Hash Functions and Applications (哈希函数及其应用)

Sheng Zhong Yuan Zhang

Computer Science and Technology Department
Nanjing University

Outline

Hash functions for compressing

- Recall we have seen PRG extends a short bit string into a long random-looking string.
- Sometimes, we might need to compress a long bit string into a short string.
- In this case, we use hash functions (sometimes called message digest functions).

Cryptographic hash functions

In this course, we study cryptographic hash functions.
By cryptographic (hash functions), we mean hash functions that can be used in cryptographic applications. Depending on the specific usage, we may require the hash functions to be:

- **collision resistant**: It is difficult to find two different messages m and m' such that $hash(m) = hash(m')$.
- **"completely unpredictable"** (a.k.a. **random oracles**): The hash function is indistinguishable from a random mapping.

Collision resistant hash functions

- We consider keyed hash functions: H is a two-input function that takes as input a key s and a string x, and outputs a string $H^s(x) = H(s, x)$.
- The key s is typically generated by a key generating algorithm *Gen* RATHER THAN being chosen uniformly.
- The key s is generally not kept secret (and the collision resistance is required even the adversary is given s).
- To emphasize the differences, we write H^s rather than H_s.

Formal definition of a hash function

DEFINITION 5.1

A **hash function** (with output length l) is a pair of PPT algorithms (Gen, H) satisfying the following:

- Gen is a probabilistic algorithm which takes as input a security parameter 1^n and outputs a key s.
- H takes as input a key s and a string $x \in \{0,1\}^*$ and outputs a string $H^s(x) \in \{0,1\}^{l(n)}$.

- If H^s is defined only for input $x \in \{0,1\}^{l'(n)}$ and $l'(n) > l(n)$, then we say (Gen, H) is a fixed-length hash function for inputs of length l'. In this case, we also call H a **compression function**.

The collision-finding experiment

The collision-finding experiment Hash-coll$_{\mathcal{A},\Pi}(n)$:

1. A key s is generated by running $Gen(1^n)$.
2. The adversary $\mathcal{A}$ is given s and outputs x, x'.
3. The output of the experiment is defined to be 1 iff $x \neq x'$ and $H^s(x) = H^s(x')$. In such a case, we say that $\mathcal{A}$ has found a collision.

The definition of collision-resistant hash function

DEFINITION 5.2

A hash function $\Pi = (Gen, H)$ is **collision resistant** if for all PPT adversaries $\mathcal{A}$ there is a negligible function *negl* such that

$$Pr[\text{Hash-coll}_{\mathcal{A},\Pi}(n) = 1] \leq negl(n).$$

- For simplicity, we sometimes omit *Gen* and refer to H or H^s as a collision-resistant hash function.
- In practice, unkeyed collision-resistant hash functions are usually used.

Weaker notations of security

In some applications, it suffices to rely on security requirements weaker than collision resistance:

- **Second-preimage or taget-collision resistance**: A hash function is second preimage resistant if given s and a uniform x it is infeasible for a PPT adversary to find $x' \neq x$ such that $H^s(x') = H^s(x)$.
- **Preimage resistance**: A hash function is preimage resistant if given s and a uniform y it is infeasible for a PPT adversary to find a value x such that $H^s(x) = y$.

In fact, we have

$$\text{c.r.} \Rightarrow \text{2nd p.r.} \Rightarrow \text{p.r.}$$

The Merkle-Damgard Transform for domain extension

- frequently used to construct hash functions (handling arbitrary-length inputs) with a collision-resistant compression function handling fixed-length inputs.
- It is used in MD5 and the SHA family (SHA0-SHA2).

Details of Merkle-Damgard Transform

Assume we have a compressing function (Gen, h) that compresses an $(a+b)$-bit input into an a-bit output $(a, b \in \mathbb{N})$, we can construct a hash function (Gen, H) as follows.

- *Gen*: same as the compressing function.
- *H*: on input a key s and a string x of arbitrary length, perform the following two major steps:
  - "Partition and Padding".
  - "Chained-Compressing".

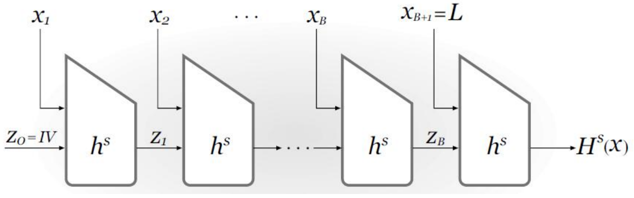


图 1: The Merkle-Damgård transform

"Partition and Padding"

Assume we have a compressing function (Gen, h) that compresses an $(a+b)$-bit input into an a-bit output $(a, b \in \mathbb{N})$. On input a key s and a string $x \in \{0,1\}^*$ of length $L < 2^b$, H does the following:

- Pad x with 0s so its length is a multiple of b.
- Partition the padded result into $B+1$ blocks of b bits ($B = \lceil \frac{L}{b} \rceil$):

$$x_1, \ldots, x_{B+1},$$

  where the last block stores the binary representation of value L.

