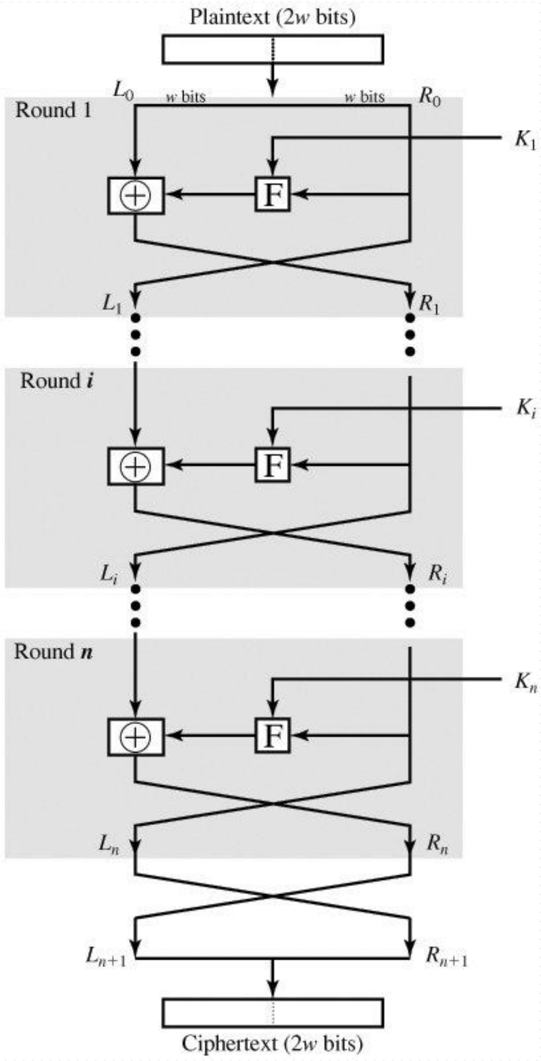
IT 422 Network Security
Authentication and Hashing

