Part I

Cryptographically Secure Hash Functions

1. Cryptographically Secure Hash Functions

Hash Functions

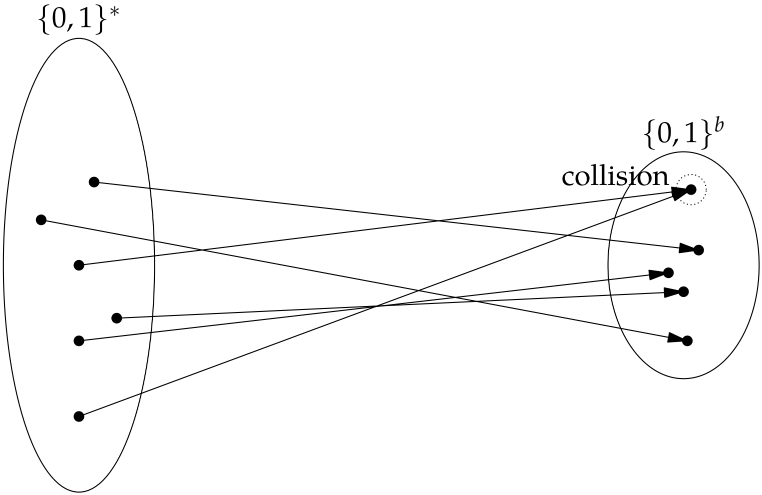
A hash function $h(\cdot)$ takes arbitrarily-length strings and compresses them into fixed-size strings.
Several uses outside cryptography: the *hash table* data structure, data partitioning, etc.

A collision is a pair of distinct inputs x, x' such that $h(x) = h(x')$.
Since the domain of $h(\cdot)$ is larger that the codomain and the codomain has finite size, then collisions always exist.
(Pigeon-hole principle)

The adversary looks for collisions. A good cryptographic hash function makes it hard to find collisions.

Hash Function

Example



Definition of Hash Function

Hash Function

An hash function h is a function satisfying:

- $h\colon \{0,1\}^* \to \{0,1\}^b$
- h can be calculated efficiently

- A hash function has no key.
- If $h\colon \{0,1\}^l \to \{0,1\}^b$ is defined only for input of fixed size $l > b$, then h is a *compression function*.
- If $h\colon \{0,1\}^l \to \{0,1\}^b$ is defined only for input of fixed size $l < b$, then h is a *expansion function*.

Definition of Security

Collision Resistance

A q-query adversary is an algorithm that can calculate the hash of at most q distinct inputs.

Collision finding experiment

1. The adversary outputs x and x'.
2. The experiment succeeds if $h(x) = h(x')$.

Given some large q, a hash function is collision resistant if for all q-query adversaries, the probability of success is negligible. Collision resistance is difficult to achieve and sometimes not necessary. Therefore we also consider other levels of security.

Definition of Security

Second Preimage Resistance

Second preimage finding experiment

1. The adversary receives x as input.
2. The adversary outputs x'.
3. The experiment succeds if $h(x) = h(x')$.

Given some large q, a hash function is second preimage resistant if for all q-query adversaries, the probability of success is negligible.

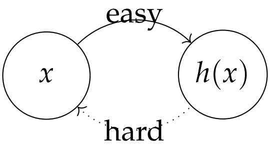
Definition of Security

Preimage Resistance

Preimage finding experiment

1. The adversary receives y as input.
2. The adversary outputs x.
3. The experiment succeds if $h(x) = y$.

Given some large q, a hash function is preimage resistant if for all efficient adversaries, the probability of success is negligible. In this case h is also said to be one-way.



About Weaker Notions of Security

A collision-resistant hash function is also second-preimage-resistant.
A second-preimage-resistant function is also preimage resistant.
We can prove that, by showing that an algorithm that can find a preimage can be used to find a second preimage. An algorithm that can find a second preimage can be used to find a collision.

Second-Preimage Resistance

Theorem

If h is second-preimage resistant, then h is preimage resistant.

We will show that an hypothetical q-query adversary $\mathcal{A}_h$ that can find a preimage with non-negligible probability, can be used to build a $(q+1)$-query $\mathcal{B}_h$ that can find a second preimage with non-negligible probability.

