RV College of Engineering®

Go, change the world

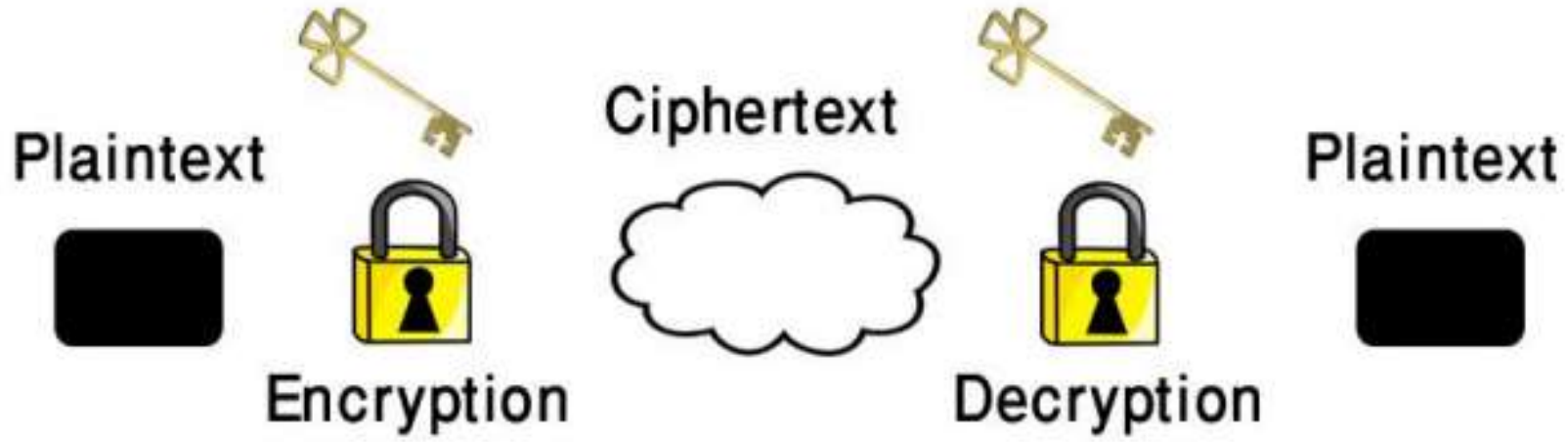
Unit 4- Traditional Block Cipher and Public key Cryptosystem

Contents

- Stream Ciphers and Block Ciphers, Feistel Cipher Structure.
- The Data Encryption Standard-Encryption and Decryption.
- Principles of Public Cryptosystems- Public-Key Cryptosystems.
- Applications for Public-Key Cryptosystems.
- Requirements for Public-Key Cryptosystems.
- Public-Key Cryptanalysis.
- The RSA algorithm-Algorithm and Computational Aspects and the security of RSA.
- Other Public key Cryptosystems: Diffie-Hellman Key Exchange.

Stream Ciphers and Block Ciphers, Feistel Cipher Structure.

- Encryption Definition: –The action of disguising information so that it can be recovered easily by the persons who have the key, but is highly resistant to recovery by persons who do not have the key.



- A message is cleartext (plaintext) is encrypted (disguised) through the use of an encryption key to create a Ciphertext.
- The encryption key may be changed from time to time to make an intruder's task more difficult.
- Restoration of a ciphertext to cleartext is achieved by the action of decryption using a decryption key.
- In symmetric (Single key) - The encryption and decryption keys are the same.
- In asymmetric (two keys) - The encryption and decryption keys are different.

Encryption Methods

- Encryption Methods - Encryption is accomplished by scrambling the bits, characters, words, or phrases in the original message.
- Scrambling involves two activities:
- Transposition - In which the order of the bits patterns, characters, words or phrases is rearranged.
- Substitution - In which new bit patterns, characters, words, or phrases are substituted for the originals without changing their order.

STREAM CIPHERS and BLOCK CIPHERS

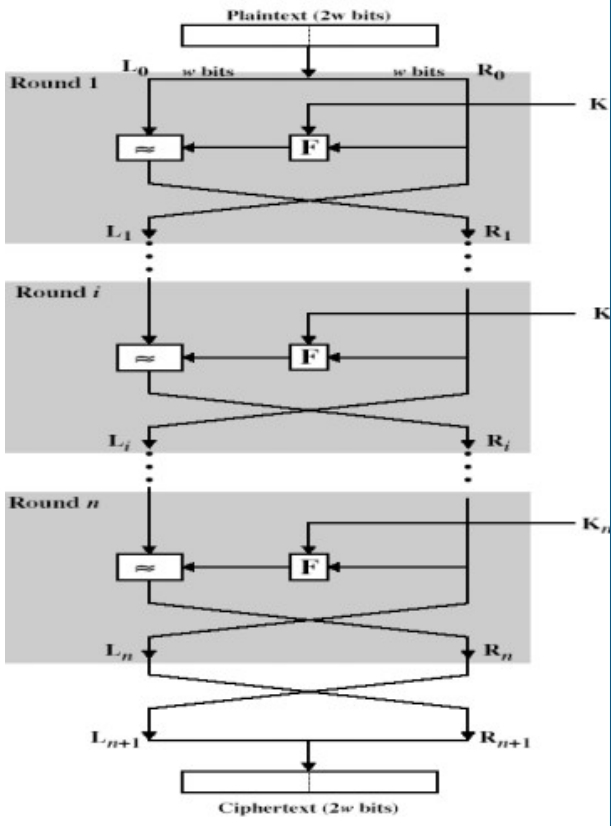
- Stream ciphers process messages a bit or byte at a time when en/decrypting.
- Block ciphers process messages into blocks, each of which is then en/decrypted – 64-bits or more
- Many current ciphers are block ciphers hence are focus of course