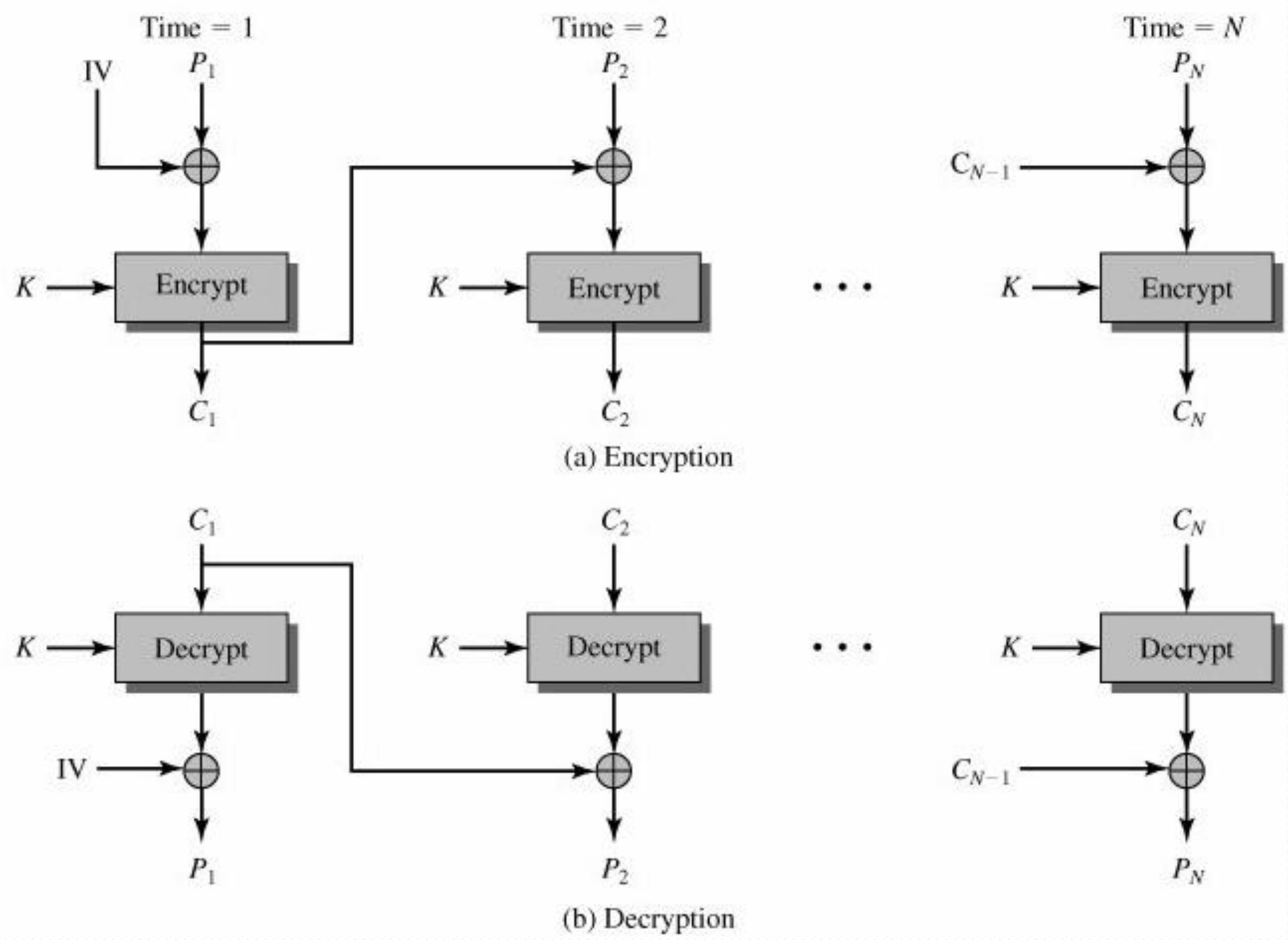
Yasser F. O. Mohammad

REMINDER 1:Fiestel Network

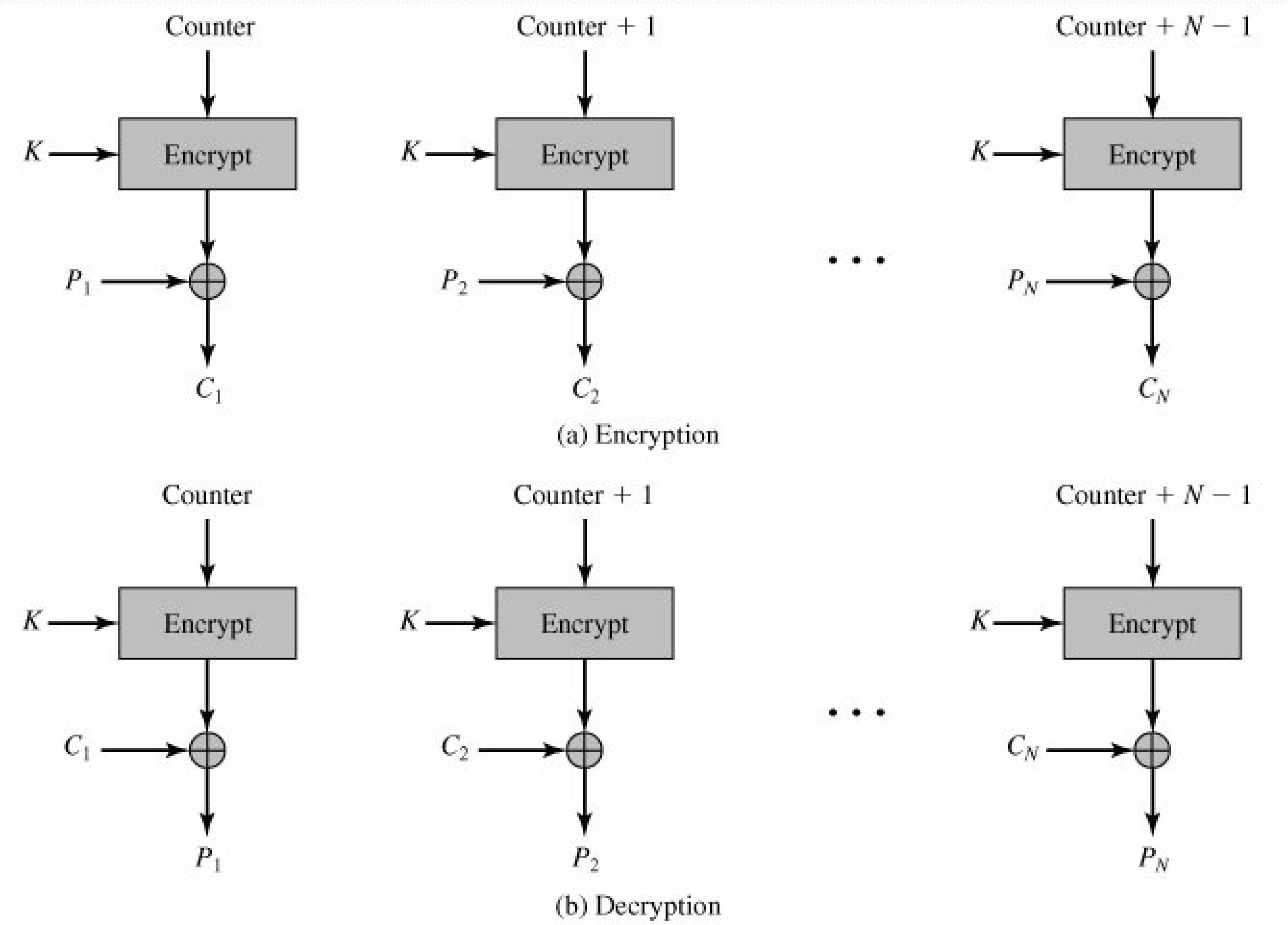
- Each round consists of:
  - Substitution on left half of text
  - Permutation of the two halves
- The substitution is controlled by the key of every round
- Factors of Security:
  - Block size
  - Key size
  - N. rounds
  - Subkey generation
  - Round Function
- Decryption = Encryption with reversed subkey order



