COMP7120/8120 Cryptography and Data Security

Hash Function
Hash Function

- H(.) is also known as
  - Message digest
  - One-way transformation
  - One-way function
  - Hash
  - Widely-used hash functions: MD5, SHA1, SHA256.

- Usually fixed lengths: 128, 160, 256 bits, ...
Desirable Property of Hash Functions: 1

1) **Performance**: Easy to compute $H(m)$

Message $m$ of arbitrary length $\rightarrow$ Hash $H(.)$ $\rightarrow$ A fixed-length short message $H(m)$

Very fast to compute $H(m)$!
2) **One-way property**: Given \( H(m) \) but not \( m \), it’s **computationally infeasible** to find \( m \).

**Desirable Property of Hash Functions: 2**

- Message \( m \) of arbitrary length
- Hash \( H(.) \)
- A fixed-length short message \( H(m) \)

Computationally difficult to compute get \( m \) from \( H(m) \)
Desirable Property of Hash Functions: 3

3) Weak collision resistance: Given $H(m)$, it’s computationally infeasible to find $m'$ such that $H(m') = H(m)$.

Collision: two or more messages have the same hash.
4) **Strong collision resistance:** computationally infeasible to find \( m_1, m_2 \) such that \( H(m_1) = H(m_2) \)

- **message** \( m_1 \)
- **message** \( m_2 \)

A fixed-length short message
\[ H(m_1) = H(m_2) \]

**computationally difficult to compute** \( m_1, m_2 \)!
Summary of Properties

1) **Performance**: Easy to compute $H(m)$
2) **One-way property**: Given $H(m)$ but not $m$, it's *computationally infeasible* to find $m$
3) **Weak collision resistance**: Given $H(m)$, it's *computationally infeasible* to find $m'$ such that $H(m') = H(m)$.
4) **Strong collision resistance**: *computationally infeasible* to find $m_1, m_2$ such that $H(m_1) = H(m_2)$
Is Encryption a Good Hash Function?

- Computationally efficient?
- One way property?
- Collision resistance?
- Any arbitrary message length?
Hash Function Applications
Application: File Authentication

• Want to detect if a file has been changed by someone after it was stored.

• Method
  - Compute the 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 $H(F)$
Application: Password Storage

- The passwords in the system must be stored in a secure way.

- **Method**
  - Compute the hash $H(P)$ of password $P$
  - Store $H(P)$ in a plain-text file (can be known to everyone)
  - Then, how the system verify if the password is correctly entered by a user?
  - Why it is secure?
Saving Password with Salt

• An enhanced way:
  - Create a random number R,
  - Get Hash $H(R|P)$
    • i.e., hash the combination of random number R and the password P.
  - Save $H(R|P)$ and R in plaintext.
  - R is called the salt.

• How to verify the password?
• Is it more secure?
Application: Commitment Protocols

- Ensure two parties commit before they communicate.

- Ex.: A and B wish to play the game of “odd or even” over the network
  1. A picks a number X and B picks a number Y
  2. A and B “simultaneously” exchange X and Y
  3. A wins if X+Y is odd, otherwise B wins
Commitment... (Cont’d)

- If A gets Y before deciding X, A can easily cheat (and vice versa for B)
  - How to prevent this using hashing to make sure A and B are both committed before seeing the other’s number?
Commitment... (Cont’d)

• Proposal: A must **commit** to X before B will send Y
• Protocol:

A picks X and computes $Z = H(X)$

B picks Y

B verifies that $H(X) = Z$
Commitment... (Cont’d)

- Why is sending $H(X)$ better than sending $X$?
- Why is sending $H(X)$ good enough to prevent $A$ from cheating?
- What problems are there if:
  - The set of possible values for $X$ is *small*?
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
   \[ Y = H(M|K|R) \]

- What if an attack changes M, R, or Y?
- Why is R needed? Why is K needed?
Message Authentication

- Many hash based MAC design:
  - keyed-hash message authentication code (HMAC)
  - Need a hash function:
    - HMAC-MD5
    - HMAC-SHA1
    - ...

- Compare HMAC with CBC-MAC?
  - Computational efficiency?
  - Security?
  - Other?
Application: User Authentication

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

- Protocol:
  - Alice picks a random number $R$
  - Computes $Y = H(R|K)$
  - Sends $Y$ to Bob
  - 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?