Unit 2: Cryptography and Cryptographic Algorithms

Lecturer: Binod Chandra Shrestha

Encryption and Decryption

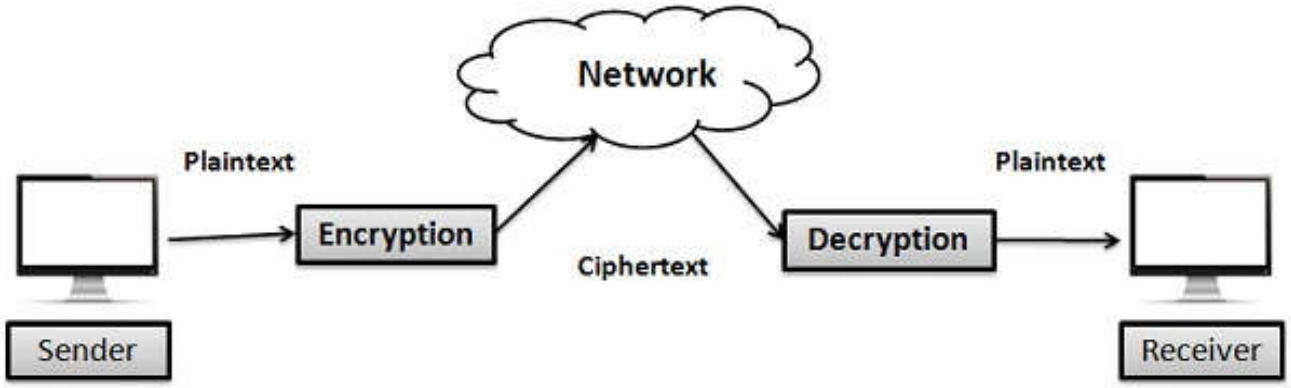


Fig. Encryption and Decryption

- The method of disguising plaintext in such a way as to hide its substance is called **encryption**. Encrypting plaintext gives the unreadable form i.e. ciphertext.
- The process of reverting ciphertext to its plaintext is called **decryption**.

Cryptography

- Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables us to store sensitive information or transmit it across insecure network e. g. internet so that it cannot be read by anyone except the intended recipient.
- There are two types of cryptography:
  - Symmetric key Cryptography: same key is used in encryption and decryption.
  - Asymmetric key Cryptography: different keys are used in encryption and decryption.

Classical Cryptography

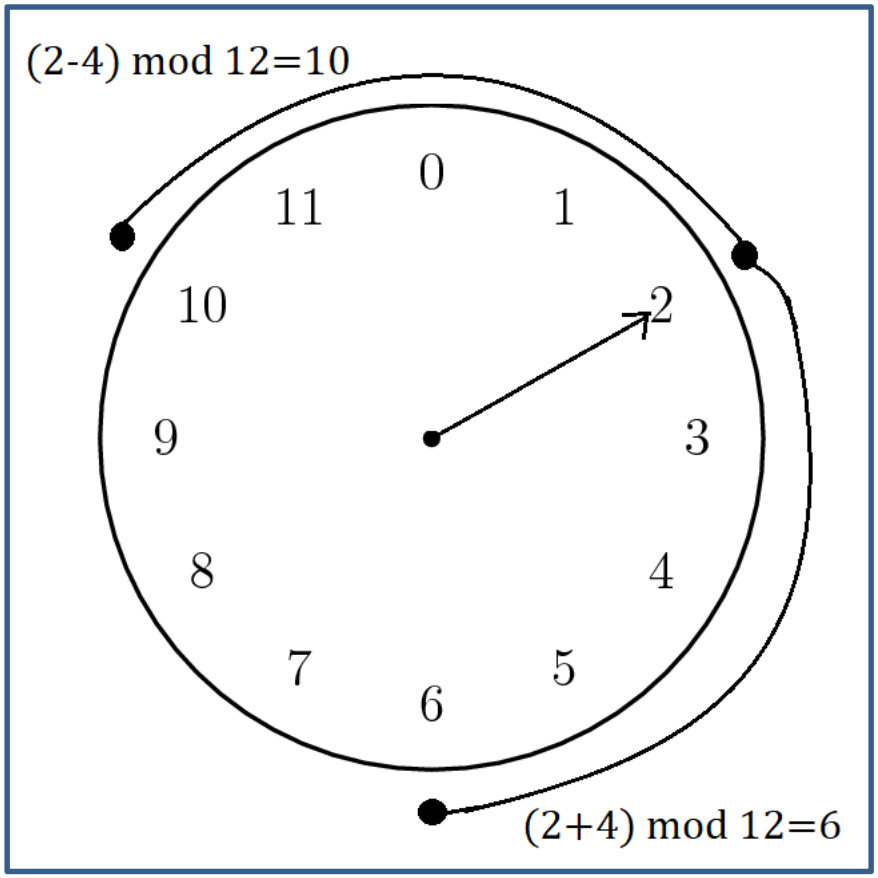
- Classical ciphers can be practically computed and solved by hand.
- They are easily breakable with modern technology.
- Some examples: Caesor Cipher, Substitution Cipher, Transposition Cipher, Simple XOR Cipher, Playfair Cipher etc.

Modular Arithmetic

- Several important cryptosystems make use of modular arithmetic. This is when the answer to a calculation is always in the range *0 – n* where n is the **modulus**.
- To calculate the value of *b mod n*, you take away as many multiples of n as possible until you are left with an answer between *0* and *n*.

Clock Method for Modular Arithmetic

- Label 0 to (n-1) for mod n on a clock.
- For addition count clockwise the value of 2nd number from the value of 1st number.
- For subtraction count anti-clockwise the value of 2nd number from the value of 1st number.
- e.g. from figure we can see:
  - ❖ (2+4) mod 12=6
  - ❖ (2-4) mod 12=10



Reminder for Modular Arithmetic

- If “b”is positive integer and its modulo n can be calculated as Reminder while dividing it by n. i.e. b mod n=r, where r is reminder while dividing “b” by “n”.
- e.g.
  - ❖14 mod 12= 2 since 14/12 we get reminder 2.
  - ❖29 mod 26= 3 since 29/26 we get reminder 2.

If b is a negative number then you add as many multiples of m as necessary to get an answer in the range 0 – n.

Examples

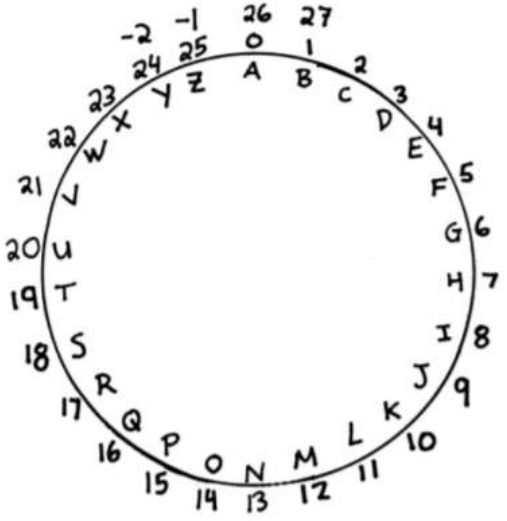
17 mod 5 = 2	7 mod 11 = 7
20 mod 3 = 2	11 mod 11 = 0
-3 mod 11 = 8	-1 mod 11 = 10
25 mod 5 = 0	-11 mod 11 = 0

Caesor Cipher

- It is developed by Julius Caesor.
- It is applicable only for English alphabets.
- It is a mono-alphabetic cipher wherein each letter of the plaintext is substituted by another letter to form the ciphertext. It is a simplest form of substitution cipher scheme.
- This cryptosystem is generally referred to as the **Shift Cipher**. The concept is to replace each alphabet by another alphabet which is 'shifted' by some fixed number between 0 and 25.
- For this type of scheme, both sender and receiver agree on a 'secret shift number' for shifting the alphabet. This number which is between 0 and 25 becomes the key of encryption.
- The name 'Caesar Cipher' is occasionally used to describe the Shift Cipher when the 'shift of three' is used.
- If m=plaintext, k=key and c=ciphertext, then
  - Encryption: $c=E_k(m)= (m+k) \bmod 26$; $0 \le m \le 25$
  - Decryption: $m=D_k(c)= (c-k) \bmod 26$; $0 \le c \le 25$

Modular Arithmetic and Caesor Ciphers

- Since there are 26 letters in the English alphabet, let's relate the letters a-z by numbers 0-25 as shown by the diagram below.



- Notice going from "A" to "D" was a shift of 3 letters over. Thus we can encrypt the word "ZEAL" by relating "Z" with 25 on the wheel, adding 3 to get 2, and then we turn this back into a letter, which gives us "C". Similarly "E" → 4 → 7 → H… So we can get ciphertext "CHDO".

Caesor Cipher Cont…

- e.g. Plaintext=ANT, Key=3 then Ciphertext=?

  Now, m1=A, m2=N and m3=T

  k=3

  We have, $c=E_k(m)= (m+k) \bmod 26; 0 \leq m \leq 25$

  c1=(m1+k) mod 26=(0+3) mod 26=3=D

  c2= =(m2+k) mod 26=(13+3) mod 26=16=Q

  c3=(m3+k) mod 26=(19+3) mod 26=22=W

  Hence, ciphertext="DQW"

Simple Substitution Cipher

- Simple substitution cipher is the most commonly used cipher and includes an algorithm of substituting every plain text character for every cipher text character.
- In this process, alphabets are jumbled in comparison with Caesar cipher algorithm.
- There are two table one is for plaintext and another is for ciphertext.
- e.g.

Plaintext	Ciphertext
A	W
B	L
C	M

- If plaintext="ABCA" then ciphertext="WLMW"

Transposition Cipher

- A method of encryption by which the positions held by units of plaintext are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext.
- The simplest such cipher is the “rail fence” technique in which the plaintext is written down as a sequence of diagonals and then read off as sequence of rows.
- e.g. encrypt message “meet me after the toga party” with a rail fence of depth 2.

→

m		**e**		**m**		**a**		**t**		**g**		**p**		**r**		**y**
	e		**t**		**e**		**f**		**o**		**a**		**a**		**t**	

- Encrypted message=mematgpryetefoaat

Simple XOR Cipher

- Bitwise XOR operation between plaintext and key to get ciphertext.

  i.e. plaintext $\oplus$key= ciphertext
- Bitwise XOR operation between ciphertext and key to get plaintext.

  i.e. ciphertext $\oplus$ key= plaintext
- e.g. plaintext=1101101 and key=1000000 then

  plaintext=1101101

  key=1000000

  ciphertext=0101101

Playfair Cipher

- Multiletter encryption cipher.
- It is based on the use of 5x5 matrix of letters constructed using keyword.
- e.g. keyword=“monarchy”, fill the matrix with letters without repeating letter. The letters ‘i’ and ‘j’ are counted as one letter.

m	**o**	**n**	**a**	**r**
c	**h**	**y**	b	d
e	f	g	i/j	k
l	p	q	s	t
u	v	w	x	z

- Take letters in pair. If there is same letter repeating in single pair, they shall be separated with a filter letter such as x. e.g in “aap”, ‘a’ and ‘a’ are in same pair sothat “aap” is treated as “ax ap”.