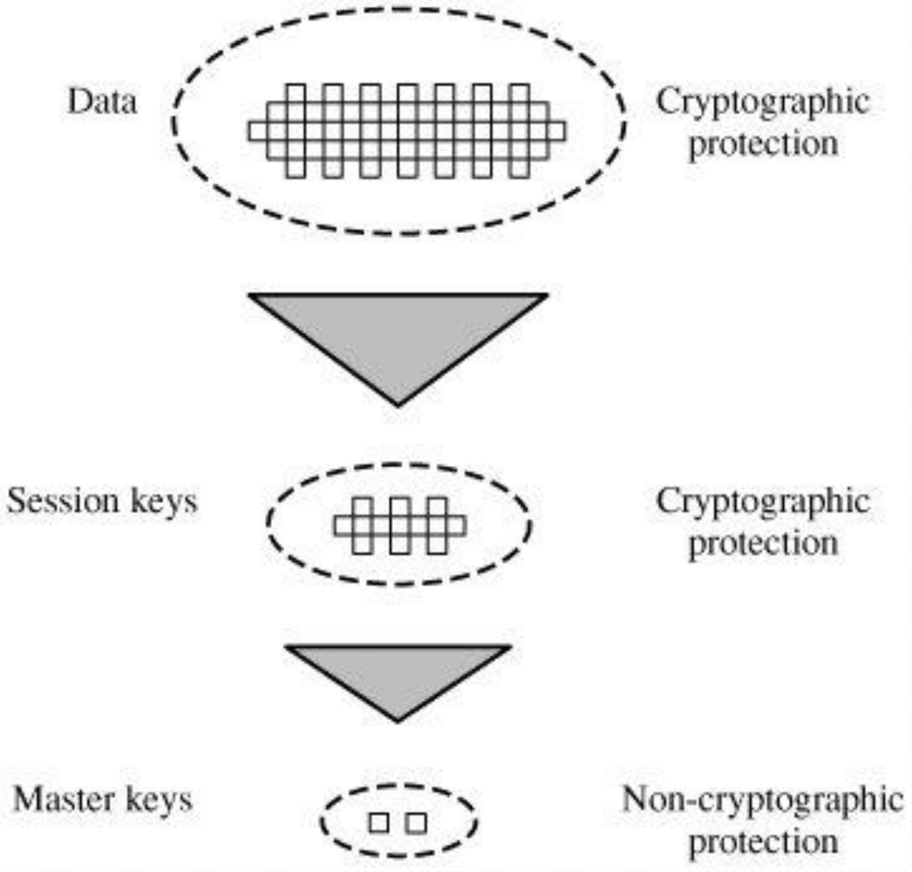
REMINDER 2: CBC (Cipher Block Chaining Mode)