“Second-preimage-resistant” means that $\mathcal{B}_h$ does not exist, therefore also $\mathcal{A}_h$ does not exist. If $\mathcal{A}_h$ does not exist, h is also “preimage-resistant”.

Proof

We show how to build $\mathcal{B}_h$, given$\mathcal{A}_h$.

function $\mathcal{B}_h(x)$ ▷ Finds a second preimage
    $y = h(x)$
    $x' = \mathcal{A}_h(y)$ ▷ May return x but it is unlikely
    return x'
end function

Collision Resistance

Theorem

If h is collision resistant, then h is second-preimage resistant.

We will show that an hypothetical q-query adversary $\mathcal{B}_h$ that can find a second preimage with non-negligible probability, can be used to build a q-query $\mathcal{C}_h$ that can find a collision with non-negligible probability.

"Collision-resistant" means that $\mathcal{C}_h$ does not exist, therefore also $\mathcal{B}_h$ does not exist. If $\mathcal{B}_h$ does not exist, h is also "second-preimage-resistant".

Proof

We show how to build $\mathcal{C}_h$, given$\mathcal{B}_h$.

function $\mathcal{C}_h$ ▷ Finds a collision
    Choose x randomly
    $x' = \mathcal{B}_h(x)$
    return x, x'
end function

Exhaustive Search

Exhaustive search is a generic algorithm to find a preimage for any hash function.

```
function EXHAUSTIVE.SEARCH(y, q)
    Choose q distinct values x_1, x_2, ..., x_q
    y_i = h(x_i) for 1 ≤ i ≤ q
    if ∃i: y_i = y then
        return x_i                          ▷ Preimage found
    else
        return fail                         ▷ Preimage not found
    end if
end function
```

What is the probability of success?

Exhaustive Search

Probability of success

We can calculate a lower bound to the probability of success assuming that

- h is a random oracle

```
function RANDOM.ORACLE(x)
    if x is seen for the first time then
        Choose y uniformly at random in {0,1}^b
        store (x, y)
    else
        load (x, y)
    end if
    return y
end function
```

- q is *small enough* for some approximations to hold.

Exhaustive Search

Probability of success

The random oracle returns q random numbers. A preimage is found if at least one of them is the value looked for.

$$\boxed{\Pr\{\text{preimage found}\} = 1 - \left(1 - \frac{1}{2^b}\right)^q \simeq \frac{q}{2^b}}$$

This is in the random oracle model. Specific functions h can have higher success probabilities. For specific functions there are attacks that have higher success probabilities.

The Birthday Attack

The birthday attack is a generic algorithm to find collisions in any hash function.

Birthday Attack

function BIRTHDAY.ATTACK(q)
  Choose q distinct values $x_1, x_2, \ldots, x_q$
  $y_i = h(x_i)$ for $1 \leq i \leq q$
  if $\exists i, j\colon i \neq j\colon$ and $h(x_i) = h(x_j)$ **then**
    return x_i, x_j ▷ Collision found
  else
    `fail` ▷ Collision not found
  end if
end function

What is the probability of success?
If $q > 2^b$ then success is certain.

The Birthday Attack

Approximate probability of success

We can calculate an approximation to the probability of success assuming that:

- h is a random oracle
- q is *small enough* for some approximations to hold. Roughly $q \leq 2^{b/2}$.

The random oracle returns q random numbers. A collision is not found if the second number is different from the first, the third is different from the first two, etc.

$$\Pr\{\text{collision found}\} = 1 - \left(\frac{2^b - 1}{2^b}\right)\left(\frac{2^b - 2}{2^b}\right)\cdots\left(\frac{2^b - (q-1)}{2^b}\right) =$$

$$= 1 - \prod_{i=1}^{q-1}\left(1 - \frac{i}{2^b}\right) \simeq 1 - \prod_{i=1}^{q-1}\exp\left(-\frac{i}{2^b}\right) = 1 - \exp\left(-\sum_{i=1}^{q-1}\frac{i}{2^b}\right) =$$

$$1 - \exp\left(-\frac{q(q-1)}{2\cdot 2^b}\right) \simeq \frac{q(q-1)}{2\cdot 2^b} \simeq \frac{q^2}{2^{b+1}}$$

Exhaustive Search

- The probability of finding a preimage grows linearly with the number of attempts
- The probability of finding a collision grows with the square of the number of attempts

Roughly, for a q-query adversary

- a one-way function needs to have at least $\log_2 q$ bits of output,
- while a collision resistant function must have at least $2\log_2 q$ bits of output.