Playfair Cipher Cont…

- If two plaintext letters in a pair that fall in same row of matrix, each letter is replaced by the letter to the right, with the first element of the row circularly following the last. e.g. “ar” is encrypted as “rm”.
- If two plaintext letters in a pair that fall in same column of matrix, each letter is replaced by the letter beneath, with the top element of the column circularly following the last. e.g. “mu” is encrypted as “cm”.
- Otherwise each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. e.g. “hs” become “bp” and “ea” become “im” or “jm”.
- e.g. plaintext= “ballon”

  We can write plaintext in letter pair as ba lx lo nx

  From given matrix with key, letters pairs are replaced as:

  ba=ib

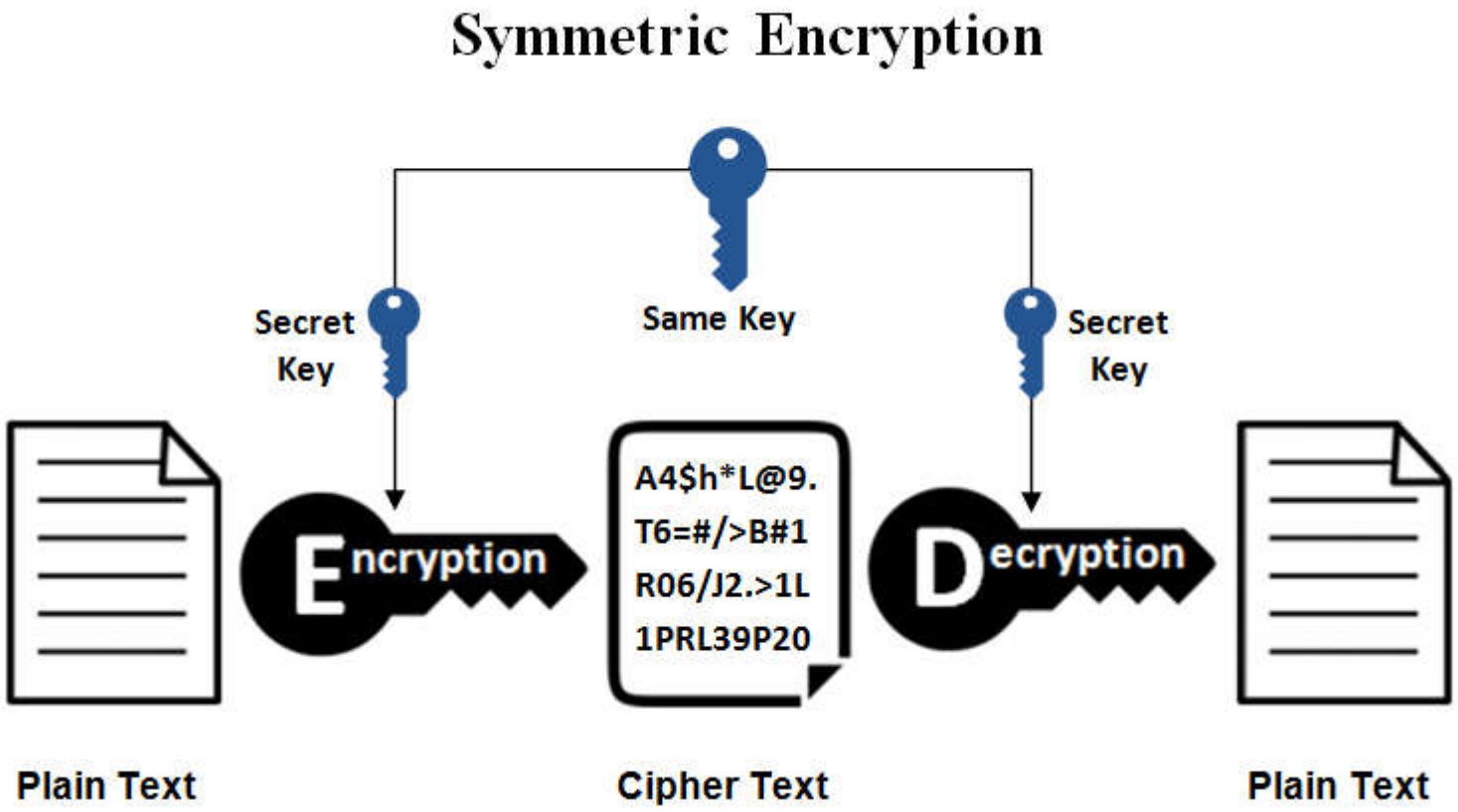
  lx=su

  lo=pm

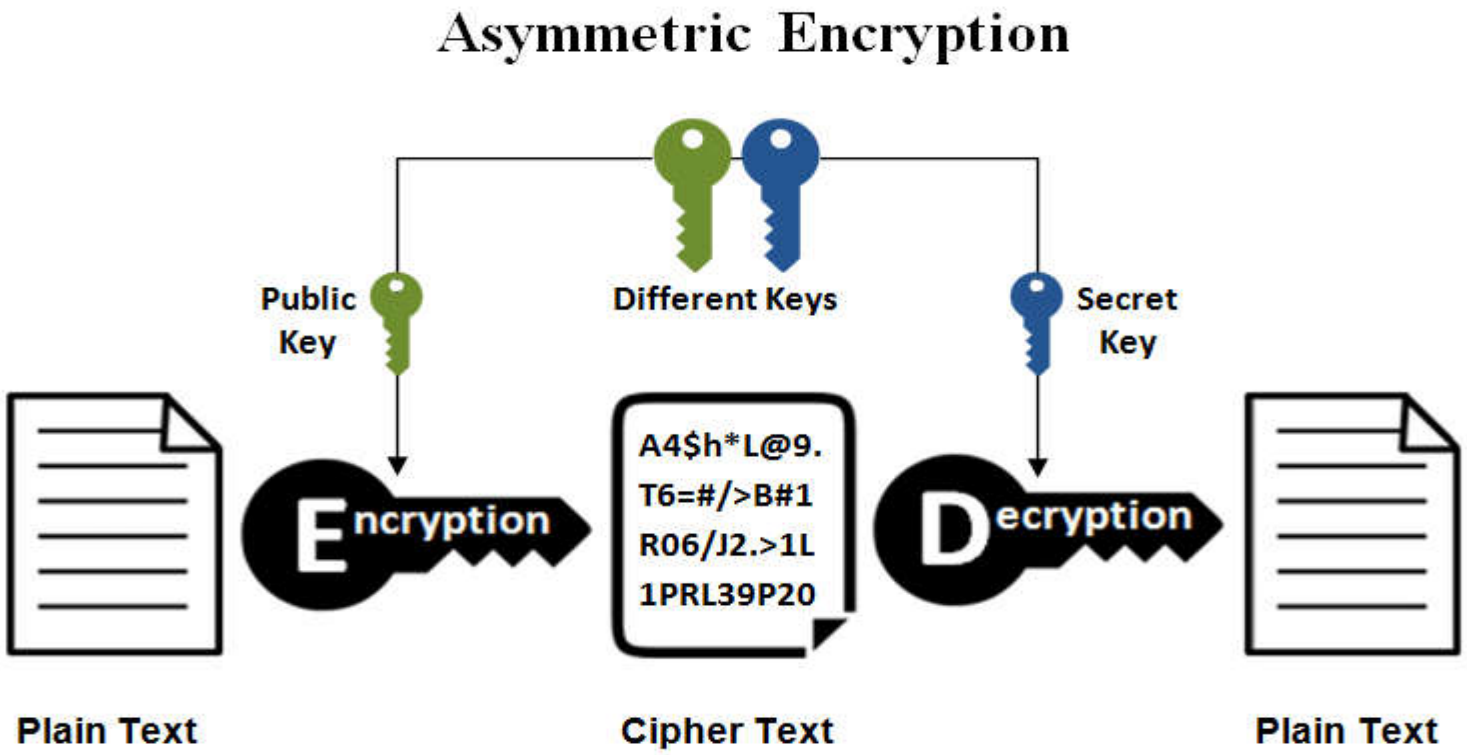
  nx=aw

  Hence, ciphertext=“ibsupmaw”

