"ql"

Lecture 31: Cryptographic Hash Functions

→ A Hash Function H accepts a variable length block of data M as input and produces a fixed size hash value $h = H(M)$

→ Principal Objective is Data Integrity.

→ A change in one bit or more bits in M results, with high probability, in a change to the Hash code.

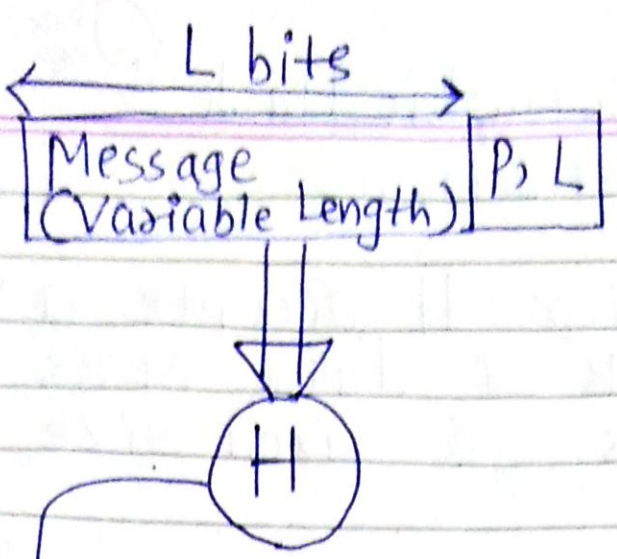
→ A Cryptographic Hash function is an algorithm for which it is computationally infeasible to find either

1. Data object that maps to a pre-specified hash result (Onewayness property)

or

2. Two data objects that map to the same hash result (the Collision-free property)

Because of these characteristics, hash functions are often used to determine whether or not data has changed.



P, L = Padding plus Length Field

Figure 1: Cryptographic Hash Function; $h = H(M)$

☆ Topic: Applications of Cryptographic Hash Function:

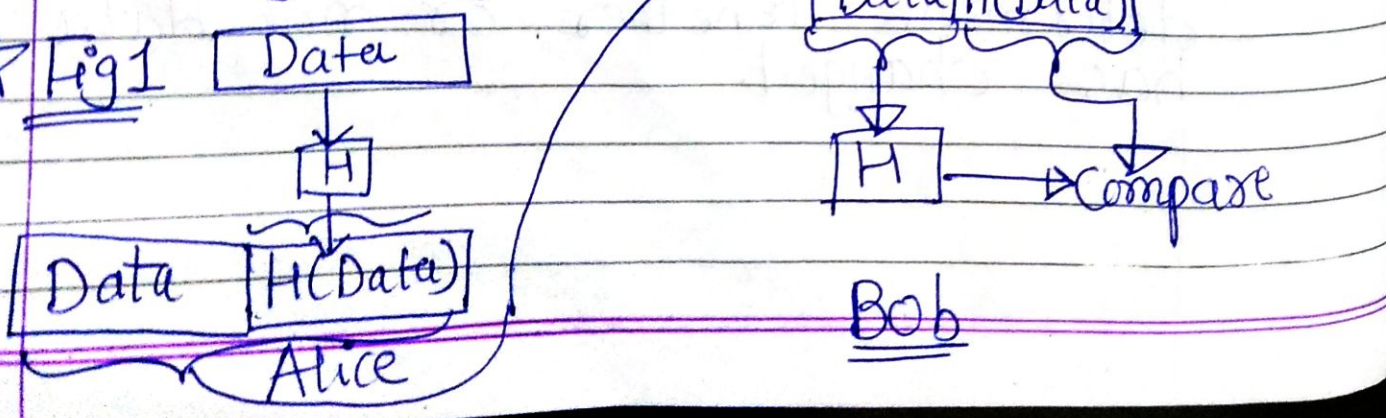
1. Message Authentication

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

→ In many cases, there is a requirement that the authentication mechanism assures the Identity of the sender.

→ When a hash function is used to provide message authentication, the hash value is often referred to as ~~hash~~ ~~value~~ message digest.

☆ Fig 1



→ In the figure Fig 1,

Alice: ① Alice creates Hash value H(Data)
② Data and Hash value H(Data) are sent to Bob i.e.

Data	H(Data)

Bob: ① Upon receiving data part (say Data'), Bob will again perform hash over it i.e.
Hash(Data')

② If Hash of Data' i.e.

H(Data') == H(Data),

H(Data'): Hash value computed for the received data i.e. data'.