Block Cipher Principles

- block ciphers look like an extremely large substitution would need table of 264 entries for a 64-bit block, arbitrary reversible substitution cipher for a large block size is not practical
- – 64-bit general substitution block cipher, key size 2 64!
- Most symmetric block ciphers are based on a Feistel Cipher Structure needed since must be able to decrypt ciphertext to recover messages efficiently

FEISTEL CIPHER STRUCTURE

- Horst Feistel devised the feistel cipher – implements Shannon's substitution-permutation network
- Concept - partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves



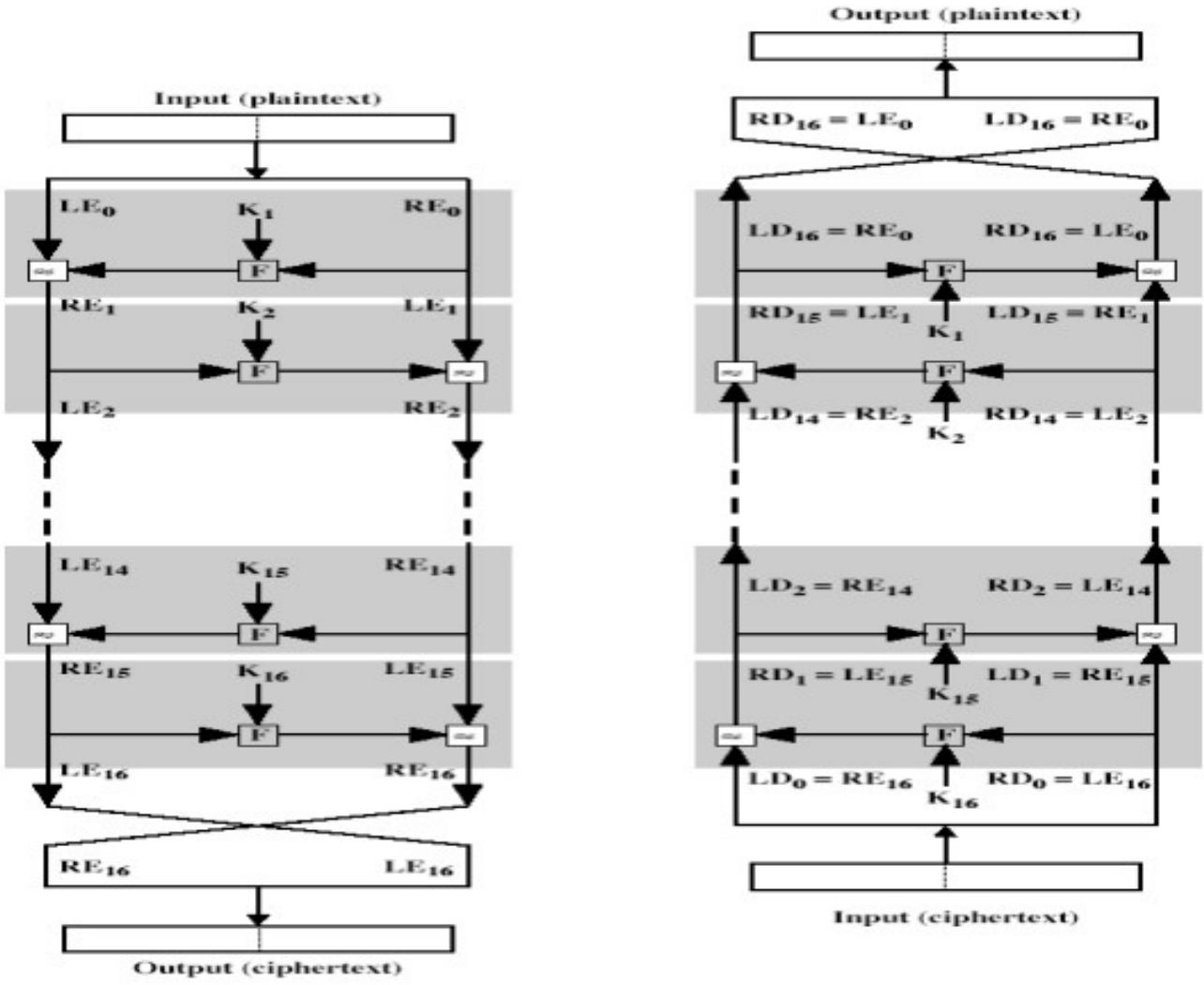
FEISTEL CIPHER

- n sequential rounds
- A substitution on the left half Li –1.
- Apply a round function F to the right half Ri and –2.
- Take XOR of the output of (1) and Li
- The round function is parameterized by the subkey Ki –Ki are derived from the overall key K