Symmetric key Cryptography

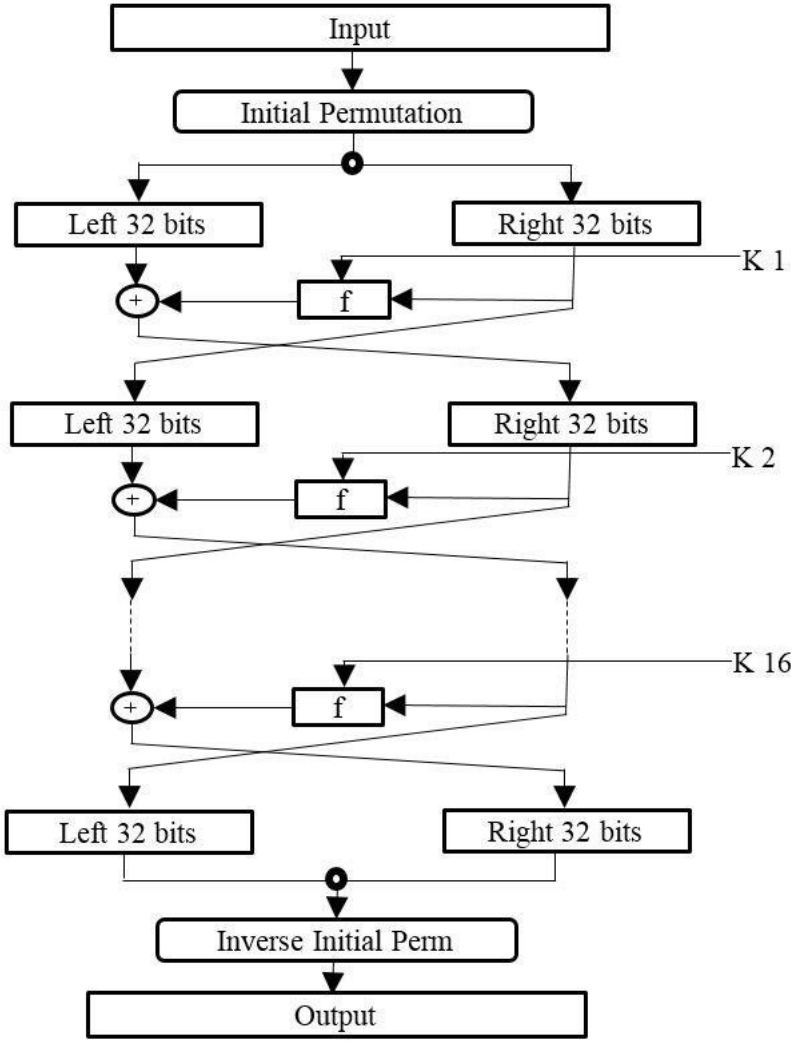


Asymmetric key Cryptography

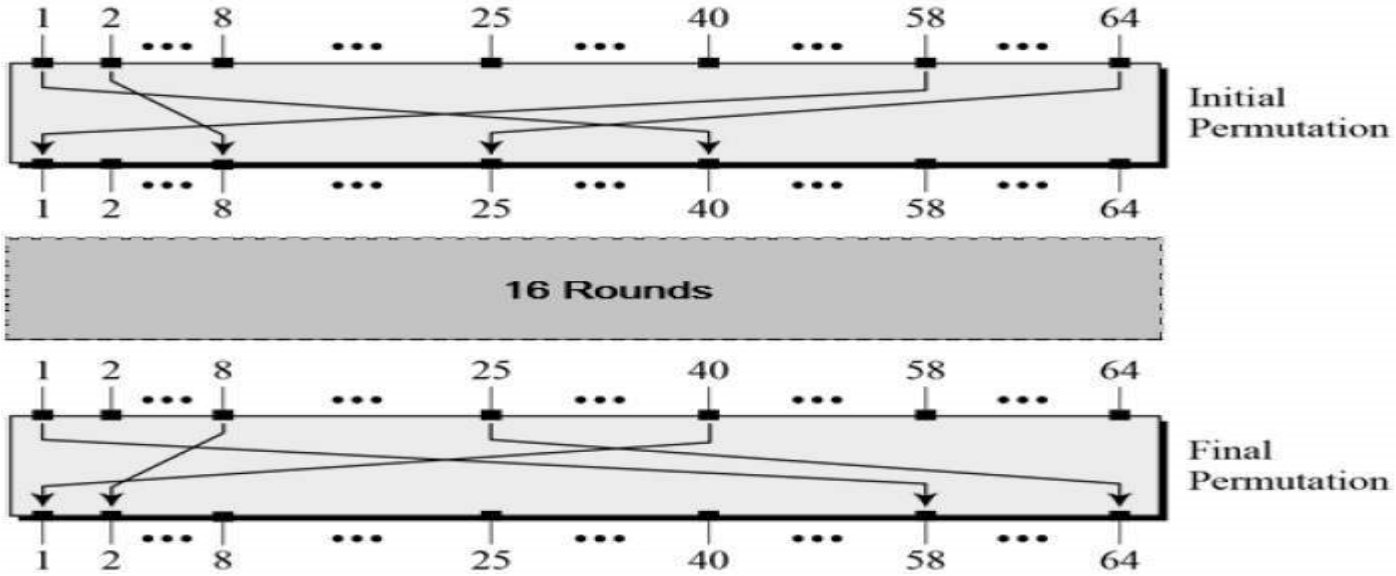


Data Encryption Standard (DES)

- The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
- DES is an implementation of a Feistel Cipher.
- It uses 16 round Feistel structure.
- The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).
- General Structure of DES is depicted in the figure as shown.
- Since DES is based on the Feistel Cipher, all that is required to specify DES is −
  - Round function
  - Key schedule
  - Any additional processing − Initial and final permutation



- The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES.
- The initial and final permutations are shown as follows:



DES Round Function

- The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.
- **Expansion Permutation Box** – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration.

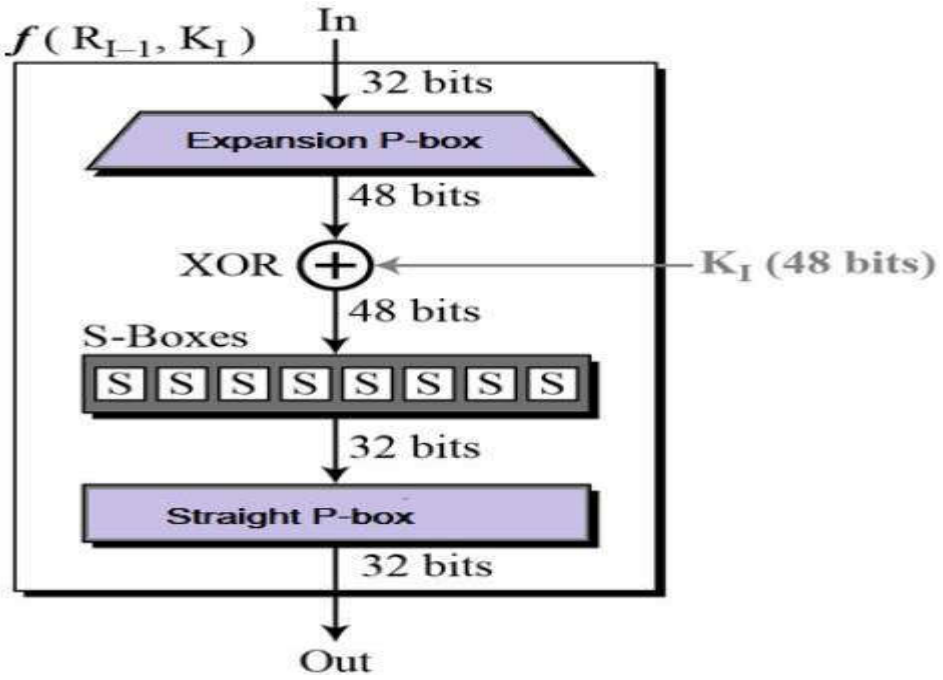


Fig. DES Round Function

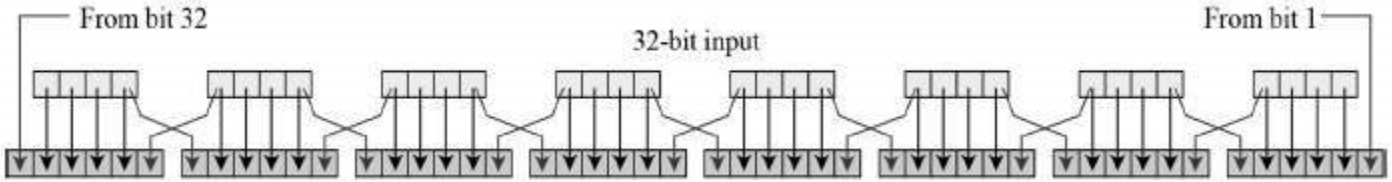


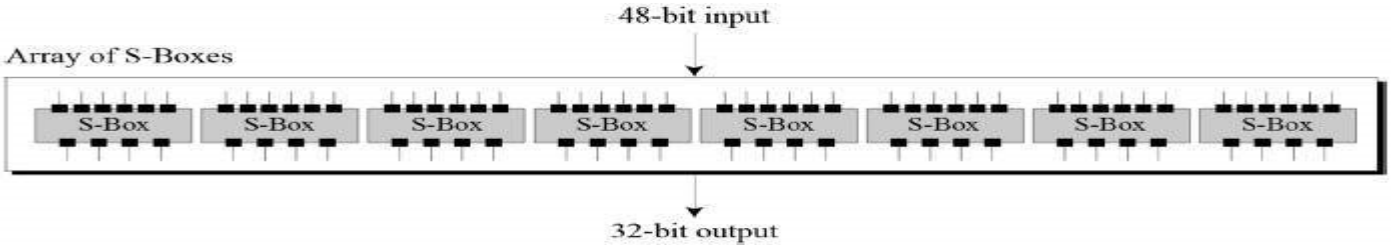
Fig. Permutation Logic for Expansion Permutation Box

- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown:

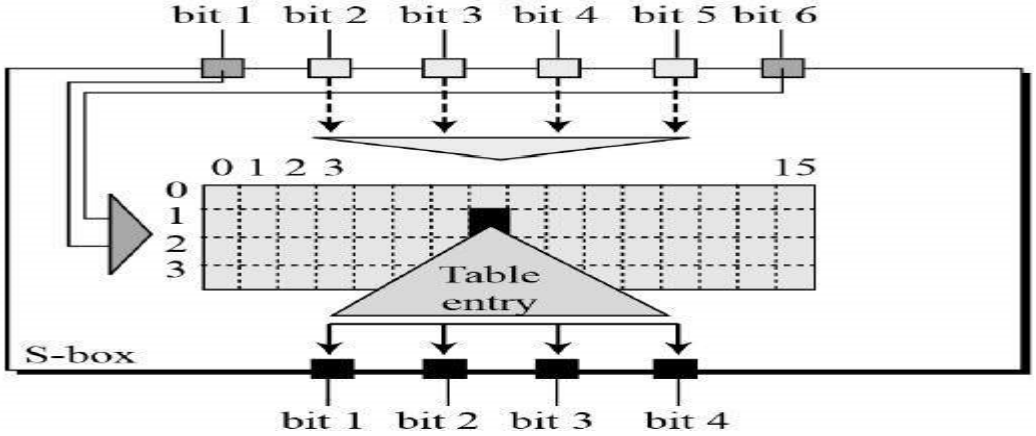
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- **XOR (Whitener).** – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration



- The S-box rule is illustrated below −



- The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

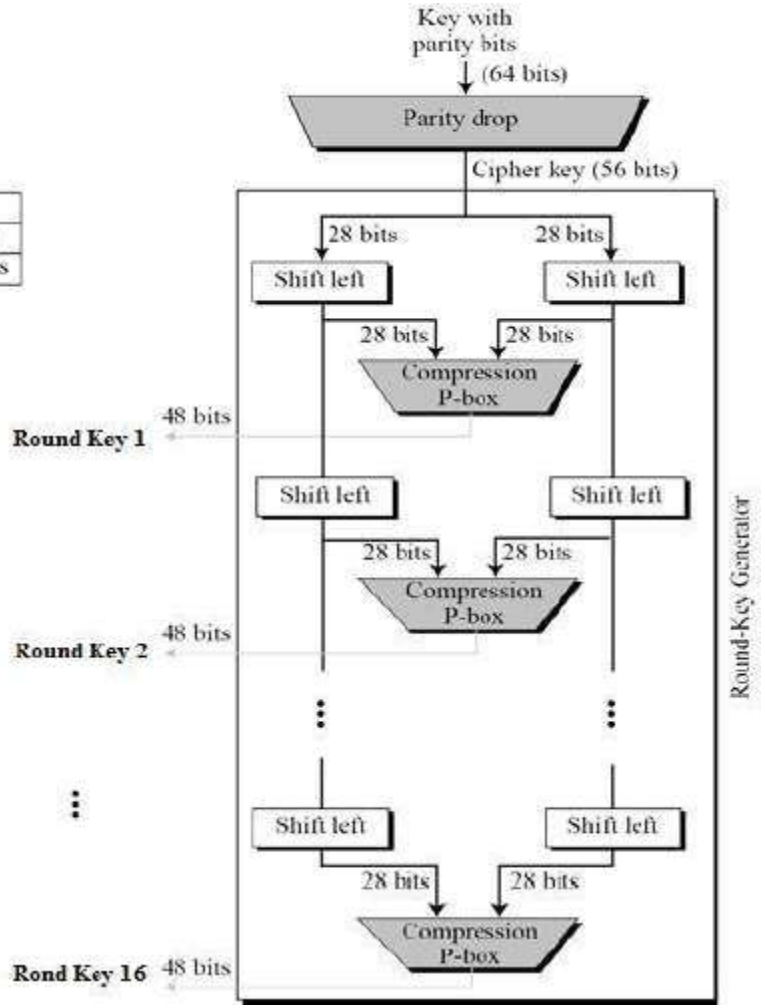
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