H(Data): Hash value received by Bob from Alice

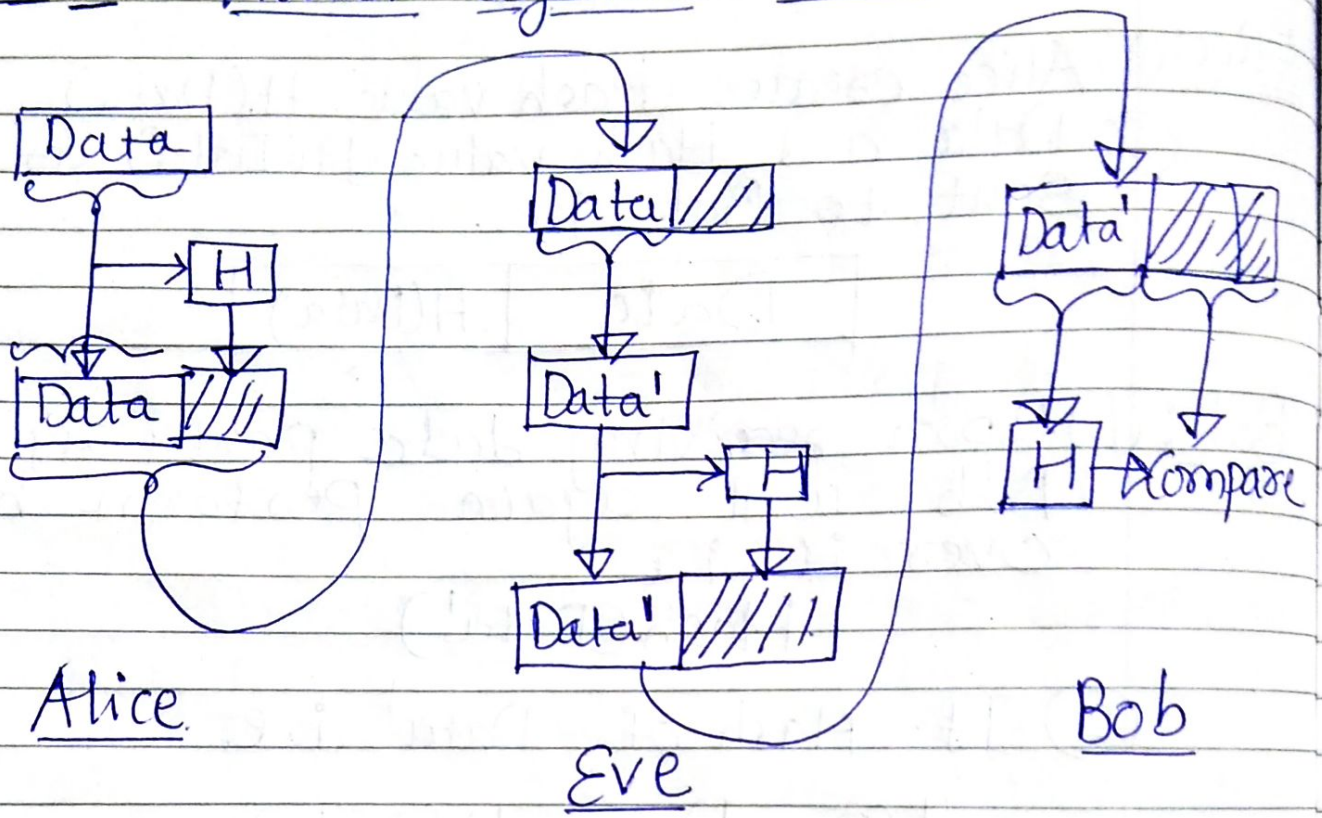
then Integrity of data is Verified

↓

How?

If Data is received without any modification then hash value will remain same, otherwise i.e. If H(Data) ≠ H(Data') then Data' (Received Data) has been modified and that indicates Violation of Integrity.

Figure 2 : Attack against Hash Function :



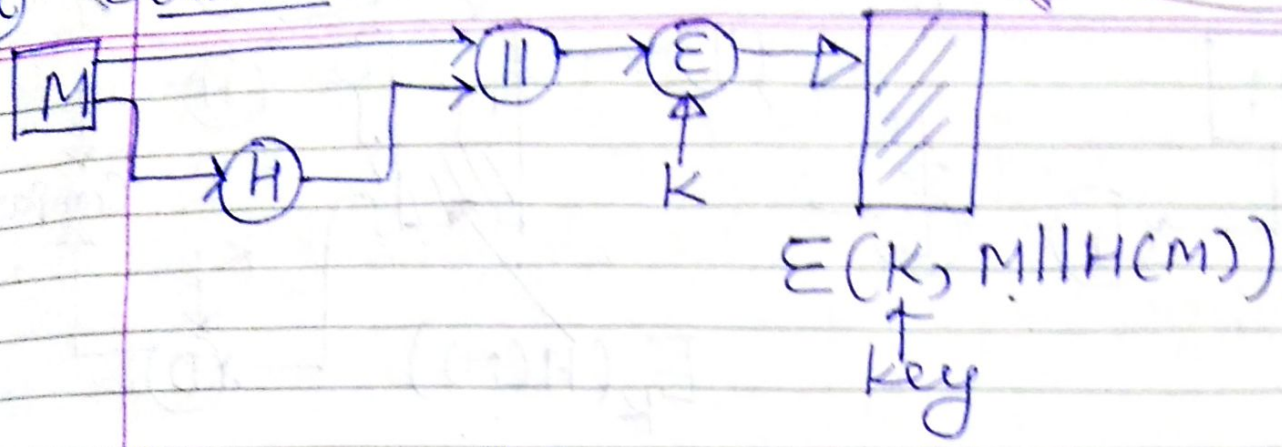
We can see that if an Adversary Eve changes the Data to Data' and Create Hash value, Eve can fool Bob because when Bob will verify Hash value, it will be compared and will produce true. This way Eve can fool Bob.

How to prevent this ?

