Cryptographic Hash Functions

Chapter-11 in Stallings' Book

Hash function $h = H(M)$

A hash function H accepts a variable-length block of data M as input and produces a fixed-size hash value h.

↳ Good hash function ≈ for a large set of inputs it will produce outputs that are evenly distributed and apparently random

Cryptographic Hash Function:

This is an algorithm for which it is computationally infeasible to find either

These are basic requirements

(a) a data object that maps to a pre-specified hash result (the one-way property) → infeasible to find y where $H(y) = h$ ← given.

or

(b) two data objects that map to the same hash result (the collision-free property) → find x and y such that $H(x) = H(y)$.

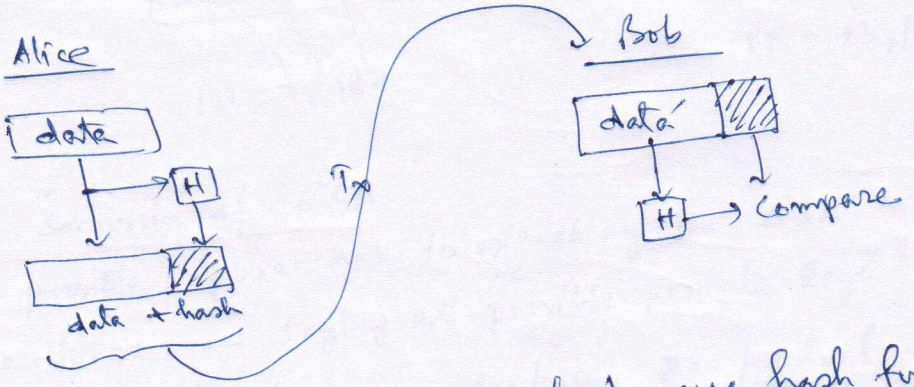
Use of Cryptographic Hash Functions

(1) Message Authentication

↳ It is a mechanism or service used to verify the integrity of a message.

(integrity means no modification, insertion, deletion or replay)

↳ when a hash function is used to provide message authentication the hash value is often referred as a message digest.



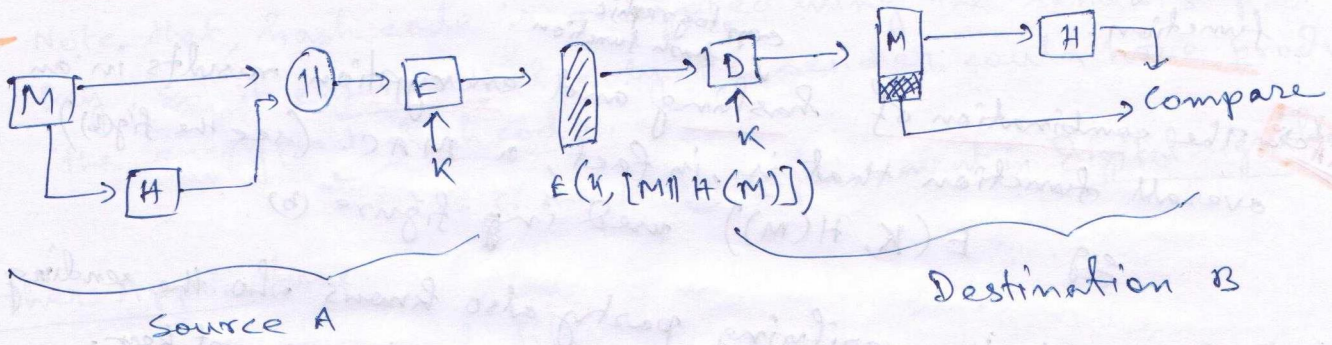
Simple approach to use hash function for data integrity

↳ However, this simple approach is vulnerable to MITM attack.