- The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.
- The process of key generation is depicted in the following illustration

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

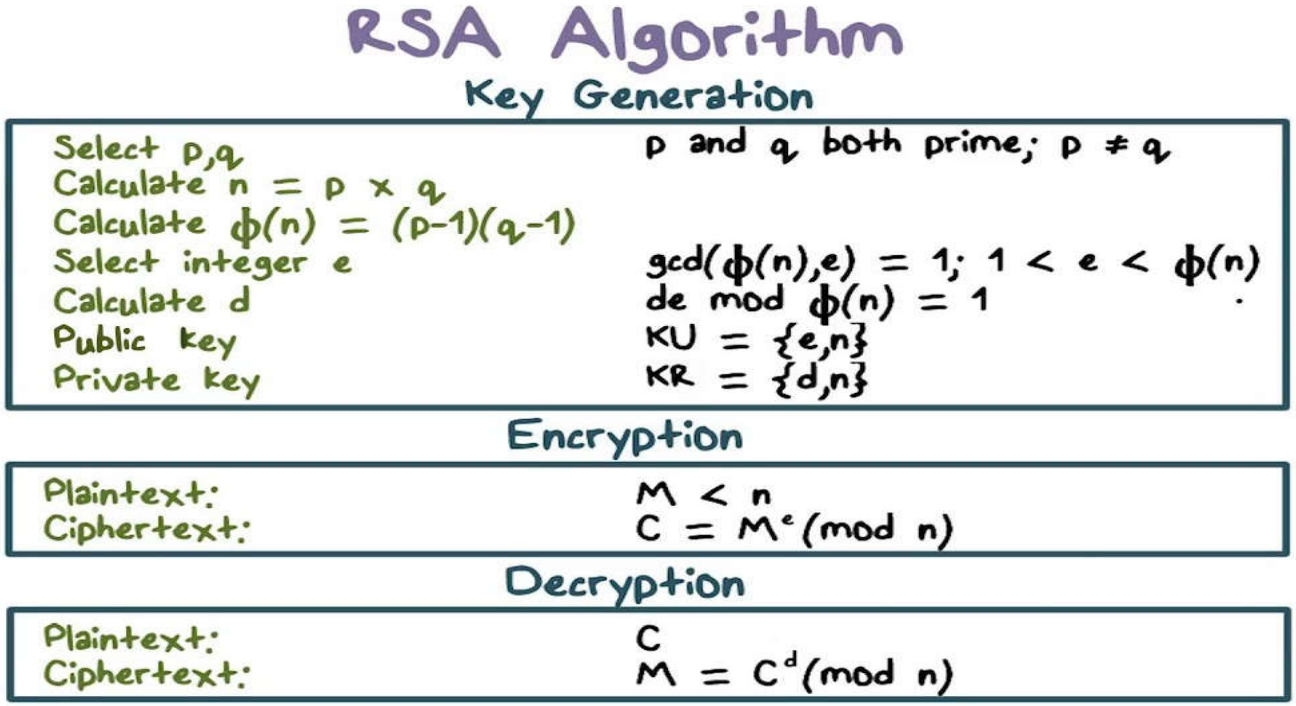


DES Analysis

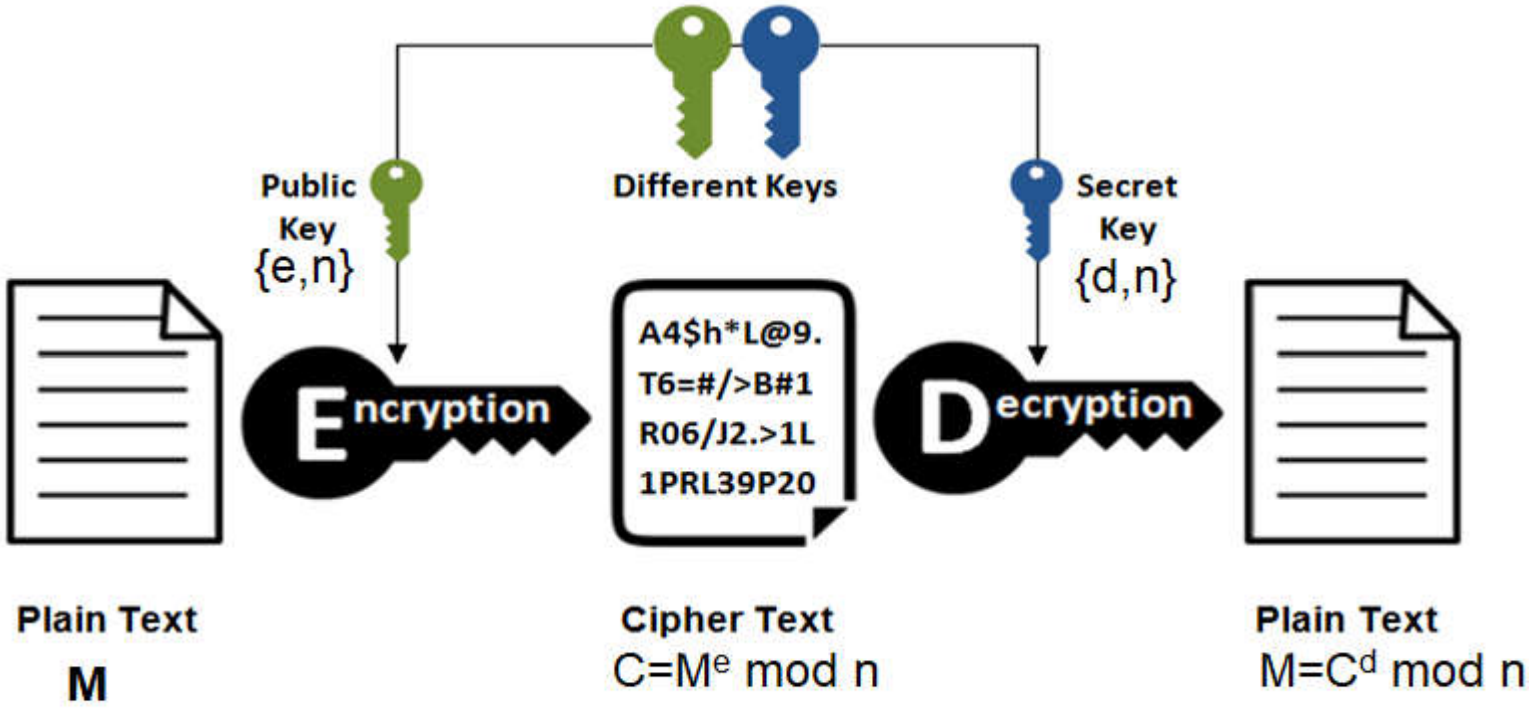
- The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.
  - **Avalanche effect** − A small change in plaintext results in the very great change in the ciphertext.
  - **Completeness** − Each bit of ciphertext depends on many bits of plaintext.
- During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.
- DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

RSA Algorithm

- RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption.
- It was invented by Rivest, Shamir and Adleman in year 1978 and hence name **RSA** algorithm.



RSA Algorithm for Encryption and Decryption

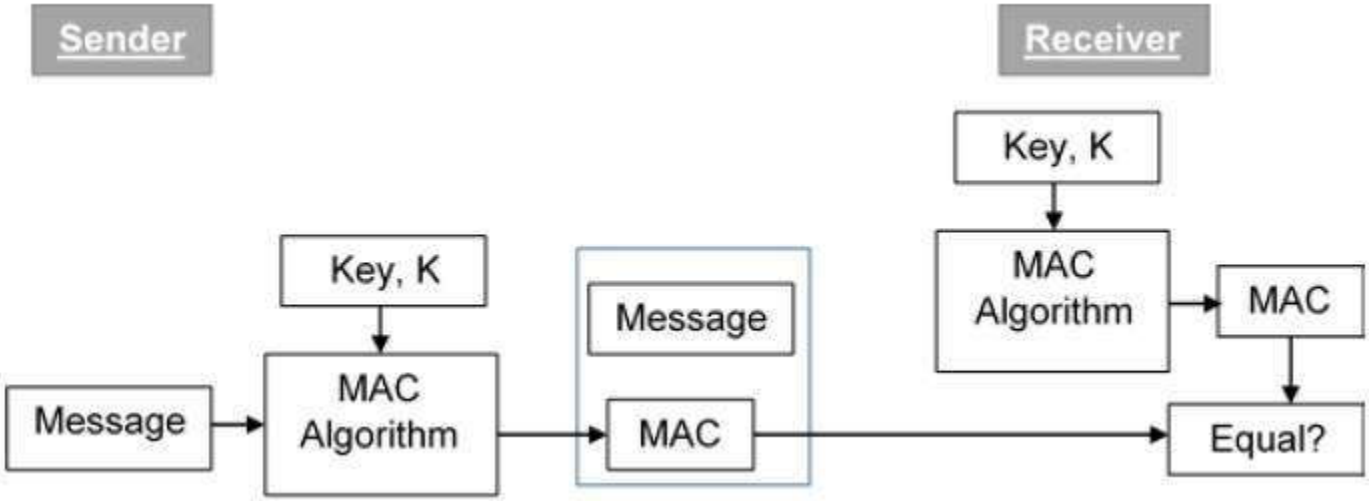


RSA Algorithm: Example

- Select two large primes: p, q, $p \neq q$
  p = 17, q = 11
- n = p×q = 17×11 = 187
- Calculate $\Phi = (p-1)(q-1) = 16x10 = 160$
- Select e, such that lcd(Φ, e) = 1; $0 < e < \Phi$
  say, e = 7
- Calculate d such that de mod Φ = 1
  - 160k+1 = 161, 321, 481, 641
  - Check which of these is divisible by 7
  - 161 is divisible by 7 giving d = 161/7 = 23
- Key 1 = {7, 187}, Key 2 = {23, 187}

Message Authentication Code (MAC)