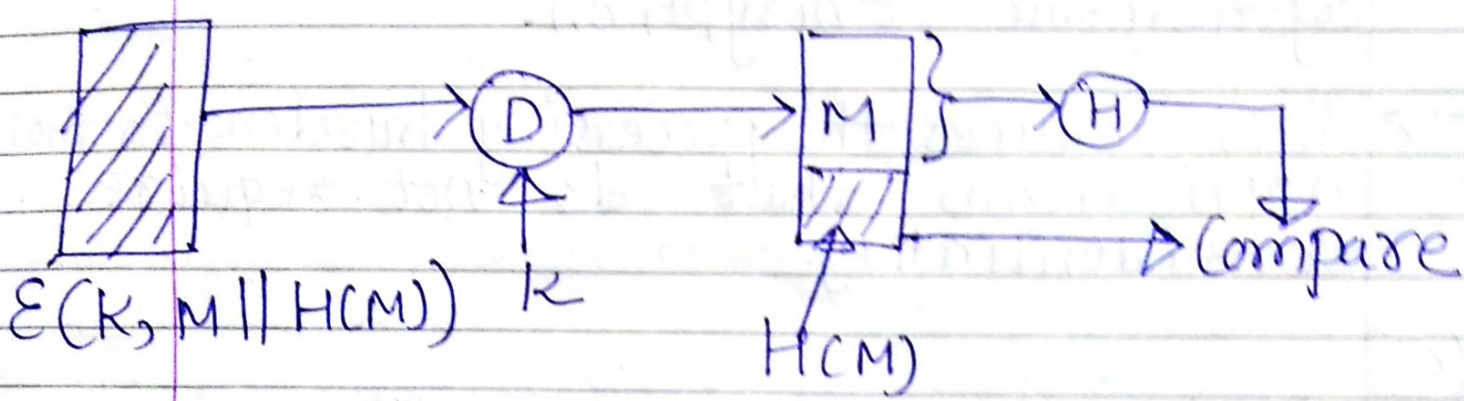
The Hash Value generated by Alice must be protected.

→ Figure 3 illustrates a variety of ways in which a Hash code can be used to provide Message Authentication, as follows.

@ Source A



Destination B

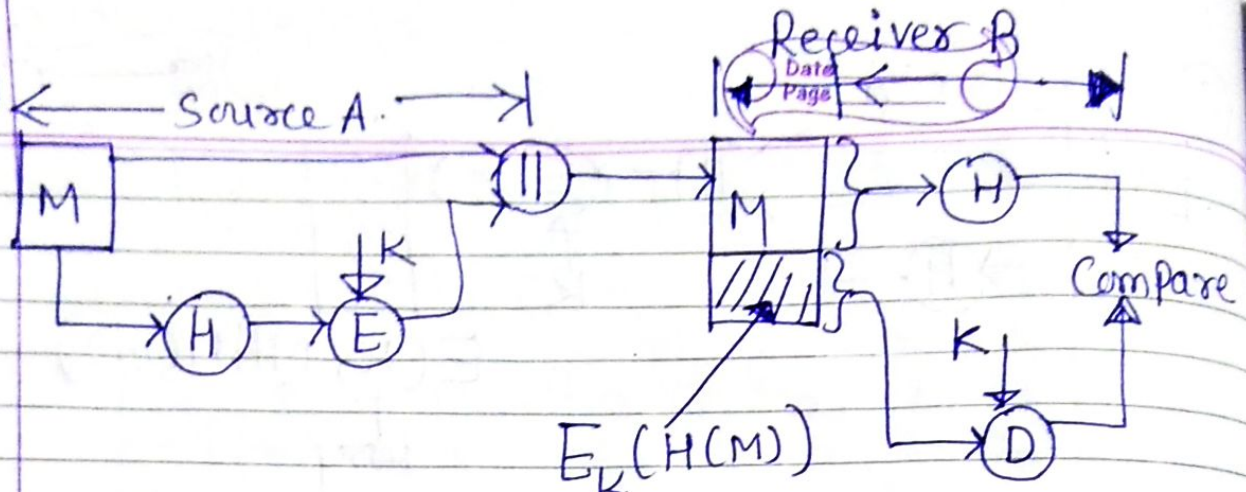


E: DES/AES like algo
D: DES/AES like Decryption algo
K: Secret key

Goals achieved

1. Confidentiality: Because M is encrypted using key K by Symmetric Cipher.
2. Authentication Of Message: We can verify that Source is A as after Decryption using shared Secret key K, Comparison returns true i.e. Hash value's are same.

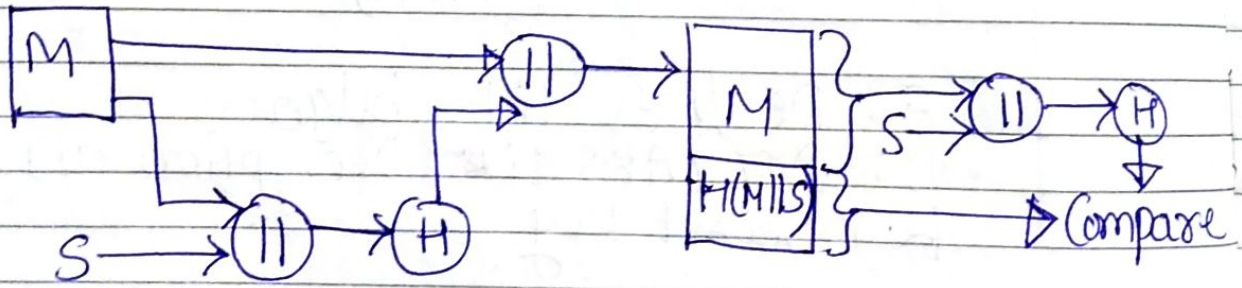
(b)



→ Only the Hash Code is Encrypted, using Symmetric Encryption.