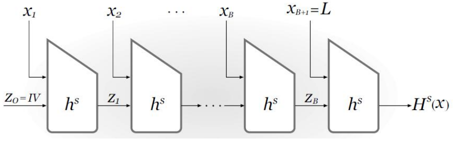


图 2: The Merkle-Damgård transform

"Chained Compression"

After the input has been partitioned, H does the following:

- Choose a constant initialization vector (IV) z_0 of length a, e.g. set $z_0 = 0^a$.
- Compute $z_i := h^s(z_{i-1} || x_i)$ for $i = 0, 1, \ldots, B+1$, and output

$$H^s(x) = z_{B+1}.$$

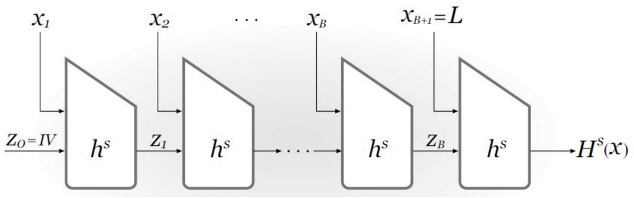


图 3: The Merkle-Damgård transform

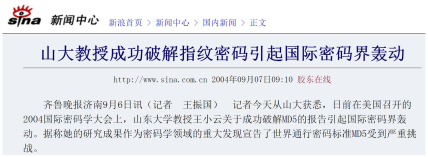
Security of the Merkle-Damgård Transform

THEOREM 5.4

If (Gen, h) is collision resistant, the (Gen, H) is also collision resistant.

MD5

- It is a widely used cryptographic hash function that is frequently used for integrity verification.
- Developed by Ron Rivest in 1992.
- Since 2004, efficient attacks on the collision resistance of MD5 are found.
- Now it is believed that MD5 is NOT secure.



新闻中心 新浪首页 > 新闻中心 > 国内新闻 > 正文

山大教授成功破解指纹密码引起国际密码界轰动

http://www.sina.com.cn 2004年09月07日09:10 胶东在线

齐鲁晚报济南9月6日讯（记者　王振国）　记者今天从山大获悉，日前在美国召开的2004国际密码学大会上，山东大学教授王小云关于成功破解MD5的报告引起国际密码界轰动。据称她的研究成果作为密码学领域的重大发现宣告了世界通行密码标准MD5受到严重挑战。

图 4: MD5 is broken by Prof. X. Wang.

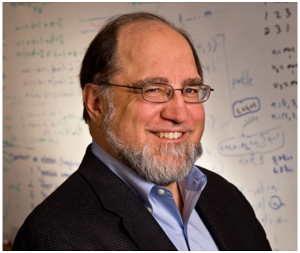


图 5: R. Rivest, co-inventor of the RSA algorithm, Turing award winner (2002), photo downloaded from https://www.soldierx.com/

SHA family

- SHA is short for the Secure Hash Algorithm which was first designed by the NSA and standardized by NIST.
- 1st version: SHA (outputs 160-bit digest, now often called SHA-0).
- 2nd version: SHA-1 (published in 1995, designed to fix SHA-0, was broken in 2005.)
- 3rd version: SHA-2 (published in 2001 and 2004, consists of SHA-224, SHA-256, SHA-384, SHA-512).
- Latest version: SHA-3 (released by NIST on August 5, 2015).

Using hash functions to construct MACs for arbitrary-length message

Q: Given a fixed-length MAC, how can we construct MACs for arbitrary-length message?
A: We use an approach that we call "hash-and-MAC". The idea is quite simple:

- Arbitrary-length message $m \in \{0,1\}^* \overset{Hash}{\to}$ fixed-length message $m \in \{0,1\}^{l(n)} \overset{MAC}{\to}$ tag t.

The hash-and-MAC paradigm

CONSTRUCTION 5.5

Let $\Pi = (Mac, Vrfy)$ be a MAC for messages of length $l(n)$, and let $\Pi_H = (Gen_H, H)$ be a hash function with output length $l(n)$. Construct a MAC $\Pi' = (Gen', Mac', Vrfy')$ for arbitrary-length messages as follows:

- Gen': on input 1^n, choose uniform $k \in \{0,1\}^n$ and run $Gen_H(1^n)$ to obtain s; the key is $k' := < k, s >$.
- Mac': on input a key $< k, s >$ and a message $m \in \{0,1\}^*$, output $t \leftarrow Mac_k(H^s(m))$.
- $Vrfy'$: on input a key $< k, s >$, a message $m \in \{0,1\}^*$, and a MAC tag t, output 1 if and only if $Vrfy_k(H^s(m), t) = 1$.

