Outline

- Hash function lengths
- Hash function applications
- MD5 standard
- SHA-1 standard
- Hashed Message Authentication Code (HMAC)

Hash Function Properties

- Hash function
  - Also known as
    - Message digest
    - One-way transformation
    - One-way function
    - Hash
  - Length of \( H(m) \) much shorter than length of \( m \)
  - Usually fixed lengths: 128 or 160 bits

Desirable Properties of Hash Functions

- Consider a hash function \( H \)
  - Performance: Easy to compute \( H(m) \)
  - One-way property: Given \( H(m) \) but not \( m \), it's computationally infeasible to find \( m \)
  - Weak collision resistance (free): Given \( H(m) \), it's computationally infeasible to find \( m' \) such that \( H(m') = H(m) \).
  - Strong collision resistance (free): Computationally infeasible to find \( m_1, m_2 \) such that \( H(m_1) = H(m_2) \)

Length of Hash Image

- Question
  - Why do we have 128 bits or 160 bits in the output of a hash function?
  - If it is too long
    - Unnecessary overhead
  - If it is too short
    - Birthday paradox
    - Loss of strong collision property
Birthday Paradox

Question:
- What is the smallest group size $k$ such that
  - The probability that at least two people in the group have the same birthday is greater than 0.5?
  - Assume 365 days a year, and all birthdays are equally likely
- $P(k$ people having $k$ different birthdays): $Q(365,k) = \frac{365!}{(365-k)!365^k}$
- $P($at least two people have the same birthday): $P(365,k) = 1 - Q(365,k)$
- $k$ is about 23

Birthday Paradox (Cont’d)

Generalization of birthday paradox
- Given
  - a random integer with uniform distribution between 1 and $n$, and
  - a selection of $k$ instances of the random variables,
- What is the least value of $k$ such that
  - There will be at least one duplicate
  - with probability $P(n,k) > 0.5$?

Birthday Paradox (Cont’d)

Generalization of birthday paradox
- $P(n,k) \approx 1 - e^{-k(k-1)/2n}$
- For large $n$ and $k$, to have $P(n,k) > 0.5$ with the smallest $k$, we have
  $k = \sqrt{2n}$
- Example
  - $1.18*(365)^{1/2} = 22.54$

Birthday Paradox (Cont’d)

Implication for hash function $H$ of length $m$
- With probability at least 0.5
  - If we hash about $2^{m/2}$ random inputs,
  - Two messages will have the same hash image
  - Birthday attack

Conclusion
- Choose $m \geq 128$

Application: File Authentication

Hash Function Applications

Want to detect if a file has been changed by someone after it was stored
- Method
  - Compute a hash $H(F)$ of file $F$
  - Store $H(F)$ separately from $F$
  - Can tell at any later time if $F$ has been changed by computing $H(F')$ and comparing to stored $H(F)$
- Why not just store a duplicate copy of $F$???
Application: User Authentication

- Alice wants to authenticate herself to Bob
  - assuming they already share a secret key $K$
- Protocol:

  Alice
  
  "I'm Alice"
  
  computes $Y=H(R|K)$
  
  picks random number $R$

  Bob
  
  verifies that $Y=H(R|K)$

User Authentication… (cont’d)

- Why not just send…
  - $...K$, in plaintext?
  - $...H(K)$? i.e., what’s the purpose of $R$?

Application: Commitment Protocols

- Ex.: A and B wish to play the game of “odd or even” over the network
  1. A picks a number $X$
  2. B picks another number $Y$
  3. A and B “simultaneously” exchange $X$ and $Y$
  4. A wins if $X+Y$ is odd, otherwise $B$ wins
- If A gets $Y$ before deciding $X$, A can easily cheat (and vice versa for $B$)
  - How to prevent this?

Commitment… (Cont’d)

- Why is sending $H(X)$ better than sending $X$?
- Why is sending $H(X)$ good enough to prevent A from cheating?
- Why is it not necessary for B to send $H(Y)$ (instead of $Y$)?
- What problems are there if:
  1. The set of possible values for $X$ is small?
  2. B can predict the next value $X$ that A will pick?