→ This reduces the processing burden for those applications that do not require Confidentiality.

(c)



S: Common Secret (Known to only Alice and Bob)

This option (c) uses a hash function but no encryption for message authentication.

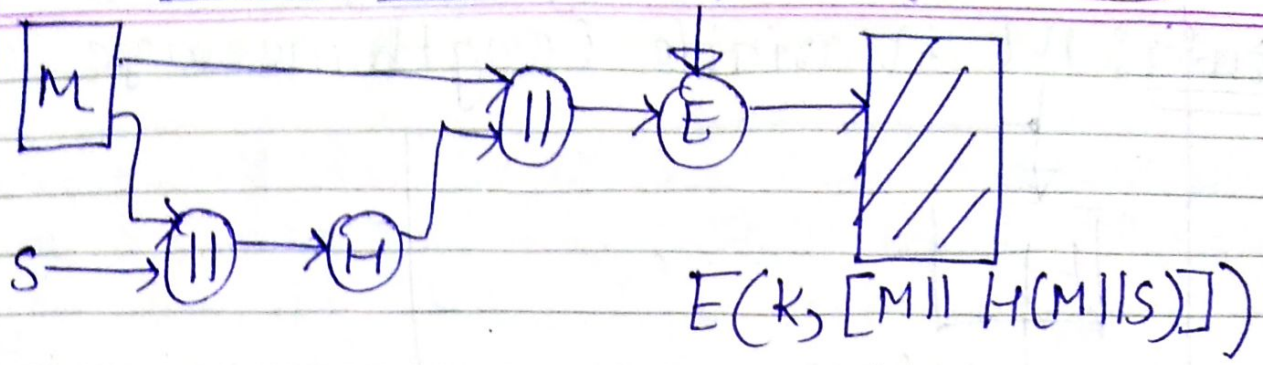
→ If Eve tries to Modify M, then Comparison of hash values will fail.

Note: This doesn't Include Confidentiality aspect

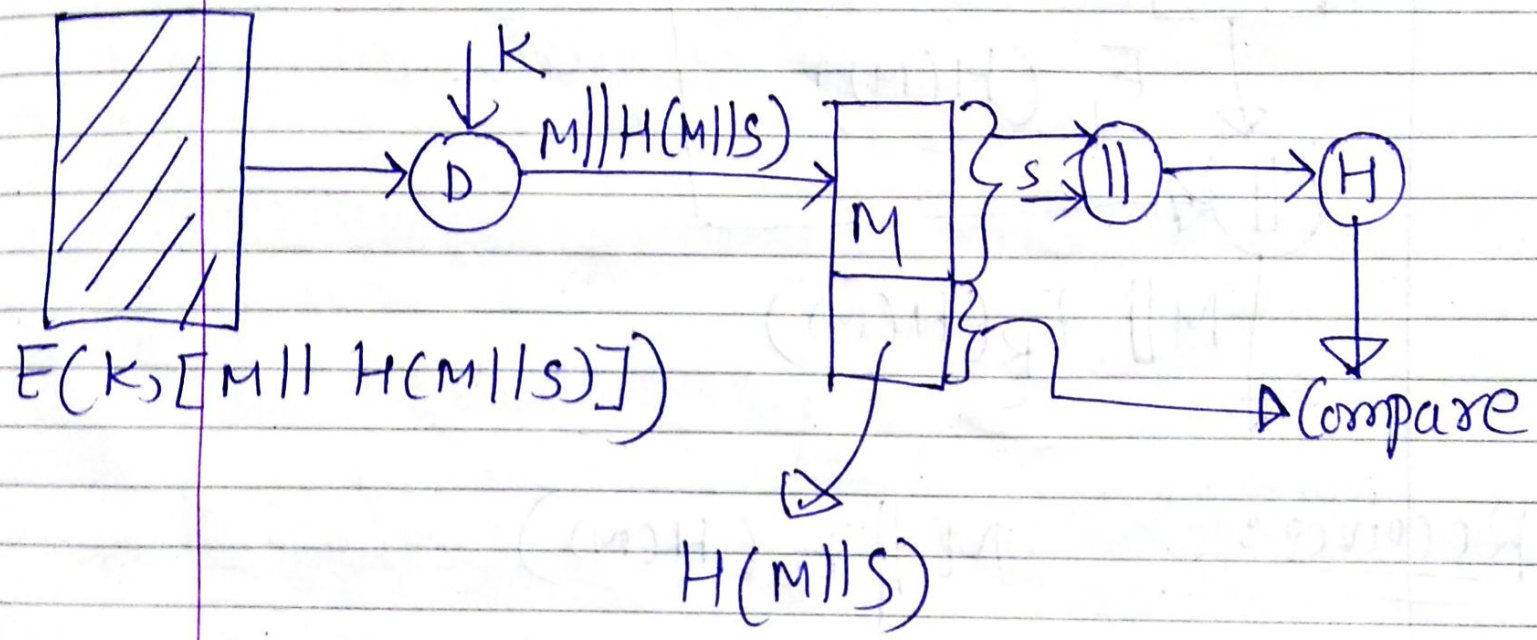
Sender Side

(d)

