Hash Functions & Applications

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY

Outline

1. Secure Hash Algorithm
2. Message Authentication Codes
3. Applications

Hash Function

- A hash function $H: \{0,1\}^* \rightarrow \{0,1\}^n$, where n is a fixed, defined as h = H(x) satisfy the properties:

x, $H(x)$

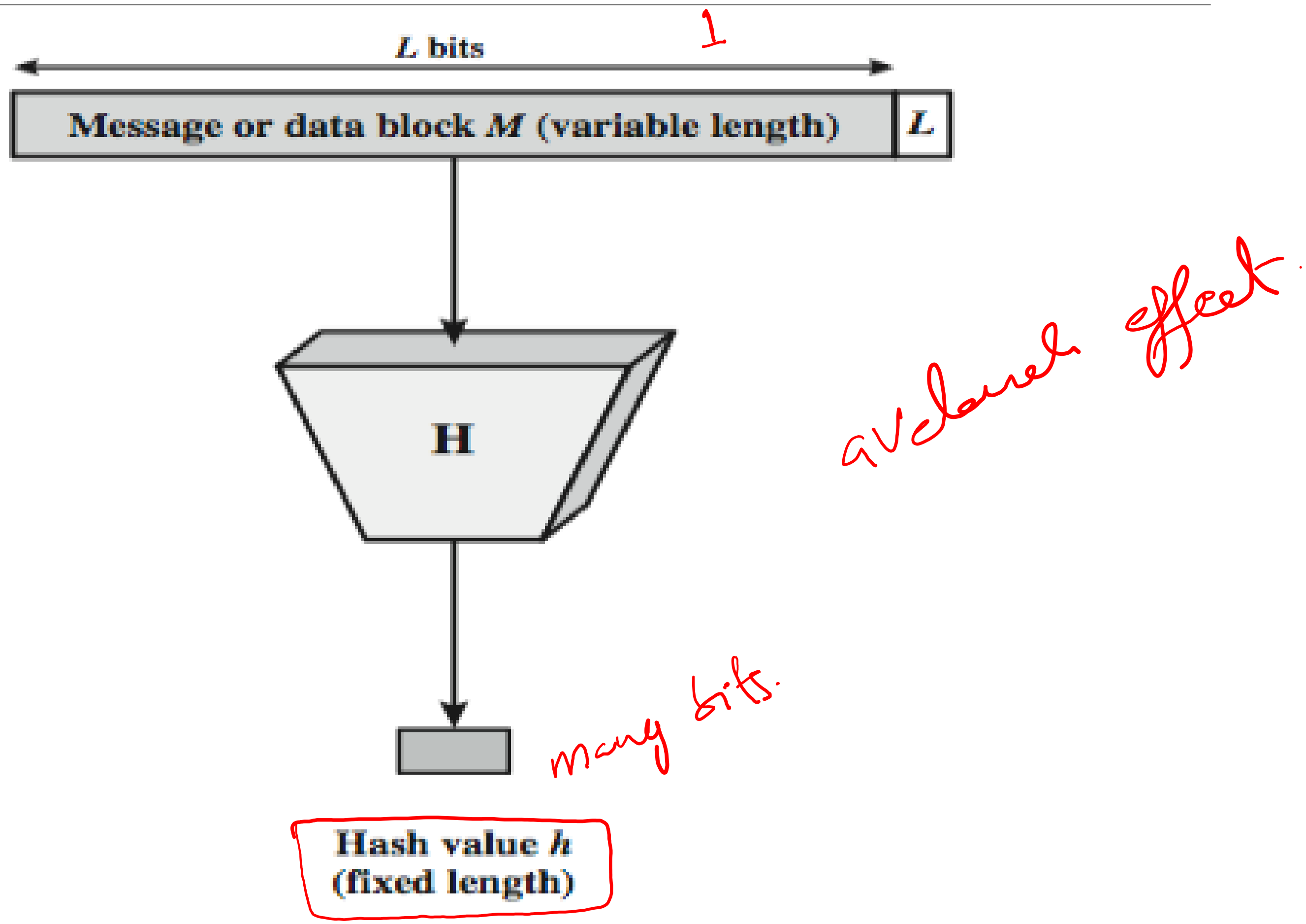
y, $H(x)=H(y)$

  - **Pre-image resistance:** Given h, computing *x* is hard.
  - **Second pre-image resistance:** Given an input *x*, difficult to find *y (≠x)* such that *H(x) = H(y)*.

$x \neq y$.

