m_5 + T_{52}) \leftrightarrow 21 \]
\[ d_0 = (d_0 + I(d_1, d_2, d_3) + m_{12} + T_{53}) \leftrightarrow 6 \]
Secure Hash Standard (SHS)

- 64 bits $\rightarrow$ 160 bits
- similar to MD5
- 5 passes
- 5 32-bit digest words $\rightarrow$ digest $A|B|C|D|E$:

\[
\begin{align*}
A &= 67452301_{16} \\
B &= efcdab89_{16} \\
C &= 98badcf6_{16} \\
D &= 10325376_{16} \\
E &= c3d2e1f0_{16}
\end{align*}
\]
SHS

- start with 512-bit block
- fill 4 more blocks with recursion \( \Rightarrow \) 80 32-bit words:
- word \( W_n = W_{n-3} \oplus W_{n-8} \oplus W_{n-14} \oplus W_{n-16} \leftrightarrow 1 \)
- modify digest:

\[
A' = E + (A \leftrightarrow 5) + W_t + K_t + f(t, B, C, D) \\
B' = A \\
C' = B \leftrightarrow 30 \\
D' = C \\
E' = D
\]

- \( K_t = \lfloor 2^{30} \sqrt{x} \rfloor \), where \( x = \{2, 3, 5, 10\}[t/20] \)
SHS

Function $f()$:

\[
\begin{align*}
  f(t, B, C, D) &= (B \land C) \lor (\sim B \land D) & \text{for } 0 \leq t \leq 19 \\
  f(t, B, C, D) &= B \oplus C \oplus D & \text{for } 20 \leq t \leq 39 \\
  f(t, B, C, D) &= (B \land C') \lor (B \land D) \lor (C' \land D) & \text{for } 40 \leq t \leq 59 \\
  f(t, B, C, D) &= B \oplus C \oplus D & \text{for } 60 \leq t \leq 79
\end{align*}
\]