Eqn:



Receiver Side:



→ This achieves

Message Authentication

+

Confidentiality

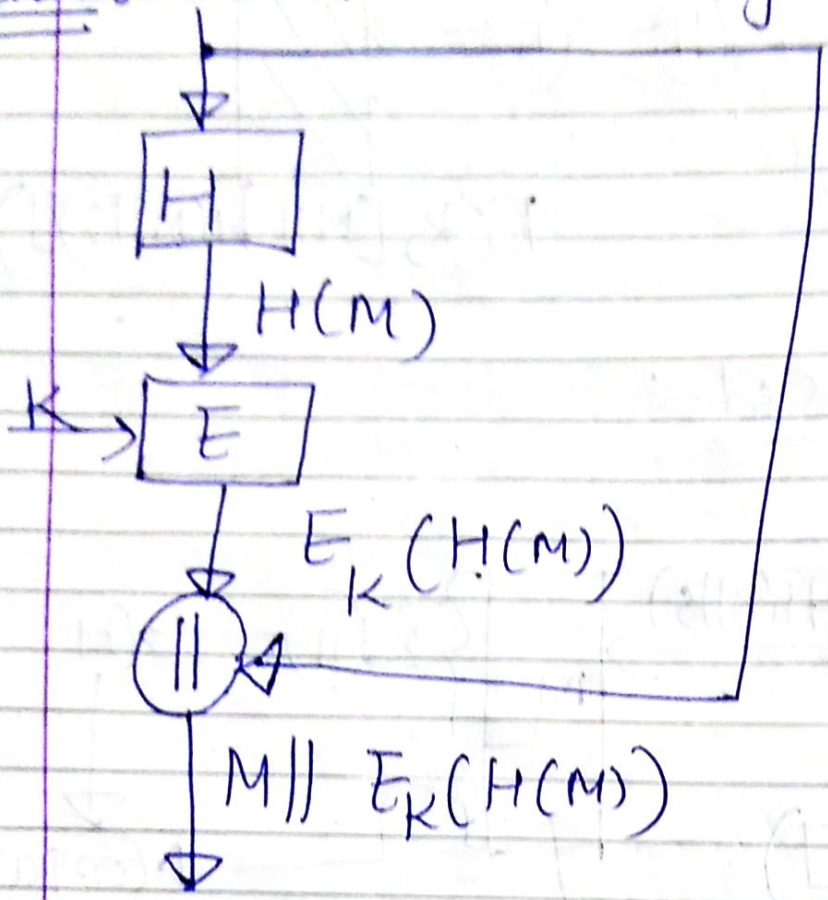
MAC : Message Authentication code

↓

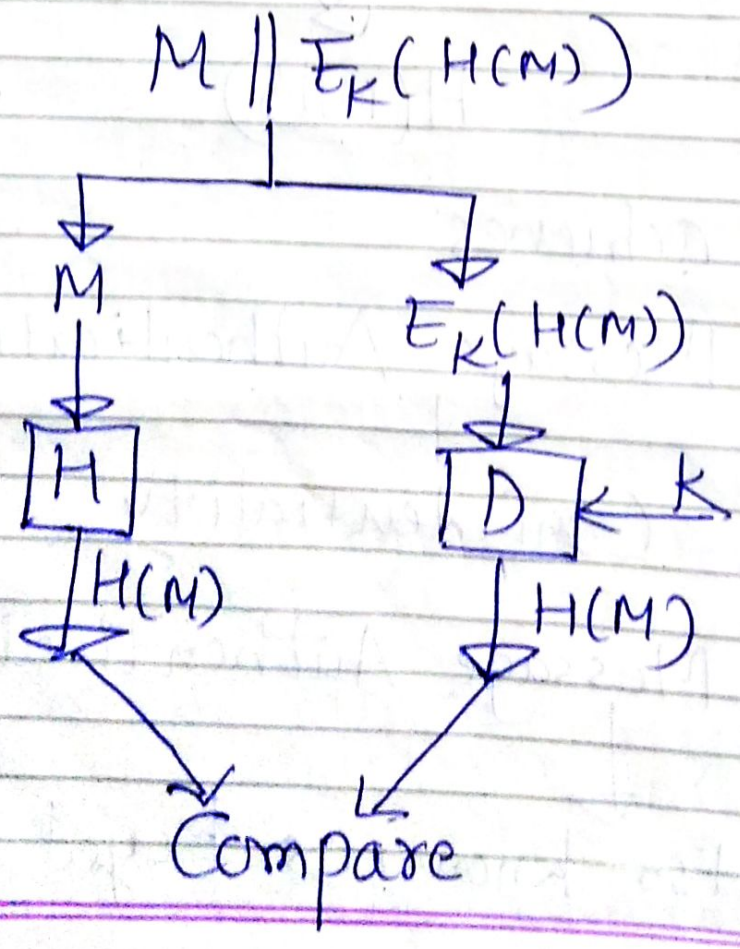
It is also known as keyed Hash Function

MAC: Hash + Symmetric Encryption

Sender: M : Variable length message



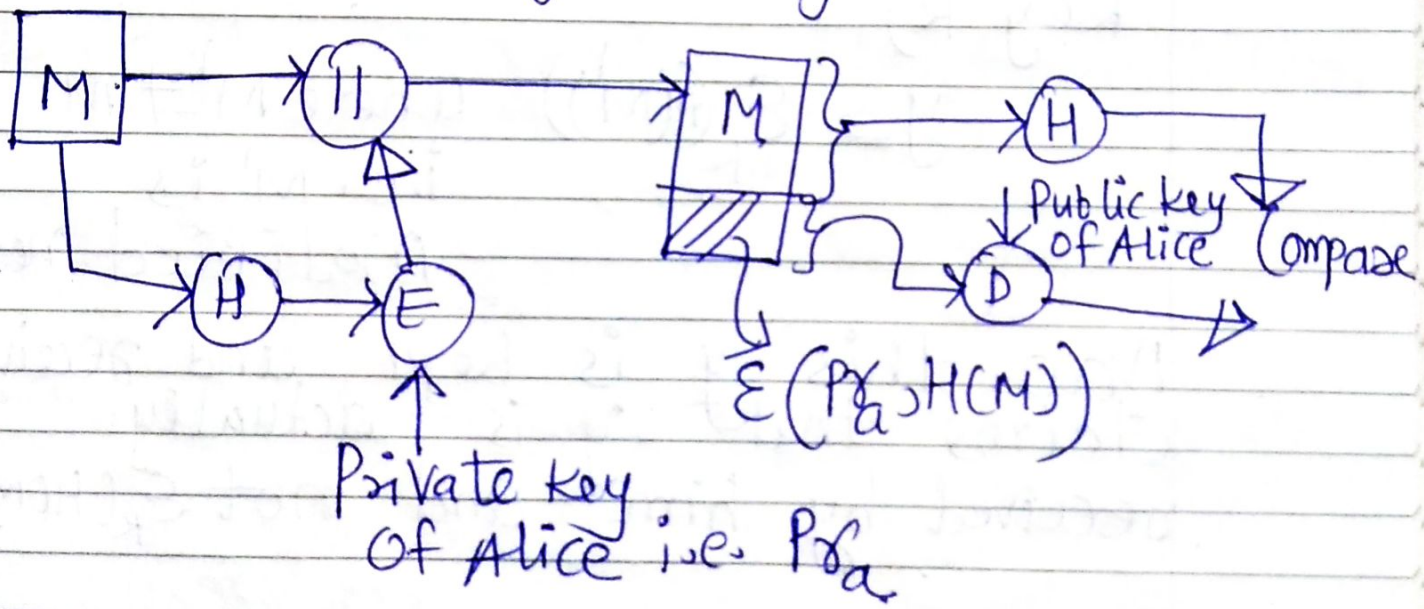
Receiver:



→ If Comparison is true i.e. returns true
⇒ Sender is Verified and Authenticated as key k is only with Alice

→ If Comparison is false i.e. returns false
⇒ Message M is ~~verified~~ modified by attacker

Topic 2 Application: Digital Signature



Goals: Provides
→ Authentication of Message
→ Digital signature (using Private key)
→ Non-Repudiation
→ Integrity

✡ How Non-Repudiation?

→ Receiver can prove that message is indeed sent by Sender because Sender has used private key for Encryption.

✡ This is not the case with MAC. In MAC, once the receiver receives data M, receiver can modify M to M' and encrypt H(M') using Symmetric Cipher E (with key K)

$$y = E_K(H(M')) \quad \text{where } M' \neq M$$

i.e. M' is Modified message

Now this y is kept and receiver claims that y is actually received by him and not $E_K(H(M))$

→ Sender can't prove that he has sent $E_K(H(M))$ and not $E_K(H(M'))$

→ This is Repudiation. It is possible in MAC

→ In Digital Signature, Sender uses private key to encrypt the data.

∴ Receiver can decrypt only using Sender's public key.

Receiver can't encrypt as he doesn't have Sender's private key.

∴ Receiver can't cheat i.e. repudiation can not happen.

Note: MAC provides

→ Integrity

→ Authentication

MAC doesn't provide Non-Repudiation i.e. Repudiation is possible.

Topic [3] Application:

Hash functions are commonly used to create a one-way password file.