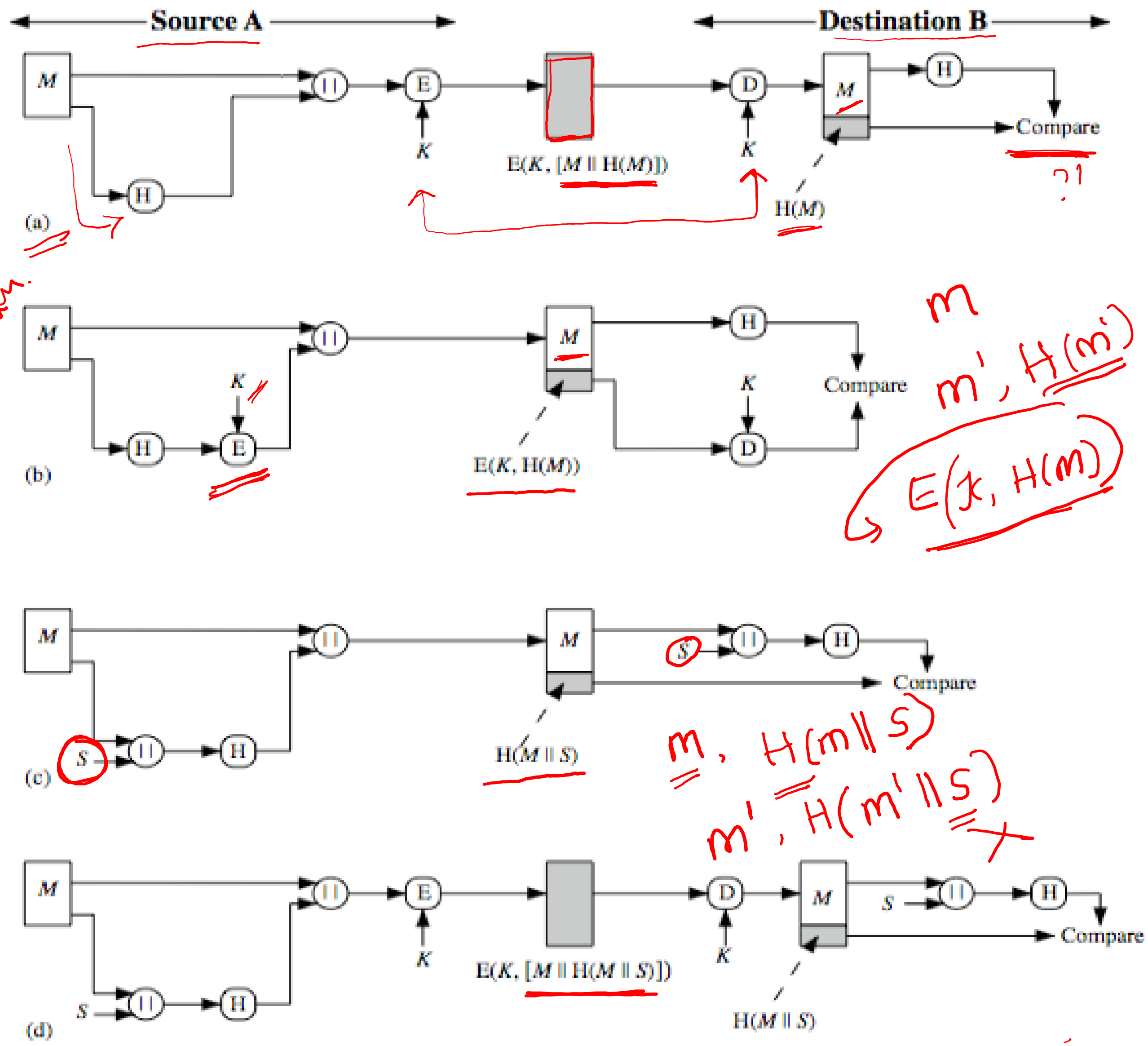
  - **Collision resistance:** Difficult to find a pair *(x, y)* of two different messages such that *H(x) = H(y)*.
  - Note that collisions may be found by a birthday attack

Cryptographic Hash Function



Hash Functions & Message Authentication

1) Confidentiality
2) Integrity
3). Message Authentication



Digital Signatures
(using public-key technique)