Security of hash-and-MAC

The Hash-and-MAC is a secure MAC if 1) a secure fixed-length MAC and a collision resistant hash function

THEOREM 5.6

If Π is a secure MAC for messages of length l and Π_H is collision resistant, then Construction 5.5 is a secure MAC for arbitrary-length messages.

Q: Why?

Why is Theorem 5.6 true?

Consider a sender uses Construction 5.5 to authenticate some set of messages $\mathcal{Q}$, and an adversary $\mathcal{A}$ is able to forge a valid tag on a new message $m^* \notin \mathcal{Q}$.
There are two possible cases:

- Case 1: There is a $m \in \mathcal{Q}$ such that $H^s(m^*) = H^s(m)$. Then $\mathcal{A}$ has found a collision in H^s, contradicting the collision resistance of H.
- Case 2: For every $m \in \mathcal{Q}$ it holds $H^s(m^*) \neq H^s(m)$. Then $\mathcal{A}$ has forged a valid tag on a new input $H(m^*)$ of Mac_k, contradicting the security assumption of Π.

Can we construct MAC solely with hash functions?

- We have seen:
  "secure fixed-len MAC $+$ c.r. hash $=$ secure arbitrary-len MAC".
- Can we construct a secure arbitrary-len MAC based directly on c.r hashes?

Simple but insecure constructions

Given a c.r. hash (Gen, H), consider the following two schemes for implementing $MAC_k(m)$:

- $H(k||m)$:
  vulnerable to “length-extension attacks”
- $H(m||k)$:
  A collision of H (e.g. H(m)=H(m’)) may result in MAC(m)=MAC(m’)

HMAC

- HMAC is an industrial standard of constructing MAC with hash functions.
- HMAC is highly efficient and easy to implement.
- HMAC is provable secure under reasonable assumptions about practical hash functions.
- HMAC has been widely used. e.g. HMAC-SHA1 and HMAC-MD5 are used within IPsec and TLS protocol. (Unlike MD5, practical vulnerability has not been found in HMAC-MD5.)

Details of HMAC

- HMAC computes $MAC_k(m)$ as

$$H\Big(k \oplus opad || H(k \oplus ipad || m)\Big),$$

where *opad* and *ipad* are specified constants.

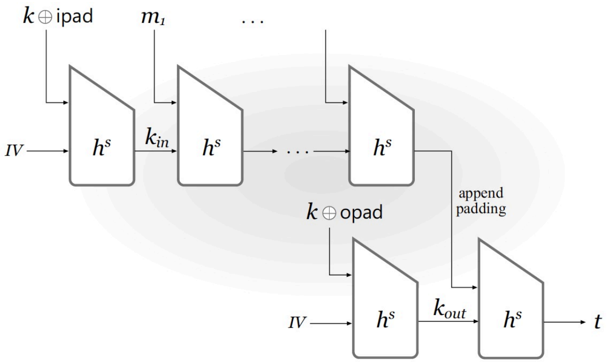


图 6: HMAC

Additional applications

Besides constructing MACs, Hash functions is also widely used in

- Fingerprinting and deduplication (e.g. virus fingerprint database, online storage deduplication, etc.)
- File consistency/integrity check (e.g. Merkle Tree)
- Password hashing
- Key derivation
- Commitment schemes

A trivial collision-finding attack

Let $H : \{0,1\}^* \to \{0,1\}^l$ be a hash function. A trivial way to find a collision is as follows:

1. Evaluate H on $2^l + 1$ distinct inputs.
2. Check for equal outputs.

- According to the pigeonhole principle, two of the outputs must be equal.
- Above attack requires $\Theta(2^l)$ hash evaluations. Can we do better?

Reducing hash evaluations

Consider the following attack with reduced hash evaluations:

Problem 1

1. Evaluate H on q distinct inputs ($q < 2^l$).
2. Check for equal outputs.

What's probability that the above attack finds a collision?

Maybe a small amount of evaluations can find a collision with a good probability?

Birthday problem

Assuming H is a random function, Problem 1 is reduced to the following:

Problem 2

If we choose $h_1, \ldots, h_q \in \{0,1\}^l$ uniformly at random, what's probability that there exist distinct i, j with $h_i = h_j$?

Problem 2 has been extensively studied and is related to the so-called birthday problem:

The birthday problem

If q people are in a room, what is the probability that two of them have the same birthday?

Success probability of birthday attack

Regarding the success probability of birthday attack, we can prove

Success prob of birthday attack

For $h_1, \ldots, h_q$ chosen uniformly in $\{1, N\}$, the probability of a collision is roughly 1/2 when $q = \Theta(N^{1/2})$.

- In the case of birthday, once there are only 23 people, the probability is greater 1/2.
- In our setting, taking $q = \Theta(2^{l/2})$ yields a collision with probability roughly 1/2.
- The above imply that a hash function with n-bit output is limited to $n/2$ bits of security.