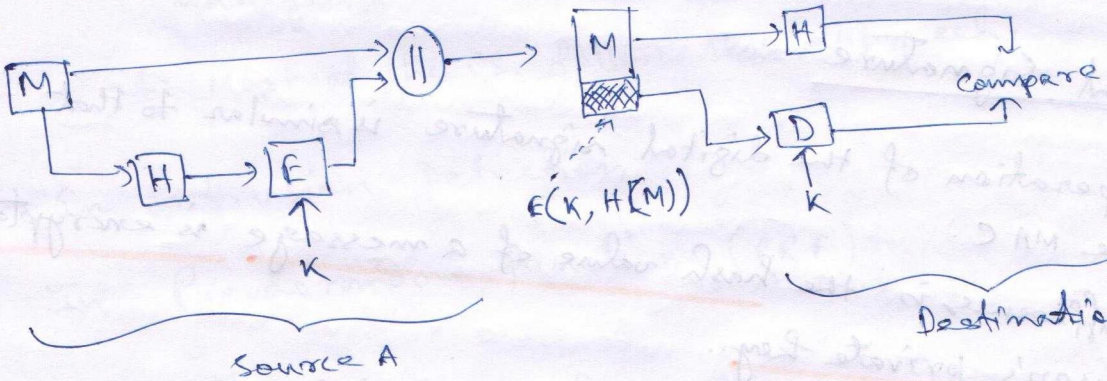
↳ So, hash value must be transmitted in a secure fashion.

Different ways in which a hash code can be used to provide message authentication.

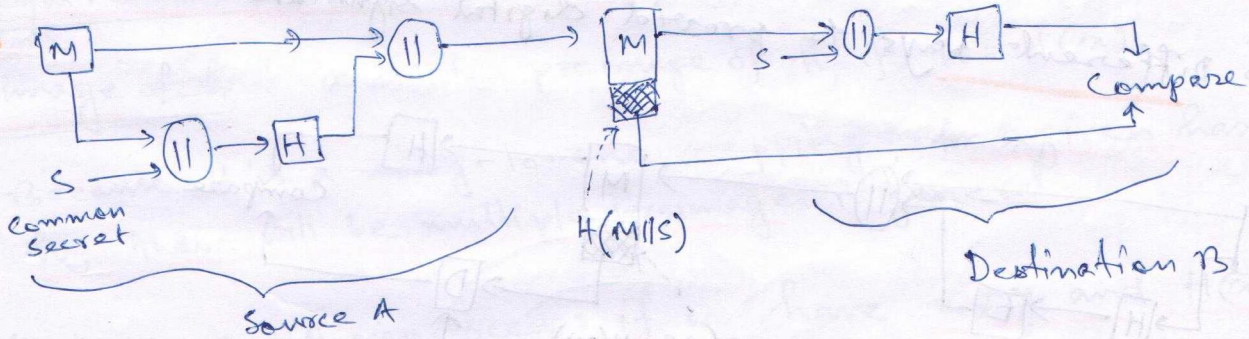
(a)



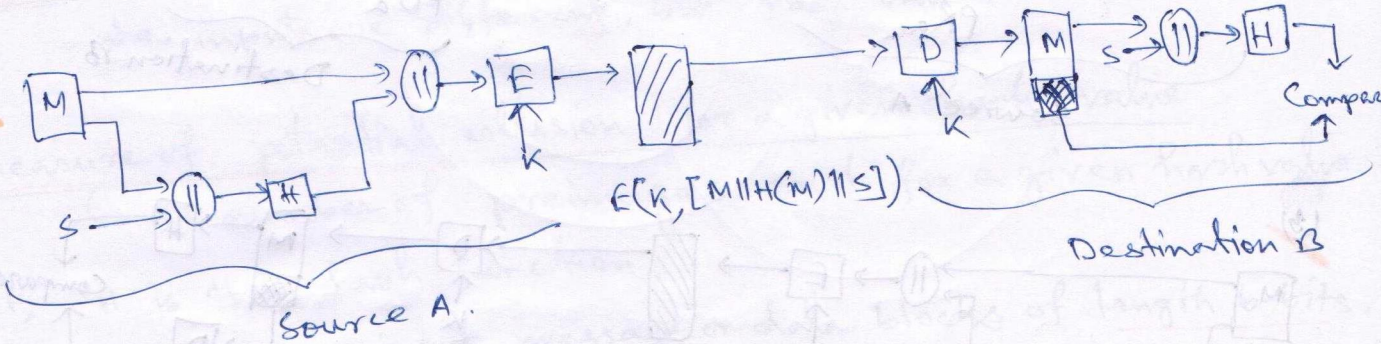
(b)