$\langle m, S \rangle$

$m' = S^e \bmod n$

$\stackrel{?}{=} m$

RSA : $n = pq$, $\phi(n)$

e, d

$c = m^e \bmod n$

$m = c^d \bmod n$

A

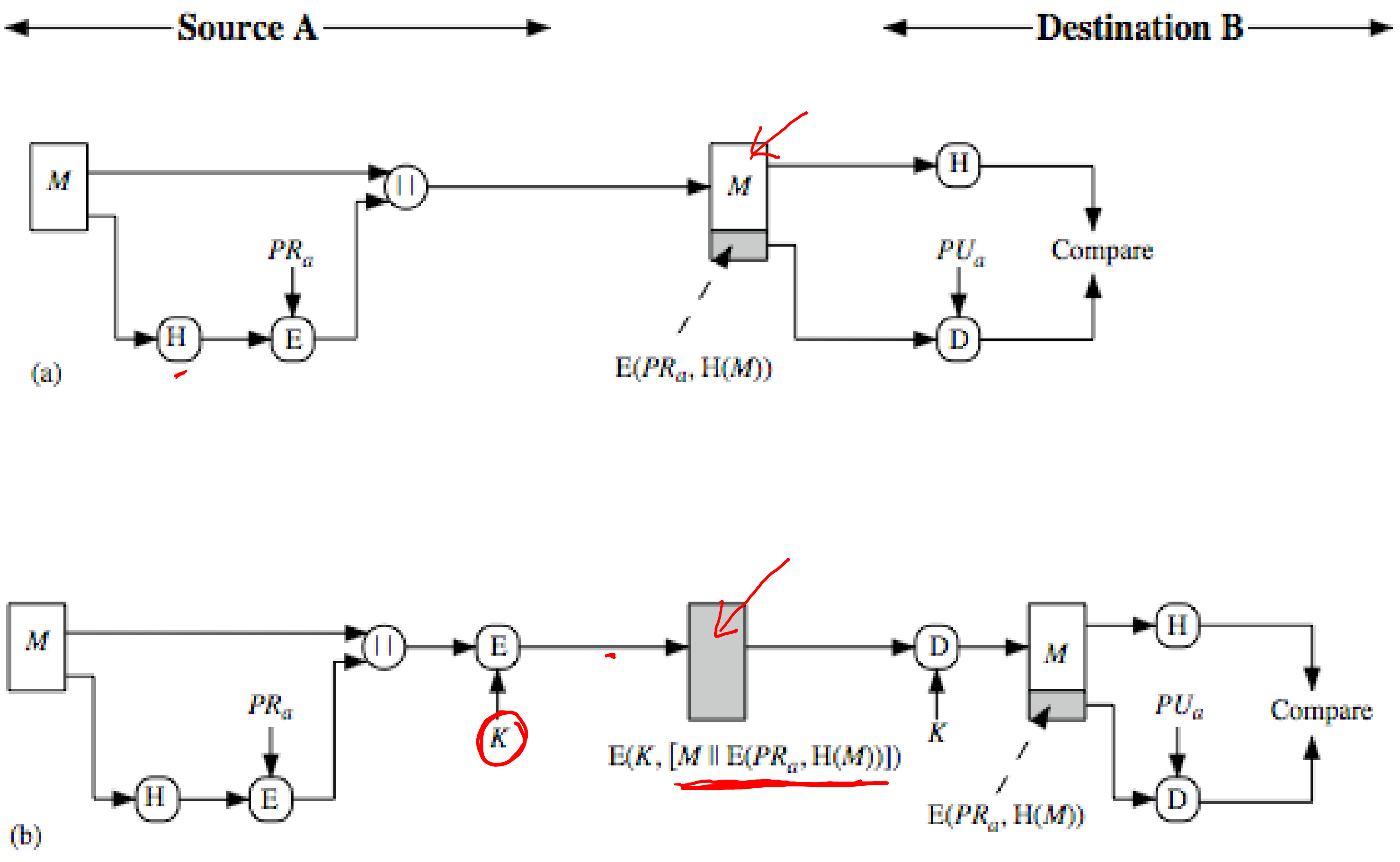
B

$\langle (n, e), (n, d) \rangle$

1) $c = m^e \bmod n \xrightarrow{\langle c \rangle} m = c^d \bmod n$

2) $m = S^e \bmod n \xleftarrow{S} S = m^d \bmod n$, $\boxed{m = H(m)}$

Hash Functions & Digital Signatures



Hash Function Uses Cases

