RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Cryptography and Network Security

Course Code - CSE643

Credits - 3:0:0

UNIT -3

Term: Oct2024 – Jan 2025

Outline

Encipherment using Modern Symmetric-Key Ciphers: (Text 1: Chapter 8)

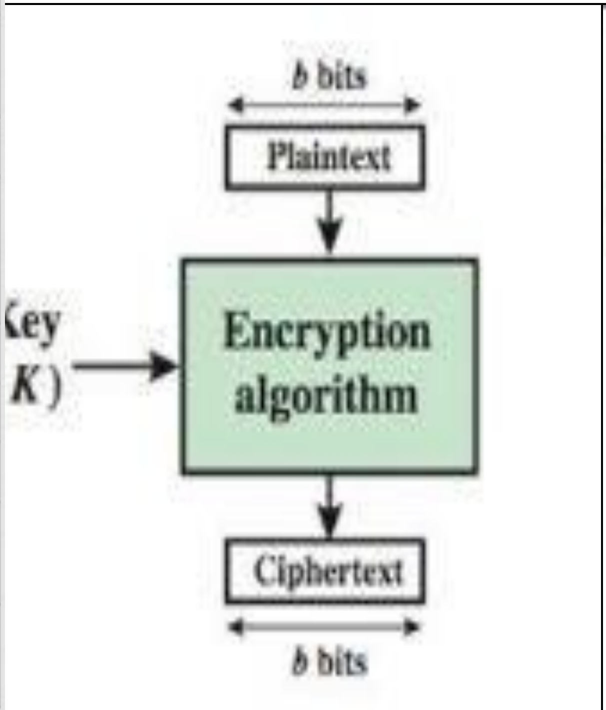
- Use of Modern Block Ciphers
- Use of Stream Ciphers
- Other Issues.

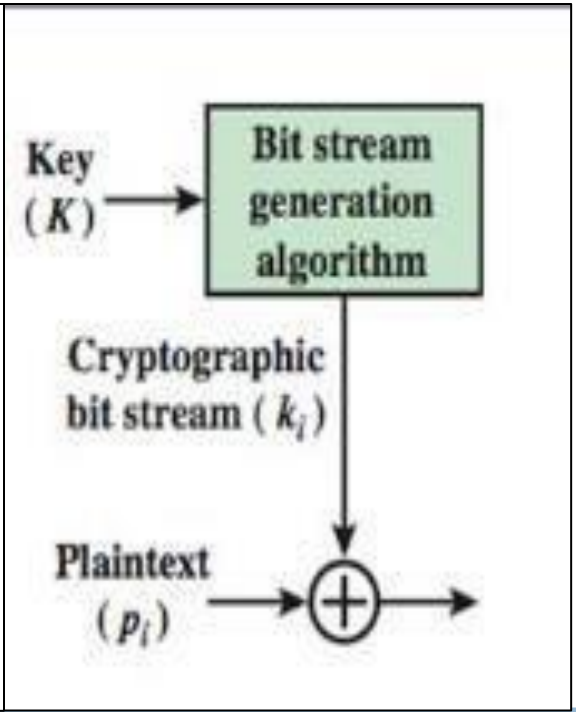
Asymmetric Key Cryptography: (Text1: Chapter 10)

- Introduction
- RSA Cryptosystem
- Rabin Cryptosystem
- Elgamal Cryptosystem

Blockcipher and Stream cipher

Stream Cipher	Block Cipher
Stream cipher operates on smaller Units of Plaintext	Block cipher operates on larger block of data
Faster than block cipher	Slower than Stream Cipher
Stream cipher processes the input element continuously producing output one element at a time	Block cipher processes the input one block of element at a time, producing an output block for each input block
Require less code	Requires more code
Only one time of key used.	Reuse of key is possible
Ex: One time pad	Ex: DES (Data Encryption Standard)
Application: SSL (secure connection on the web)	Application: Database, file encryption.
Stream cipher is more suitable for hardware implementation	Easier to implement in software.

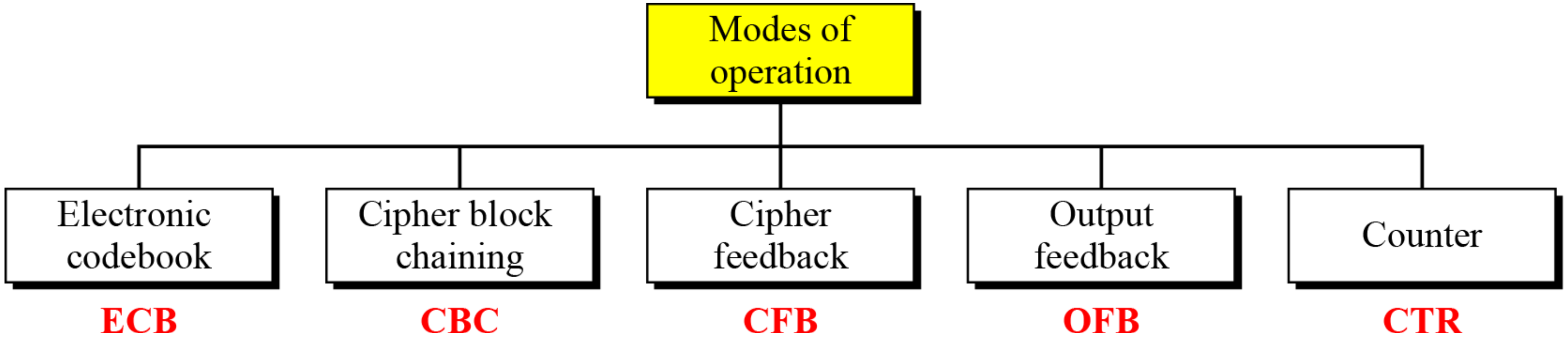




Use of Modern Block Ciphers

- Symmetric-key encipherment can be done using modern block ciphers.
- DES (64bit block) and AES (128bit block)
- Modes of operation have been devised to encipher text of any size employing either DES or AES

Figure : Modes of operation



Electronic Codebook (ECB) Mode

- ECB Mode of Operation
- Properties - Error Propagation

  - Security Issues
- Algorithm
- Ciphertext Stealing
- Applications

Electronic Codebook (ECB) Mode

- It is one of the simplest modes of operation.
- In this mode, the plain text is divided into a block where each block is 64 bits.
- Then each block is encrypted separately.
- The same key is used for the encryption of all blocks.
- Each block is encrypted using the key and makes the block of ciphertext.
- The same key which is used for encryption is used for decryption.
- It takes the 64-bit ciphertext and, by using the key convert the ciphertext into plain text.

Electronic Codebook (ECB) Mode

- As the same key is used for all blocks’ encryption, if the block of plain text is repeated in the original message, then the ciphertext’s corresponding block will also repeat.

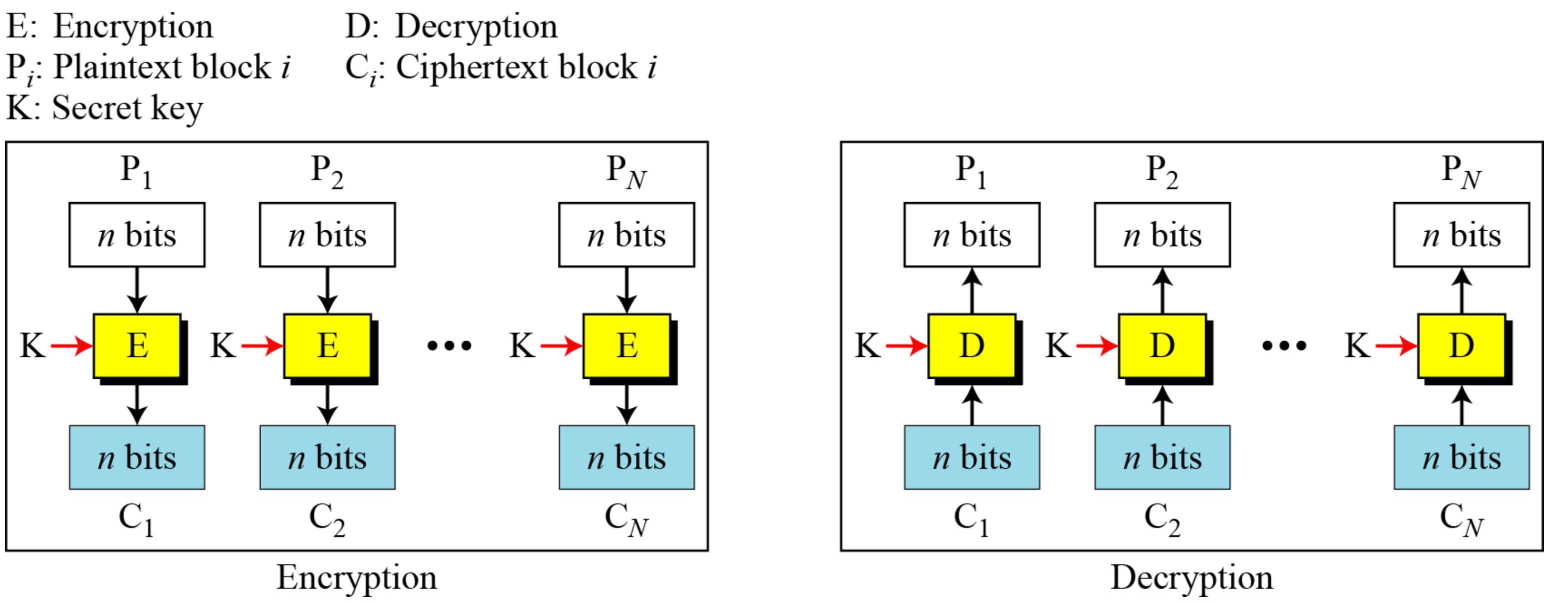
- As the same key used for all block, to avoid the repetition of block ECB mode is used for an only small message where the repetition of the plain text block is less.

Electronic Codebook (ECB) Mode

The relationship between Plaintext and Ciphertext

Encryption: $C_i = E_K(P_i)$ Decryption: $P_i = D_K(C_i)$

Figure : Electronic codebook (ECB) mode



Security Issues

1 Same blocks encrypts to the same ciphertext

2The block independency creates opportunities for Eve to exchange some ciphertext blocks without knowing the key.

Error Propagation

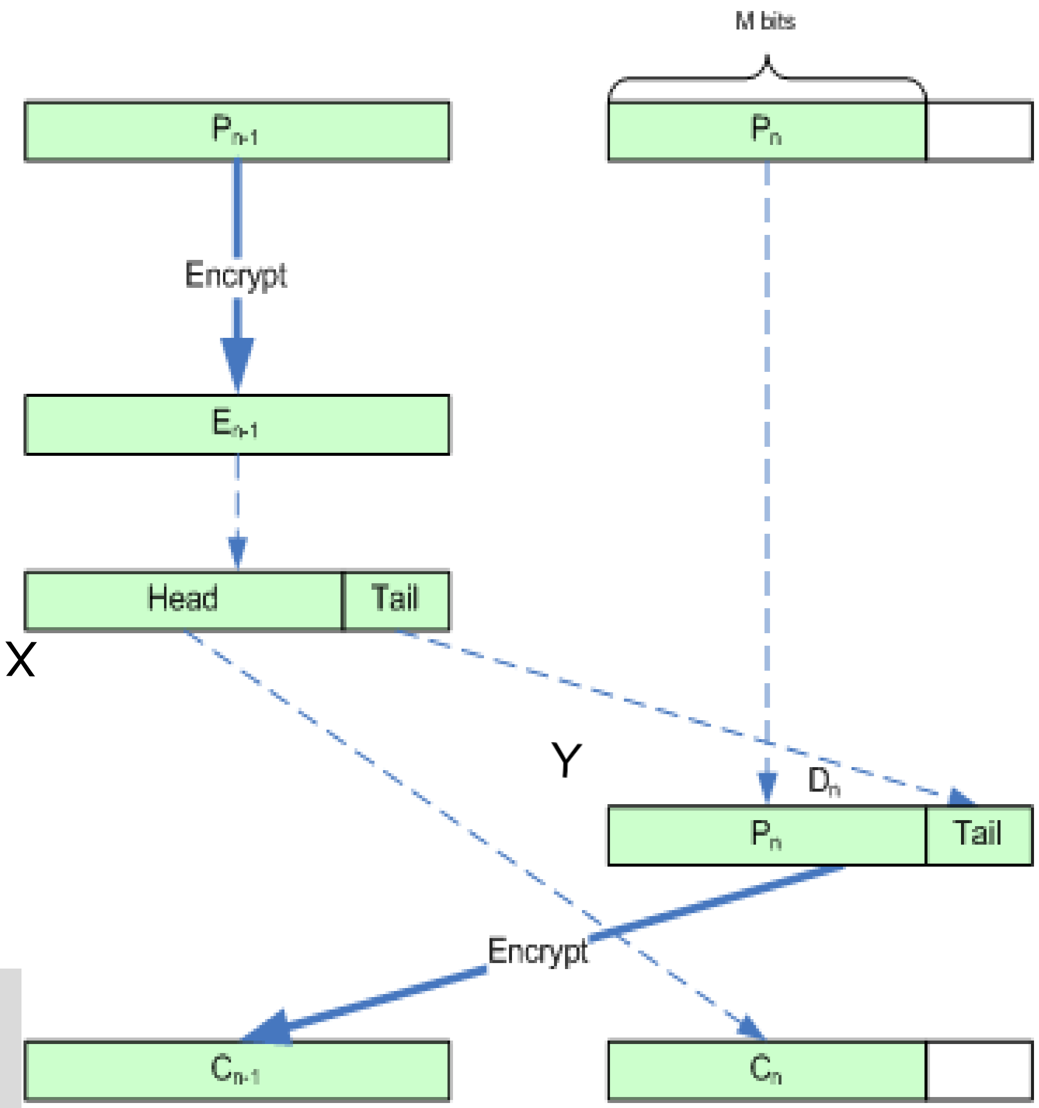
A single bit error in transmission can create errors in several in the corresponding block. However, the error does not have any effect on the other blocks.

RAMAIAH Institute of Technology

Ciphertext Stealing

A technique called ciphertext stealing (CTS) can make it possible to use ECB mode without padding.