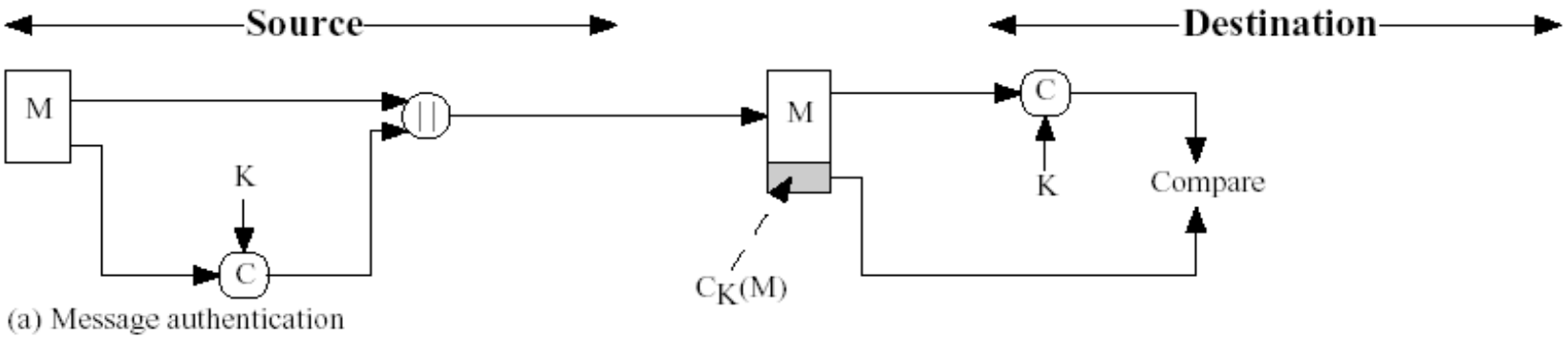
- A message authentication code (MAC) is a cryptographic checksum on data that uses a session key to detect both accidental and intentional modifications of the data.
- A MAC requires two inputs: a message and a secret key known only to the originator of the message and its intended recipient(s). This allows the recipient of the message to verify the integrity of the message and authenticate that the messege's sender has the shared secret key. If a sender doesn't know the secret key, the hash value would then be different, which would tell the recipient that the message was not from the original sender.



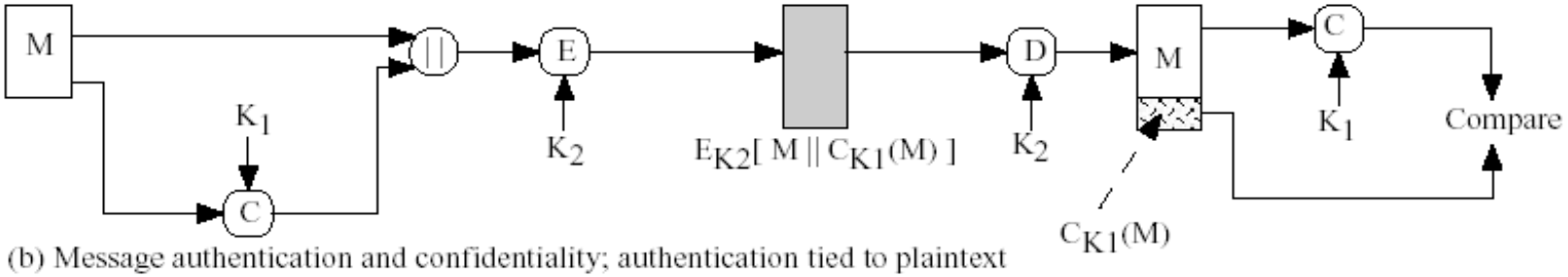
MAC Cont..

- The sender uses some publicly known MAC algorithm, inputs the message and secret key and produce a MAC value.
- Similar to hash, MAC function also compress a arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during compression and hash does not.
- The sender forwards the message along with MAC. Here, we assume that the message is sent in the clear, as we concerned of providing message origin authentication, not confidentiality. If confidentiality is required the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and shared secret key into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself/ herself that the message has been sent by the intended sender.
- If computed MAC does not match MAC sent by sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line a receiver safely assumes that the message is not genuine.

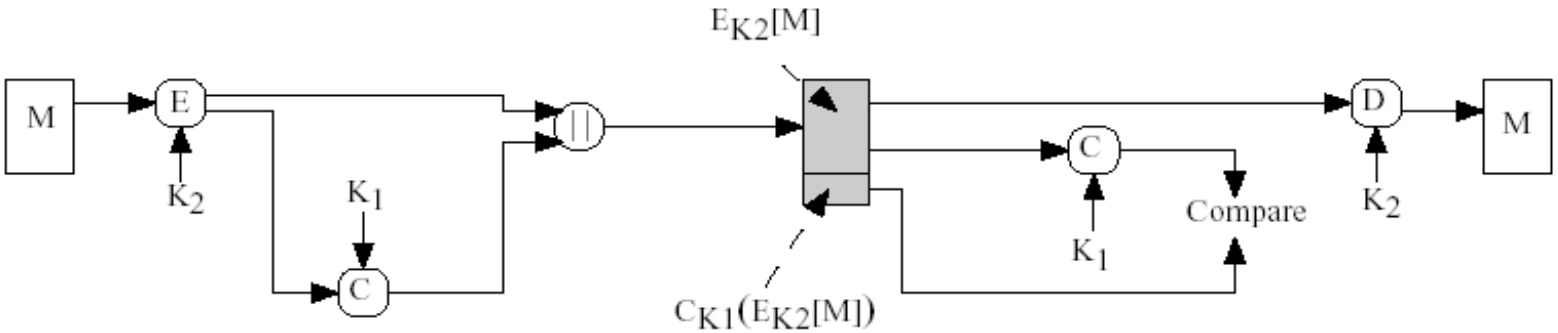
Basic Uses of MAC



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Basic Uses of MAC

Table 8.2 Basic Uses of Message Authentication Code C

(a) $A \rightarrow B: M \| C_K(M)$ •Provides authentication —Only A and B share K
(b) $A \rightarrow B: E_{K_2}\left[M \| C_{K_1}(M)\right]$ •Provides authentication —Only A and B share K_1 •Provides confidentiality —Only A and B share K_2
(c) $A \rightarrow B: E_{K_2}[M] \| C_{K_1}\left(E_{K_2}[M]\right)$ •Provides authentication —Using K_1 •Provides confidentiality —Using K_2

Why Use MACs?

– i.e., why not just use encryption?

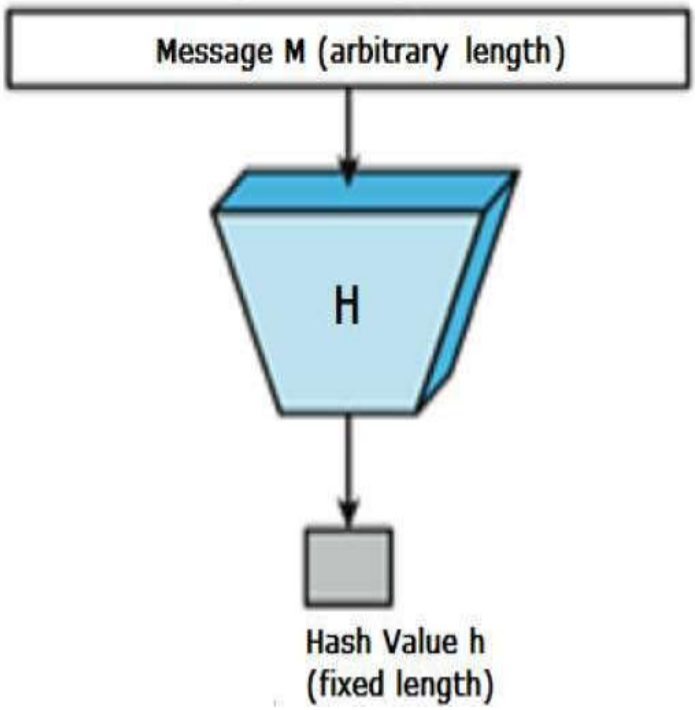
- Cleartext stays clear
- MAC might be cheaper
- Broadcast
- Authentication of executable codes
- Architectural flexibility
- Separation of authentication check from message use

Limitation of MAC

- There are two major limitations of MAC, both due to its symmetric nature of operation.
  - Establishment of shared secret
  - Inability to provide non repudiation

Hash Function

- Hash functions are extremely useful and appear in almost all information security applications.
- A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.
- Values returned by a hash function are called **message digest** or simply **hash values**.



Hash Function

- Converts a variable size message M into fixed size hash code H(M) (Sometimes called a *message digest*)
- Can be used with encryption for authentication
  - E(M || H)
  - M || E(H)
  - M || signed H
  - E(M || signed H) gives confidentiality
  - M || H(M || K)
  - E(M || H(M || K))

Basic Uses of Hash Function

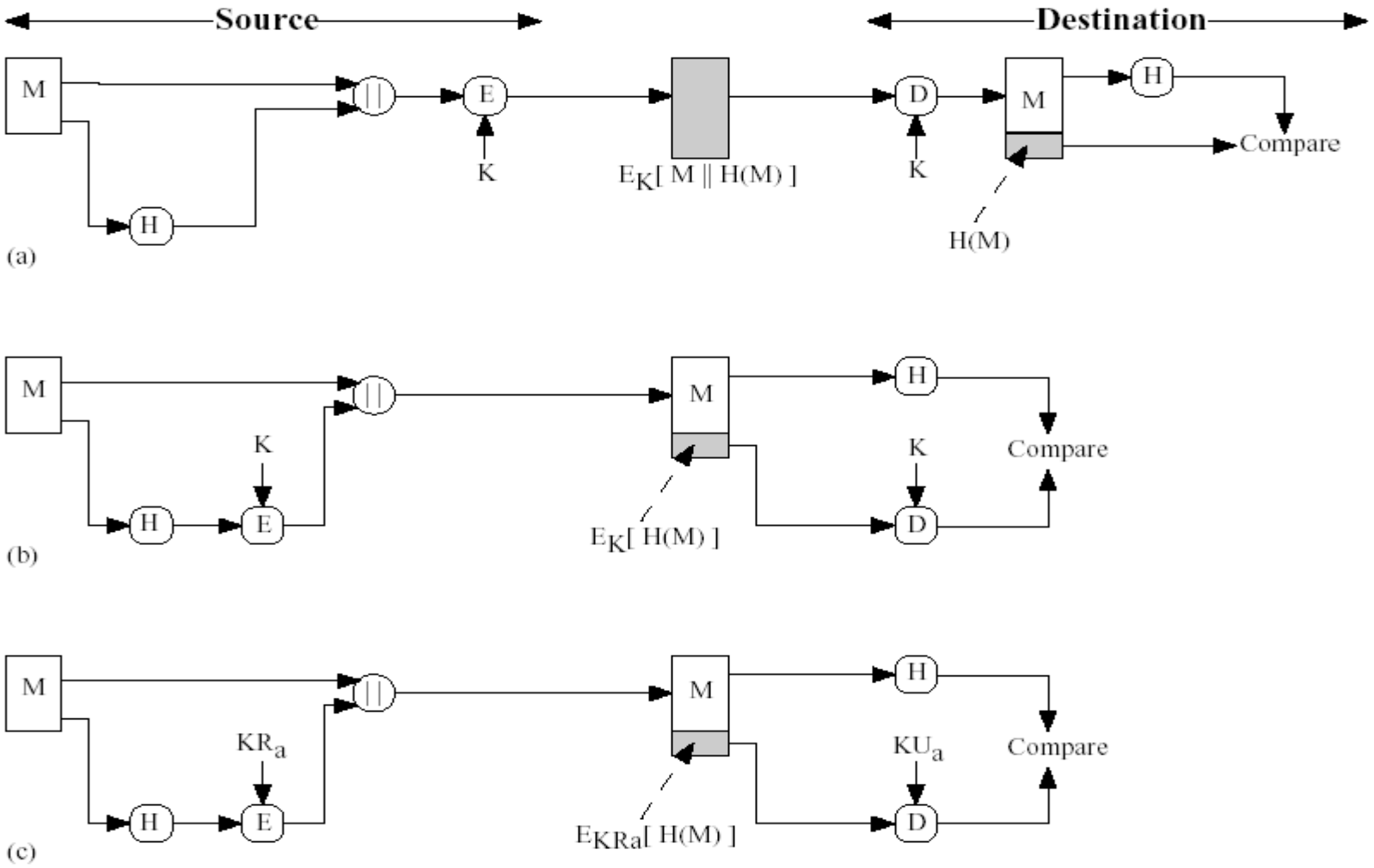


Figure 8.5 Basic Uses of Hash Function (page 1 of 2)