(c)



(d)



↳ when confidentiality is not required, method (b) is the good choice as no encryption is there.

↳ Encryption software is relatively slow.
Encryption hardware costs are not negligible.

↳ More commonly, Message Authentication is achieved using MAC (Message Authentication code) also known as keyed hash function.

↳ The combination of hashing (cryptographic hash function) and encryption results in an overall function that is, in fact, a MAC. (see the fig(b))

eg. E(K, H(M)) used in figure (b).

↳ Note that the verifying party also knows who the sending party is because no one else knows the secret key.

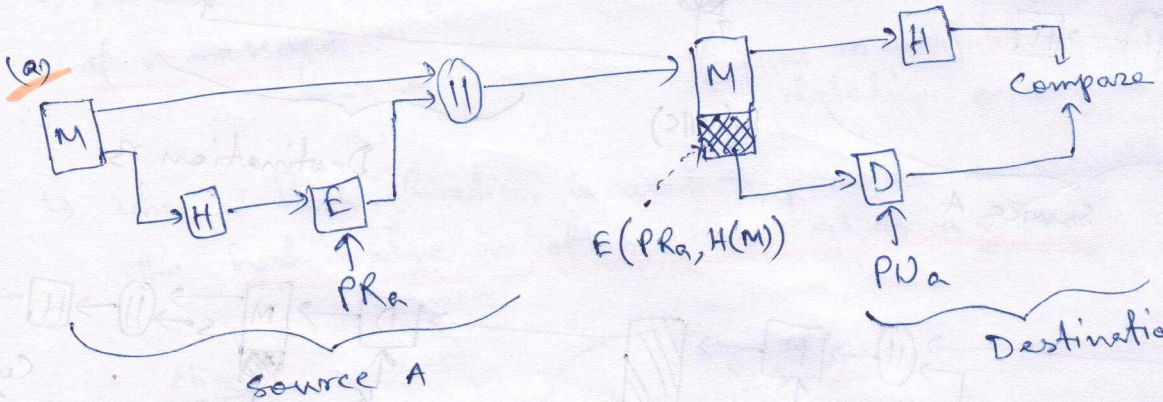
(2) Digital Signature

↳ The operation of the digital signature is similar to that of the MAC.