Application: Message Encryption

- Assume A and B share a secret key $K$
  - but don’t want to just use encryption of the message with $K$
- A sends B the (encrypted) random number $R_1$
- B sends A the (encrypted) random number $R_2$
- And then…
Application: Message Authentication

- A wishes to authenticate (but not encrypt) a message M (and A, B share secret key K).

1. picks random number R
2. computes $Y = H(M|K|R)$

- Why is R needed? Why is K needed?

Application: Digital Signatures

Generating a signature

- Only one party (Bob) knows the private key

Verifying a signature

Building Hash Using Block Chaining Techniques (Cont’d)

- Meet-in-the-middle attack
  - Get the correct hash value $G$
  - Construct any message in the form $Q_1, Q_2, \ldots, Q_{n^2}$
  - Compute $H_i = E_{c_i}(H_{i-1})$ for $1 \leq i \leq (n^2)$.
  - Generate $2^{n^2}$ random blocks; for each block $X$, compute $E_{c_i}(H_{i-1})$.
  - Generate $2^{n^2}$ random blocks; for each block $Y$, compute $D_{c_i}(G)$.
  - With high probability there will be an $X$ and $Y$ such that $E_{c_i}(H_{i-1}) = D_{c_i}(G)$.
  - Form the message $Q_1, Q_2, \ldots, Q_{n^2}, X, Y$. It has the hash value $G$.

Modern Hash Functions

- MD5
  - Previous versions (i.e., MD2, MD4) have weaknesses.
  - Broken; collisions published in August 2004
  - Too weak to be used for serious applications
- SHA (Secure Hash Algorithm)
  - Weaknesses were found
- SHA-1
  - Broken, but not yet cracked
  - Collisions in $2^{39}$ hash operations, much less than the brute-force attack of $2^{80}$ operations
  - Results were circulated in February 2005, and published in CRYPTO ’05 in August 2005
- SHA-256, SHA-384, …
The MD5 Hash Function

MD5: Message Digest Version 5

• MD5 at a glance

Processing of A Single Block

Called a compression function

MD5: A High-Level View

Padding

• There is always padding for MD5, and padded messages must be multiples of 512 bits
• To original message M, add padding bits “10...0” – enough 0’s so that resulting total length is 64 bits less than a multiple of 512 bits
• Append L (original length of M), represented in 64 bits, to the padded message
• Footnote: the bytes of each 32-bit word are stored in little-endian order (LSB to MSB)

Padding… (cont’d)

• How many 0’s if length of M =
  • n * 512?
  • n * 512 – 64?
  • n * 512 – 65?
Preliminaries

- The four 32-bit words of the output (the digest) are referred to as $d_0$, $d_1$, $d_2$, $d_3$.
- Initial values (in little-endian order)
  - $d_0 = 0x67452301$
  - $d_1 = 0xEFCDAB89$
  - $d_2 = 0x98BADCFE$
  - $d_3 = 0x10325476$
- The sixteen 32-bit words of each message block are referred to as $m_0$, …, $m_{15}$
  - $(16*32 = 512$ bits in each block)

Notation

- $\sim x = \text{bit-wise complement of } x$
- $x \land y, x \lor y, x \oplus y = \text{bit-wise AND, OR, XOR of } x$ and $y$
- $x << y = \text{left circular shift of } x \text{ by } y \text{ bits}$
- $x + y = \text{arithmetic sum of } x$ and $y$ (discarding carry-out from the msb)
- $\lfloor x \rfloor = \text{largest integer less than or equal to } x$

Processing a Block -- Overview

- Every message block $Y_i$ contains 16 32-bit words:
  - $m_0, m_1, m_2, \ldots, m_{15}$
- A block is processed in 4 consecutive passes, each modifying the MD5 buffer $d_0, \ldots, d_3$.
  - Called $F, G, H, I$
- Each pass uses one-fourth of a 64-element table of constants, $T[1\ldots64]$:
  - $T[i] = \lfloor 2^{32} * \text{abs(sin}(i)) \rfloor$, represented in 32 bits
  - why these values?
- Output digest = input digest + output of 4th pass