C {UN, pwd} S

{UN, pwd}

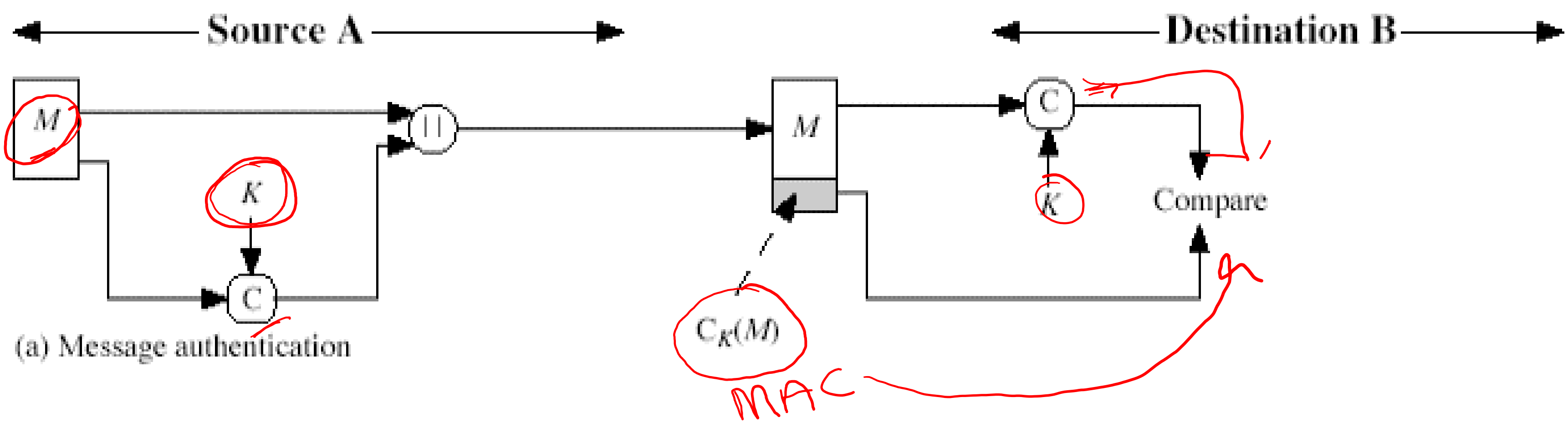
H(pwd)

- To create a one-way password file
  - Store hash of password not actual password
- For intrusion detection and virus detection
  - keep & check hash of files on system
- Pseudorandom function (PRF) or pseudorandom number generator (PRNG)

Message Authentication Code (MAC)

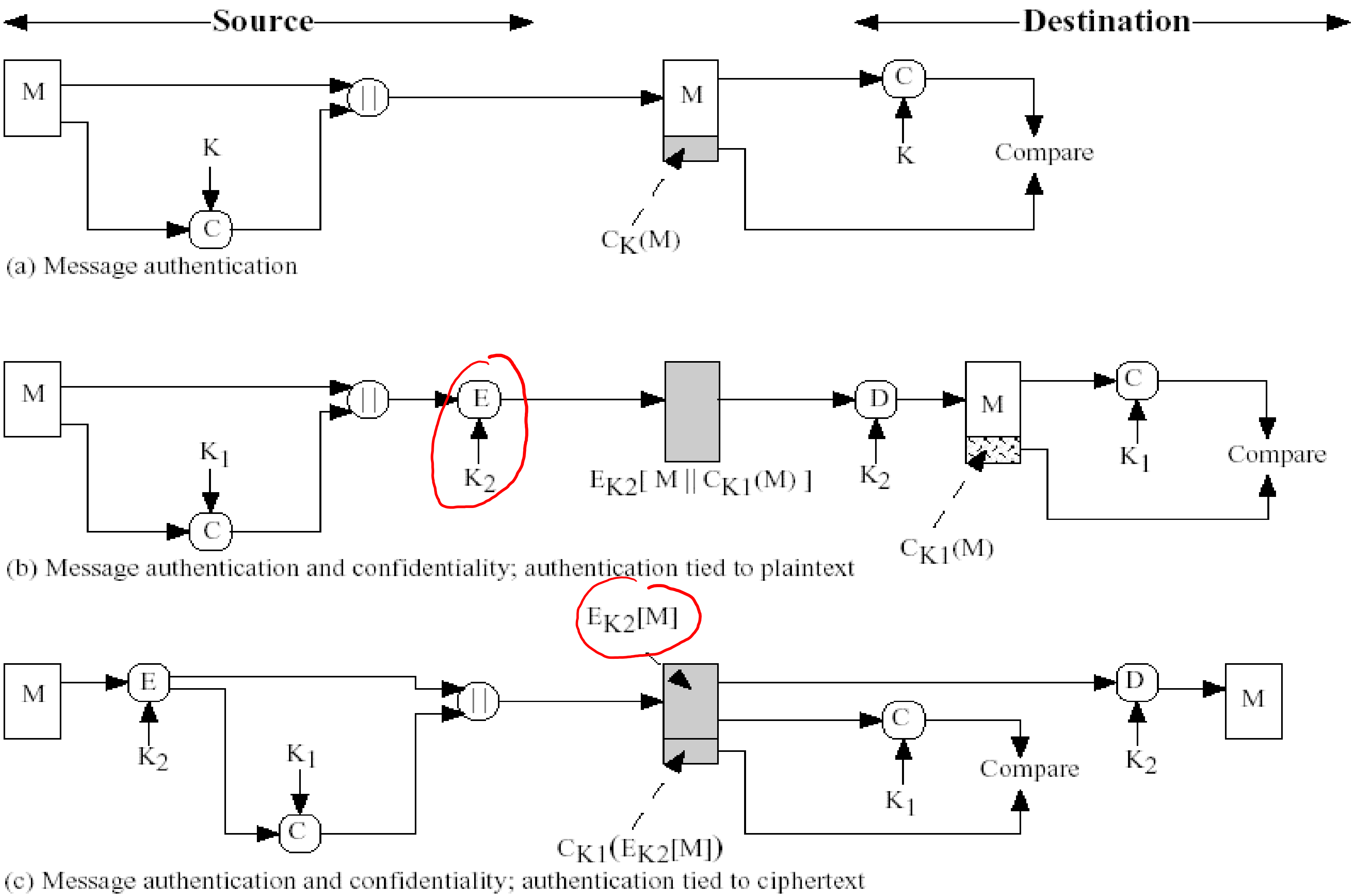
Message Authentication Code (MAC)