REMINDER 3: CTR (Counter Mode)



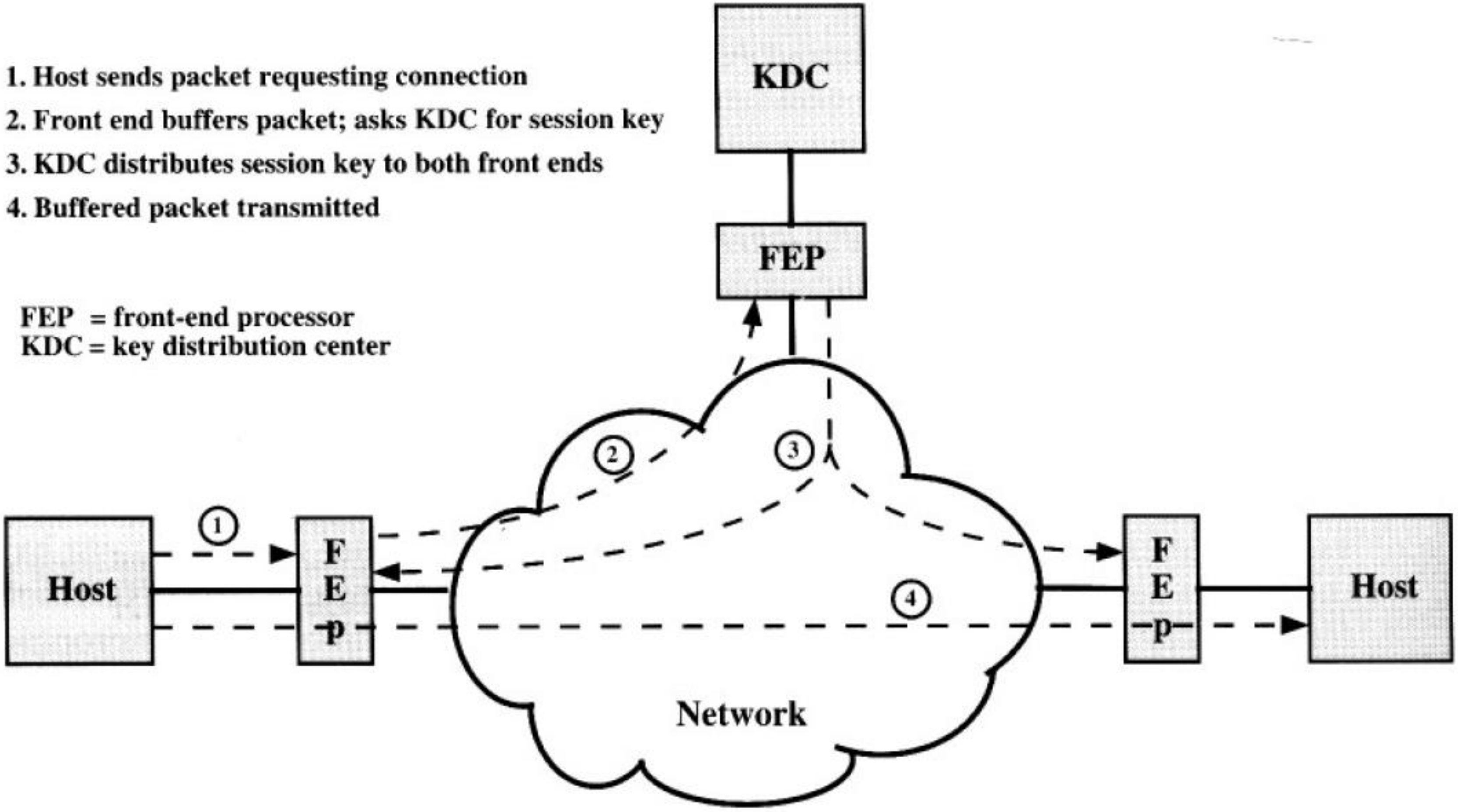
REMINDER 4: Key Hierarchy



REMINDER 5: Key Distribution Center

1. Host sends packet requesting connection
2. Front end buffers packet; asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted

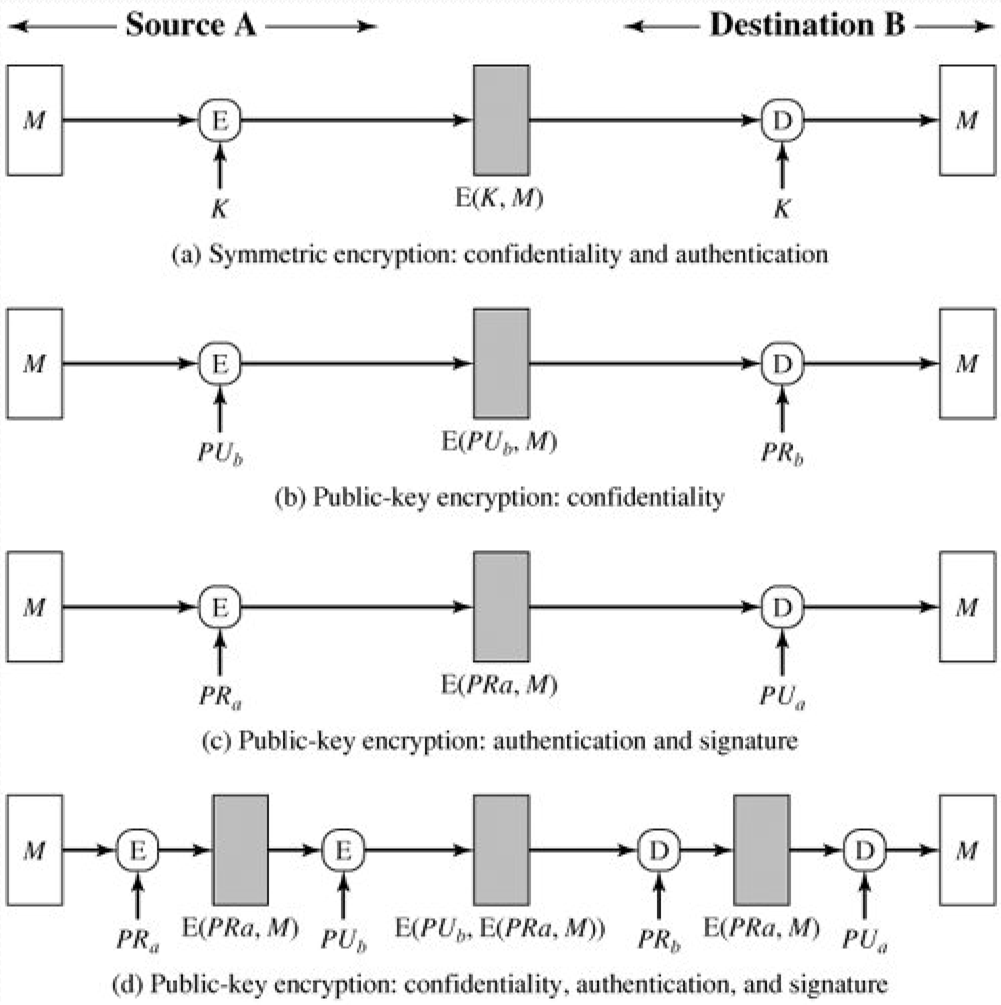
FEP = front-end processor
KDC = key distribution center



Rule of Authentication

- Encryption protects against passive attacks
- Authentication protects against active attacks
- Authentication uses encryption

Different Uses of Encryption



Authentication Without Confidentiality

- Why?
  - Broadcasting
  - I am too busy to encrypt
  - Authentication of programs (no need to decrypt every time)