FEISTEL CIPHER DESIGN PRINCIPLES

- block size - increasing size improves security, but slows cipher
- key size - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- number of rounds - increasing number improves security, but slows cipher
- subkey generation - greater complexity can make analysis harder, but slows cipher
- round function - greater complexity can make analysis harder, but slows cipher
- fast software en/decryption & ease of analysis - are more recent concerns for practical use and testing

FEISTEL CIPHER DECRYPTION

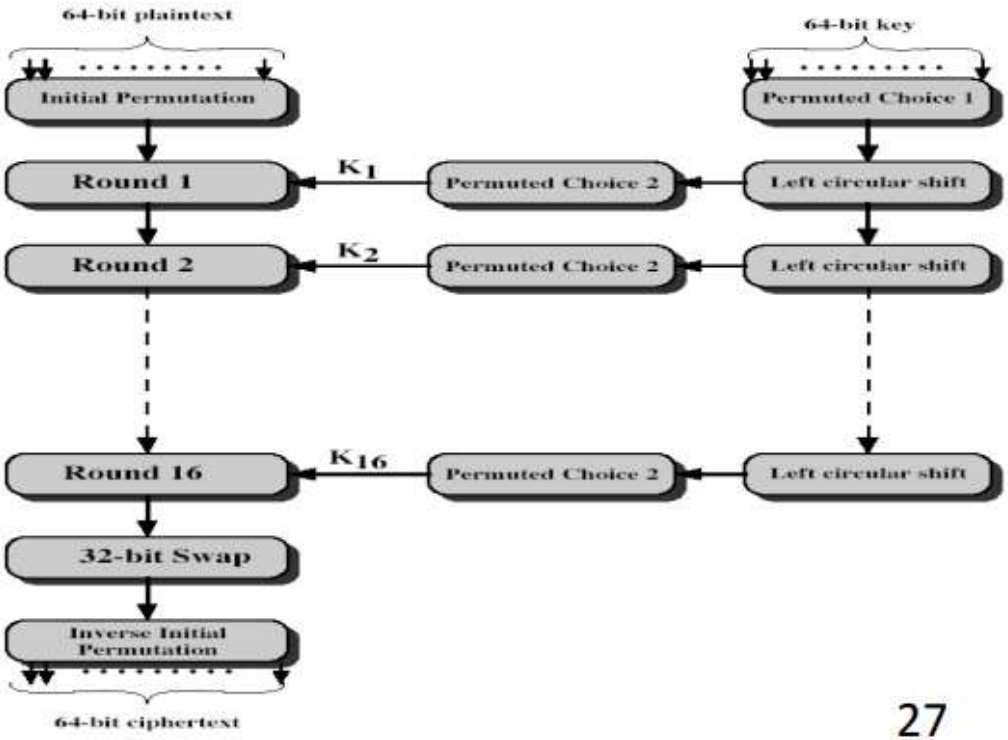


DATA ENCRYPTION STANDARD (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST) - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use- IBM developed Lucifer cipher - by team lead by Feistel – used 64-bit data blocks with 128-bit key then redeveloped as a commercial cipher with input from NSA and others
- In 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES DESIGN CONTROVERSY

- Although DES standard is public was considerable controversy over design -in choice of 56-bit key (vs Lucifer 128-bit)
- Subsequent events and public analysis show in fact design was appropriate
- DES has become widely used, especially in financial applications



INITIAL PERMUTATION IP

- first step of the data computation IP reorders the input data bits quite regular in structure example:
- IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

DES ROUND STRUCTURE

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as: Li = Ri–1 Ri = Li–1 xor F(Ri–1 , Ki)
- takes 32-bit R half and 48-bit subkey and: – expands R to 48-bits using Expansion Permutation E