Overview (Cont’d)

Logic of Each Step

1st Pass of MD5

- $F(x, y, z) \overset{def}{=} (x \land y) \lor (\sim x \land z)$
- 16 processing steps, producing $d_0, d_1, d_2, d_3$ output:
  - $d_i = d_j + (d_k + F(d_l, m_0, d_m) + m_o + T_p) << s$
  - values of subscripts, in this order

Overview (Cont’d)

Logic of Each Step

1st Pass of MD5

- $F(x, y, z) \overset{def}{=} (x \land y) \lor (\sim x \land z)$
- 16 processing steps, producing $d_0, d_1, d_2, d_3$ output:
  - $d_i = d_j + (d_k + F(d_l, m_0, d_m) + m_o + T_p) << s$
  - values of subscripts, in this order
Logic of Each Step (Cont’d)

- Within each pass, each of the 16 words of \( m_i \) is used exactly once
  - Round 1, \( m_i \) are used in the order of \( i \)
  - Round 2, in the order of \( p_2(i) \), where \( p_2(i) = (i + 5) \mod 16 \)
  - Round 3, in the order of \( p_3(i) \), where \( p_3(i) = (i + 3) \mod 16 \)
  - Round 4, in the order of \( p_4(i) \), where \( p_4(i) = (i + 7) \mod 16 \)
- Each word of \( T[i] \) is used exactly once throughout all passes
- Number of bits \( s \) to rotate to get \( d_j \)
  - Round 1, \( s(d_j) = 7 \), \( s(d_j) = 17 \), \( s(d_j) = 12 \)
  - Round 2, \( s(d_j) = 5 \), \( s(d_j) = 14 \), \( s(d_j) = 9 \)
  - Round 3, \( s(d_j) = 4 \), \( s(d_j) = 16 \), \( s(d_j) = 11 \)
  - Round 4, \( s(d_j) = 6 \), \( s(d_j) = 15 \), \( s(d_j) = 10 \)

\( \text{Insecurity of MD5} \)

- A few recently discovered methods can find collisions in a few hours
  - A few collisions were published in 2004
  - Can find many collisions for 1024-bit messages
  - More discoveries afterwards
  - In 2005, two X.509 certificates with different public keys and the same MD5 hash were constructed
    - This method is based on differential analysis
    - Much faster than birthday attack
    - 8 hours on a 1.6GHz computer

\( \text{2nd Pass of MD5} \)

- Form of processing (16 steps):
  \( d_j = d_j + (d_k + \mathcal{G}(d_i, d_m, d_o) + m_o + T_j) \ll s \)

\( \text{3rd Pass of MD5} \)

- Form of processing (16 steps):
  \( d_j = d_j + d_k + \mathcal{H}(d_i, d_m, d_o + m_o + T_j) \ll s \)

\( \text{4th Pass of MD5} \)

- Form of processing (16 steps):
  \( d_j = d_j + d_k + \mathcal{J}(d_i, d_m, d_o) + m_o + T_j \ll s \)

\( \text{The SHA-1 Hash Function} \)
Secure Hash Algorithm (SHA)

- Developed by NIST, specified in the Secure Hash Standard, 1993
- SHA is specified as the hash algorithm in the Digital Signature Standard (DSS)
- SHA-1: revised (1995) version of SHA

SHA-1 Parameters

- Input message must be \(< 2^{64}\) bits
- Input message is processed in 512-bit blocks, with the same padding as MD5
- Message digest output is 160 bits long
  - Referred to as five 32-bit words \(A, B, C, D, E\)
  - IV: \(A = \text{0x67452301}, B = \text{0xEFCDAB89}, C = \text{0x98BADCFE}, D = \text{0x10325476}, E = \text{0xC3D2E1F0}\)
  - Footnote: bytes of words are stored in big-endian order

Preprocessing of a Block

- Let 512-bit block be denoted as sixteen 32-bit words \(W_0..W_{15}\)
- Preprocess \(W_0..W_{15}\) to derive an additional sixty-four 32-bit words \(W_{16}..W_{79}\), as follows:
  
  for \(16 \leq t \leq 79\)
  
  \[W_t = (W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}) \ll 1\]

