Cryptographic Hash Algorithms

Sachin Tripathi

IIT(ISM), Dhanbad

Outline

- Cryptographic hash function
- Important properties of hash function
- Security analysis of hash function
- Overview of different families of hash functions

Introduction

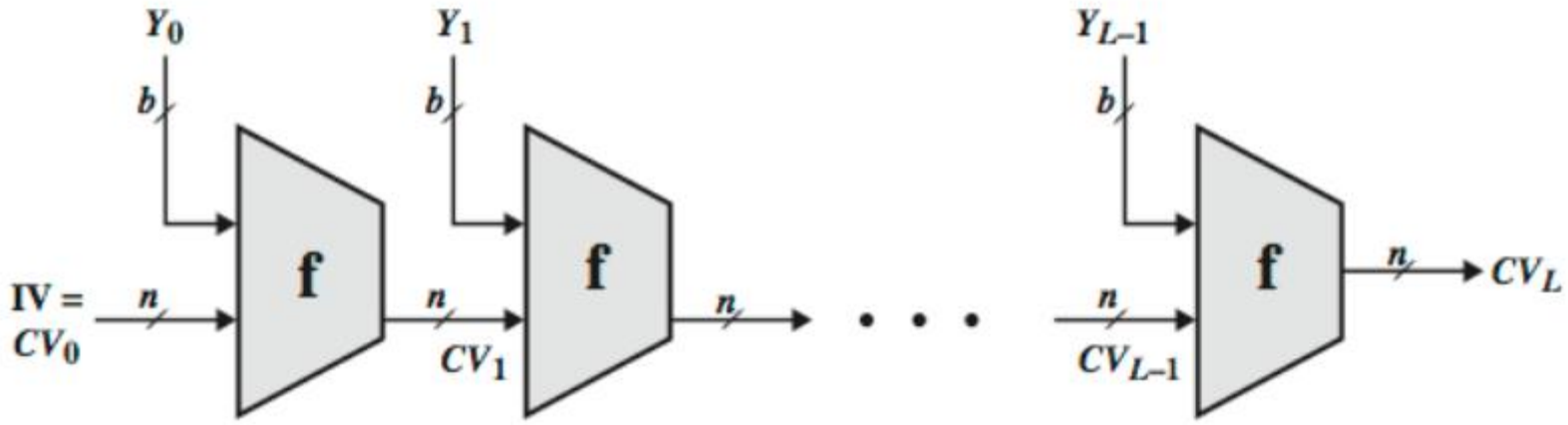
- A cryptographic hash function takes a message of arbitrary length and creates a message digest of fixed length.

- For a particular message, the message digest, or hash value, can be seen as the fingerprint of a message, i.e., a unique representation of a message.
- All cryptographic hash functions need to create a fixed-size digest out of a variable-size message.

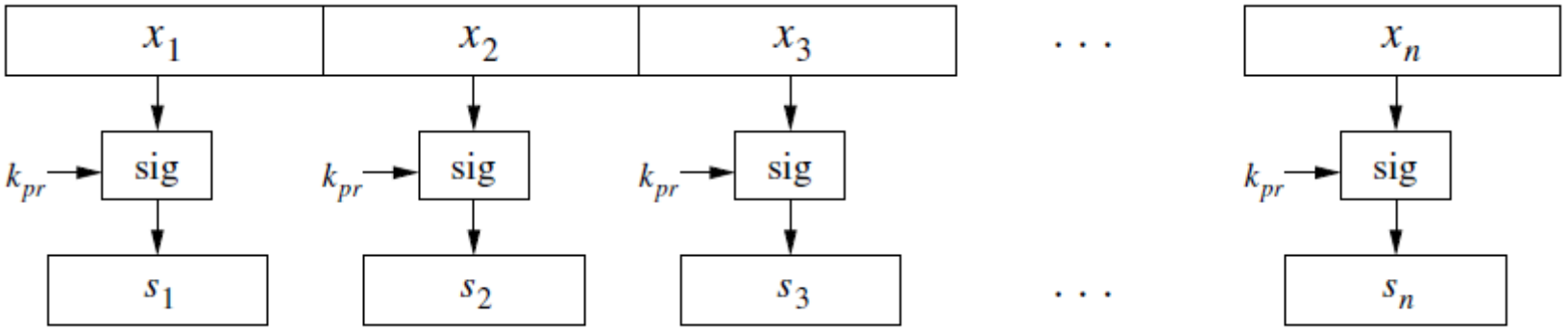
- The best way to create such function is using iteration, and used a necessary number of times.

- The fixed-size input function is referred to as a compression function.