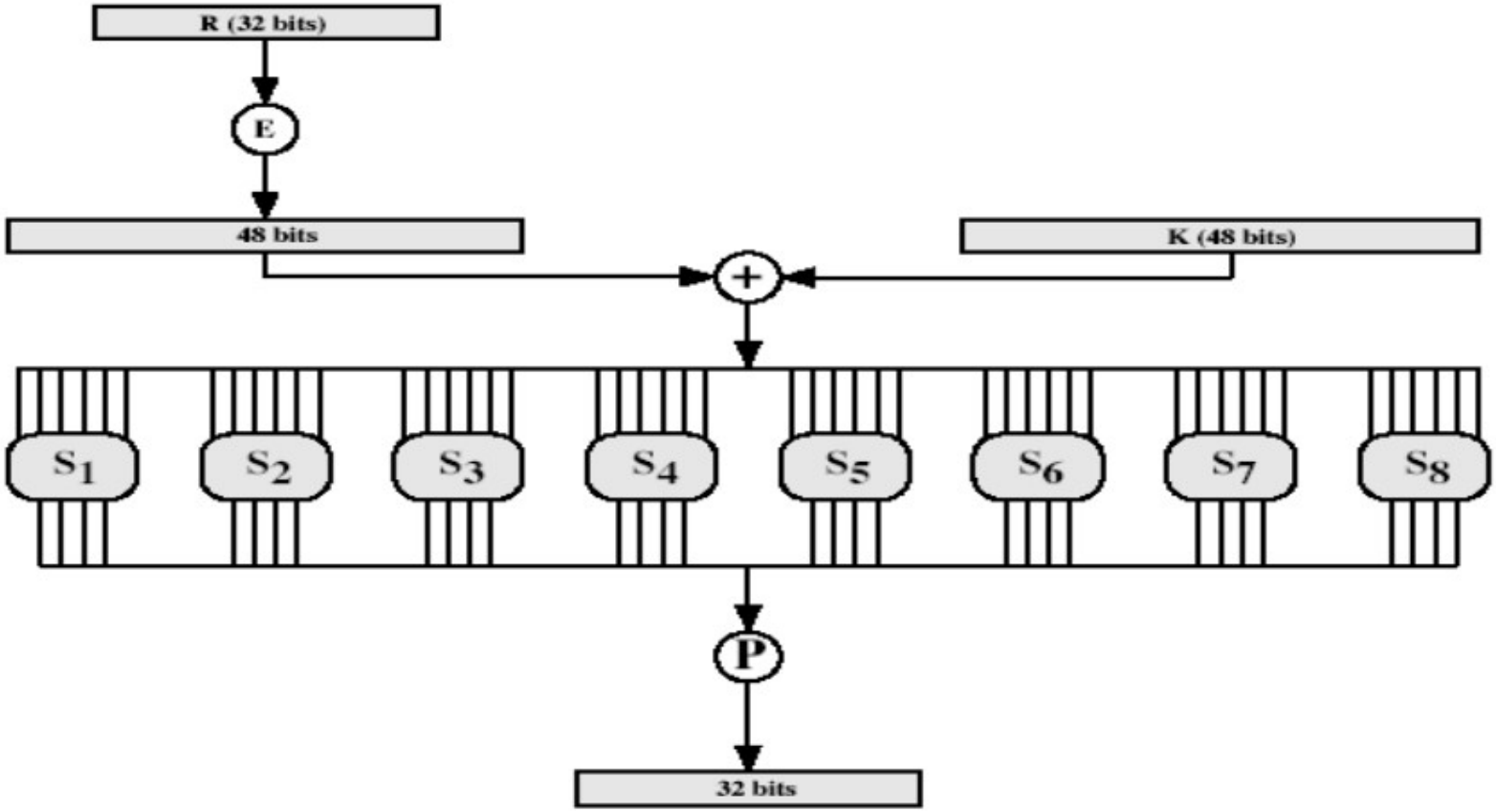
– adds to subkey

– passes through 8

S-boxes to get 32-bit result

– finally permutes this using 32-bit Permutation Function P

THE ROUND FUNCTION F(R,K)



SUBSTITUTION BOXES S

- 8 S-boxes
- Each S-Box mapps 6 to 4 bits
  - – outer bits 1 & 6 (row bits) select the row
  - – inner bits 2-5 (col bits) select the column
  - – For example, in S1, for input 011001,
  - • the row is 01 (row 1)
  - • the column is 1100 (column 12)
  - .• The value in row 1, column 12 is 9
  - • The output is 1001. • result is 8 X 4 bits, or 32 bits