Basic Uses of Hash Function

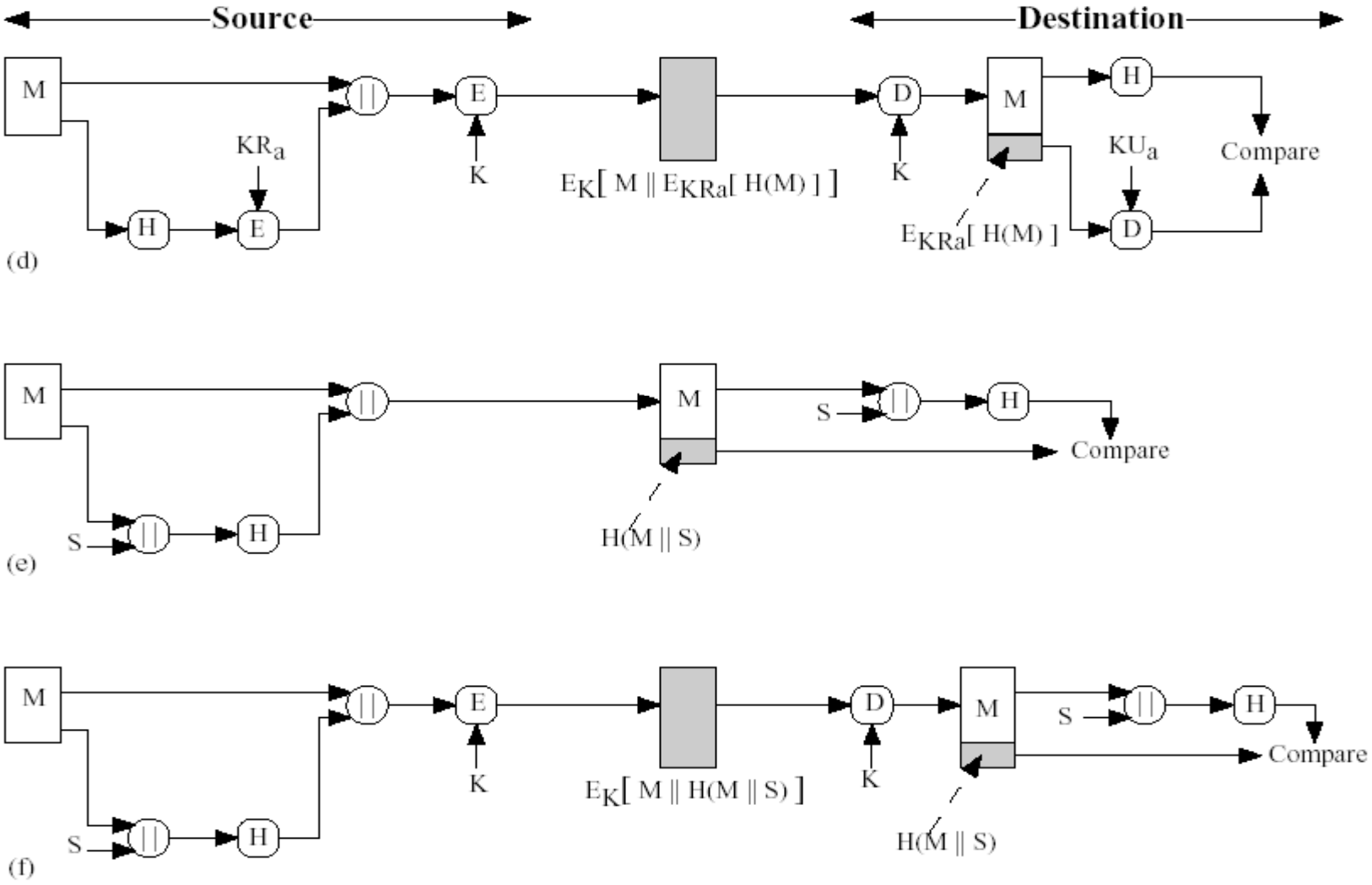


Figure 8.5 Basic Uses of Hash Function (page 2 of 2)

Basic Uses of Hash Function

Table 8.3 Basic Uses of Hash Function H

(a) $A \rightarrow B: E_K[M \parallel H(M)]$ •Provides confidentiality —Only A and B share K •Provides authentication —H(M) is cryptographically protected	**(d)** $A \rightarrow B: E_K[M \parallel E_{KRa}[H(M)]]$ •Provides authentication and digital signature •Provides confidentiality —Only A and B share K
(b) $A \rightarrow B: M \parallel E_K[H(M)]$ •Provides authentication —H(M) is cryptographically protected	**(e)** $A \rightarrow B: M \parallel H(M \parallel S)$ •Provides authentication —Only A and B share S
(c) $A \rightarrow B: M \parallel E_{KRa}[H(M)]$ •Provides authentication and digital signature —H(M) is cryptographically protected —Only A could create $E_{KR_a}[H(M)]$	**(f)** $A \rightarrow B: E_K[M \parallel H(M) \parallel S]$ •Provides authentication —Only A and B share S •Provides confidentiality —Only A and B share K

Message Authentication Codes

- MAC= $C_K(M)$
- Key length requirements
  - Sufficient key length to thwart brute force attack

Hash

- Also known as
  - Message digest
  - One-way function
- Function: input message -> output
- One-way: d=h(m), but not h'(d) = m
  - Computationally infeasible find the message given the digest
- Cannot find m_1 and m_2, where $d_1 = d_2$
- Randomness:
  - Any bit in the output '1' half the time
  - Each output: 50% '1' bits

Hash Functions

- h = H(M)
- M is a variable-length message, h is a fixed-length hash value, H is a hash function
- The hash value is appended at the source
- The receiver authenticates the message by recomputing the hash value
- Because the hash function itself is not considered to be secret, some means is required to protect the hash value

Hash Function Requirements

1. H can be applied to any size data block
2. H produces fixed-length output
3. H(x) is relatively easy to compute for any given x
4. H is *one-way*, i.e., given h, it is computationally infeasible to find any x s.t. h = H(x)
5. H is *weakly collision resistant*: given x, it is computationally infeasible to find any $y \neq x$ s.t. H(x) = H(y)
6. H is *strongly collision resistant*: it is computationally infeasible to find any x and y s.t. H(x) = H(y)

Hash Function Requirements

- One-way property is essential for authentication
- Weak collision resistance is necessary to prevent forgery
- Strong collision resistance is important for resistance to birthday attack

Simple Hash Functions

- Operation of hash functions
  - The input is viewed as a sequence of n-bit blocks
  - The input is processed one block at a time in an iterative fashion to produce an n-bit hash function
- Simplest hash function: Bitwise XOR of every block
  - $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$
    - C_i = i-th bit of the hash code, $1 \leq i \leq n$
    - m = number of n-bit blocks in the input
    - b_{ij} = i-th bit in j-th block
  - Known as longitudinal redundancy check

Simple Hash Functions

- **Improvement over the simple bitwise XOR**
  - **Initially set the n-bit hash value to zero**
  - **Process each successive n-bit block of data as follows**
    - » **Rotate the current hash value to the left by one bit**
    - » **XOR the block into the hash value**

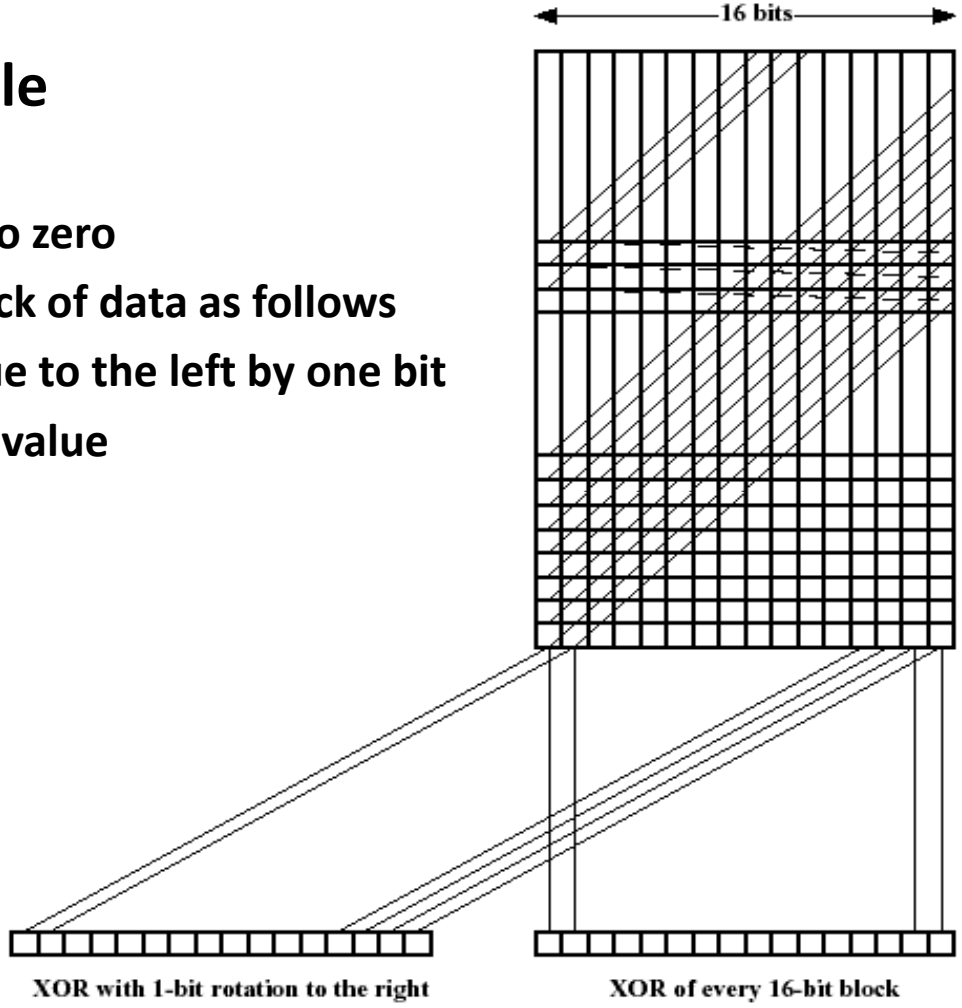


Figure 8.8 Two Simple Hash Functions

Birthday Paradox

- What is the minimum value of n such that the probability is greater than 0.5 that at least two people in a group of n people have the same birthday?
  - Ignore Feb. 29 and assume each birthday is equally likely
- Probability of n people having n different birthdays:

$$\frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdots \frac{365 - n + 1}{365}$$