- How?
  - Message Authentication Code (MAC)
  - One Way Hash function

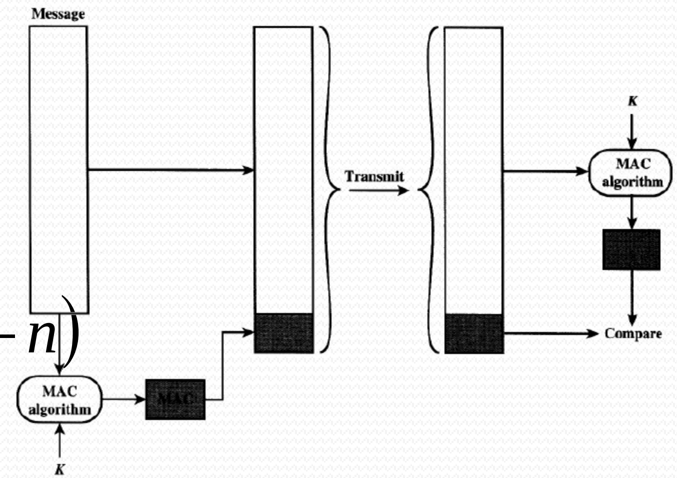
MAC

$$A \rightarrow B : M + MAC$$

$$MAC \equiv Substring\left(E\left(k_{A-B}, M\right), n\right)$$

$$B : M_1 = Substring\left(M_{received}, strlen\left(M_{received}\right) - n\right)$$

$$Test\left(MAC == Substring\left(E\left(k_{A-B}, M_1\right), n\right)\right)$$



- B knows that the message was not altered. Why?
- B knows that the message is from A. Why?
- If the message contains a sequence number, B knows that the order was not altered

- Usually DES is used and *n* equals 16 or 32

Authentication using shared key

$$A \rightarrow B : M_1 = E\left(k_{A-B}, 'hello' + M\right)$$

$$B : \text{if } Substring\left(D\left(k_{A-B}, M_{1-received}\right), 5\right) == 'hello' \text{ then}$$

$$M_{1-received} == M_1$$

$$Sender\left(M_1\right) == A$$

$$\text{if } E \neq A \text{ then } A \text{ cannot read } M$$

How can we use this exchange to agree on a new key? Why would we want to do that?

One Way Hash Functions

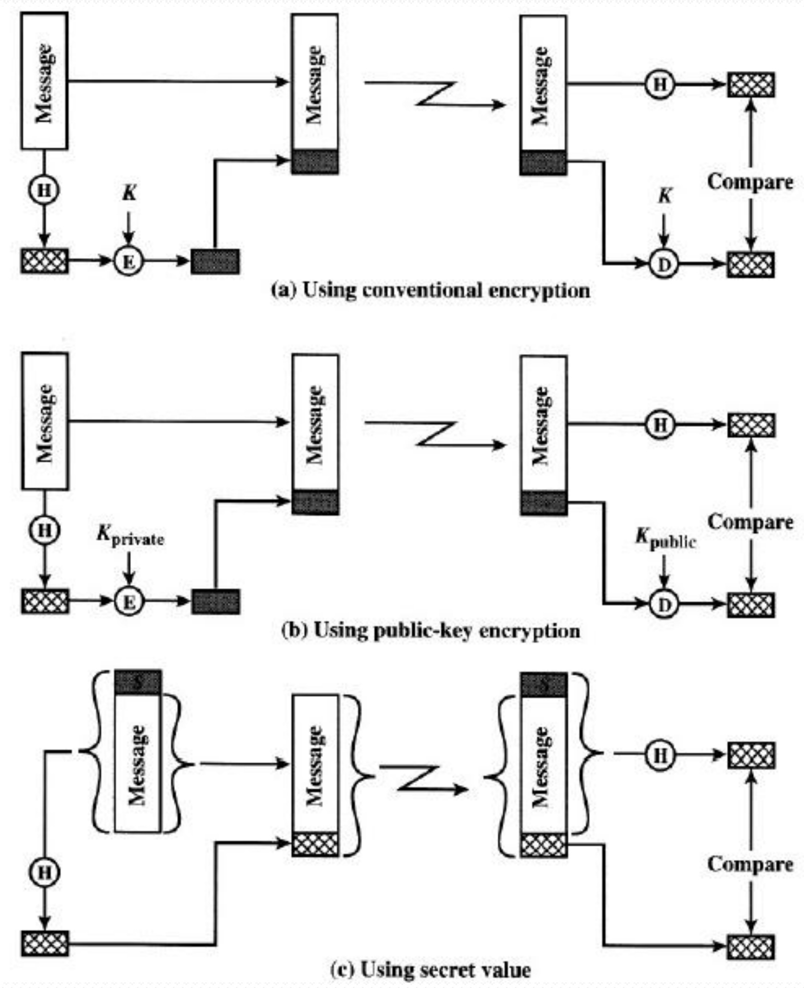
a) Only we know *k*
- *Most conventional*

b) Uses Public Keys only
- *Offers Nonrepudiation*
- *No key distribution*

c) Only we know the secret
- *No encryption*
- *Used in HMAC adopted by IP security*