In this technique the last two plaintext blocks, P_{N-1} and P_N, are encrypted differently and out of order, as shown below, assuming that P_{N-1} has n bits and P_N has m bits, where $m \leq n$.



$$X = E_K(P_{N-1}) \rightarrow C_N = head_m(X)$$

$$Y = P_N \mid tail_{n-m}(X) \rightarrow C_{N-1} = E_K(Y)$$

Applications

- The ECB mode is not recommended for encryption of messages more than one block.
- Access to the database can be random.
- Use parallel processing if we need to create a very huge encrypted database.

Cipher Block Chaining (CBC) Mode

In CBC mode, each plaintext block is exclusive-ored with the previous ciphertext block before being encrypted.

Encryption:	**Decryption:**
$C_0 = IV$	$C_0 = IV$
$C_i = E_K(P_i \oplus C_{i-1})$	$P_i = D_K(C_i) \oplus C_{i-1}$

Initialization Vector (IV)

The initialization vector (IV) should be known by the sender and the receiver.

Cipher Block Chaining (CBC) Mode

- At the sender side, the plain text is divided into blocks.
- In this mode, IV(Initialization Vector) is used, which can be a random block of text.
- IV is used to make the ciphertext of each block unique.
- The first block of plain text and IV is combined using the XOR operation and then encrypted the resultant message using the key and form the first block of ciphertext.
- The first block of ciphertext is used as IV for the second block of plain text. The same procedure will be followed for all blocks of plain text.

RAMAIAH Institute of Technology

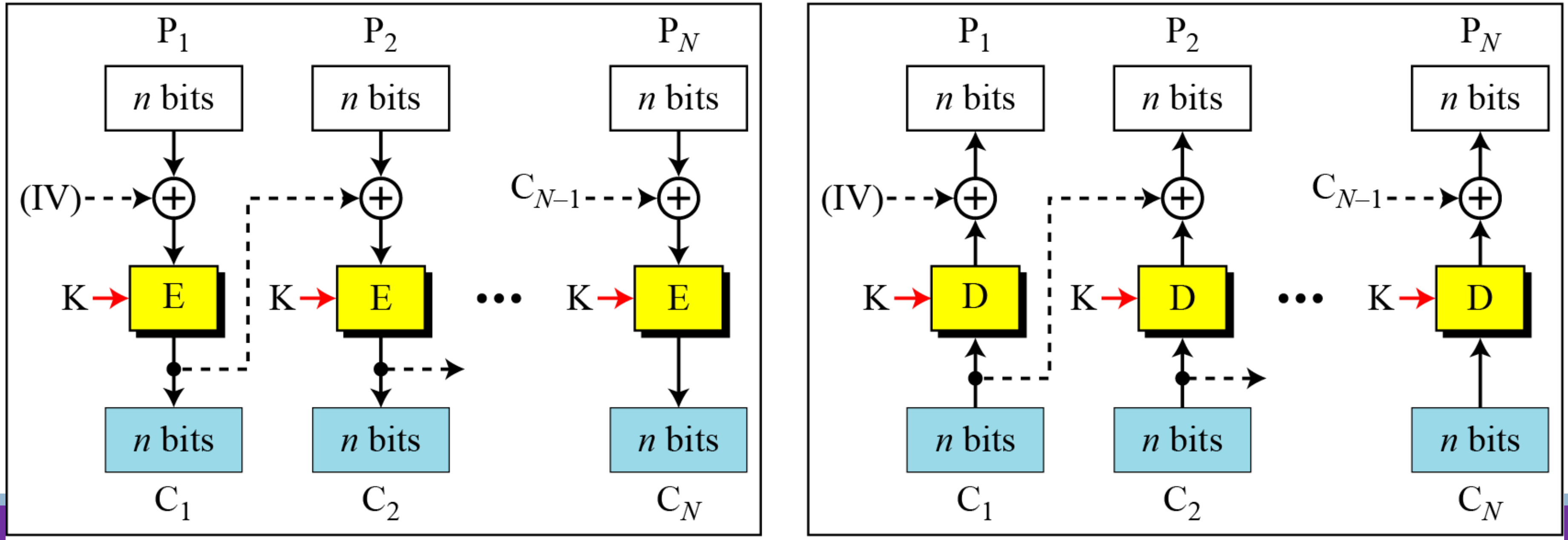
Cipher Block Chaining (CBC) Mode

Encryption:	**Decryption:**
$C_0 = IV$	$C_0 = IV$
$C_i = E_K(P_i \oplus C_{i-1})$	$P_i = D_K(C_i) \oplus C_{i-1}$

E: Encryption
P_i: Plaintext block i
K: Secret key

D : Decryption
C_i : Ciphertext block i
IV: Initial vector (C_0)

Figure: Cipher block chaining (CBC) mode



Encryption

Decryption

RAMAIAH
Institute of Technology

Security Issues

- Patterns at the block level are not preserved. However, if the first M blocks in two different messages are equal, they are enciphered into equal blocks unless different IVs are used. Hence, recommend the use of timestamp as an IV.
- Eve can add some ciphertext blocks to the end of the ciphertext stream.

Error Propagation

In CBC mode, a single bit error in ciphertext block C_j during transmission may create error in most bits in plaintext block P_j during decryption.

Disadvantages

- Parallel processing is not possible.
- CBC mode is not used to encrypt and decrypt random-access files records because of the need to access the previous records.

Applications

- CBC mode is used for authentication.

Ciphertext Stealing

The ciphertext stealing technique described for ECB mode can also be applied to CBC mode, as shown below.

$$
\begin{array}{lllll}
U = P_{N-1} \oplus C_{N-2} & \rightarrow & X = E_K(U) & \rightarrow & C_N = head_m(X) \\
V = P_N \mid pad_{n-m}(0) & \rightarrow & Y = X \oplus V & \rightarrow & C_{N-1} = E_K(Y)
\end{array}
$$

The head function is the same as described in ECB mode; the pad function inserts 0's.

Cipher Feedback (CFB) Mode

- In this mode, the data is encrypted in the form of units where each unit is of 8 bits.
- Like cipher block chaining mode, IV is initialized.
- The IV is kept in the shift register.
- It is encrypted using the key and form the ciphertext.

Cipher Feedback (CFB) Mode

- Now the leftmost j bits of the encrypted IV is XOR with the plain text's first j bits. This process will form the first part of the ciphertext, and this ciphertext will be transmitted to the receiver.

- Now the bits of IV is shifted left by j bit. Therefore the rightmost j position of the shift register now has unpredictable data.

- These rightmost j positions are now filed with the ciphertext. The process will be repeated for all plain text units.

Cipher Feedback (CFB) Mode

In some situations, we need to use DES or AES as secure ciphers, but the plaintext or ciphertext block sizes are to be smaller.

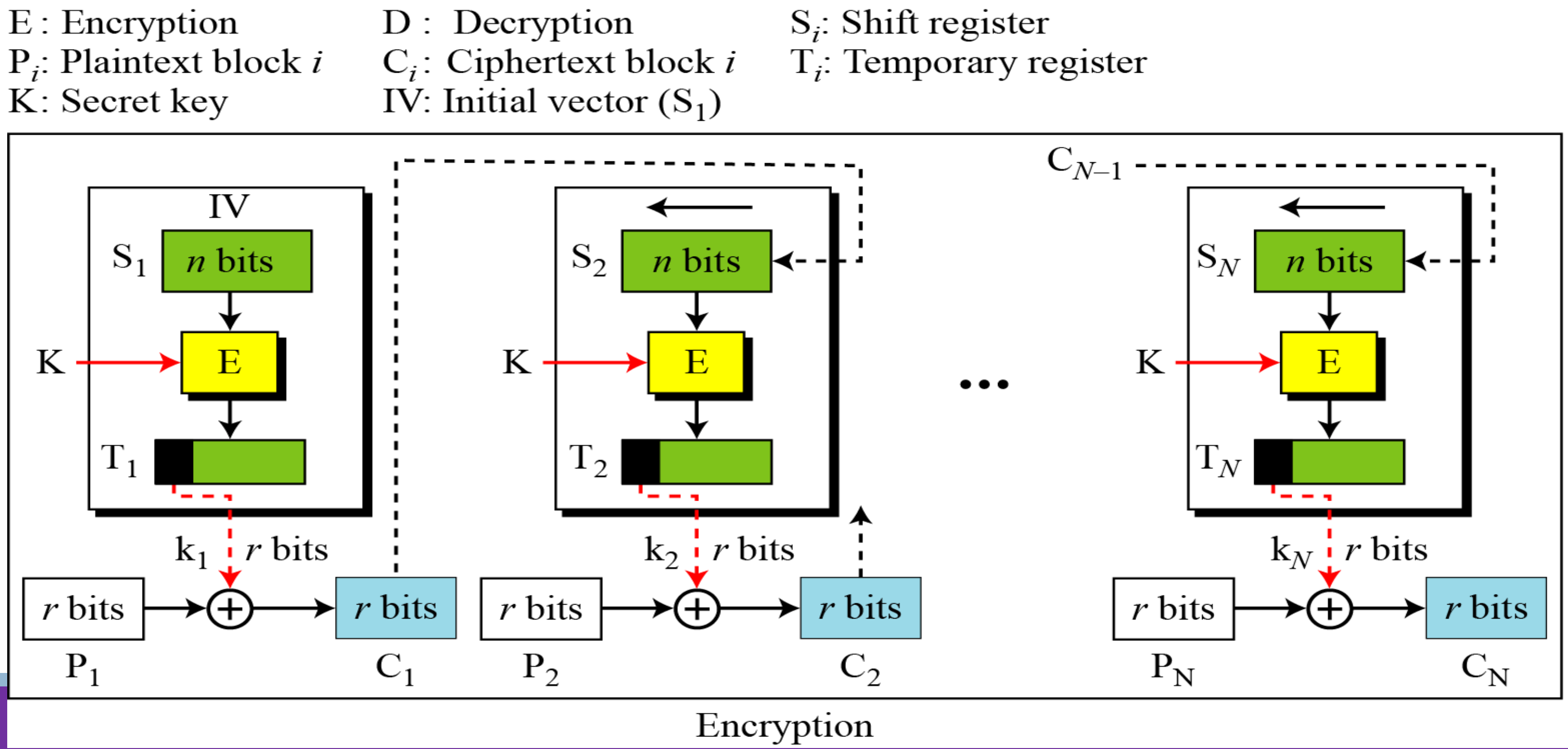


Figure
Encryption in cipher feedback (CFB) mode

Note

In CFB mode, encipherment and decipherment use the encryption function of the underlying block cipher.

The relation between plaintext and ciphertext blocks is shown below:

$$\textbf{Encryption: } C_i = P_i \oplus \text{SelectLeft}_r\{E_K[\text{ShiftLeft}_r(S_{i-1}) \mid C_{i-1})]\}$$
$$\textbf{Decryption: } P_i = C_i \oplus \text{SelectLeft}_r\{E_K[\text{ShiftLeft}_r(S_{i-1}) \mid C_{i-1})]\}$$

Advantages

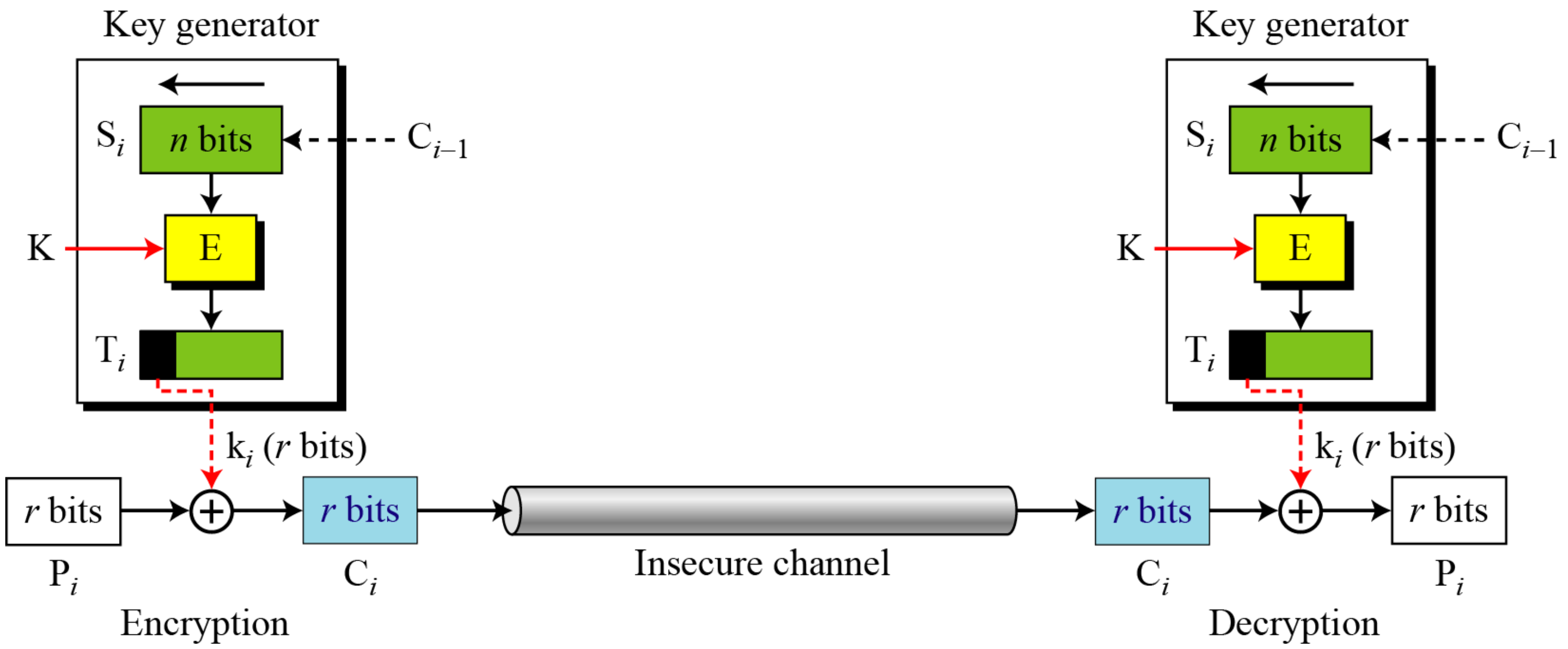
- This mode does not need padding because the size of the block r, is normally chosen to fit the data unit to be encrypted (a character for example).
- The system does not have to wait until It has received a large block of data (64 or 128 bits) before starting the encryption.