Block Processing

- Consists of 80 steps! (vs. 64 for MD5)
- Inputs for each step \(0 \leq t \leq 79\):
  - \(W_t\)
  - \(K_t\): a constant
  - \(A, B, C, D, E\): current values to this point
- Outputs for each step:
  - \(A, B, C, D, E\): new values
- Output of last step is added to input of first step to produce 160-bit Message Digest

Constants \(K_t\)

- Only 4 values (represented in 32 bits), derived from \(2^{30} \ast i/2\), for \(i = 2, 3, 5, 10\)
  - for \(0 \leq t \leq 19\): \(K_t = \text{0x5A827999}\)
  - for \(20 \leq t \leq 39\): \(K_t = \text{0x6ED9EBA1}\)
  - for \(40 \leq t \leq 59\): \(K_t = \text{0x8F1BBCDC}\)
  - for \(60 \leq t \leq 79\): \(K_t = \text{0xCA62C1D6}\)

Function \(f(t,B,C,D)\)

- 3 different functions are used in SHA-1 processing

<table>
<thead>
<tr>
<th>Round</th>
<th>Function (f(t,B,C,D))</th>
<th>Compare with MD-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \leq t \leq 19</td>
<td>((B \land C) \lor (\neg B \land D))</td>
<td>(\overline{f} = (x \land y) \lor (\neg x \land z))</td>
</tr>
<tr>
<td>20 \leq t \leq 39</td>
<td>(B \lor C \lor D)</td>
<td>(H = x \oplus y \oplus z)</td>
</tr>
<tr>
<td>40 \leq t \leq 59</td>
<td>((B \land C) \lor (B \land D) \lor (C \land D))</td>
<td>(H = x \oplus y \oplus z)</td>
</tr>
<tr>
<td>60 \leq t \leq 79</td>
<td>(B \lor C \lor D)</td>
<td></td>
</tr>
</tbody>
</table>

- No use of MD5’s \(g((x \land z) \lor (y \land \neg z))\) or \(i(y \oplus (x \land z))\)
Processing Per Step

- Everything to right of “=” is input value to this step

```plaintext
for t = 0 upto 79
    A = E + (A <<< 5) + W_t + K_t + f(t, B, C, D)
    B = A
    C = B <<< 30
    D = C
    E = D
endfor
```

Comparison: SHA-1 vs. MD5

- SHA-1 is a stronger algorithm
  - brute-force attacks require on the order of $2^{80}$ operations vs. $2^{64}$ for MD5
- SHA-1 is about twice as expensive to compute
- Both MD-5 and SHA-1 are much faster to compute than DES

Security of SHA-1

- SHA-1
  - Broken, but not yet cracked
  - Collisions in 269 hash operations, much less than the brute-force attack of 280 operations
  - Results were circulated in February 2005, and published in CRYPTO ’05 in August 2005
- SHA-256, SHA-384, …

The Hashed Message Authentication Code (HMAC)

Extension Attacks

- Given M1, and secret key K, can easily concatenate and compute the hash: H(K|M1|padding)
- Given M1, M2, and H(K|M1|padding) easy to compute H(K|M1|padding|M2|newpadding) for some new message M2
- Simply use H(K|M1|padding) as the IV for computing the hash of M2|newpadding
  - does not require knowing the value of the secret key K

Extension Attacks (Cont’d)

- Many proposed solutions to the extension attack, but HMAC is the standard
- Essence: digest-inside-a-digest, with the secret used at both levels
- The particular hash function used determines the length of the message digest = length of HMAC output
HMAC Processing

Key K

pad on right with 0's
512 bits in length

0x363636…36

compute message digest

concatenate

Message M

0x5c5c5c…5c

compute message digest

concatenate

HMAC(key, message)

Summary

• Hashing is fast to compute
• Has many applications (some making use of a secret key)
• Hash images must be at least 128 bits long
  – but longer is better
• Hash function details are tedious ☹
• HMAC protects message digests from extension attacks