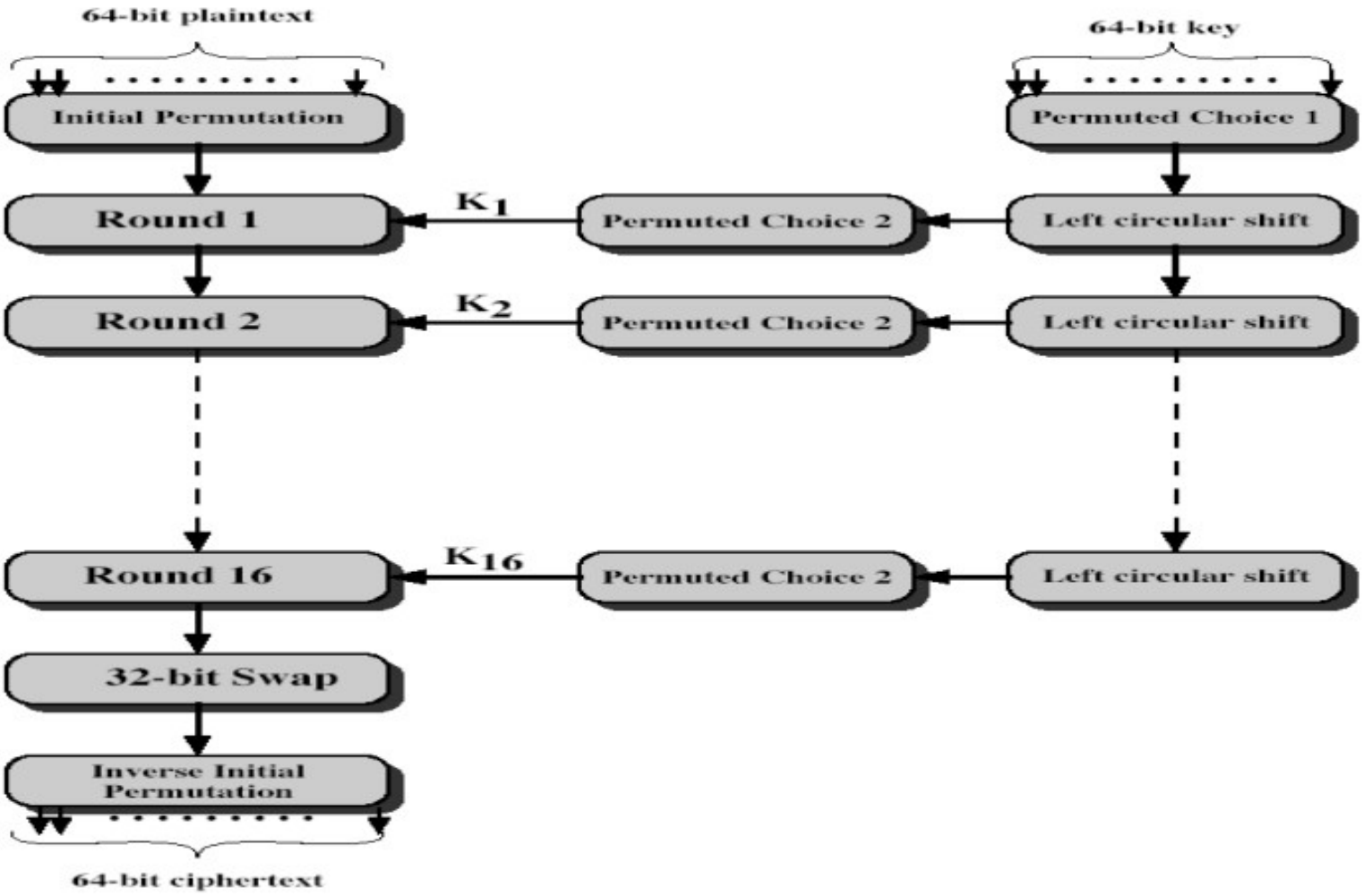
DES Key Schedule

- forms subkeys used in each round
- 1. initial permutation of the key PC1
- 2. divide the 56-bits in two 28-bit halves
- 3. at each round – 3.1. Left shift each half (28bits) separately either 1 or 2 places based on the left shift schedule
- Shifted values will be input for next round – 3.2. Combine two halfs to 56 bits, permuting them by PC2 for use in function f
- PC2 takes 56-bit input, outputs 48 bits

DES DECRYPTION

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using subkeys in reverse order (SK16 … SK1)
- note that IP undoes final FP step of encryption
- 1st round with SK16 undoes 16th encrypt round ….
- 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP
- thus recovering original data value

DES Decryption (Reverse encryption)



AVALANCHE EFFECT

- key desirable property of encryption alg
- DES exhibits strong avalanche
- where a change of one input or key bit results in changing approx half output bits