Disadvantages

CFB is less efficient than CBC and ECB because it needs to apply the encryption function for each small block of size r.

CFB as a Stream Cipher

Figure: **Cipher feedback (CFB) mode as a stream cipher**



RAMAIAH
Institute of Technology

Security Issues

- The patterns are not preserved.
- The IV should be changed for each message
- Eve can add some ciphertext block to the end of the ciphertext stream.

Error Propagation

A single bit error in ciphertext block Ci during transmission creates a single bit eror in plaintext block Pi. However most of the bits in the following plaintext blocks are in error.

Application

This mode can be used to encipher blocks of small size such as characters or bit at a time.

Output Feedback (OFB) Mode

- In OFB mode, ciphertext is used for the next stage of the encryption process, whereas in OFB, the output of the IV encryption is used for the next stage of the encryption process.
- The IV is encrypted using the key.
- Plain text and leftmost 8 bits of encrypted IV are combined using XOR and produce the ciphertext.
- For the next stage, the ciphertext, which is the form in the previous stage, is used as an IV for the next iteration. The same procedure is followed for all blocks.

Output Feedback (OFB) Mode

In this mode each bit in the ciphertext is independent of the previous bit or bits. This avoids error propagation.

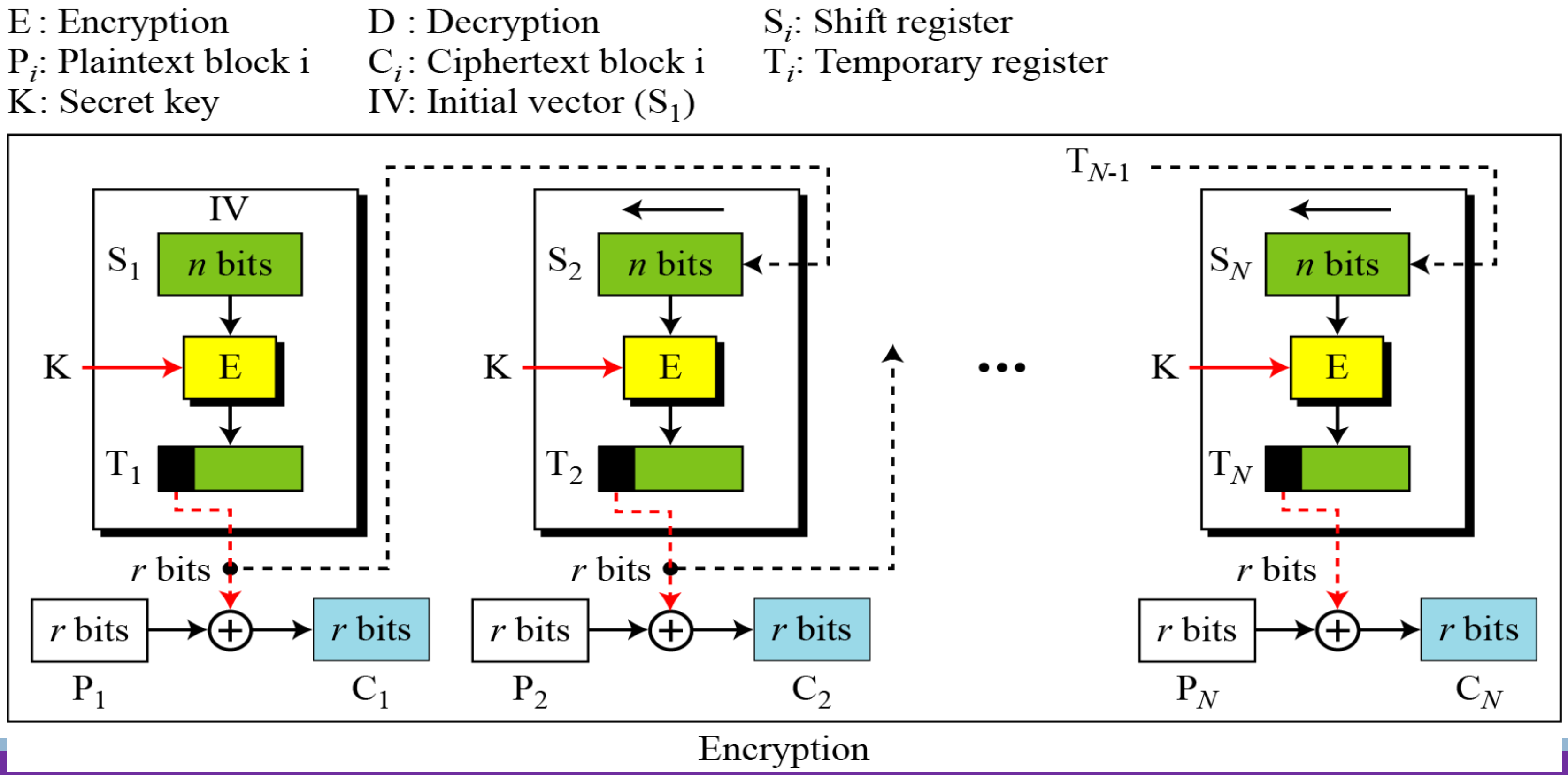
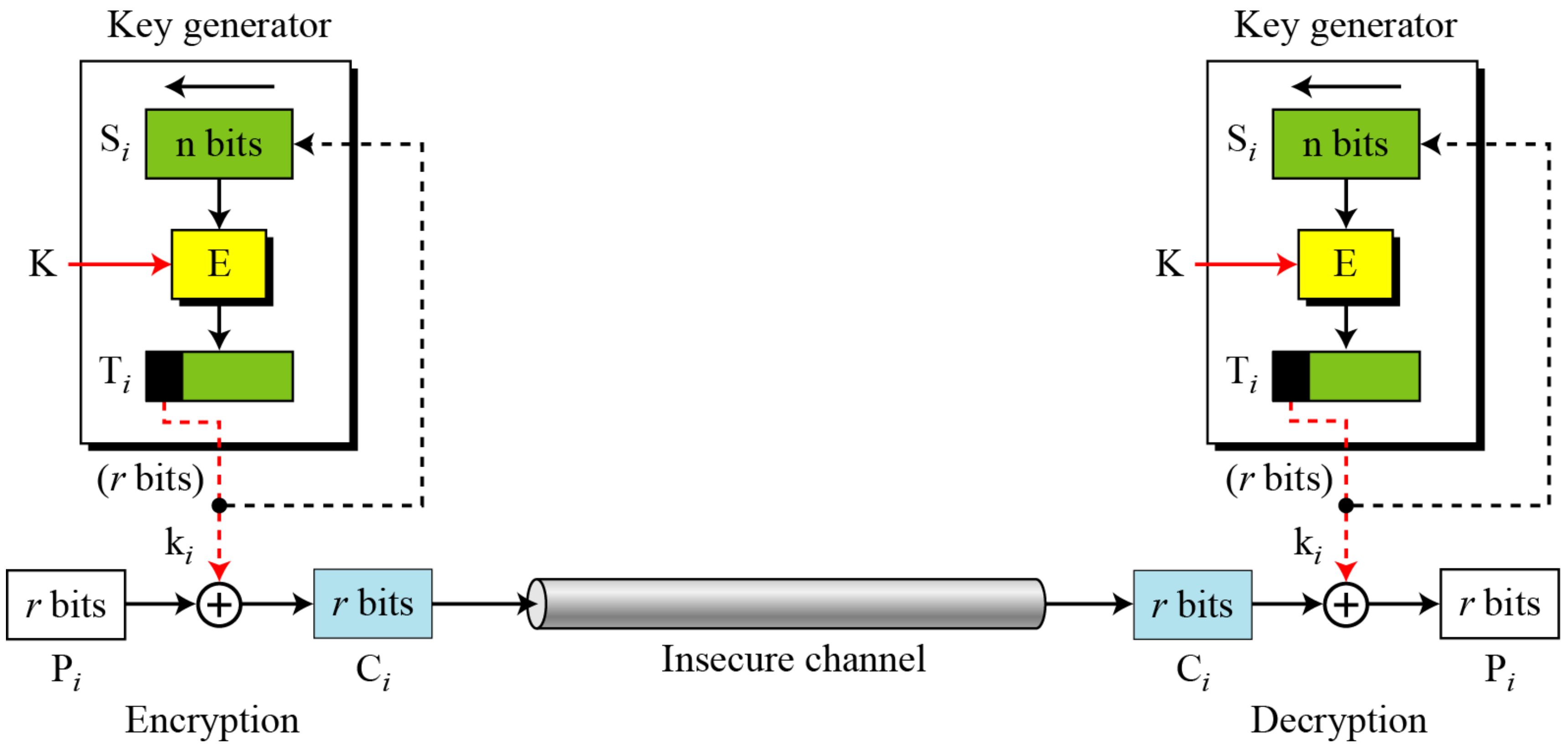


Figure
Encryption in output feedback (OFB) mode

OFB as a Stream Cipher

Figure :Output feedback (OFB) mode as a stream cipher



Security Issues

The patterns are not preserved.

Error Propagation

A single bit error in the ciphertext affects only the corresponding bit in the plaintext.

Counter (CTR) Mode

- Every time a counter-initiated value is encrypted and given as input to XOR with plaintext which results in ciphertext block.

- The CTR mode is independent of feedback use and thus can be implemented in parallel.

Note: the counter value will be incremented by 1.

Counter (CTR) Mode

- The counter will be incremented by 1 for the next stage, and the same procedure will be followed for all blocks.

- For decryption, the same sequence will be used. Here to convert ciphertext into plain text, each ciphertext is XOR with the encrypted counter.

- For the next stage, the counter will be incremented by the same will be repeated for all Ciphertext blocks.

Counter (CTR) Mode

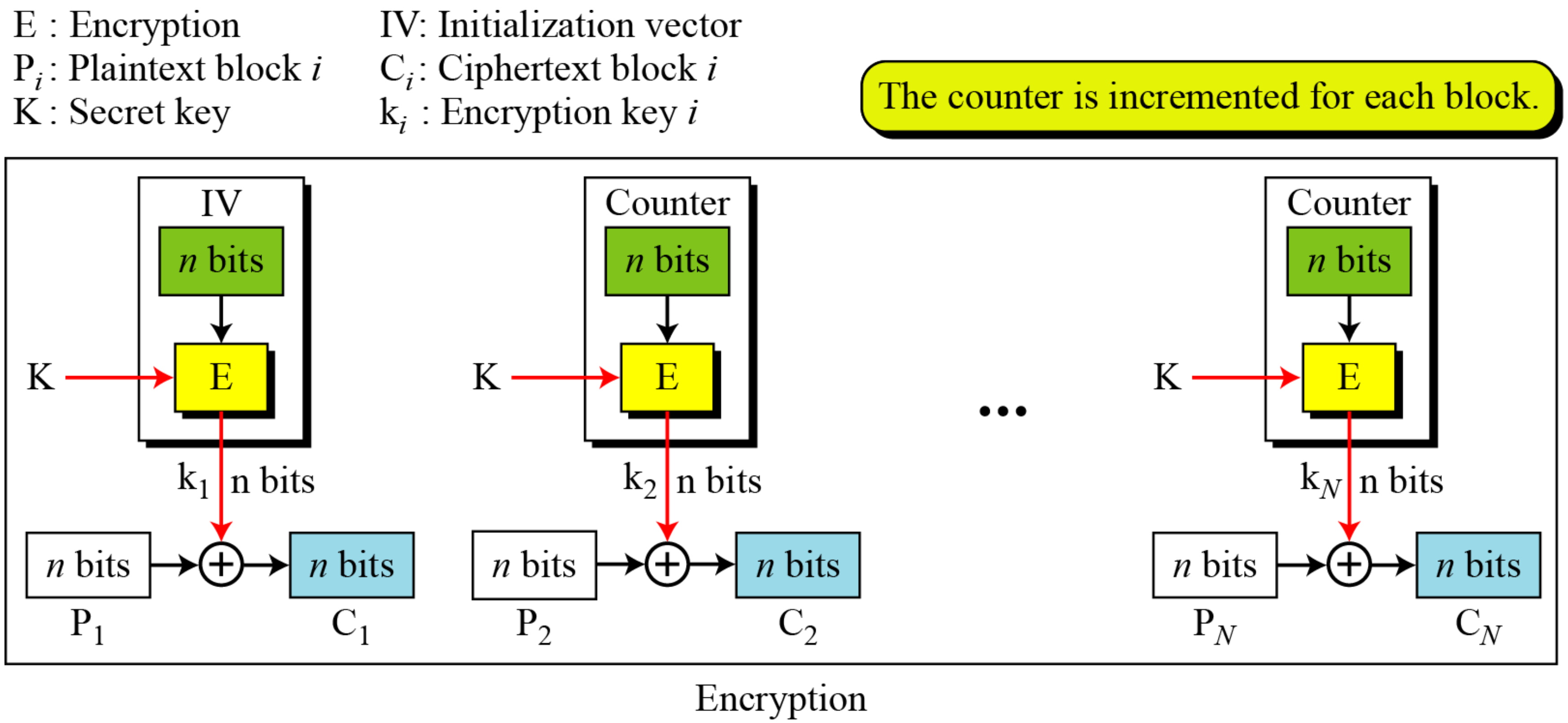
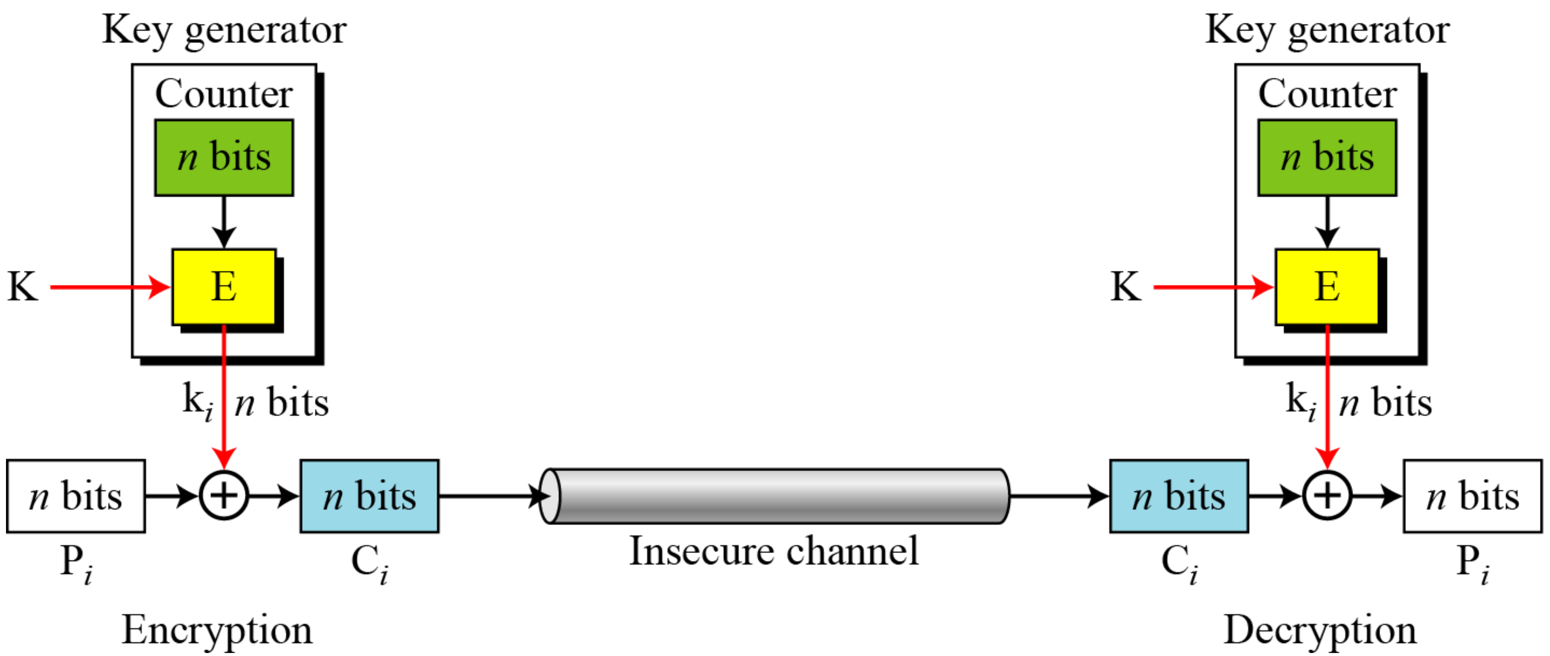