- Probability that at least two people have the same birthdays:
  - 1 - $\frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdots \frac{365 - n + 1}{365}$
- n is about 23

Generalization of Birthday Problem

- Compute probability of *different* birthdays
- Random samples of n people (birthdays) taken from d (365) days
- What is the minimum value of n such that the probability is greater than 0.5 that there is at least one duplicate?
  - $P(n, d) = 1 - \frac{d-1}{d}\frac{d-2}{d}\cdots\frac{d-(n-1)}{d}$
- For large n and d, we have
  - $n = 1.2 * d^{1/2}$
- Implication
  - We expect to obtain the same output after about $1.2 * d^{1/2}$ trials

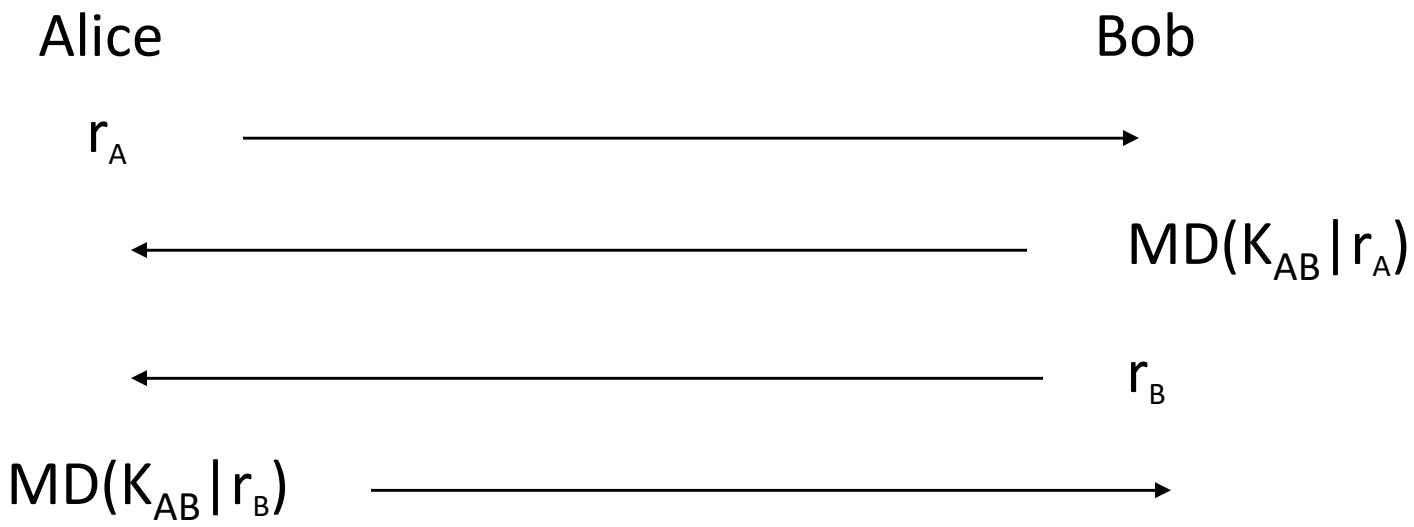
http://www.rsasecurity.com/rsalabs/node.asp?id=2205

How many bits for hash?

- *m* bits, takes $2^{m/2}$ to find two with the same hash
- 64 bits, takes 2^{32} messages to search (doable)
- Need at least 128 bits
- Example use
  - Fingerprint a program/document: attackers cannot find a different program with the same message digest

Hash used for Authentication

- Alice and Bob share a secret K_{AB}



Computing a MAC with a HASH

- Cannot just compute MD(m)
  - Anyone can compute MD(m)
- MAC: MD(K_{AB}|m)
  - Allows concatenation with additional message: MD($K_{AB}|m|m'$)
    - MD through chunk n depends on MD through chunks *n*-1 and the data in chunk *n*
    - 512-bit blocks, append (message length, pad)
- How to solve?
  - Put secret at the end of message:
    - MD($m|K_{AB}$)
  - Use only half the bits of the message digest as the MAC
  - Concatenate the secret to both the front and the back of the message

Encryption with a Message Digest

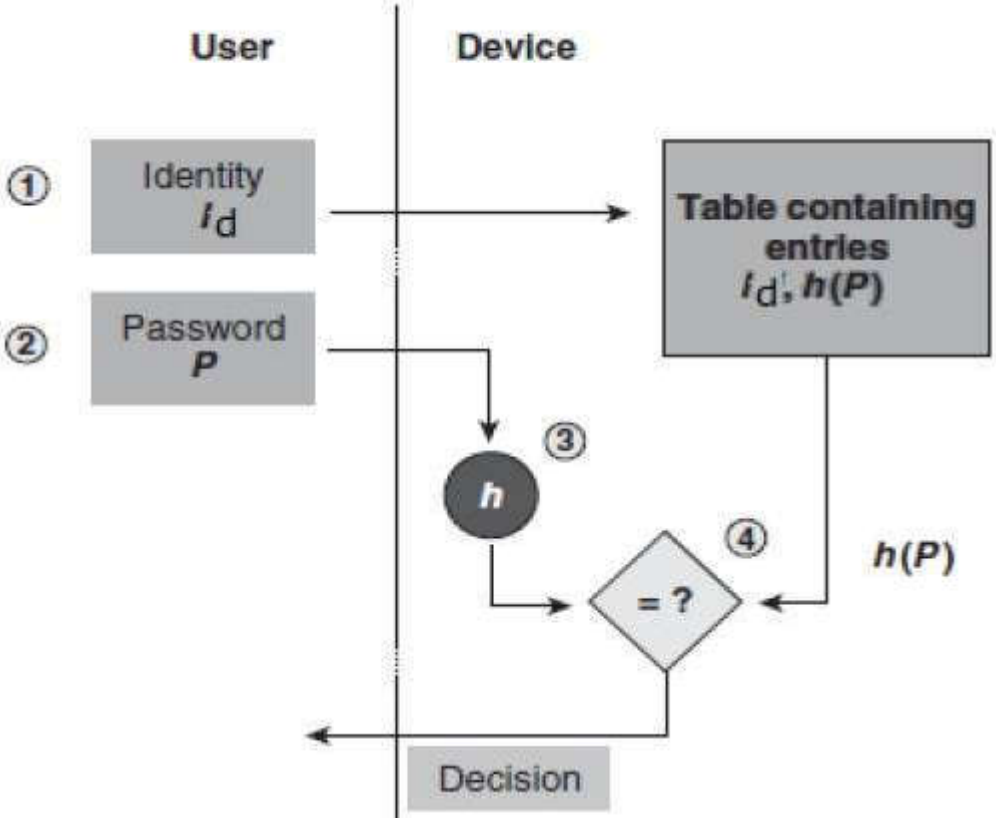
- One-time pad:
  - compute bit streams using MD, K, and IV
    - b_1=MD(K_{AB}|IV), b_i=MD(K_{AB}|b_{i-1}), …
  - $\oplus$ with message blocks
- Mixing in the plaintext
  - similar to cipher feedback mode (CFB)
    - b_1=MD(K_{AB}|IV), $c_1 = p_1 \oplus b_1$
    - b_2=MD(K_{AB}| c_1), $c_2 = p_2 \oplus b_2$

Applications of Hash Function

- There are two direct applications of hash function based on its cryptographic properties.
  - Password Storage
  - Data Integrity Check

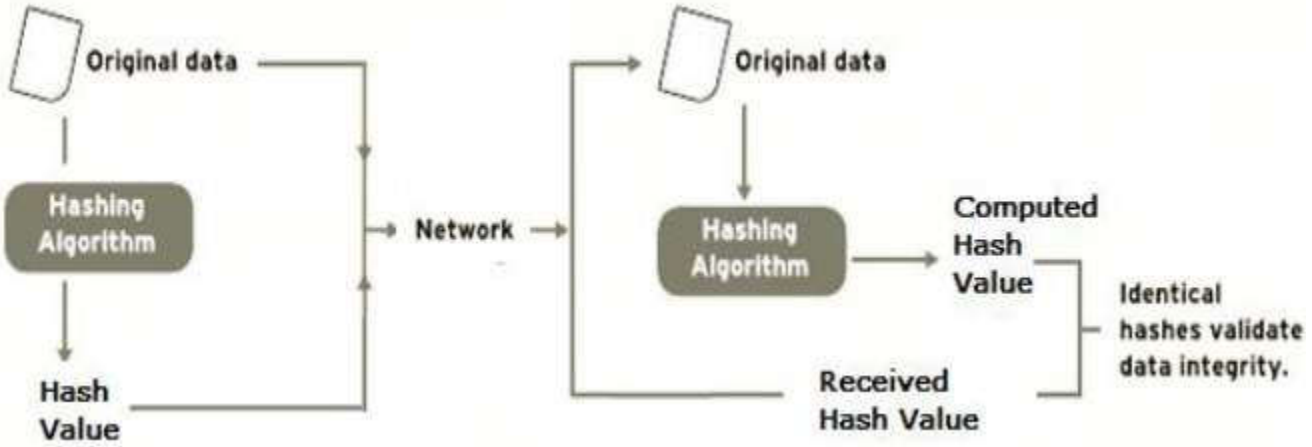
- Password Storage
  - Hash functions provide protection to password storage.
  - Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.
  - The Password file consists of a table of pairs which are in the form (user id, h(P)).

