Fundamentals of Cryptography

David Jao

Topics in Quantum-Safe Cryptography

CryptoWorks21

UNIVERSITY OF WATERLOO

June 28, 2016

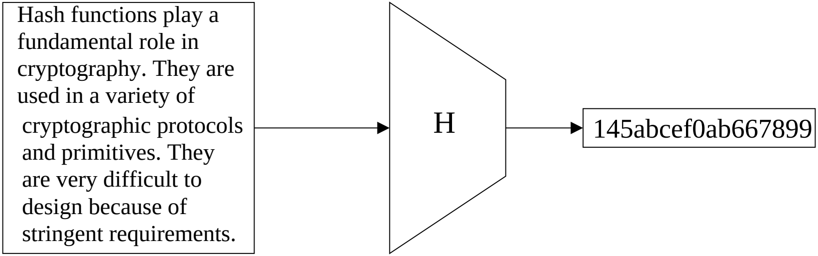
Part VI

Hash functions

Definitions and terminology

- Hash functions play a fundamental role in cryptography.
- They are used in a variety of cryptographic primitives and protocols.
- They are very difficult to design because of very stringent security and performance requirements.
- Examples: SHA-1, RIPEMD-160, SHA-224, SHA-256, SHA-384, SHA-512, BLAKE, SHA-3.

What is a hash function?



Definition of a hash function

- A hash function is a mapping H such that:
  (i) H maps inputs of arbitrary lengths to outputs of a fixed length n: $H : \{0,1\}^* \longrightarrow \{0,1\}^n$.
  (More generally, H maps elements of a set S to a set T where $|S| > |T|$.)
  (ii) $H(x)$ can be efficiently computed for all $x \in \{0,1\}^*$.
- H is called an n-bit hash function.
- $H(x)$ is called the hash value, hash, or message digest of x.
- Note: The description of a hash function is public. There are no secret keys.

Typical cryptographic requirements

- Preimage resistance: Given a hash value $y \in_R \{0,1\}^n$, it is computationally infeasible to find (with non-negligible probability of success) any input x such that $H(x) = y$.
  - x is called a preimage of y.
  - $y \in_R \{0,1\}^n$ means that y is chosen uniformly at random from $\{0,1\}^n$.
- 2nd preimage resistance: Given an input $x \in_R \{0,1\}^*$, it is computationally infeasible to find (with non-negligible probability of success) a second input $x' \neq x$ such that $H(x) = H(x')$.
- Collision resistance: It is computationally infeasible to find (with non-negligible probability of success) two distinct inputs x, x' such that $H(x) = H(x')$.
  - The pair (x, x') is called a collision for H.

Examples

Examples of hash functions:

- MD5 (RSA Laboratories; 1991)
- SHA-1 (NIST/NSA; 1995)
- SHA-2 (NIST/NSA; 2001)
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- SHA-3 (Bertoni, Daemen, Peeters, Van Assche; 2012)
  - SHA3-224
  - SHA3-256
  - SHA3-384
  - SHA3-512

NOT a hash function: "Hash tables", CRC8, CRC16, CRC32, ...

Generic attacks

- A generic attack on hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^n$ does not exploit any properties a specific hash function may have.
- In the analysis of a generic attack, we view H as a random function in the sense that for each $x \in \{0,1\}^*$, the value $y = H(x)$ was chosen by selecting y uniformly at random from $\{0,1\}^n$ (written $y \in_R \{0,1\}^n$).
- From a security point of view, a random function is an ideal hash function. However, random functions are not suitable for practical applications because they cannot be compactly stored.

Generic attack for finding preimages

- Given $y \in \{0,1\}^n$, select arbitrary $x \in \{0,1\}^*$ until $H(x) = y$.
- Expected number of steps is $\approx 2^n$.
  (Here, a step is a hash function evaluation.)
- This attack is infeasible if $n \geq 80$.

Note: It has been proven that this generic attack for finding preimages is optimal, i.e., no better *generic* attack exists.

Generic attack for finding collisions

- Select arbitrary $x \in \{0,1\}^*$ and store $(H(x), x)$ in a table sorted by first entry. Continue until a collision is found.
- Expected number of steps: $\sqrt{\pi 2^n/2} \approx \sqrt{2^n}$ (by birthday paradox). (Here, a step is a hash function evaluation.)
- It has been proven that this generic attack for finding collisions is optimal in terms of the number of hash function evaluations.
- Expected space required: $\sqrt{\pi 2^n/2} \approx \sqrt{2^n}$.
- This attack is infeasible if $n \geq 160$.
- If $n = 128$, the expected running time is about 2^{64} steps.

Some applications of hash functions

1. Password protection on a multi-user computer system:
   - Server stores (userid, H(password)) in a password file. Thus, if an attacker gets a copy of the password file, she does not learn any passwords.
   - Requires preimage-resistance.

2. Modification Detection Codes (MDCs).
   - To ensure that a message m is not modified by unauthorized means, one computes $H(m)$ and protects $H(m)$ from unauthorized modification.
   - e.g. virus protection.
   - Requires 2nd preimage resistance.

Some applications of hash functions

3. Message digests for digital signature schemes:
   - For reasons of efficiency, instead of signing a (long) message, the (much shorter) message digest is signed.
   - Requires preimage-resistance, 2nd preimage resistance and collision resistance. (More on this later)
   - To see why collision resistance is required:
     - Suppose that Alice can find two messages x_1 and x_2, with $x_1 \neq x_2$ and $H(x_1) = H(x_2)$.
     - Alice can sign x_1 and later claim to have signed x_2.