Figure
Encryption in counter (CTR) mode

RAMAIAH Institute of Technology

Figure : **Counter (CTR) mode as a stream cipher**



Advantages

- CTR creates n-bit blocks that are independent from each other; they depend only on the value of the counter.
- CTR cannot be used for real- time processing.
- CTR can be used to encrypt and decrypt random access files as long as the value of the counter can be related to the record number in the file.

Comparison of Different Modes

Table 8.1 *Summary of operation modes*

Operation Mode	*Description*	*Type of Result*	*Data Unit Size*
ECB	Each n-bit block is encrypted independently with the same cipher key.	Block cipher	n
CBC	Same as ECB, but each block is first exclusive-ored with the previous ciphertext.	Block cipher	n
CFB	Each r-bit block is exclusive-ored with an r-bit key, which is part of previous cipher text	Stream cipher	$r \leq n$
OFB	Same as CFB, but the shift register is updated by the previous r-bit key.	Stream cipher	$r \leq n$
CTR	Same as OFB, but a counter is used instead of a shift register.	Stream cipher	n

Use of Stream Ciphers

Although the five modes of operations enable the use of block ciphers for encipherment of messages or files in large units and small units,

But sometimes pure stream are needed for enciphering small units of data such as characters or bits.

- **RC4**
- **A5/1**

Rivest Cipher 4 (RC4)

- RC4 invented by Ron Rivest in 1987 for RSA Security.
- RC4 is byte oriented stream cipher, in which a byte is exclusive-ored with a byte of key to produce a byte of cipher
- The secret key, from which one byte keys in the key stream are generated, can contain from 1 to 256 bytes.
- The algorithm operates on a user-selected variable-length key(K) of 1 to 256 bytes (8 to 2048 bits)
- Most widely used stream ciphers because of its simplicity and speed of operation.
- It is generally used in applications such as Secure Socket Layer (SSL), Transport Layer Security (TLS), and also used in IEEE 802.11 wireless LAN std.

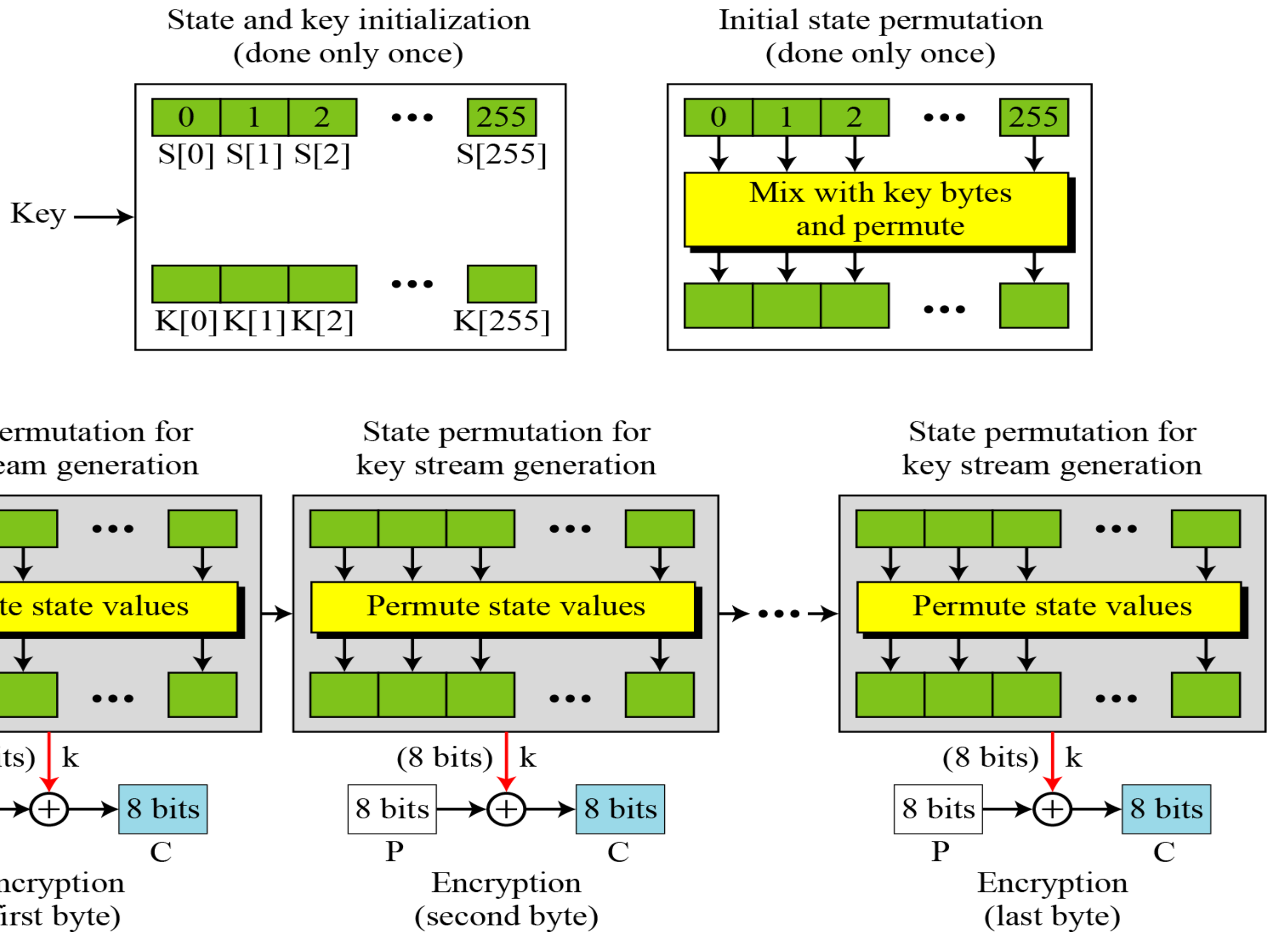
Rivest Cipher 4 (RC4)

State

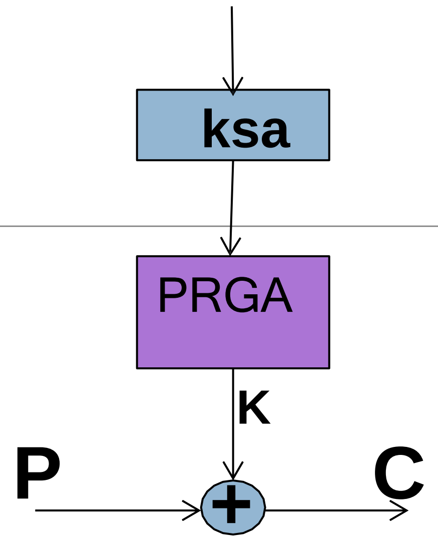
- RC4 is based on the concept of a state.
- At each moment, a state of 256byte is active, from which one of the byte is randomly selected to serve as key for encryption.
- Shown as an array of bytes

S[0]	S[1]	S[2]	⋯	S[255]

Rivest Cipher 4 (RC4)



Rivest Cipher 4 (RC4)



RC4 has two main parts:

- **KSA (Key Scheduling Algorithm)**
- **PRGA (Pseudo Random Generation Algorithm)**

KSA: A key-scheduling algorithm initializes the process in an array typically referred to as "S." That "S" is processed 256 times, and bytes from the key are mixed in too.

PRGA: Data is fed in byte by byte, and a mathematical model modifies it. The model looks up values, add them to 256, and uses the sum as the byte within the keystream. It swaps each element with another at least once every 256 rounds.

Rivest Cipher 4 (RC4)

Initialization: Done in two steps

The first step

```
for (i = 0 to 255)
{
    S[i]  ← i
    K[i]  ← Key [i mod KeyLength]
}
```

The second step

```
j ← 0
for (i = 0 to 255)
{
    j ← (j + S[i] + K[i]) mod 256
    swap (S[i] , S[j])
}
```

Rivest Cipher 4 (RC4)

Key stream Generation

The keys in the key stream are generated, one by one.

i and j are initialized to 0.

$$
\begin{aligned}
&i \leftarrow (i + 1) \bmod 256 \\
&j \leftarrow (j + S[i]) \bmod 256 \\
&\textbf{swap}\ (S[i], S[j]) \\
&k \leftarrow S[(S[i] + S[j]) \bmod 256]
\end{aligned}
$$

Encryption or Decryption

After k has been created,

the plaintext byte is encrypted with k

to create the ciphertext byte.

RAMAIAH
Institute of Technology

Rivest Cipher 4 (RC4)

Encryption algorithm for RC4

```
RC4_Encryption (K)
{
    // Creation of initial state and key bytes
    for (i = 0 to 255)
    {
        S[i] ← i
        K[i] ← Key [i mod KeyLength]
    }
    // Permuting state bytes based on values of key bytes
    j ← 0
    for (i = 0 to 255)
    {
        j ← (j + S[i] + K[i]) mod 256
        swap (S[i] , S[j])
    }
    // Continuously permuting state bytes, generating keys, and encrypting
    i ← 0
    j ← 0
    while (more byte to encrypt)
    {
        i ← (i + 1) mod 256
        j ← (j + S[i]) mod 256
        swap (S [i] , S[j])
        k ← S [(S[i] + S[j]) mod 256]
        // Key is ready, encrypt
        input P
        C ← P ⊕ k
        output C
    }
}
```

RC4 Encryption Example

Lets consider the stream cipher RC4, but instead of the full 256 bytes, we will use 8 x 3-bits.

That is, the state vector **S** is 8 x 3-bits.

We will operate on 3-bits of plaintext at a time since S can take the values 0 to 7, which can be represented as 3 bits.

Assume we use a 4 x 3-bit key of **K** = [1 2 3 6]

And a plaintext **P** = [1 2 2 2]

RC4 PRGA and Encryption

Initialise the state vector **S** and temporary vector **T**.

S is initialised so the S[i] = i,

S = [0 1 2 3 4 5 6 7]

The first step

```
for (i = 0 to 255)
{
    S[i] ← i
    K[i] ← Key [i mod KeyLength]
}
```

T is initialised so it is the key **K** (repeated as necessary).

T = [1 2 3 6 1 2 3 6]

RAMAIAH Institute of Technology

RC4 PRGA and Encryption

Now perform the initial permutation on S.

```
j = 0;
for i = 0 to 7 do
    j = (j + S[i] + T[i]) mod 8
    Swap(S[i],S[j]);
end
```

The second step

$j \leftarrow 0$
for ($i = 0$ to 255)
{
  $j \leftarrow (j + S[i] + K[i]) \bmod 256$
  swap $(S[i], S[j])$
}

For i = 0:

j = (0 + 0 + 1) mod 8 = 1

Swap(S[0],S[1]);

S = [1 0 2 3 4 5 6 7]

S = [0 1 2 3 4 5 6 7]

T = [1 2 3 6 1 2 3 6]

RC4 PRGA and Encryption

S = [1 0 2 3 4 5 6 7]

For i = 1:
j =(1 + 0 + 2) mod 8 = 3
Swap(S[1],S[3])
S = [1 3 2 0 4 5 6 7];

For i = 2:
j =(3 + 2 + 3) mod 8 = 0
Swap(S[2],S[0]);
S = [2 3 1 0 4 5 6 7];

For i = 3:
j =(0 + 0 + 6) mod 8 = 6
Swap(S[3],S[6])
S = [2 3 1 6 4 5 0 7];

For i = 4:
j = 3
Swap(S[4],S[3])
S = [2 3 1 4 6 5 0 7];

For i = 5:
j = 2
Swap(S[5],S[2]);
S = [2 3 5 4 6 1 0 7];

For i = 6:
j = 5;
Swap(S[6],S[4])
S = [2 3 5 4 0 1 6 7];

For i = 7:
j = 2;
Swap(S[7],S[2])
S = [2 3 7 4 0 1 6 5];

Hence, our initial permutation of **S = [2 3 7 4 0 1 6 5];**

T = [1 2 3 6 1 2 3 6]

The second step

```
j ← 0
for (i = 0 to 255)
{
    j ← (j + S[i] + K[i]) mod 256
    swap (S[i] , S[j])
}
```

RC4 PRGA and Encryption

Now we generate 3-bits at a time, k, that we XOR with each 3-bits of plaintext to produce the ciphertext.