- *Why No Encryption?*

1. *Encryption is slow*
2. *Encryption is expensive*
3. *Encryption is optimized for large*
4. *Patents & export control*



Hash function Requirements

- Arbitrary Data Size
- Fixed length output
- Easy to compute
- One Way: Given the hash we should not recover the message
- Weak collision resistance: given x we cannot find y so that H(x)=H(y)
- Strong collision resistance: we cannot find any (x,y) so that H(x)=H(y)

General Hashing algorithm

- n bits hash
  - Treat the message as a sequence of n bit blocks
  - Process each block in some order
  - Output the final n bits

Simplest hash function (XOR)

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

where

C_i	= ith bit of the hash code, $1 \leq i \leq n$
m	= number of n-bit blocks in the input
b_{ij}	= ith bit in jth block
$\oplus$	= XOR operation

- How to break this?

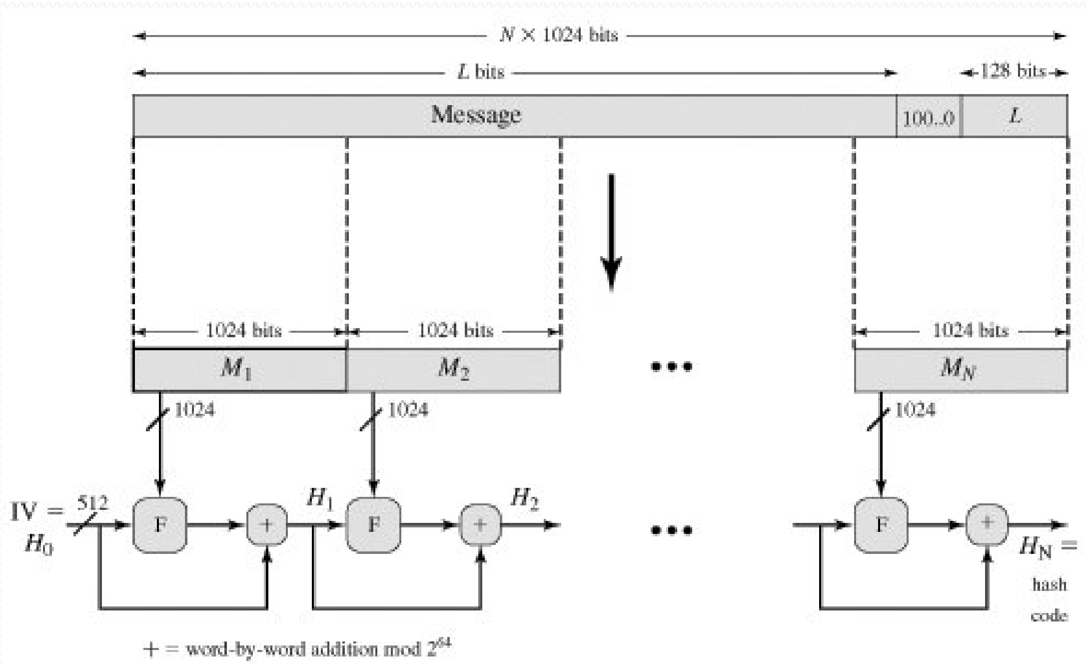
First Improvement (RXOR)