4. Message Authentication Codes (MACs).
   - Provides data integrity & data origin authentication.

Some applications of hash functions

5. Pseudorandom bit generation:
   - Distilling random bits s from several "random" sources $x_1, x_2, \ldots, x_t$.
   - Output $s = H(x_1, x_2, \ldots, x_t)$.

6. Key derivation function (KDF): Deriving a cryptographic key from a shared secret. (More on this later)

Notes:

- Collision resistance is not always necessary.
- Depending on the application, other properties may be needed, for example 'near-collision resistance', 'partial preimage resistance', ...

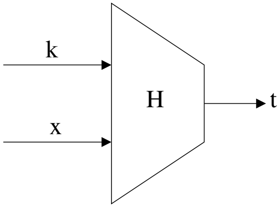
Part VII

Message authentication codes

Definition

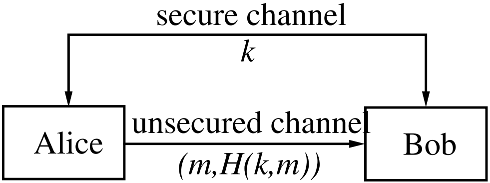
- A message authentication code (MAC) scheme is a family of functions $H_k : \{0,1\}^* \longrightarrow \{0,1\}^n$ parameterized by an ℓ-bit key k, where each function H_k can be efficiently computed.

- $H_k(x)$ is called the MAC or tag of x.



- MAC schemes are used for providing (symmetric-key) data integrity and data origin authentication.

Applications of MAC Schemes



- To provide data integrity and data origin authentication:
  1. Alice and Bob establish a secret key $k \in \{0,1\}^\ell$.
  2. Alice computes $t = H_k(x)$ and sends (x, t) to Bob.
  3. Bob verifies that $t = H_k(x)$.
- Note: No confidentiality or non-repudiation.
- To avoid replay, add a timestamp, or sequence number.
- Widely used in banking applications.

Security Definition

- Let k be the secret key shared by Alice and Bob.
- The adversary does not know k, but is allowed to obtain (from Alice or Bob) tags for messages of her choosing. The adversary's goal is to obtain the tag of any message whose tag she did not already obtain from Alice or Bob.
- Definition: A MAC scheme is secure if given some MAC tags $H_k(x_i)$ for x_i's of one's own choosing, it is computationally infeasible to compute (with non-negligible probability of success) a pair $(x, H_k(x))$ for any new message x.
  - That is, the MAC scheme must be existentially unforgeable against chosen-message attack.
- Note: A secure MAC scheme can be used to provide data integrity and data origin authentication.

Generic Attacks

Guessing the MAC of a message x:

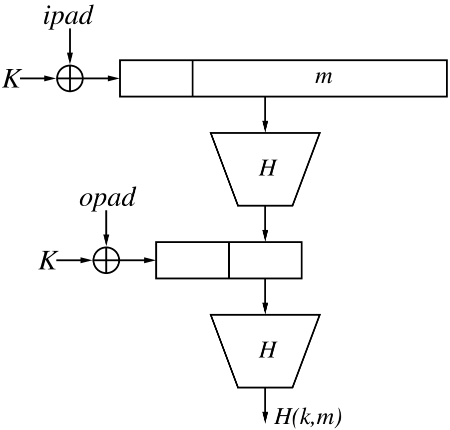
- Select $y \in \{0,1\}^n$ and guess that $H_k(x) = y$.
- Assuming that H_k is a random function, the probability of success is $1/2^n$.
- Note: Guesses cannot be directly checked.
- Depending on the application where the MAC algorithm is employed, one could choose n as small as 32 (say). In general, $n \geq 80$ is preferred.

Exhaustive search on the key space:

- Given r known message-MAC pairs: $(x_1, t_1), \ldots, (x_r, t_r)$, one can check whether a guess k of the key is correct by verifying that $H_k(x_i) = t_i$, for $i = 1, 2, \ldots, r$.
- Assuming that the H_k's are random functions, the expected number of keys for which the tags verify is $\mathrm{FK} = (2^\ell - 1)/2^{nr}$.
  - Example: If $\ell = 56$, $n = 64$, $r = 2$, then $\mathrm{FK} \approx 1/2^{72}$.
- Expected number of s is $\approx 2^\ell$.
- Exhaustive search is infeasible if $\ell \geq 80$.

HMAC

- "Hash-based" MAC; Bellare, Canetti & Krawczyk (1996).
- Define 2 r-bit strings (in hexadecimal notation): ipad $=$ 0x36, opad $=$ 0x5C; each repeated $r/8$ times.



- MAC definition: $H_K(x) = H(K \oplus \text{opad}, H(K \oplus \text{ipad}, x))$.

HMAC

- Proved secure:
  Theorem: Suppose that the compression function used in H is a secure MAC with fixed length messages and a secret IV as the key. Then HMAC is a secure MAC algorithm.
- HMAC is specified in IETF RFC 2104 and FIPS 198.
- SHA-3 is designed to be safe to use directly without HMAC: $H_K(x) = H(K, x)$. Other hash functions are **unsafe** to use directly.