Working Procedure

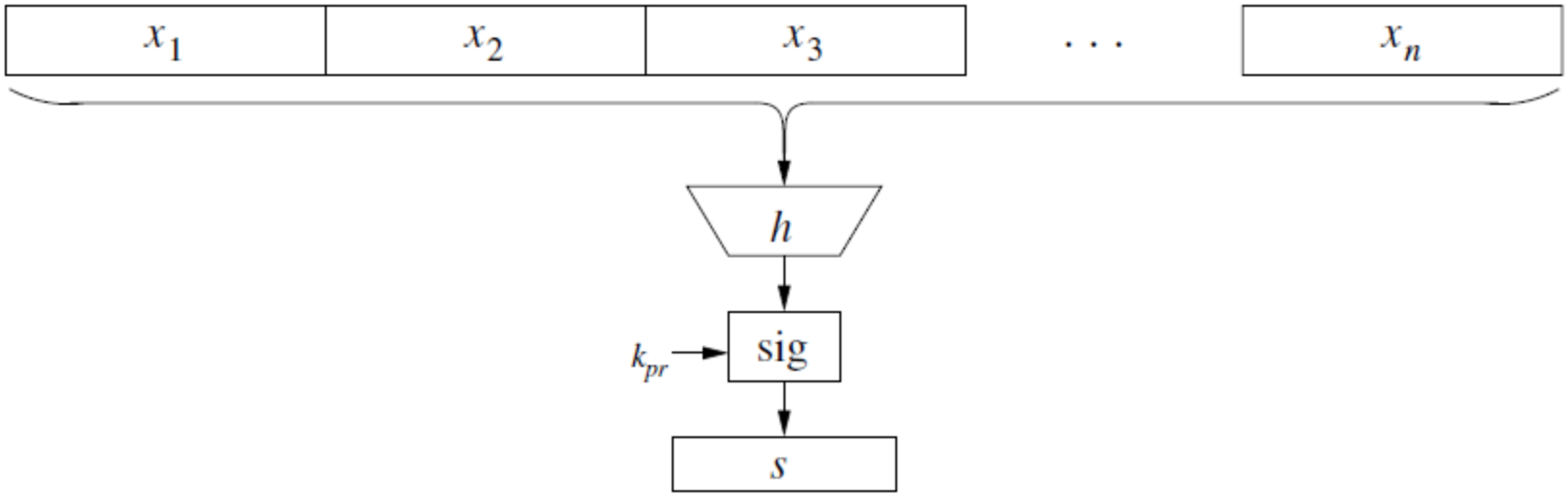


Motivation: Signing Long Messages



Problem 1: High Computational Load
Problem 2: Message Overhead
Problem 3: Security Limitations

Signing of long messages with a hash function



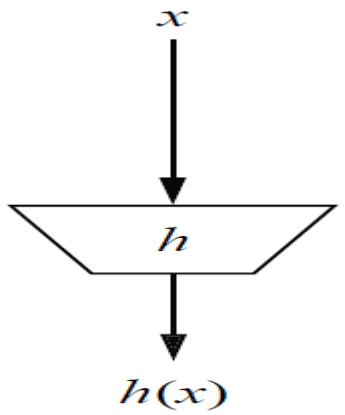
Security Requirements of Hash Function

- There are three central properties which hash functions need to possess in order to be secure:
- Preimage resistance (or one-wayness)
- Second preimage resistance (or weak collision resistance)
- Collision resistance (or strong collision resistance)

Preimage Resistance

- Hash functions need to be *one-way*

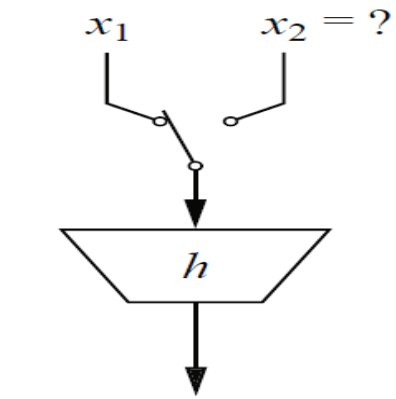
Given a hash output z it must be computationally infeasible to find an input message *x such that z = h(x).*



preimage resistance

Second Preimage Resistance

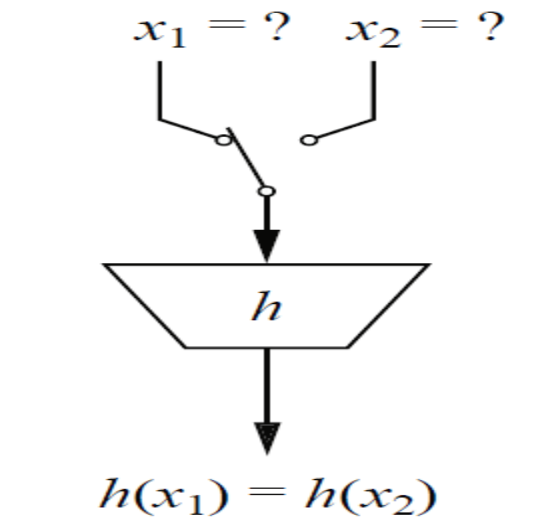
- It is essential that two different messages do not hash to the same value.
- It should be computationally infeasible to create two different messages *x1≠x2 with equal hash values h(x1) = h(x2).*