<u>The 3-bits k is generated by:</u>

```
i, j = 0;
while (true)
{
    i = (i + 1) mod 8;
    j = (j + S[i]) mod 8;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 8;
    k = S[t];
}
```

$i \leftarrow (i + 1) \bmod 256$

$j \leftarrow (j + S[i]) \bmod 256$

swap $(S[i], S[j])$

$k \leftarrow S[(S[i] + S[j]) \bmod 256]$

RC4 PRGA and Encryption

The first iteration:

S = [2 3 7 4 0 1 6 5]

i = (0 + 1) mod 8 = 1
j = (0 + S[1]) mod 8 = 3
Swap(S[1],S[3])
S = [2 4 7 3 0 1 6 5]
t = (S[1] + S[3]) mod 8 = 7
k = S[7] = 5

$i \leftarrow (i + 1) \bmod 256$
$j \leftarrow (j + S[i]) \bmod 256$
swap $(S[i], S[j])$
$k \leftarrow S[(S[i] + S[j]) \bmod 256]$

Remember, **P** = [1 2 2 2]
3-bits of ciphertext is obtained by:
= k XOR P
=5 XOR 1
=101 XOR 001
=100
= 4

The second iteration:

S = [2 4 7 3 0 1 6 5]

i = (1 + 1) mod 8 = 2
j = (2 + S[2]) mod 8 = 1
Swap(S[2],S[1])

S = [2 7 4 3 0 1 6 5]

t = (S[2] + S[1]) mod 8 = 3

k = S[3] = 3

$$i \leftarrow (i + 1) \bmod 256$$
$$j \leftarrow (j + S[i]) \bmod 256$$
$$\textbf{swap}\ (S[i], S[j])$$
$$k \leftarrow S[(S[i] + S[j]) \bmod 256]$$

Remember, **P** = [1 2 2 2]
3-bits of ciphertext is obtained by:
= k XOR P
=3 XOR 2
= 011 XOR 010
=001
= 1

RC4 PRGA and Encryption

After 4 iterations:

To encrypt the
plaintext stream **P** = [1 2 2 2] with key **K** = [1 2 3 6]
cipher we get **C** = [4 1 2 0].

RC4 Example

Encrypt the
Plaintext stream **P** = [6 1 5 4] with key **K** = [1 0 0 2]
Cipher   **C** = [   ]

RC4 Example

Encrypt the
Plaintext stream **P** = [6 1 5 4] with key **K** = [1 0 0 2]
Cipher **C** = [7 1 5 6]

Rivest Cipher 4 (RC4)

Security Issues

- It is believed that the cipher is secure if the key size is at least 128 bits (16 bytes).
- There are some reported attacks for smallest key sizes (less than 5 bytes), but the protocols that use RC4 today all use key size that make RC4 secure.
- However, to protect against differential cryptanalysis, it is recommended the different keys be used for different sessions.

A5/1

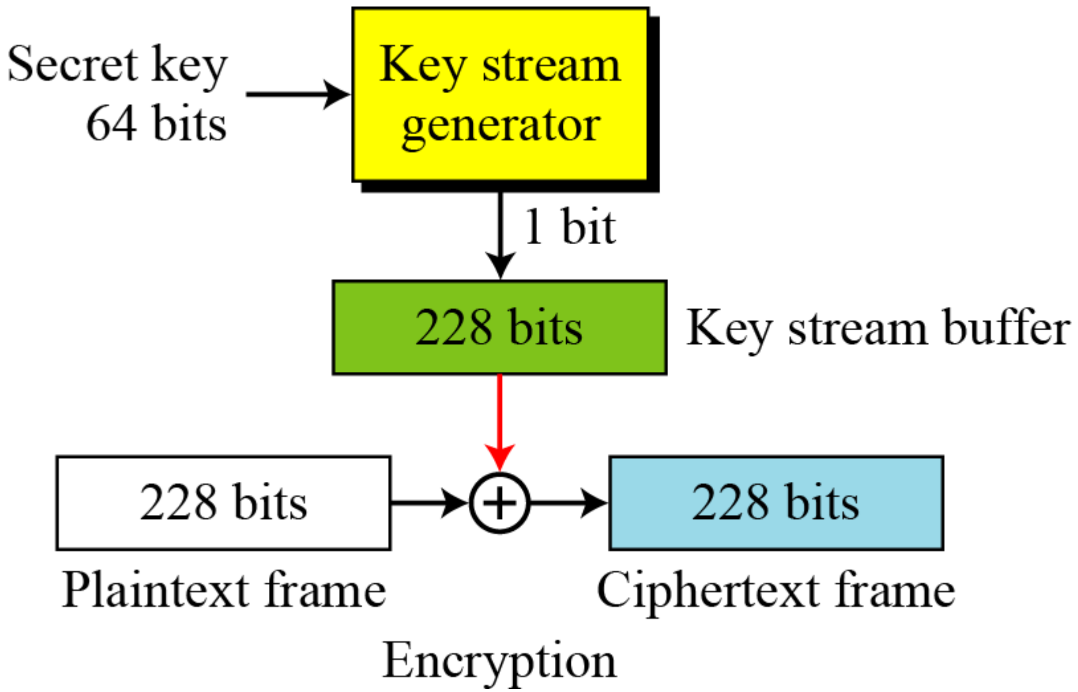
- A5/1 is a symmetric stream encryption algorithm (cipher) which is hardware-based and is used for confidentiality in GSM cell phones.
- Phone communication in GSM done as a sequence of 228 bit frames in which each frame last 4.6milliseconds.

RAMAIAH Institute of Technology

- It is a stream cipher that uses Linear Feedback Shift Registers (LFSRs) to create a bit stream.
- A5/1 creates a bit stream out of 64-bit key
- 1-bit output is fed to the 228 bit buffer to be used for encryption or decryption

- Bit stream collected in 228bit buffer is exclusive ored with a 228 bit frame.

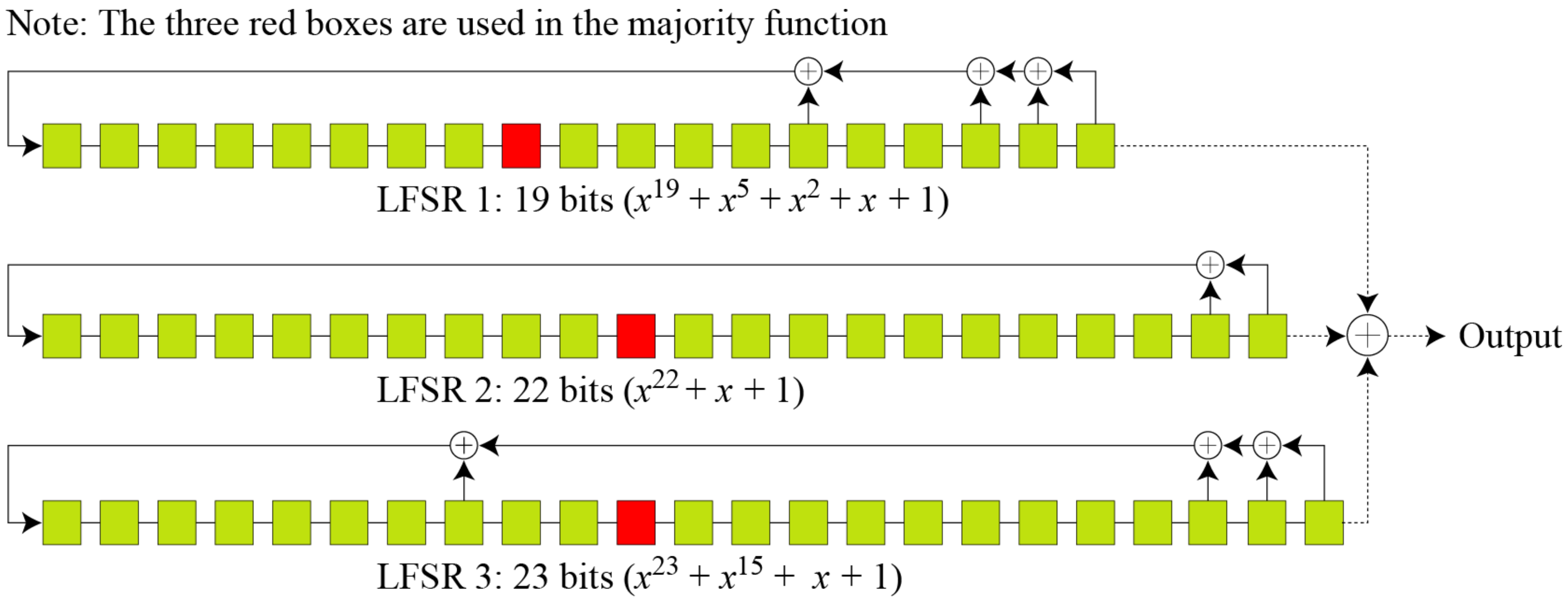
Figure : General outline of A5/1



Key Generator

A5/1 uses three LFSRs with 19, 22, and 23 bits.

Figure : Three LFSR's in A5/1



Other Issues

- Encipherment using symmetric-key block or stream ciphers requires discussion of other issues.
  - Key Management
  - Key Generation

Key Management

Key management refers to **managing cryptographic keys within a cryptosystem**.

It deals with generating, exchanging, storing, using and replacing keys as needed at the user level.

A key management system will also include key servers, user procedures and protocols, including cryptographic protocol design.

Key Generation

- Different symmetric-key ciphers need keys of different sizes.

- The selection of the key must be based on a systematic approach to avoid a security leak.

- The keys need to be chosen randomly.

- This implies that there is a need for random (or pseudorandom) number generator.

Asymmetric Key Cryptography

- Distinguish between two cryptosystems: symmetric-key and asymmetric-key
- Trapdoor one-way functions and their use in asymmetric- key cryptosystems
- Knapsack cryptosystem as one of the first ideas in asymmetric-key cryptography
- RSA cryptosystem
- Rabin cryptosystem
- ElGamal cryptosystem

Introduction

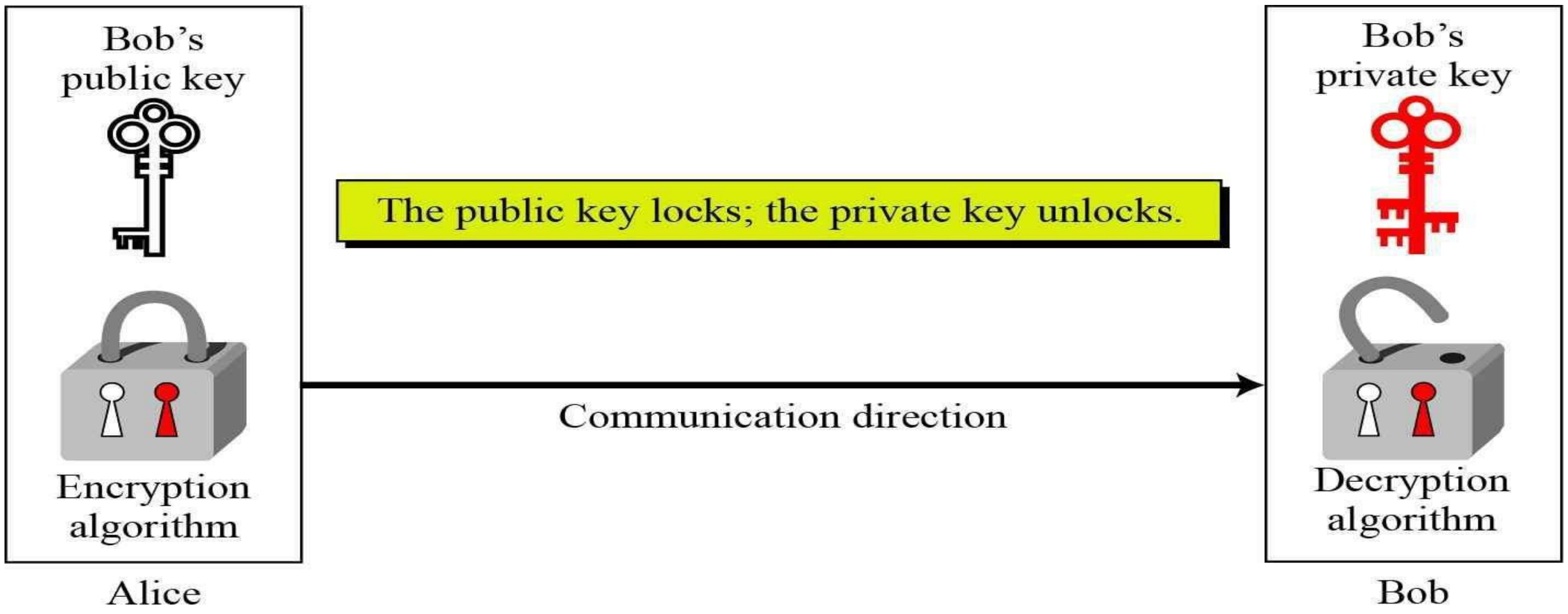
- Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community.
- We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.
- Symmetric-key cryptography is based on sharing secrecy.
- Asymmetric-key cryptography is based on personal secrecy.

Introduction

- Keys
- General Idea
- Trapdoor One-Way Function
- Knapsack Cryptosystem

Keys

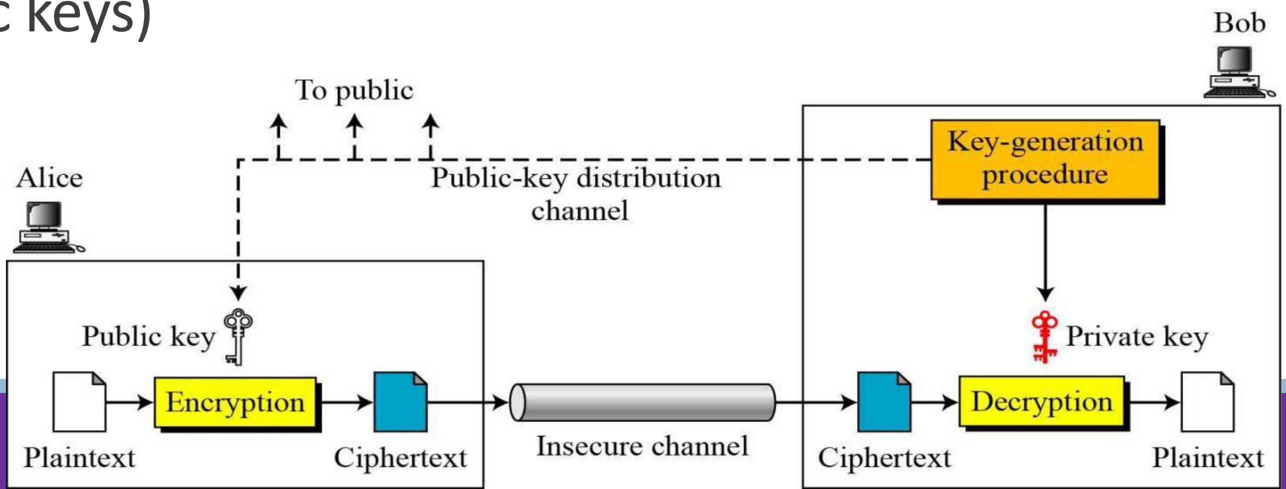
Asymmetric key cryptography uses two separate keys: one private and one public.