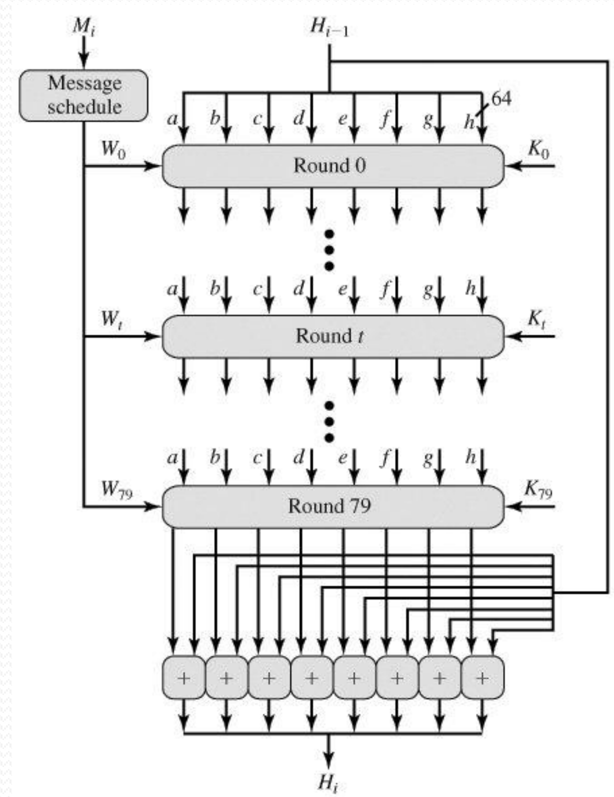
1. Initially set the *n*-bit hash value to zero.
2. Process each successive *n*-bit block of data as follows:
    a. Rotate the current hash value to the left by one bit.
    b. XOR the block into the hash value.

- How to break this?

Modern Hash Functions

- SHA-1 (self read the algorithm)
  - Maximum input is 2^{64}
  - Digest size = 160 bits
  - Block size is 512 or 1024 bits





Other Hash functions

- MD5
  - By Ron Rivest
  - 128 bit digest
  - 512 bit blocks
  - Arbitrary input length

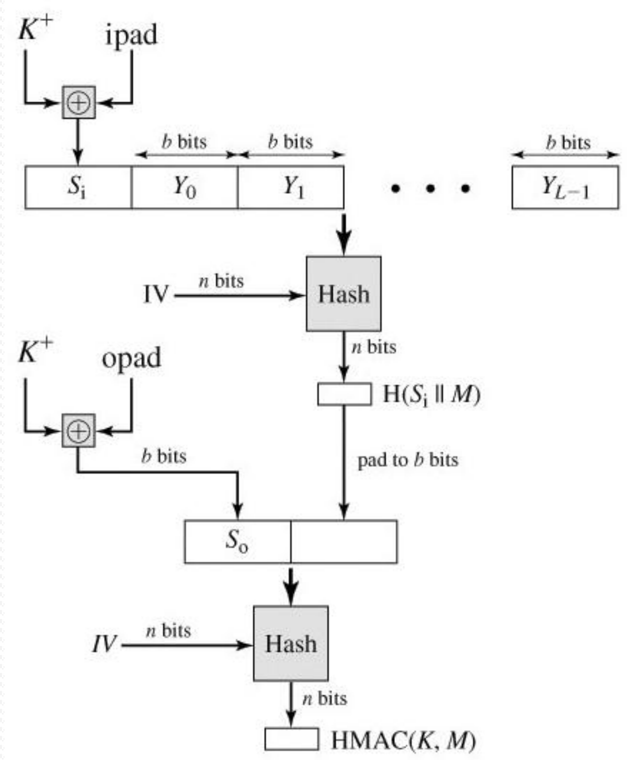
- RIPMOD 160
  - 160 bit digest
  - 512 bit block

HMAC

$\text{HMAC}(K,M) = H[(K^+ \oplus \text{opad})||H[(K^+ \oplus \text{ipad})||M]]$

In words,

1. Append zeros to the left end of K to create a b-bit string K^+(e.g., if K is of length 160 bits and b = 512 then K will be appended with 44 zero bytes 0 x 00).
2. XOR (bitwise exclusive-OR) K^+ with ipad to produce the b-bit block S_i.
3. Append M to S_i.
4. Apply H to the stream generated in step 3.
5. XOR K^+ with opad to produce the b-bit block S_o
6. Append the hash result from step 4 to S_o
7. Apply H to the stream generated in step 6 and output the result.



- A hash function that uses a key but does not require slow encryption.