second preimage resistance

Collision Resistance

It is computationally infeasible to find two different inputs *x1≠x2 with h(x1) = h(x2).*



collision resistance

Properties of Hash Functions

1. **Arbitrary message size** $h(x)$ can be applied to messages x of any size.
2. **Fixed output length** $h(x)$ produces a hash value z of fixed length.
3. **Efficiency** $h(x)$ is relatively easy to compute.
4. **Preimage resistance** For a given output z, it is impossible to find any input x such that $h(x) = z$, i.e, $h(x)$ is one-way.
5. **Second preimage resistance** Given x_1, and thus $h(x_1)$, it is computationally infeasible to find any x_2 such that $h(x_1) = h(x_2)$.
6. **Collision resistance** It is computationally infeasible to find any pairs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.

Design Goals

Cryptographic hash function must be

- computationally infeasible to find data mapping to specific hash (**one-way property**)
- computationally infeasible to find two data to same hash (**collision-free property**)

Birthday Attack

- Due to the pigeonhole principle, collisions always exist.The question is how difficult it is to find them.
- Our first guess is probably that this is as difficult as finding second preimages, i.e., if the hash function has an output length of 80 bits, we have to check about 2^{80} messages. However, it turns out that an attacker needs only about 2^{40} messages due to the *birthday attack.*

Birthday Paradox

Problem Statement: How many people are needed at a party such that there is a reasonable chance that at least two people have the same birthday?

Solution:

$$P(\text{no collision among 2 people}) = \left(1 - \frac{1}{365}\right)$$

If a third person joins the party, he or she can collide with both of the people already there, hence:

$$P(\text{no collision among 3 people}) = \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right)$$

Consequently, the probability for t people having no birthday collision is given by:

$$P(\text{no collision among } t \text{ people}) = \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{t-1}{365}\right)$$

Contd...

For $t = 366$ people we will have a collision with probability 1 since a year has only 365 days. We return now to our initial question: how many people are needed to have a 50% chance of two colliding birthdays? Surprisingly—following from the equations above—it only requires 23 people to obtain a probability of about 0.5 for a birthday collision since:

$$\begin{aligned} P(\text{at least one collision}) &= 1 - P(\text{no collision}) \\ &= 1 - \left(1 - \frac{1}{365}\right) \cdots \left(1 - \frac{23-1}{365}\right) \\ &= 0.507 \approx 50\%. \end{aligned}$$

Note that for 40 people the probability is about 90%. Due to the surprising outcome it is often referred to as the birthday paradox.

Birthday Attack

- Collision search for a hash function h() is exactly the same problem as finding birthday collisions among party attendees.

- For a hash function there are not 365 values each element can take but 2^n, where n is the output width of h(). In fact, it turns out that n is the crucial security parameter for hash functions.
- The question is how many messages (x1,x2, . . . ,xt) does Eve need to hash until he has a reasonable chance that h(xi) = h(x j) for some xi and xj that he picked.

$$P(\text{no collision}) = \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{2}{2^n}\right)\cdots\left(1 - \frac{t-1}{2^n}\right)$$
$$= \prod_{i=1}^{t-1}\left(1 - \frac{i}{2^n}\right)$$

We recall from our calculus courses that the approximation

$$e^{-x} \approx 1 - x,$$

holds[1] since $i/2^n << 1$. We can approximate the probability as:

$$P(\text{no collision}) \approx \prod_{i=1}^{t-1} e^{-\frac{i}{2^n}}$$
$$\approx e^{-\frac{1+2+3+\cdots+t-1}{2^n}}$$

The arithmetic series

$$1 + 2 + \cdots + t - 1 = t(t-1)/2,$$

is in the exponent, which allows us to write the probability approximation as

$$P(\text{no collision}) \approx e^{-\frac{t(t-1)}{2\cdot 2^n}}.$$

Recall that our goal is to find out how many messages $(x_1, x_2, \ldots, x_t)$ are needed to find a collision. Hence, we solve the equation now for t. If we denote the probability of at least one collision by $\lambda = 1 - P(\text{no collision})$, then

$$\begin{aligned} \lambda &\approx 1 - e^{-\frac{t(t-1)}{2^{n+1}}} \\ \ln(1-\lambda) &\approx -\frac{t(t-1)}{2^{n+1}} \\ t(t-1) &\approx 2^{n+1} \ln\left(\frac{1}{1-\lambda}\right). \end{aligned}$$