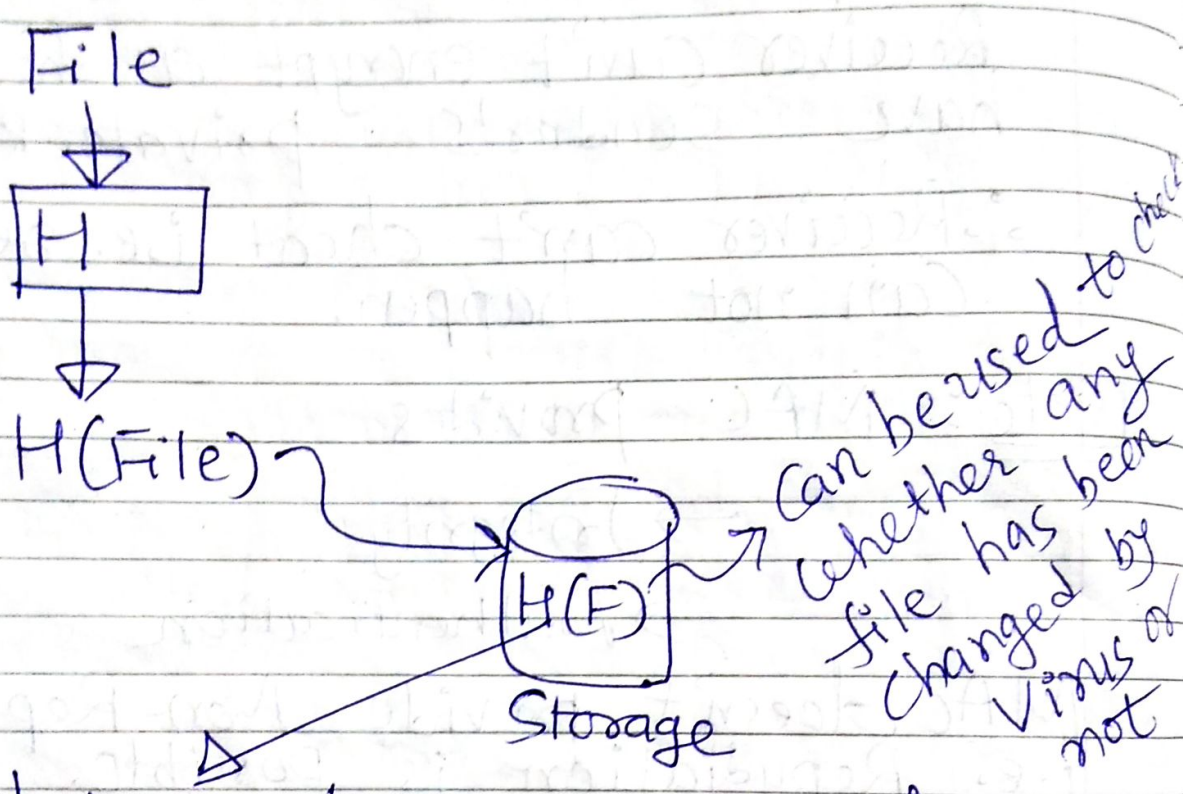
→ OS uses Hash of password in the password file.

→ When user enters password, its hash is compared against stored hash value.

→ This will Identify user is authorized or not.

☆ Hash functions can also be used

for intrusion detection and virus detection.



Later on H(F) can be used to see whether file has been modified or not.

★

→ Cryptographic Hash function can be used to construct

① PRF (Pseudo-Random Function)

② PRNG (Pseudo-Random Number Generator)

→ Common application of Hash based PRF is for the generation of Symmetric keys

✡ Terminology:

* **Image:**
$$y = H(x)$$
y is Image of x under H

* **Pre-Image:**
x is Pre-Image of y under H.

* **Oneway ness / Pre-Image Resistant:**

→ Given x → Should be Easy to find → $y = H(x)$

Given $y = H(x)$ → Should be difficult to find → x

This is Known as Oneway ness / Pre-Image Resistance property of Hash function H.

* **Second Pre-Image Resistance;**
or
Weak Collision Resistant:

→ For any given block x, it is Computationally infeasible to find

$y \neq x$ such that $H(y) = H(x)$

* **Strong Collision Resistant:** (Collision Resistant Property)

→ It is computationally infeasible to find any pair (x, y) such that

$$H(x) = H(y)$$

* **Pseudo Randomness:**

Output of H should look like random i.e. it should meet standard tests of Pseudo-Randomness

* **Variable Input size:**

H can be applied to a block of data of any size.

* **Fixed Output Size:**

H produces a fixed-length output.

* **Efficieney:** $H(x)$ is relatively easy to compute for any given x, making both software and hardware implementations practical.

* **Collision:** It occurs if $x \neq y$ but $H(x) = H(y)$

✱ Two Simple Hash Functions

One of the simplest Hash function is the bit by bit XOR of every block.

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

Where
C_i = i^{th} bit of Hash code
m = number of n-bit blocks
b_{ij} = i^{th} bit in j^{th} block
$\oplus$ = XOR

→ This Operation produces a Simple Parity for each bit position and is known as longitudinal redundancy check.

→ Each n-bit value is equally likely.

Pr. that a data error will result in an Unchanged Hash value

$= \frac{1}{2^n}$ (∵ only 1 out of 2^n possibility is such where hash remains unchanged even if data is changed i.e. erroneous)

$\frac{1}{2^{-n}}$

✱ Improvement: Perform a one-bit circular shift, or rotation, on the hash value after each block is processed.

✡ Procedure :-

1. Initially set the n-bit Hash value to 0.
2. Process each successive n-bit block of data as follows:
   - (a) Rotate the current Hash value to the left by one bit.
   - (b) XOR the block into the Hash value.