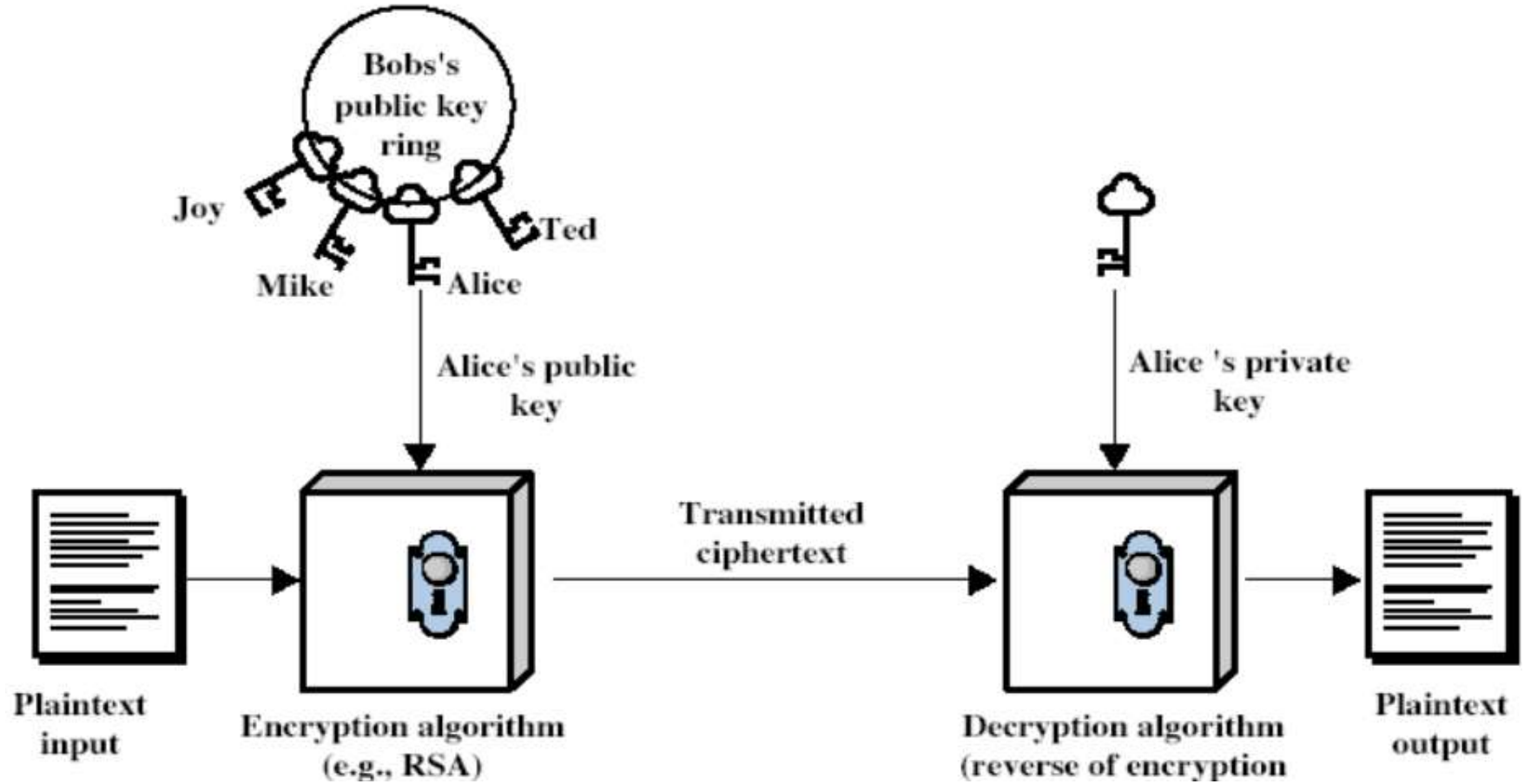
Principles of Public Cryptosystems

- Traditional private/secret/single-key cryptography uses one key shared by both sender and receiver.
- if this key is disclosed communications are compromised • also is symmetric, parties are equal hence does not protect sender from receiver forging a message & claiming it's sent by sender
- probably most significant advance in the 3000 year history of cryptography uses two keys – a public & a private key
- asymmetric since parties are not equal

Public-Key Cryptosystems

- Uses clever application of number theoretic concepts to function
- Complements rather than replaces private key crypto
- Public-key/two-key/asymmetric cryptography involves the use of two keys:
  – a public-key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
  – a private-key, known only to the recipient, used to decrypt messages, and sign (create) signatures
- is asymmetric because – those who encrypt messages or verify signatures cannot decrypt messages or create signatures

Public-Key Cryptosystems



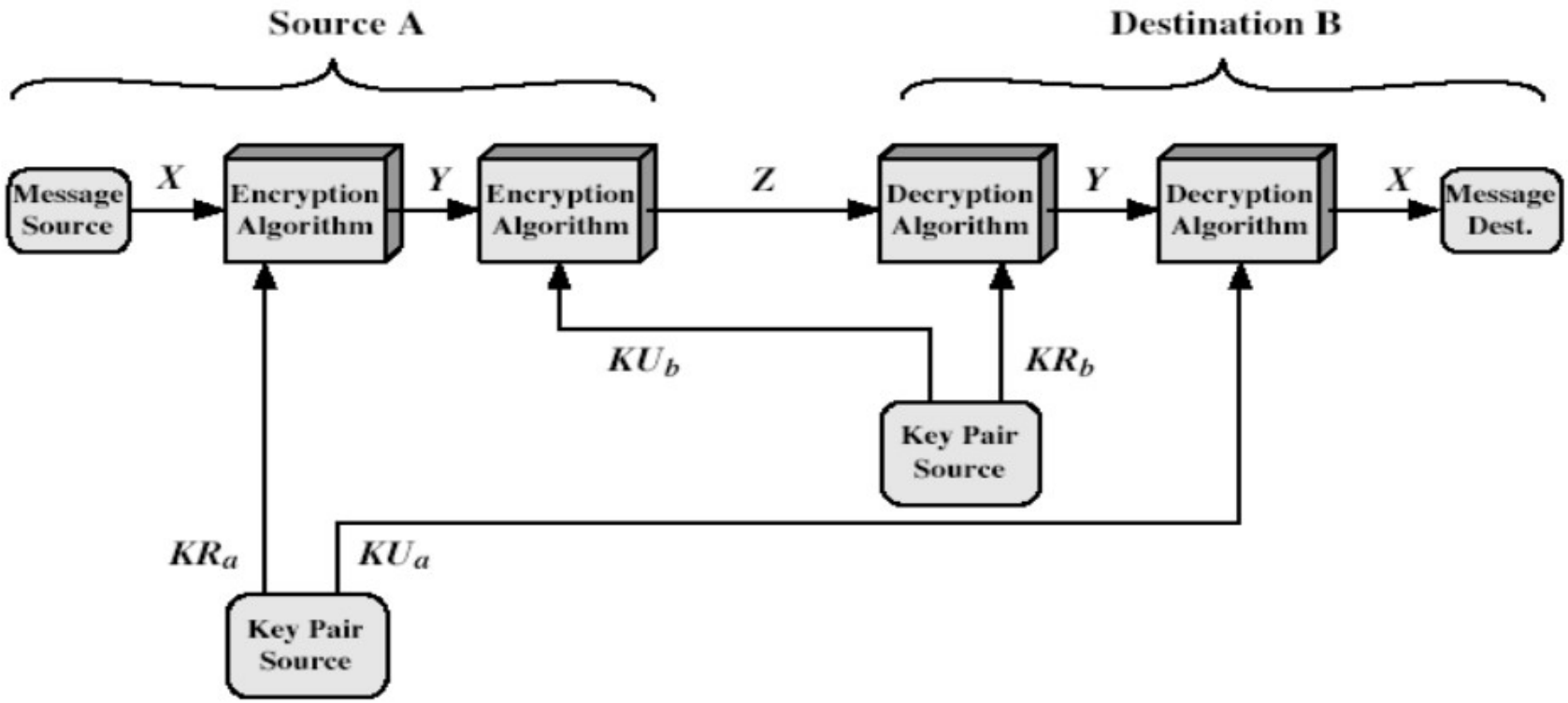
Applications for Public-Key Cryptosystems