Since in practice $t >> 1$, it holds that $t^2 \approx t(t-1)$ and thus:

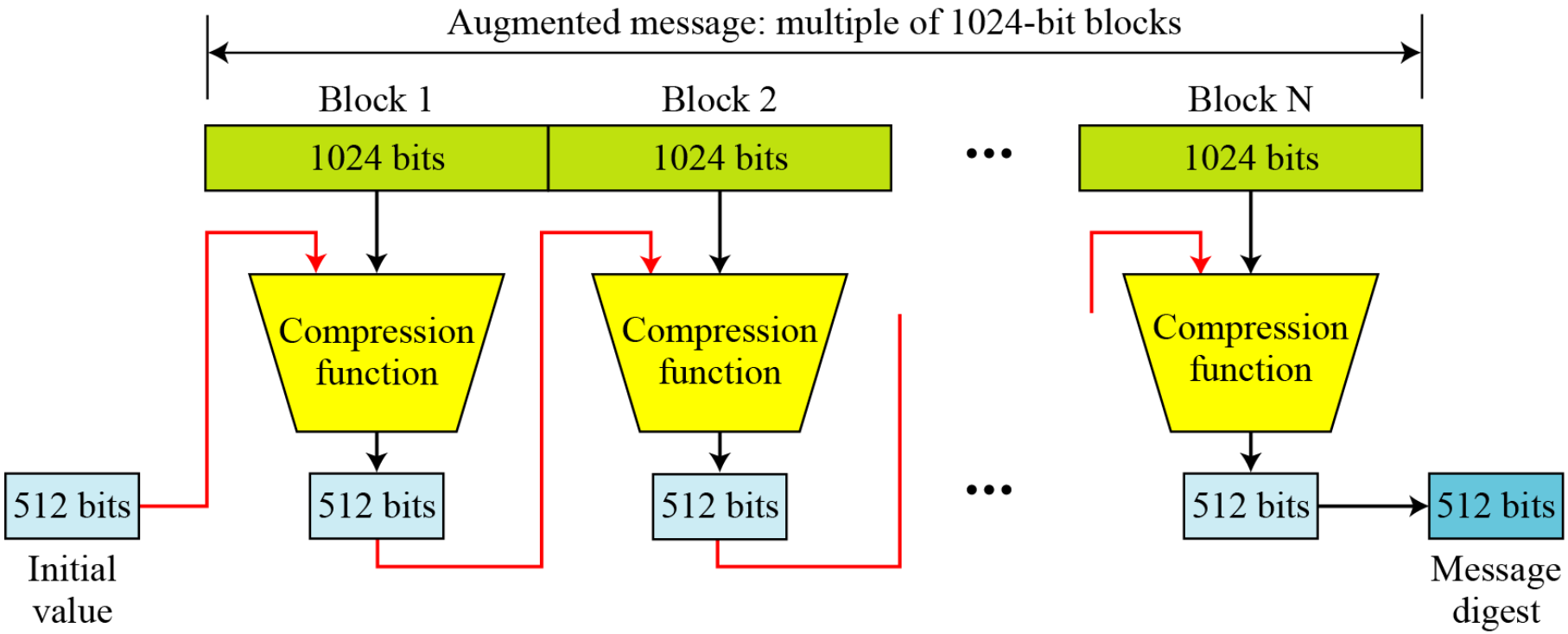
$$\begin{aligned} t &\approx \sqrt{2^{n+1} \ln\left(\frac{1}{1-\lambda}\right)} \\ t &\approx 2^{(n+1)/2} \sqrt{\ln\left(\frac{1}{1-\lambda}\right)}. \end{aligned}$$

The most important consequence of the birthday attack is that the number of messages we need to hash to find a collision is roughly equal to the square root of the number of possible output values

Block Ciphers as Hash Functions

- can use block ciphers as hash functions
  - using H_0=0 and zero-pad of final block
  - compute: $H_i = E_{M_i} [H_{i-1}]$
  - and use final block as the hash value
  - similar to CBC but without a key
- resulting hash is too small (64-bit)
  - both due to direct birthday attack
  - and to “meet-in-the-middle” attack
- other variants also susceptible to attack

SHA-512


Augmented message: multiple of 1024-bit blocks
Block 1
Block 2
Block N
1024 bits
1024 bits
1024 bits
Compression function
Compression function
Compression function
512 bits
512 bits
512 bits
512 bits
512 bits
Initial value
Message digest