- The process of logon is depicted in the following illustration –



– An intruder can only see the hashes of passwords, even if he accessed the password. He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

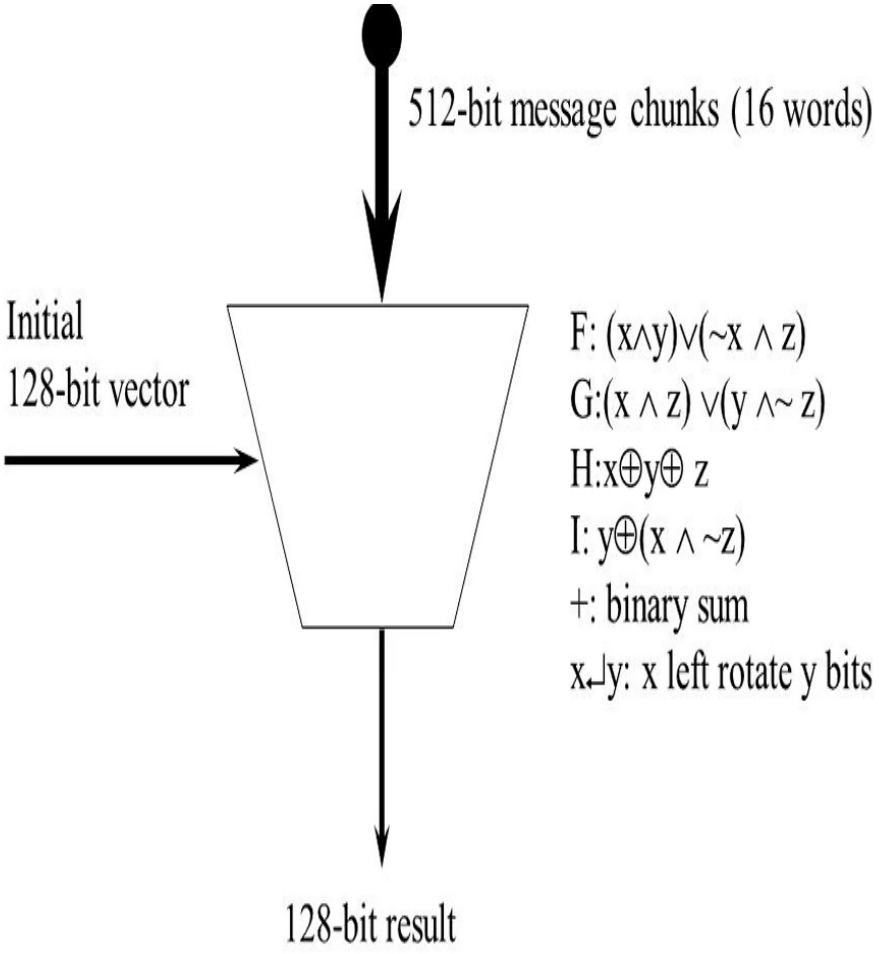
- Data Integrity Check
  - Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files. This application provides assurance to the user about correctness of the da
  - Th
    ill



– The integrity check helps the user to detect any changes made to original file. It however, does not provide any assurance about originality. The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver. This integrity check application is useful only if the user is sure about the originality of file.

Message Digest-5 (MD-5)

- Designed and developed by Roald Rivest (the R in RSA).
- It is widely used secure hash algorithm that takes as input message of arbitrary length and provide 128 bit message digest as an output.
- It processes input in 512 bit blocks.
- Pad message so its length is 448 mod 512.
- Append a 64 bit length value to message.
- Initialize 4-word (128 bit) MD buffer (A, B, C, D).
- Process message in 16 word (512 bit) block:
  - Using 4 rounds of 16 bit operations on message block and buffer.
  - Add output buffer input to form new buffer value.
- Output hash value is the final buffer value.



MD-5 Cont..

- e.g. Given message is “abc” consisting of three 8-bit ASCII characters with total length l=24 bits.

  a=01100001, b=01100010 and c=01100011

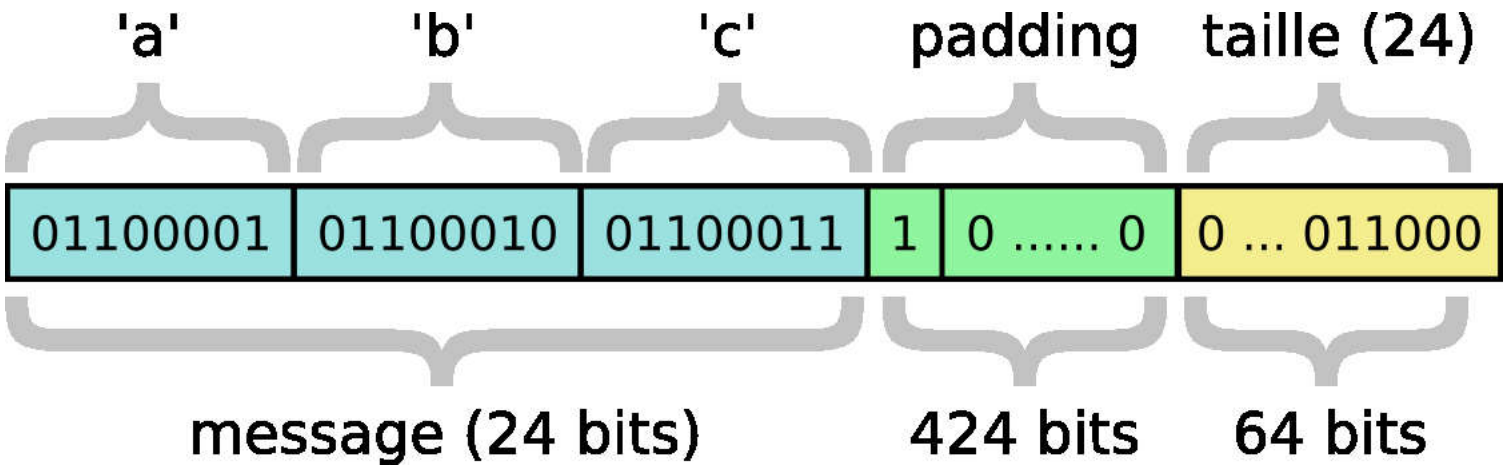
  padding length is given as:

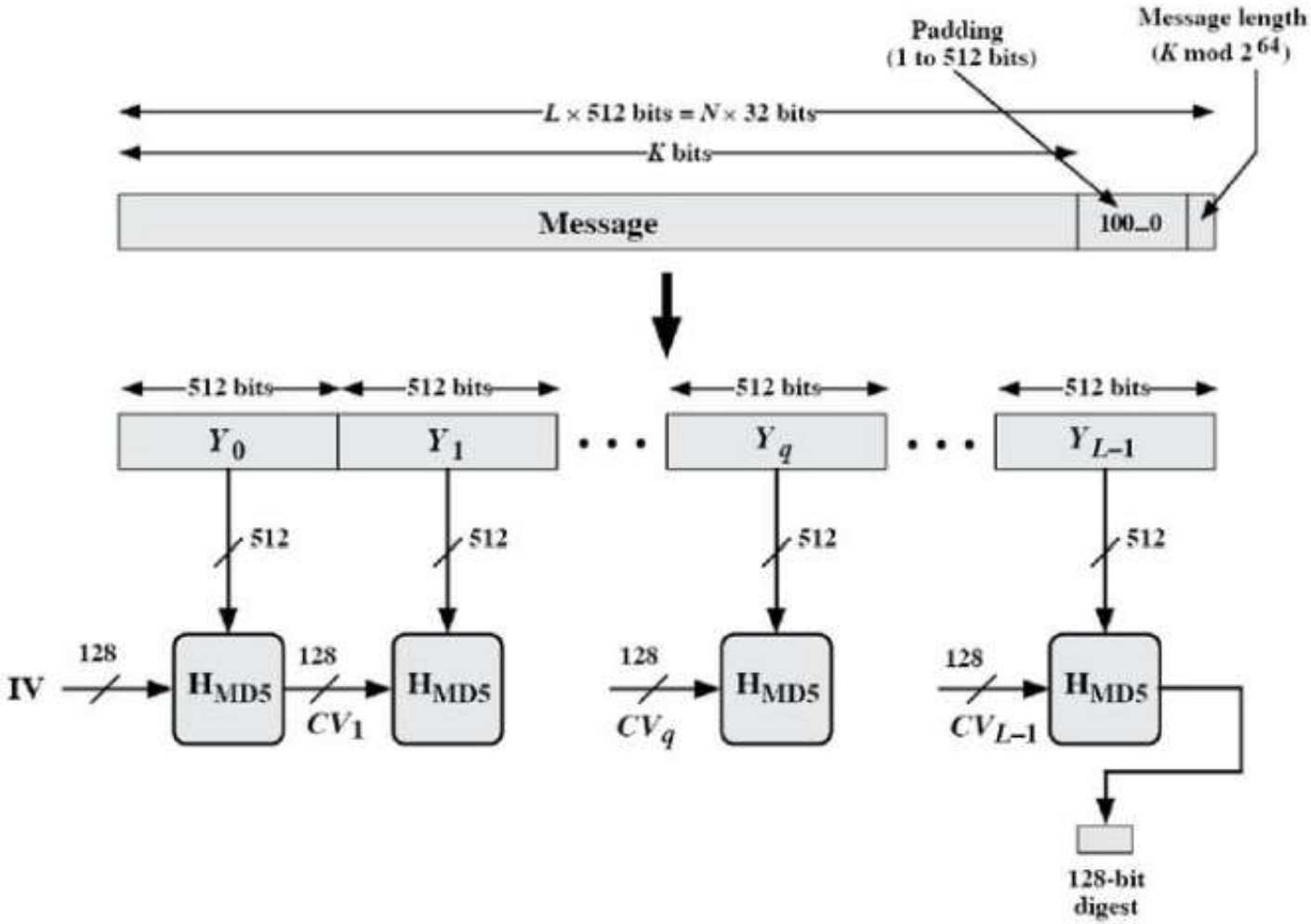
  k=448-(l+1)=448-(24+1)=423 mod 512

  $l=24_{10}=1100_2$

  The padded message is

  01100001 01100010 01100011 1





Message Digest Generation Using MD5

Secure Hash Algorithm (SHA-1)

- Developed by NIST
- SHA is specified as the hash algorithm in the Digital Signature Standard (DSS), NIST
- Take a message of length at most 2^{64} bits and produces a 160-bit output.
- SHA design is similar to MD5, but a lot stronger
- Make five passes over each block of data

SHA-1 cont'd

- Step 1: Message Padding – same as MD5
- Step 2: Initialize MD buffer 5 32-bit words

  A|B|C|D|E

  A = 67452301

  B = efcdab89

  C = 98badcfe

  D = 10325476

  E = c3d2e1f0

Assignment 2

1. What do you mean by Symmetric Cryptography?
1. What is SHA-1? Illustrate padding message of SHA-1.
2. Explain Caesor Cipher with a suitable example.
3. Define public key cryptography. Explain RSA algorithm with a suitable example.
4. What do you mean by classical cryptography?
5. Encrypt the message “college” using Playfair cipher technique with key word “voyage”.
6. How does block cipher differ from stream cipher?
7. What do you mean by cryptographic hash function?
8. What are the block sizes for plaintext, ciphertext and key for DES?
9. Encrypt the message “Cheating in the exam is unethical” using transposition cipher technique having rail fence depth of 3.
10. Explain the structure of DES with block diagram.
11. Explain the ingredients of symmetric key cryptography.
12. Define public key cryptography. Perform Encryption and decryption for p =7, q = 11, e = 13 and m = 2 using RSA.