- Developed to address two key issues:
  - – key distribution – how to have secure communications in general without having to trust a KDC with your key
  - – digital signatures – how to verify a message comes intact from the claimed sender
  - Public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976

Requirements for Public-Key Cryptosystems

- Public-Key algorithms rely on two keys with the characteristics that it is:
- – computationally infeasible to find decryption key knowing only algorithm & encryption key
- – computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
- – either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key Cryptanalysis



PUBLIC-KEY APPLICATIONS

- Can classify uses into 3 categories:
- –encryption/decryption (provide secrecy)
- –digital signatures (provide authentication)
- –key exchange (of session keys)
- some algorithms are suitable for all uses, others are specific to one

SECURITY OF PUBLIC KEY SCHEMES

- Like private key schemes brute force exhaustive search attack is always theoretically possible
- But keys used are too large (>512bits)
- Security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalyse) problems
- More generally the hard problem is known, its just made too hard to do in practise
- Requires the use of very large numbers hence is slow compared to private key schemes

Introduction RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime – nb. exponentiation takes O((log n)3) operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers – nb. factorization takes operations

$$O(e^{\log n \log \log n})$$

RSA Key Setup:

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus N=p.q
  - note ø(N)=(p-1)(q-1)
- selecting at random the encryption key e
  - where 1<e, gcd(e,ø(N))=1
- solve following equation to find decryption key d
  - e.d=1 mod ø(N) and 0≤d≤N
- publish their public encryption key: KU={e,N}
- keep secret private decryption key: KR={d,p,q}

RSA Use :

- to encrypt a message M the sender:
  - obtains public key of recipient KU={e,N}
  - computes: C=Me mod N, where 0≤M
- to decrypt the ciphertext C the owner:
  - uses their private key KR={d,p,q}
  - computes: M=Cd mod N
- note that the message M must be smaller than the modulus N (block if needed)

RSA Use :

- to encrypt a message M the sender:
  - obtains public key of recipient KU={e,N}
  - computes: C=Me mod N, where 0≤M
- to decrypt the ciphertext C the owner:
  - uses their private key KR={d,p,q}
  - computes: M=Cd mod N
- note that the message M must be smaller than the modulus N (block if needed)

RSA Example :

1. Select primes: p=17 & q=11

2. Compute n = pq =17 11=187

3. Compute ø(n)=(p–1)(q-1)=16 10=160

4. Select e : gcd(e,160)=1; choose e=7

5. Determine d: de=1 mod 160 and d < 160 Value is d=23 since 23 7=161=

10 160+1

6. Publish public key KU={7,187}

7. Keep secret private key KR={23,17,11}

RSA Example (cotd...)

sample RSA encryption/decryption is:

given message M = 88 (nb. 88<187)

Encryption:

$$C = 88^7 \bmod 187 = 11$$

Decryption:

$$M = 11^{23} \bmod 187 = 88$$

RSA KEY GENERATION

- users of RSA must:
  - determine two primes at random - p, q
  - select either e or d and compute the other
- primes p,q must not be easily derived from modulus N=p.q
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents e, d are inverses, so use Inverse algorithm to compute the other

SECURITY OF RSA:

- How to attack RSA?
  - we have public key (n,e)
  - compute ϕ (n) and get d
- Easier said than done!!
  - If we have n and ϕ(n) then we can factor n
  - If we have e and d then we can factor n

Various attacks on RSA :

Five possible approaches to attacking the RSA algorithm are

- Brute force: This involves trying all possible private keys.
- Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.
- Timing attacks: These depend on the running time of the decryption algorithm. • Hardware fault-based attack: This involves inducing hardware faults in the processor that is generating digital signatures.
- Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

Various attacks on RSA :

The Factoring Problem

We can identify three approaches to attacking RSA mathematically.