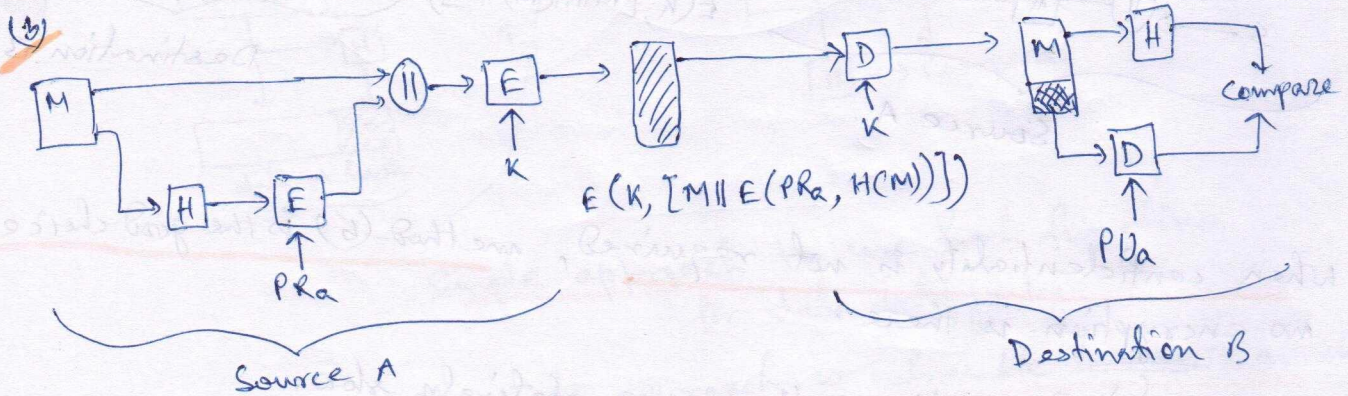
↳ only difference is the hash value of a message is encrypted with a user's private key.

↳ Different ways to provid digital signature service

(a)



(b)



↳ If confidentiality and digital signature both are required, then method (b) is the good choice.

↳ Note that, hash code is encrypted using the sender's private key. So, it ensures that only the sender could have produced the encrypted hash code.

This is the main requirement in digital signature.

Other Applications

(3) One-way password file

↳ So, the actual password is not retrievable by a hacker who gains access to the password file.

↳ Most of the OS. follow this approach.

(4) In Intrusion Detection

(5) In Pseudorandom function (PRF)

Requirements and Security

Two terms:

(a) Preimage of h : x is the preimage of h, if $h = H(x)$.

↳ Because H is a many-to-one mapping, for a given hash value h, there will be multiple preimages in general.

Collision : A collision occurs if we have $x \neq y$ and $H(x) = H(y)$

i.e. inputs are different, but hash values are same.

↳ Measure of potential collision for a given hash value

↳ ≈ number of preimages exist for a given hash value

Let, H is the hash function.

It takes as input message or data blocks of length b-bits.

It produces a hash code of length n-bits, $n < b$.

⇒ Total number of possible messages = 2^b

⇒ Total " " " hash value = 2^n

So, on average, each hash value corresponds to 2^{b-n} pre-images.

↳ If H tends to uniformly distribute hash values, then, in fact, each hash value will have close to 2^{b-n} preimages.

Requirements for H (w.r.t. security & performance)

① variable input size :- H can be applied to a block of data of any size.

② fixed output size :- H produces a fixed-length output

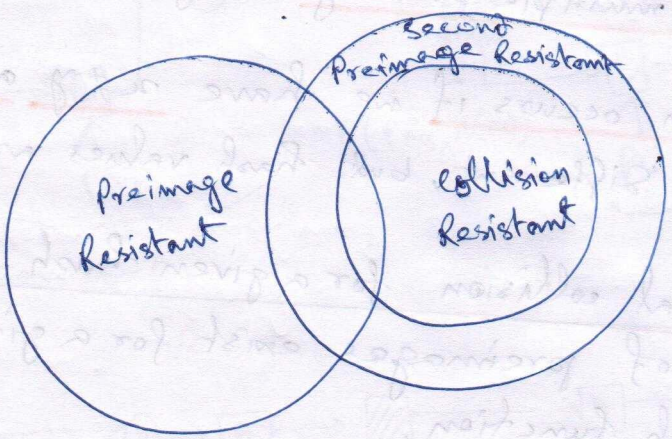
③ Efficiency :- H(x) is relatively easy to compute for any x

④ Preimage resistant (one-way property) :-
- for a given hash value h, it is computationally infeasible to find y such that H(y) = h.
- Message → code is easy, but code → message is infeasible.

⑤ second preimage resistant (weak collision resistant) :-
- For a given block x, it is computationally infeasible to find y ≠ x with H(y) = H(x)
- Infeasible to find alternative message with same hash value of given x.