E.g. 8 bit blocks

Way 1

Block 1 : 0110 1111
Block 2 : 1010 1011
Block 3 : 0010 1000
Block 4 : 1011 1101

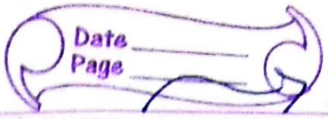
Way 1

XOR of all blocks : 0101 0001 ↪ Hash code

Way 2

Initially 0000 0000 is Hash code

① 0000 0000
↓ Rotate Left (1 bit)
0000 0000



XOR: 0000 0000 ⊕ 0110 1111

Hash = 0110 1111

(2) 0110 1111

↓ Rotate Left

1101 1110 → Block 2

XOR: 1101 1110 ⊕ 1010 1011

∴ Hash = 0111 0101

(3) 0111 0101

↓ Rotate Left

1110 1010 → Block 3

Hash = 1110 1010 ⊕ 0010 1000

= 1100 0010

(4) 1100 0010

↓ Rotate

1000 0101 → Block 4

Hash = 1000 0101 ⊕ 1011 1101

= 0011 1000

Way 2 has effect of randomizing the input more completely and overcoming any

regularities that appear in the input.

Issues: It is easy for Eve to create Message $M' \neq M$ such that $H(M') = H(M)$ (∵ procedure is simple)

Issue: Ex: Message + Hash code is encrypted

Given Message M consisting of a sequence of 64-bit blocks

$$X_1, X_2, \ldots, X_N$$

define $h = H(M)$ as the block by block XOR of all blocks and append the hash code as final block

Now

→ Encrypt the entire message + Hash code using CBC mode to produce

$$Y_1, Y_2, \ldots Y_N, Y_{N+1}$$

↑ Encrypted hash code

→ JUEN 85 points several ways in which Cipher text of this message can be manipulated in such a way that it is not detectable by the Hash code.

e.g. By defⁿ of CBC

$$y_1 = \varepsilon_K(IV \oplus x_1)$$

$$y_2 = \varepsilon_K(y_1 \oplus x_2)$$

$$\vdots$$

$$y_{N+1} = \varepsilon_K(y_N \oplus x_{N+1})$$

↳ Hashcode

and

$$x_1 = D_K(y_1) \oplus IV$$

$$x_2 = D_K(y_2) \oplus y_1$$

$$\vdots$$

$$x_{N+1} = D_K(y_{N+1}) \oplus y_N$$

But $x_{N+1} = x_1 \oplus x_2 \oplus \ldots \oplus x_N$

$$= [D_K(y_1) \oplus IV] \oplus [D_K(y_2) \oplus y_1] \oplus \ldots \oplus [D_K(y_N) \oplus y_{N-1}]$$

Terms can be XORed in any order

∴ It follows that the Hash code would not change if Ciphertext blocks are permuted

Table: Hash function Resistance properties required for various Data Integrity Applications

	①	②	③
① Hash + Digital Signature	yes	yes	yes
② MAC	yes	yes	yes
③ One way password files	yes	—	—
④ Intrusion detection + Virus detection	—	yes	—

☆ Here columns indicate

① = Pre Image Resistance (one way ness)
② = Second Pre Image Resistance
③ = Collision Resistance

Ex: length of hash code = n bits
length of data blocks = b bits
$b > n$

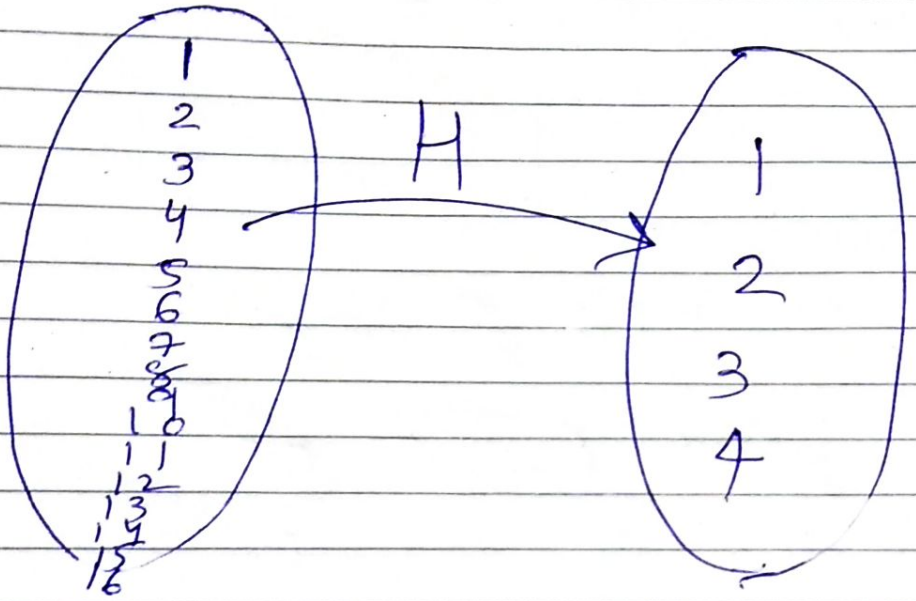
∴ Number of messages = 2^b
and
Number of Hashcodes = 2^n

→ On average, each hash value corresponds to 2^{b-n} pre-images

↓ How?

Say $b=4$

$n=2$



Domain has $2^b = 2^4 = 16$ messages

Codomain has $2^n = 2^2 = 4$ Hash codes

∴ We can say on an average, each has code corresponds to

$2^{b-n} = 2^{4-2} = 2^2 = 4$ Preimages

i.e. 4 messages.

~~∴ Many to one behaviour~~