1. Factor n into its two prime factors. This enables calculation of f(n) = (p - 1) * (q - 1), which in turn enables determination of d K e-1 (mod f(n)).
2. Determine f(n) directly, without first determining p and q. Again, this enables determination of d K e-1 (mod f(n)).
3. Determine d directly, without first determining f(n).

Various attacks on RSA :

A **timing attack** is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.

We can explain the attack using the modular exponentiation algorithm, but the attack can be adapted to work with any implementation that does not run in fixed time.

In this algorithm, modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit.

•Although the timing attack is a serious threat, there are simple countermeasures that can be used, including the following.

•Constant exponentiation time: Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.

•Random delay: Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. Kocher points out that if defenders don't add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays.

•Blinding: Multiply the ciphertext by a random number before performing exponentiation.

•This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

RSA algorithm is vulnerable to a **chosen ciphertext attack** (CCA).

•CCA is defined as an attack in which the adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key.

•Thus, the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key.

•Clearly, this provides the adversary with no new information.

•Instead, the adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis.

RSA algorithm is vulnerable to a **chosen ciphertext attack** (CCA).

A simple example of a CCA against RSA takes advantage of the following property of RSA:

$$\mathrm{E}(PU, M_1) \times \mathrm{E}(PU, M_2) = \mathrm{E}(PU, [M_1 \times M_2])$$

We can decrypt C = Me mod n using a CCA as follows.

1. Compute $X = (C \times 2^e) \bmod n.$
2. Submit X as a chosen ciphertext and receive back $Y = X^d \bmod n.$

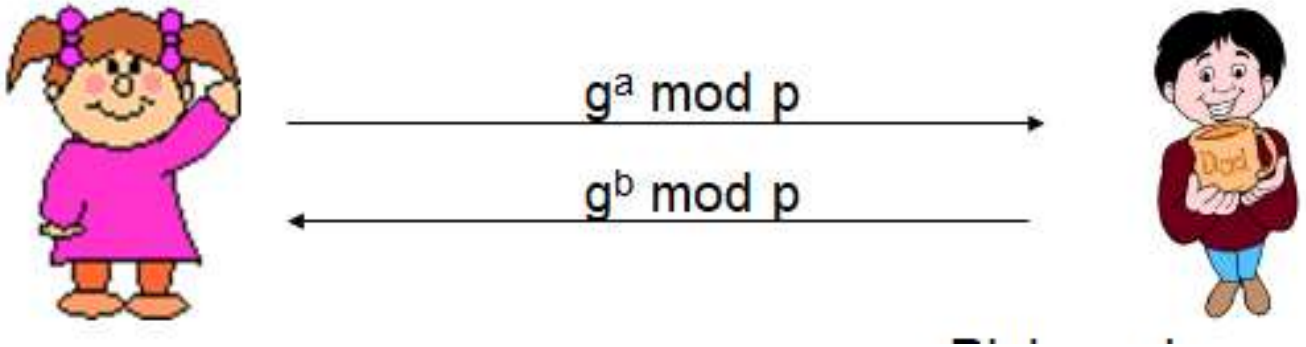
   But now note that

$$\begin{aligned} X &= (C \bmod n) \times (2^e \bmod n) \\ &= (M^e \bmod n) \times (2^e \bmod n) \\ &= (2M)^e \bmod n \end{aligned}$$

Therefore, Y = (2M) mod n. From this, we can deduce M.

Other Public key Cryptosystems: Diffie-Hellman Key Exchange.

- Not a Public Key Encryption system, but can allow A and B to agree on a shared secret in a public channel (against passive, i.e., eavesdropping only adversaries)
- Setup: p prime and g generator of Zp*, p and g public.



Pick random, secret a

Compute and send g^a mod p

$K = (g^b \bmod p)^a = g^{ab} \bmod p$

Pick random, secret b

Compute and send g^b mod p

$K = (g^a \bmod p)^b = g^{ab} \bmod p$

Diffie-Hellman Algorithm

Example: Let p=11, g=2, then

a	1	2	3	4	5	6	7	8	9	10	11
g^a	2	4	8	16	32	64	128	256	512	1024	2048
g^a mod p	2	4	8	5	10	9	7	3	6	1	2

A chooses 4, B chooses 3, then shared secret is $(2^3)^4 = (2^4)^3 = 2^{12} = 4 \pmod{11}$

Adversaries sees $2^3=8$ and $2^4=5$, needs to solve one of $2^x=8$ and $2^y=5$ to figure out the shared secret.