General Idea for encipherment

Several facts

1. The burden of providing security is mostly on the shoulders of the receiver (Bob, in this case). Bob creates private key and public key
2. Alice and Bob cannot use same set of keys for two waycommunication
3. Bob needs only one private key to receive all correspondence from anyone in the community, but Alice needs n public key to communicate with n entities in the community(Ring of Public keys)



General Idea

Plaintext/Ciphertext

Unlike in symmetric-key cryptography, plaintext and ciphertext are treated as integers in asymmetric-key cryptography.

Encryption – Mathematical functions

$$C = f\ (K_{public}\ ,\ P)$$

Decryption

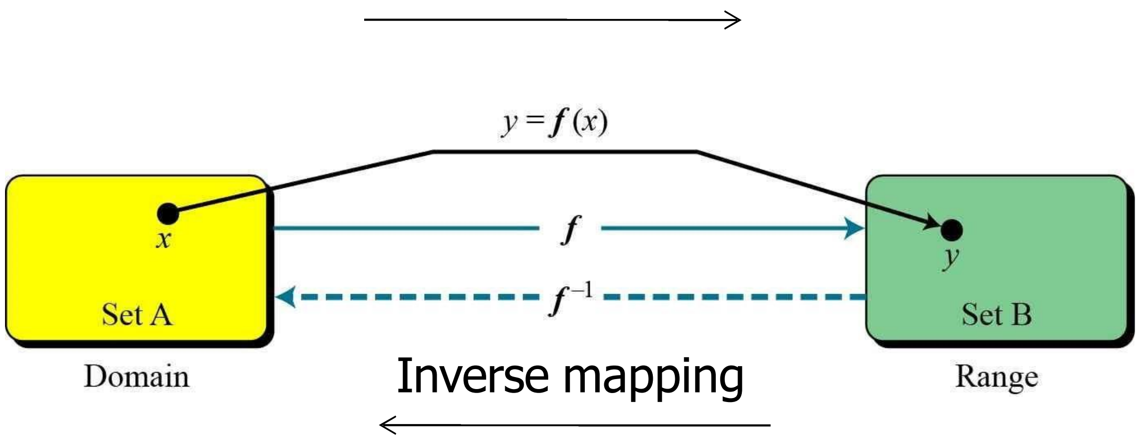
$$P = g\ (K_{private}\ ,\ C)$$

A trapdoor is a door set into a floor or ceiling. An entrance to secret passageway.

The Function f needs to be a trap door one- way function to allow receiver to decrypt the message but prevent Eve(Attacker) from doing so.

Trapdoor One-way function

- The main idea behind asymmetric-key cryptography is the concept of the trapdoor one-way function.
- A function is a rule that maps one element in set A (called the Domain), to one element in set B (called the Range)
- An Invertible function is a function that associates each element in the range with exactly one element in the domain



Inverse mapping

Trapdoor One-way function

One - way function(OWF)

1. f is easy to compute, if given x , y=f(x) can be easily computed
2. f^{-1} is difficult to compute, if given y, it is computationally infeasible to calculate $x = f^{-1}(y)$

Trap door One - way function(TOWF)

3. It is a one-way function, Given y and a trapdoor (secret), x can be computed secretly

Trapdoor One-way function

Example :

When n is large, $n = p \times q$ is a one-way function.

Given p and q , it is always easy to calculate n,
Given n, it is very difficult to compute p and q.
This is the factorization problem.

Trapdoor One-way function

Example

When n is large, the function $y = x^k \bmod n$ is a trapdoor one-way function.

Given x, k, and n, it is easy to calculate y.

Given y, k, and n, it is very difficult to calculate x.

This is the discrete logarithm problem.

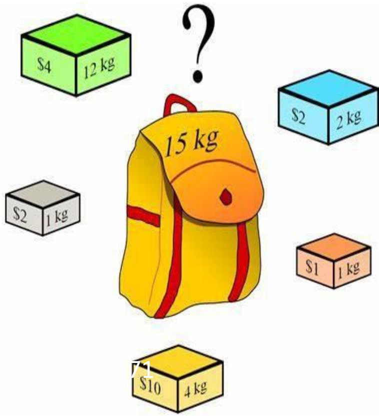
However, if we know the trapdoor, k′ such that $k \times k' = 1 \bmod \Phi(n)$,

we can use $x = y^{k'} \bmod n$ to find x.

RAMAIAH
Institute of Technology

Knapsack Cryptosystem

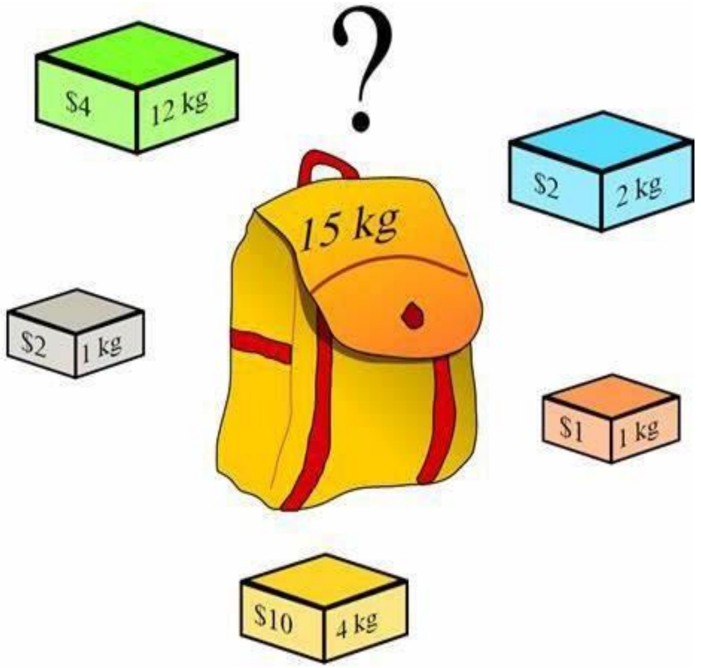
- The knapsack cryptosystem is a public-key cryptosystem based on a special case of the classic problem known as the knapsack problem.
- It is developed by Ralph Merkle and Mertin Hellman in 1978.
- The Knapsack Cryptosystem is first Public-Key cryptography



Knapsack Cryptosystem

The knapsack algorithm works like this:

Imagine you have a set of different weights which you can use to make any total weight that you need by adding combinations of any of these weights together.



Knapsack Cryptosystem

Let us look at an example:

- Imagine you had a set of weights 1, 6, 8, 15 and 24.
- To pack a knapsack weighing 30, you could use weights 1, 6, 8 and 15.

$$1+6+8+15 = 30$$

- Represent the weight 30 by the binary code – 1 1 1 1 0
- So, if someone sends the code 30 this can only have come from the plain text 11110 .

Plain text	1 0 0 1 1	1 1 0 1 0	0 1 0 1 1	0 0 0 0 0
Knapsack	1 6 8 15 24	1 6 8 15 24	1 6 8 15 24	1 6 8 15 24
Cipher text	1 + 15 + 24 = 40	1 + 6 + 15 = 22	6 + 15 + 24 = 45	0 = 0

Public key

Knapsack Cryptosystem

- When the Knapsack Algorithm is used in public key cryptography, the idea is to create two different knapsack problems.
- One is easy to solve, the other not.
- Using the easy knapsack, the hard knapsack is derived from it. The hard knapsack becomes the public key.
- The easy knapsack is the private key.
- The public key can be used to encrypt messages, but cannot be used to decrypt messages.
- The private key decrypts the messages.

Knapsack Cryptosystem

Suppose given two k-tuples

$a=[a_1,a_2,.......a_k]$ and $x=[x_1,x_2.............x_k]$

The first tuple is the predefined set

The second tuple in which x_i is only 0 or 1

The sum of elements in the knapsack is

s=knapsackSum(a,x)
x=inv_knapsackSum(s,a)

s=knapsackSum(a,x) $=x_1a_1+x_2a_2+............x_ka_k$,

given a and x it is easy to calculate to S, however given the value of s and a it is difficult to compute x

Knapsack Cryptosystem

Superincreasing tuple

- Easy to calculate knapsackSum(a,x) and inv_knapsackSum(s,a) if the k-tuple a is superincreasing
- In superincreasing tuple $a_i \geq a_1 + a_2 + \ldots\ldots\ldots\ldots\ldots a_{i-1}$
- Every element except a_1 is greater than or equal to the sum of previous elements.

Knapsack Cryptosystem

- An easy knapsack problem is one in which the weights are in a superincreasing sequence.
- A superincreasing sequence is one in which the next term of the sequence is greater than the sum of all preceding terms.
- For example,
- The set {1, 2, 4, 9, 20, 38} is superincreasing,
- The set {1, 2, 3, 9, 10, 24} is not because 10 < 1+2+3+9.

Knapsack Cryptosystem -Algorithm

```
knapsackSum (x [1 … k], a [1 … k])
{
    s ← 0
    for (i = 1 to k)
    {
      s ← s + a_i × x_i
    }
    return s
}
```

Knapsack Cryptosystem

Assume that a=[17,25,46,94,201,400] and x[0,1,1,0,1,0]. Find S using knapsack sum.

i	a_i	x_i	$s \longleftarrow s + a_i * x_i$	s
1	17	0	0 + 17 * 0	0
2	25	1	0 + 25 * 1	25
3	46	1	25 + 46 * 1	71
4	94	0	71 + 94 * 0	71
5	201	1	71 + 201 * 1	272
6	400	0	272 + 400 * 0	**272**

```
knapsackSum (x [1 … k], a [1 … k])
{
   s ← 0
   for (i = 1 to k)
   {
     s ← s + a_i × x_i
   }
   return s
}
```

Knapsack Cryptosystem

```
inv_knapsackSum (s, a [1 … k])
{
    for (i = k down to 1)
    {
        if s ≥ a_i
        {
            x_i ← 1
            s   ← s − a_i
        }
        else x_i ← 0
    }
    return x [1 … k]
}
```

RAMAIAH Institute of Technology

Knapsack Cryptosystem

Assume that a=[17,25,46,94,201,400] and s = 272 . Find tuple x using inv_knapsack sum.

i	a_i	s	$s \geq a_i$	x_i	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

```
inv_knapsackSum (s, a [1 … k])
{
    for (i = k down to 1)
    {
        if s ≥ a_i
        {
            x_i ← 1
            s ← s − a_i
        }
        else x_i ← 0
    }
    return x [1 … k]
}
```

x[0,1,1,0,1,0] = 25 , 46, 201 are in the knapsack

Knapsack Cryptosystem

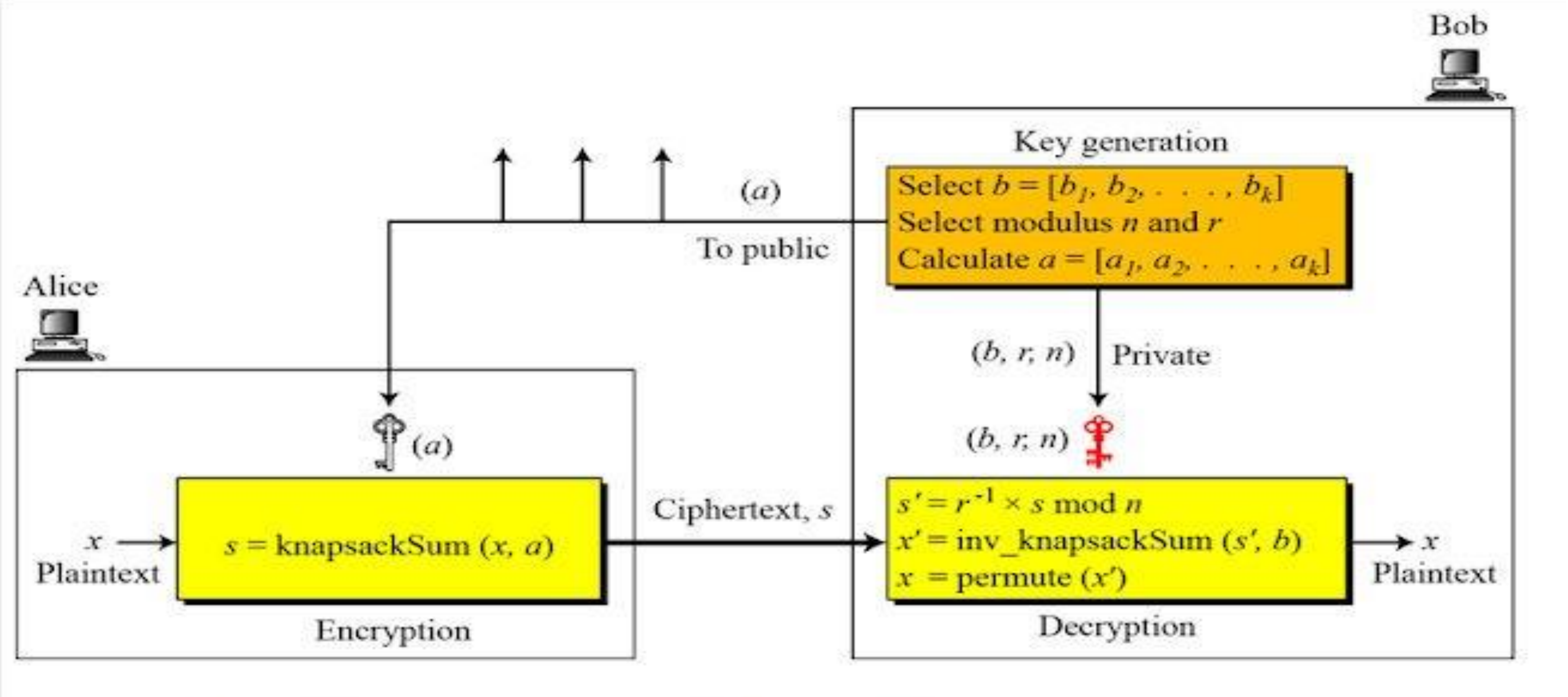
Secret Communication Process with Knapsack Cryptosystem

i. Key Generation at Receiver End

ii. Encryption at Sender End

iii. Decryption at Receiver End

Knapsack Cryptosystem



Secret Communication with Knapsack Cryptosystem

Knapsack Cryptosystem

Key Generation at Receiver End

a. Create a superincreasing k-tuple $b = [b_1, b_2, \ldots, b_k]$ **b={1, 2, 4, 10, 20, 40}.**

b. Choose a modulus n, such that $n > b_1 + b_2 + \cdots + b_k$ for example, n=110

c. Select a random integer r that is relatively prime with n and $1 \le r \le n-1$. for example, r=31

d. Create a temporary k-tuple $t = [t_1, t_2, \ldots, t_k]$ in which $t_i = r \times b_i \bmod n$.

e. Select a permutation of k objects and find a new tuple $a = permute(t)$.

f. The public key is the k-tuple a. The private key is n, r, and the k-tuple b.