⑥ Collision Resistant (= strong collision Resistant) :-
- It is computationally infeasible to find any pair (x, y) with x ≠ y, such that H(x) = H(y).

⑦ Pseudorandomness :- Output of H meets standard tests for pseudorandomness.



Relationship among the three resistant properties.

Hash function Resistance Properties Required for various data integrity applications / hash function applications:

		Preimage Resistant	Second Preimage Resistant	Collision Resistant
(1)	Hash + digital signature	Yes	Yes	Yes (*)
(2)	Intrusion detection and virus detection	×	Yes	×
(3)	Hash + Symmetric Encryption	×	×	×
(4)	One-way password file	Yes	×	×
(5)	MAC	Yes	Yes	Yes (*)

(*) Resistance required if attacker is able to mount a chosen message attack.

↳ Keep in mind that for any hash function there must ~~be~~ exist collisions, because we are mapping a message of length block size b into a hash code of length n, where $b \geq n$.

⇒ So, what is required is that it must be computationally infeasible to find collisions.

Attacks on Hash Functions:

Two types of attacks

↳ Brute-force Attacks — depends only on the bit length of hash value.

↳ Cryptanalysis — depends on weaknesses in a particular algo.

(A) ↳ Preimage and Second Preimage Attacks:

↳ Adversary wishes to find a value y such that $H(y)$ is equal to a given hash value h.

↳ Brute-force approach: pick values of y at random and try each value until a collision occurs.

↳ for m-bit hash value, ~~proportional~~ level of effort is proportional to 2^m.

↳ In fact, the adversary need to try only 2^{m-1} values of y.

Proof is in Appendix.

(3) Collision Resistant Attacks

↳ Adversary wishes to find two messages or data blocks x and y, that yield same hash code $H(x) = H(y)$

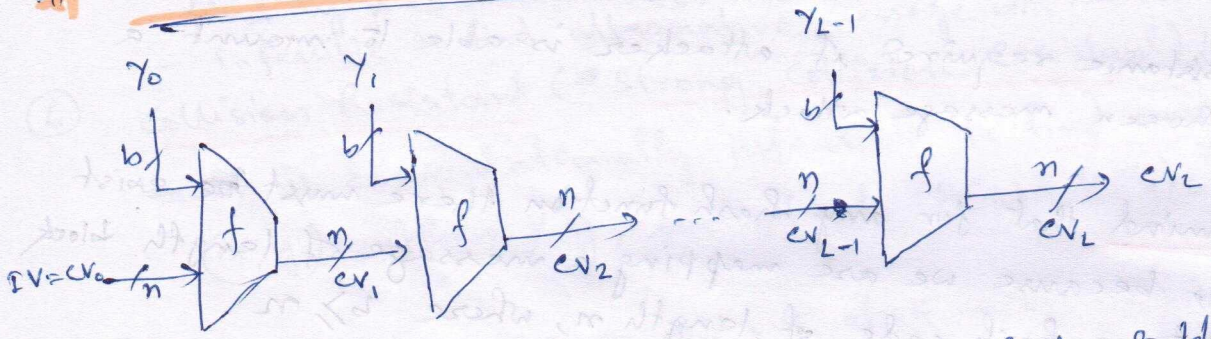
↳ for an m-bit hash value, if we pick data blocks a random, we can expect to get two data blocks with the same hash value within $\sqrt{2^m} = 2^{m/2}$ attempts

↳ ≈ Birthday Paradox ↳ proof in Appendix.

Cryptanalysis Attacks

↳ seeks to exploit some property of the algorithm.

General Structure of Secure Hash Function



IV = Initial Value

CV_i = Chaining Variable

Y_i = i-th input block

f = Compression Algorithm

L = Number of input blocks

n = Length of hash code

b = Length of input block

So, the problem of designing a secure hash function reduces to that of designing a collision-resistant compression function.