These are necessary conditions for any hash function.

1. Cryptographically Secure Hash Functions

The Merkle-Damgård Construction

Designing collision resistant hash functions is difficult. But assume we have a collision resistant compression function `compress`.

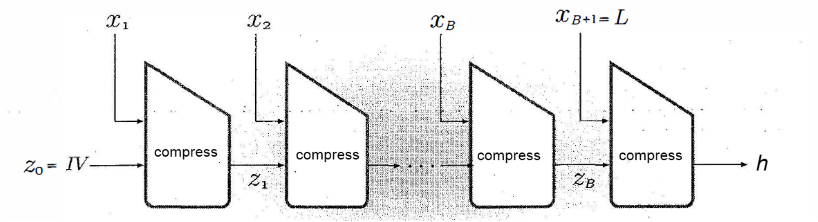
The Merkle-Damgård Construction enables to build a collision resistant hash function from a collision resistant compression function.

We see the special case in which `compress`: $\{0,1\}^{2b} \to \{0,1\}^{b}$. It can be generalized to any compression function.

Construction of h using Merkle Damgård

function $h(x)$
 $L = \text{len}(x), \quad B = \lceil L/b \rceil$ ▷ length in bits and in blocks
 Pad x with zeros and split in B blocks
 Let x_i be the ith block, with $1 \leq i \leq B$.
 $x_{B+1} = L$ ▷ encoded in bits
 $z_0 = IV$ ▷ can be any constant
 for $0 \leq i \leq B$ **do**
  $z_i = \texttt{compress}(z_{i+1} \| x_i)$
 end for
 return z_{B+1}
end function

The Merkle-Damgård Iterative Construction



Security of the Merkle-Damgård Construction

Theorem

If `compress` *is a collision resistant compression function, then h is a collision resistant hash function.*

We show that a collision in h yields a collision in `compress`.

The Merkle-Damgård Construction

Proof of security (1)

We prove by contradiction. Assume we know x and x' with lengths L and L' such that $x \neq x'$ and $h(x) = h(x')$. Recall that $x_{B+1} = L$ and $x'_{B+1} = L'$. There are two cases.

If $L \neq L'$, then the last step of the computation is $\texttt{compress}(z_B \| L) = H(x)$. Therefore

$$\texttt{compress}(z_B \| L) = \texttt{compress}(z'_B \| L')$$

Since $L \neq L'$ we have a collision in the compression function.

The Merkle-Damgård Construction

Proof of security (2)

If $L = L'$, then and x and x' must be different in at least one block. Let i^* be the highest index for which

$$z_{i^*-1} \| x_{i^*} \neq z'_{i^*-1} \| x'_{i^*}.$$

Since $z_{i^*} = z'_{i^*}$ we have a collision in the compression function.

So, we have an efficient algorithm to find a collision in `compress`, which contradicts the hypotesis. $\square$

MD Hash Functions in Practice

Name	year	output size	message block size	best known attack time
SHA-0	1993	160	512	2^{39}
SHA-1	1995	160	512	2^{63}
SHA-256	2002	256	512	2^{128}
SHA-512	2002	512	1024	2^{256}
MD4	1990	128	512	2^{1}
MD5	1992	128	512	2^{30}

Generally the security threshold is assumed to be 2^{80} meaning that a collision resistant hash function must have at least 160 bits of output.

SHA-3

The new NIST standard for hash functions does not belong to the MD family but to the new sponge family.

The standard defines four fixed-size hash functions: SHA3-224(m), SHA3-256(m), SHA3-384(m), SHA3-512(m).

The standard also defines two variable-size hash functions:

- SHAKE128(m, v) having the collision resistance performance of a 256-bit random oracle and output size v
- SHAKE256(m, v) having the collision resistance performance of a 512-bit random oracle and output size v

SHA-3

Name	output size	best known attack time
SHA-3-256	256	2^{128}
SHA-3-512	512	2^{512}
SHAKE128	v	$\min(2^{v/2}, 2^{128})$
SHAKE256	v	$\min(2^{v/2}, 2^{256})$