$1\times31 \bmod(110) = 31$
$2\times31 \bmod(110) = 62$
$4\times31 \bmod(110) = 14$
$10\times31 \bmod(110) = 90$
$20\times31 \bmod(110) = 70$
$40\times31 \bmod(110) = 30$

Private key → **b={1, 2, 4, 10, 20 40}.**

Public key → **a={31, 62, 14, 90, 70, 30}**

RAMAIAH
Institute of Technology

Knapsack Cryptosystem

Encryption at Sender End

Suppose Alice needs to send a message to Bob.

a. Alice converts her message to a k-tuple $x = [x_1, x_2, \ldots, x_k]$ in which x_i is either 0 or 1. The tuple x is the plaintext.

b. Alice uses the *knapsackSum* routine to calculate s. She then sends the value of s as the ciphertext.

x=100100111100101110

The knapsack contains six weights so we need to split the message into groups of six:

100100

111100

101110

This corresponds to three sets of weights with totals as follows

100100 = 31 + 90 = 121

111100 = 31+62+14+90 = 197

101110 = 31+14+90+70 =205

So the coded cipher message is **s={121, 197, 205}**, will transfer using some channel.

Knapsack Cryptosystem

Decryption at Receiver End

- Receiver receive cipher message **s={121, 197, 205}**, Now the receiver has to decode the message
- The person decoding must know the two numbers n=110 and r= 31
- We need r^{-1}, which is a multiplicative inverse r^{-1}mod n = 71

121×71 mod(110) = 11 = 1+10=>100100
197×71 mod(110) = 17 = 1+2+4+10=>111100
205×71 mod(110) = 35 = 1+4+10+20=>101110

Bob receives the ciphertext s.

a. Bob calculates $s' = r^{-1} \times s \bmod n$.

b. Bob uses *inv_knapsackSum* to create x'.

c. Bob permutes x' to find x. The tuple x is the recovered plaintext.

1. Key generation:
   a. Bob creates the superincreasing tuple $b = [7, 11, 19, 39, 79, 157, 313]$.
   b. Bob chooses the modulus $n = 900$ and $r = 37$, and [4 2 5 3 1 7 6] as permutation table.
   c. Bob now calculates the tuple $t = [259, 407, 703, 543, 223, 409, 781]$.
   d. Bob calculates the tuple a = permute $(t) = [543, 407, 223, 703, 259, 781, 409]$.
   e. Bob publicly announces a; he keeps n, r, and b secret.

2. Suppose Alice wants to send a single character "g" to Bob.
   a. She uses the 7-bit ASCII representation of "g", $(1100111)_2$, and creates the tuple $x = [1, 1, 0, 0, 1, 1, 1]$. This is the plaintext.
   b. Alice calculates $s = knapsackSum\ (a, x) = 2165$. This is the ciphertext sent to Bob.

3. Bob can decrypt the ciphertext, $s = 2165$.
   a. Bob calculates $s' = s \times r^{-1} \bmod n = 2165 \times 37^{-1} \bmod 900 = 527$.
   b. Bob calculates $x' = Inv\_knapsackSum\ (s', b) = [1, 1, 0, 1, 0, 1, 1]$.
   c. Bob calculates $x = permute\ (x') = [1, 1, 0, 0, 1, 1, 1]$. He interprets the string $(1100111)_2$ as the character "g".

Knapsack Cryptosystem

Assume that a=[3, 7, 12, 30, 60,115] and s = 82 . Find tuple x using inv_knapsack sum.

X = [1,1,1,0,1,0]

DES/AES/RSA

TABLE I. COMPARATIVE ANALYSIS BETWEEN AES, DES AND RSA

Features	DES	AES	RSA
Developed	1977	2000	1977
Key Length	56 bits	128,192,256 bits	More than 1024 bits
Cipher Type	Symmetric block cipher	Symmetric block cipher	Asymmetric block cipher
Block size	64 bits	128 bits	Minimum 512 bits
Security	Not secure enough	Excellent secured	Least secure
Hardware & Software Implementation	Better in hardware than software	Better in both	Not efficient
Encryption and Decryption	Moderate	Faster	Slower

RSA Cryptosystem

- Introduction
- Procedure
- Attacks on RSA
- Optimal Asymmetric Encryption Padding (OAEP)
- Applications

RSA Cryptosystem

- RSA cryptosystem, named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.
- The RSA algorithm is an asymmetric cryptography algorithm; uses a public key and a private key (i.e two different, mathematically linked keys).

Key size	Key strength
512 bits	Low-strength key
1024 bits	Medium-strength key
2048 bits	High-strength key
4096 bits	Very high-strength key

RSA Cryptosystem

- The length of the plain text is less than or equal to the key length (Bytes)-11.
- RSA is only able to encrypt data to a maximum amount equal to key size (2048 bits = 256 bytes), minus any padding and header data (11 bytes for padding).
- The padding standards we generally use are NoPPadding, OAEPPadding, PKCS1Padding, etc., among which the padding suggested by PKCS#1 occupies 11 bytes.

RSA Cryptosystem

Length of ciphertext

The length of the ciphertext is the bit length of the key

RAMAIAH
Institute of Technology

RSA Cryptosystem

Characteristics of RSA

It is a public key encryption technique.

It is safe for exchange of data over internet.

It maintains confidentiality of the data.

RSA has high toughness as breaking into the keys by interceptors is very difficult.

RSA Cryptosystem

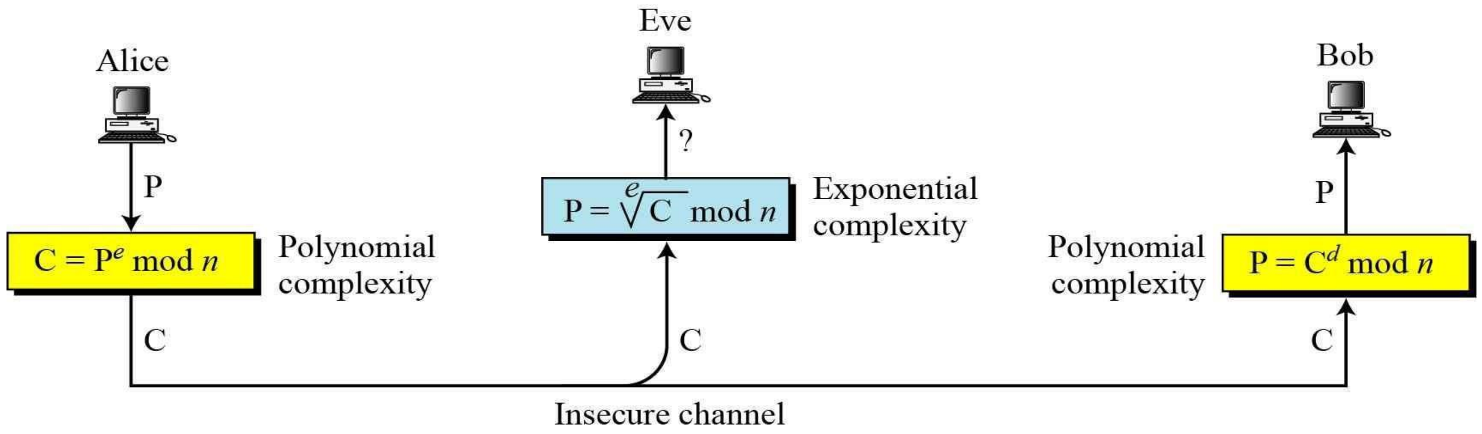
Advantages of RSA

- It is very easy to implement RSA algorithm.
- RSA algorithm is safe and secure for transmitting confidential data.
- Cracking RSA algorithm is very difficult as it involves complex mathematics.
- Sharing public key to users is easy.

Introduction

Complexity of operations in RSA

- P is Plaintext and C is Ciphertext
- RSA uses two exponents e and d, where e is public and d is private.
- Alice uses $C = P^e \bmod n$ to create Ciphertext
- Bob uses $P = C^d \bmod n$ to revert Plaintext



RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.

Introduction

Procedure used in RSA

i. Key Generation
ii. Encryption/Decryption Function

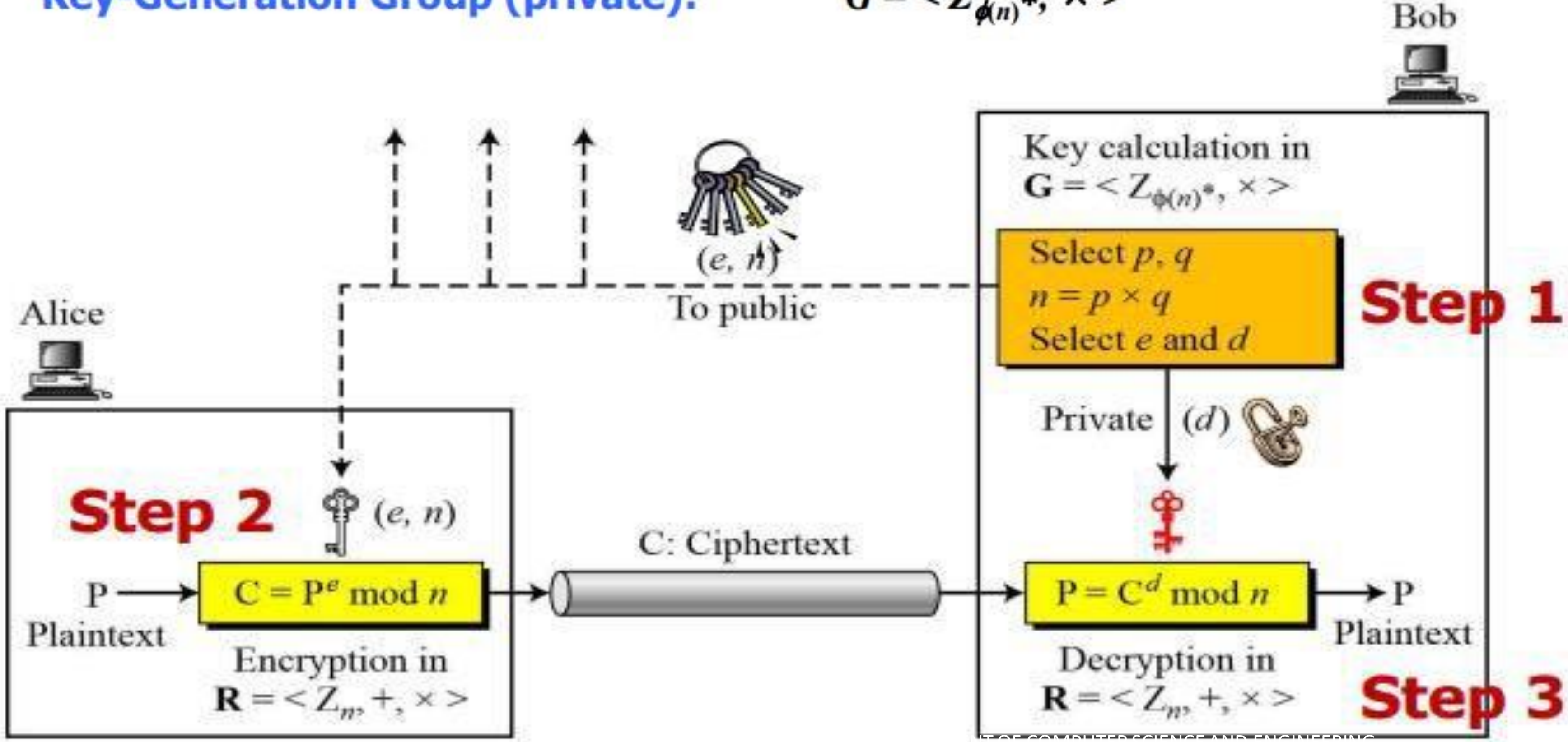
Introduction

In RSA, the tuple (e, n) is the public key; the integer d is the private key.

RSA uses two algebraic structures :

Encryption/Decryption Ring (public): $R = < Z_n, +, \times >$

Key-Generation Group (private): $G = < Z_{\phi(n)}*, \times >$



Introduction

> In RSA, p and q must be at least 512 bits; n must be at least 1024 bits.

RSA_Key_Generation

{
  Select two large primes p and q such that $p \neq q$.
  $n \leftarrow p \times q$
  $\phi(n) \leftarrow (p-1) \times (q-1)$
  Select e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$
  $d \leftarrow e^{-1} \bmod \phi(n)$ // d is inverse of e modulo $\phi(n)$
  Public_key $\leftarrow$ (e, n) // To be announced publicly
  Private_key $\leftarrow d$ // To be kept secret
  return Public_key and Private_key
}

RSA_Encryption (P, e, n) // P is the plaintext in Z_n and P < n

{ // Calculation of ($P^e \bmod n$)
  C $\leftarrow$ **Fast_Exponentiation** (P, e, n)
  return C
}

RSA_Decryption (C, d, n) //C is the ciphertext in Z_n

{ // Calculation of ($C^d \bmod n$)
  P $\leftarrow$ **Fast_Exponentiation** (C, d, n)
  return P
}

Introduction

Key Generation

- Choose two large prime numbers (p and q)
- Calculate n = p*q and

  $\Phi(n) = (p-1)(q-1)$
- Choose a number e , where $1 < e < \Phi(n)$

  e is coprime to $\Phi(n)$, $\gcd(e, \Phi(n)) = 1$
- Calculate $d = e^{-1} \bmod n$, or $de \bmod \Phi(n) = 1$
- Public key pair as (e,n)
- Private key is d

In RSA, the tuple (e, n) is the public key; the integer d is the private key.

