Co-funded by the Erasmus+ Programme of the European Union

Cryptographic Hash Functions

Truong Tuan Anh
CSE-HCMUT

Outline

- Data integrity
- Hash functions
- …

Data Integrity

- Data usually transferred over insecure channel
- Many factors affecting the data
- Example: Alice sends to Bob a message. Bob needs to know if the message has been changed over the channel?

→ Need to assure data integrity

Hash Functions

- Provide assurance of data integrity
- Compute a short "fingerprint" of some data
  - If the data is changed, the fingerprint will no longer valid
  - Check the data integrity by re-computing the fingerprint and verify that the fingerprint has not changed
- The fingerprint = message digest

Hash Functions

- Let x be some data, its message digest is $h(x)$ where h is a hash function
- The message digest should be short, commonly 160 bits
- Keyed and unkeyed hash functions
  - Keyed hash functions are often used as message authentication codes (MACs)

Hash Function: Key and Unkeyed

Example: Alice wants to send to Bob message x

- Unkeyed hash function:
  - Alice computes $y = h(x)$, then sends x over the channel and stores y in a secure channel
  - Bob receives x and reads y and verifies if $y = h(x)$, if yes, Bob is confident that x is originated from Alice
- Keyed hash function: Alice and Bob share a secret key K which determines function h_K:
  - Compute $y = h_K(x)$, then send (x, y) over the channel
  - Bob receives (x, y) and verifies if $y = h_K(x)$, if yes, Bob is confident that neither x or y was modified by attackers

Hash Family

A *hash family* is a four-tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$, where the following conditions are satisfied:

1. $\mathcal{X}$ is a set of possible *messages*
2. $\mathcal{Y}$ is a finite set of possible *message digests* or *authentication tags*
3. $\mathcal{K}$, the *keyspace*, is a finite set of possible *keys*
4. For each $K \in \mathcal{K}$, there is a *hash function* $h_K \in \mathcal{H}$. Each $h_K : \mathcal{X} \to \mathcal{Y}$.

A pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is said to be a *valid pair* under the key K if $h_K(x) = y$.

Hash Family

- Unkeyed hash function:
  - Also a hash family in which there is only one possible key
- All we want is to prevent the construction of certain types of valid pairs by an adversary

Security of Hash Function

- It is desirable that the only way to produce a valid pair *(x, y)* is to first choose x and then compute $y = h(x)$ by applying the function h to x
- Other security requirement depending on the particular applications
- Usually, a hash function is considered to be secured if the following problems are **<u>difficult to solve</u>**

Hash Function: Security Requirement

Preimage

Instance: A hash function $h : \mathcal{X} \to \mathcal{Y}$ and an element $y \in \mathcal{Y}$.

Find: $x \in \mathcal{X}$ such that $h(x) = y$.

A hash function for which **Preimage** cannot be efficiently solved is often said to be *one-way* or *preimage resistant*.

Hash Function: Security Requirement

Second Preimage

Instance:	A hash function $h : \mathcal{X} \rightarrow \mathcal{Y}$ and an element $x \in \mathcal{X}$.
Find:	$x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

A hash function for which **Second Preimage** cannot be efficiently solved is often said to be *second preimage resistant*.

Hash Function: Security Requirement

Collision

Instance:	A hash function $h : \mathcal{X} \to \mathcal{Y}$.
Find:	$x, x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

A hash function for which **Collision** cannot be efficiently solved is often said to be *collision resistant*.

Presentations

- The Random Oracle Model and Algorithms (Sections 4.2.1 and 4.2.2)
- Cryptosystems RC2 and RC5
- Hash functions MD5 and RIPEMD