- Generated by an algorithm that creates a small fixed-sized block
  - depending on both message and some key
  - like encryption though need not be reversible
- Appended to message as a **signature**



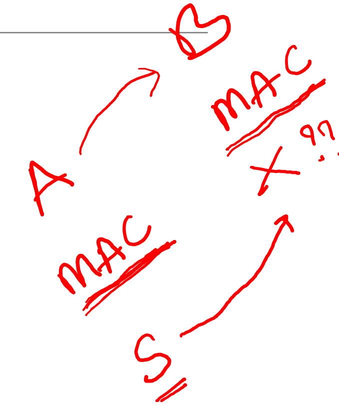
(a) Message authentication

Basic Uses of MAC



MAC Properties

$Auth = H(K \| Q \| P \| x)$



- MAC is a cryptographic checksum

  MAC = C_K(M)

  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- It is a many-to-one function
  - potentially many messages have same MAC
  - But, finding those needs to be very difficult

  - **Note that a MAC is not a digital signature** ??

HMAC

$H(K \| m)$

- Specified as Internet standard RFC2104
- Uses hash function on the message:

$$\mathbf{HMAC_K(M) = Hash[\,(K^+\ XOR\ opad)\ ||}$$
$$\mathbf{Hash[\,(K^+\ XOR\ ipad)\ \ ||\ M)\,]\,]}$$

  - where K^+ is the key padded out to size
  - opad, ipad are specified padding constants

- Any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

IOT → High
→ medium
→ low security

HMAC Security

- Proved security of HMAC relates to that of the underlying hash algorithm
- Attacking HMAC requires either:
  - brute force attack on key used   K
  - birthday attack (but since keyed would need to observe a very large number of messages)
- Choose hash functions based on speed verses security constraints

SECURE HASH ALGORITHM

Hash Function Requirements

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	H(*x*) is relatively easy to compute for any given *x*, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value *h*, it is computationally infeasible to find *y* such that H(*y*) = *h*.
Second preimage resistant (weak collision resistant)	For any given block *x*, it is computationally infeasible to find *y* ≠ *x* with H(*y*) = H(*x*).
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (*x*, *y*) such that H(*x*) = H(*y*).
Pseudorandomness	Output of H meets standard tests for pseudorandomness

Block Ciphers as Hash Functions

- Can use block ciphers as hash functions
  - Using H_0=0 and zero-pad of final block
  - Compute: $H_i = E_{M_i} [H_{i-1}]$
  - Use final block as the hash value
- Resulting hash is too small (64-bit)
  - to direct birthday attack
  - to "meet-in-the-middle" attack

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993, and revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - Standard is FIPS 180-1 1995, also Internet RFC3174
  - Algorithm is SHA, the standard is SHS
  - Produces 160-bit hash output.
- In 2005, results on security of SHA-1 have raised concerns on its use in future applications

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
  - adds 3 additional versions of SHA: SHA-256, SHA-384, SHA-512
- Designed for compatibility with increased security provided by the AES cipher
- Structure & detail is similar to SHA-1

SHA Versions

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

SHA-3

SHA-1 not yet "broken"

- but similar to broken MD5 & SHA-0
- so considered insecure

SHA-2 (esp. SHA-512) seems secure

- shares same structure and mathematical operations as predecessors so have concern.

In 2015, NIST announced that SHA-3 had become a hashing standard

THANK YOU