Encryption/Decryption Function

- If the plaintext is P, ciphertext

  $C = P^e \bmod n$
- If the ciphertext is C, plaintext

  $P = C^d \bmod n$

Introduction

Key Generation

- Choose two large prime numbers (p and q)
- Calculate $n = p*q$ and

  $\Phi(n) = (p-1)(q-1)$
- Choose a number e , where $1 < e < \Phi(n)$

  e is coprime to $\Phi(n)$, $gcd(e, \Phi(n)) = 1$
- Calculate $d = e^{-1} mod\ n$, or $de\ mod\ \Phi(n) = 1$
- Public key pair as (e,n)
- Private key is d

Choose p = 3 and q = 11

Compute $n = p * q = 3 * 11 = 33$

Compute $\varphi(n) = (p - 1) * (q - 1) = 2 * 10 = 20$

Choose e = ? → 3

$gcd(e, \Phi(n)) = 1$

$gcd(2, 20) = 2$

$gcd(3, 20) = 1$

Introduction

Key Generation

- Choose two large prime numbers (p and q)
- Calculate n = p*q and

  $\Phi(n) = (p-1)(q-1)$
- Choose a number e , where $1 < e < \Phi(n)$

  e is coprime to $\Phi(n)$, $\gcd(e, \Phi(n)) = 1$
- Calculate $d = e^{-1} \bmod n$, or $de \bmod \Phi(n) = 1$
- Public key pair as (e,n)
- Private key is d

Let e = 3

d e mod $\Phi(n)$ = 1

(1 * 3) mod 20 = 3

(2 * 3) mod 20 = 6

(3 * 3) mod 20 = 9

(4* 3) mod 20 = 12

(5 * 3) mod 20 = 15

(6 * 3) mod 20 = 8

(7 * 3) mod 20 = 1

d = 7

Public key is (e, n) => (3, 33)

Private key is d => 7

Introduction

<u>Encryption/Decryption Function</u>

- If the plaintext is P, ciphertext

  $C = P^e \bmod n$

- If the ciphertext is C, plaintext

  $P = C^d \bmod n$

The encryption of P = 2

$C = 2^3 \bmod 33$

$= 8 \bmod 33 \rightarrow 8$

The decryption of C =

$P = 8^7 \bmod 33$

$= 2$

Introduction

Key Generation

- Choose two large prime numbers (p and q)
- Calculate $n = p*q$ and

  $\Phi(n) = (p-1)(q-1)$
- Choose a number e , where $1 < e < \Phi(n)$

  e is coprime to $\Phi(n)$, $gcd(e, \Phi(n)) = 1$
- Calculate $d = e^{-1} \bmod n$, or $de \bmod \Phi(n) = 1$
- Public key pair as (e,n)
- Private key is d

Choose p = 3 and q = 11

Compute $n = p * q = 3 * 11 = 33$

Compute $\varphi(n) = (p - 1) * (q - 1) = 2 * 10 = 20$

Let e = 7

$d = (3 * 7) \bmod 20 = 1$, d=3

Public key is (e, n) => (7, 33)

Private key is d => 3

Introduction

Encryption/Decryption Function

- If the plaintext is P, ciphertext

  $C = P^e \bmod n$

- If the ciphertext is C, plaintext

  $P = C^d \bmod n$

The encryption of P = 2

$C = 2^7 \bmod 33$

$= 29$

The decryption of C

$P = 29^3 \bmod 33$

$= 2$

Introduction

Key Generation

- Choose two large prime numbers (p and q)
- Calculate n = p*q and

  $\Phi(n) = (p-1)(q-1)$
- Choose a number e , where $1 < e < \Phi(n)$

  e is coprime to $\Phi(n)$, $gcd(e, \Phi(n)) = 1$
- Calculate $d = e^{-1} \bmod n$, or $de \bmod \Phi(n) = 1$
- Public key pair as (e,n)
- Private key is d

Choose p = 3 and q = 5

Compute n = p * q = 3 * 5= 15

Compute φ(n) = (p - 1) * (q - 1) = 2 * 4 = 8

Let e = 3

d = (3 * 3) mod 8 → 9 mod 8 = 1

Public key is (e, n) => (3, 15)
Private key is d => 3

Introduction

Encryption/Decryption Function

- If the plaintext is P, ciphertext

  $C = P^e \bmod n$

- If the ciphertext is C, plaintext

  $P = C^d \bmod n$

The encryption of P = 4

$C = 4^3 \bmod 15$

$= 64 \bmod 15 = 4$

The decryption of C =

$P = 4^3 \bmod 15$

$= 4$

72^{24} mod 131

Binary of exponent 24 is 1 1 0 0 0

b = 72, q=131

Assume d=1	1	1	0	0	0
d	1	72	29	55	12
d^2 mod q	1	75	55	12	13 (Ans)
d^2b mod q	72	29			

72^{24} mod 131=13

$2^{97} \bmod 131$

Binary of exponent 97 is 1 1 0 0 0 0 1

b = 2, q=131

Assume d=1	1	1	0	0	0	0	1
d	1	2	8	64	35	46	20
$d^2 \bmod q$	1	4	64	35	46	20	7
$d^2 b \bmod q$	2	8					14(Ans)

Example

Example 10.5 **Step 1 :** Bob chooses 7 and 11 as p and q and calculates $n = 77$. The value of $\phi(n) = (7 - 1)(11 - 1)$ or 60. Now he chooses two exponents, e and d, from $Z_{60}{}^*$. If he chooses e to be 13, then d is 37. Note that $e \times d \bmod 60 = 1$ (they are inverses of each.) **Step 2 :** Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5	$C = 5^{13} = 26 \bmod 77$	Ciphertext: 26

Step 3 : Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26	$P = 26^{37} = 5 \bmod 77$	Plaintext: 5

Example 10.6 **Step 2 :** Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

Plaintext: 63	$C = 63^{13} = 28 \bmod 77$	Ciphertext: 28

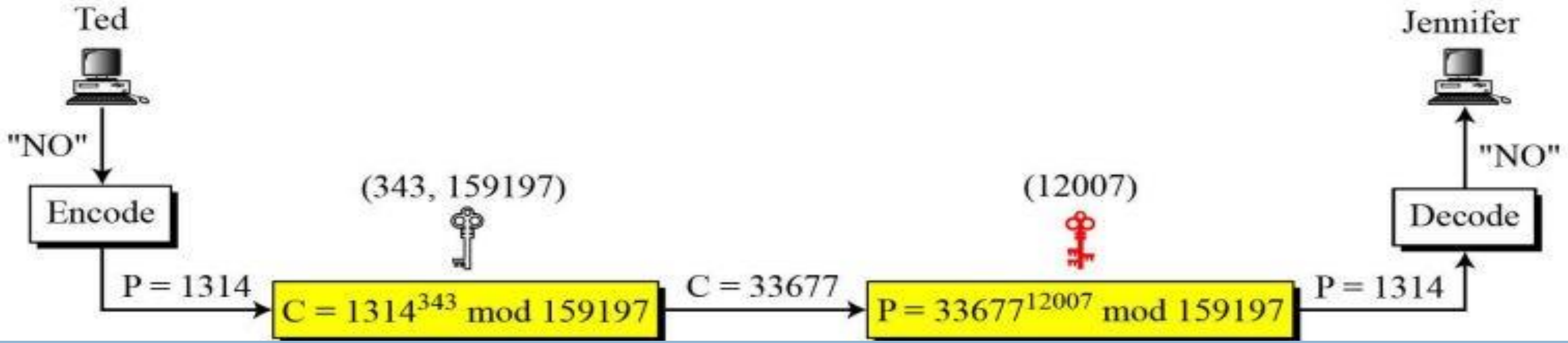
Step 3 : Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28	$P = 28^{37} = 63 \bmod 77$	Plaintext: 63

Example

Example 10.7 **Step 1 :** Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159197$. She then calculates $\phi(n) = 158400$. She then chooses $e = 343$ and $d = 12007$. Show how Ted can send a message to Jennifer if he knows e and n.

Step 2 : Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314.



Example

Bob chooses 7 and 11 as p and q and calculates n value. Find the value of $\phi(n)$. Now choose the two exponents e and d. Now assume that Alice wants to send the plain text 5 to Bob. Find the cipher text and decrypt it on receiving side to get plaintext using RSA algorithm.

Example

Show the steps of RSA Algorithm. If the RSA public key is (31, 3599), what is the corresponding private key?

Example

Show the steps of RSA Algorithm. If the RSA public key is (31, 3599), what is the corresponding private key?

e=31 and n=3599

p=59 and q=61

phi(n)= 3480

d*e=1 mod 3480

d= 3031

Private key = 3031

Example

Bob chooses 13 and 11 as p and q and calculates n value. Find the value of ϕ(n). Find the two exponents e and d. Now assume that Alice wants to send the plain text 13 to Bob. Find the cipher text and decrypt it on receiving side to get plaintext using RSA algorithm.

Example

Bob chooses 61 and 53 as p and q and calculates n value. Find the value of $\phi(n)$. Let e= 17, Find the exponents d. Now assume that Alice wants to send the plain text 65 to Bob. Find the cipher text and decrypt it on receiving side to get plaintext using RSA algorithm.

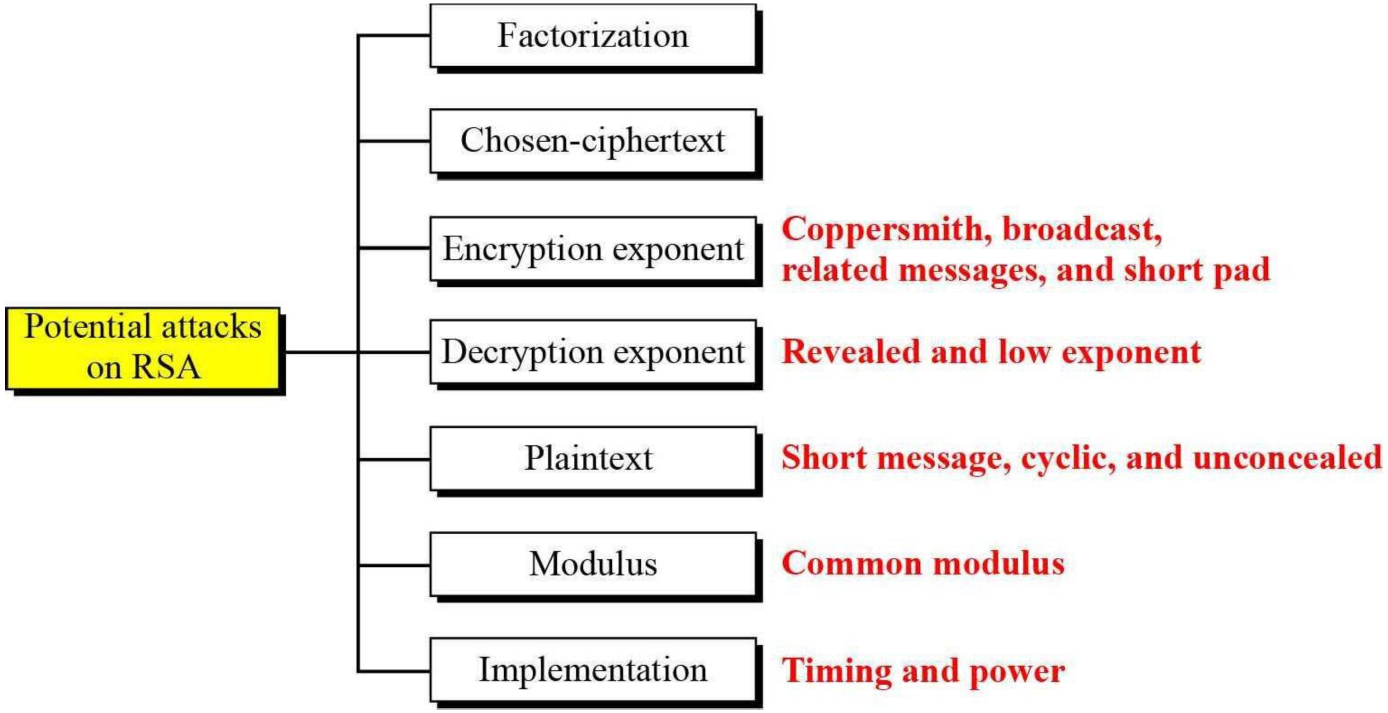
RAMAIAH
Institute of Technology

Example

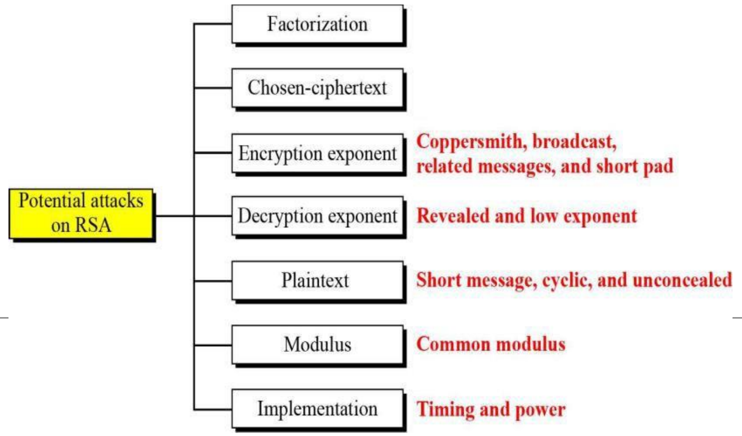
In a RSA cryptosystem a particular A uses two prime numbers p = 13 and q =17 to generate her public and private keys. If the public key of A is 35. Then the private key of A is_______________.

(A) 11
(B) 13
(C) 16
(D) 17

Attacks on RSA



Attacks on RSA



Factoring attacks

Factoring is the act of splitting an integer into a set of smaller integers (factors) which, when multiplied together, form the original integer.

The factoring problem is to find 3 and 5 when given 15.

Factoring an RSA would allow an attacker to figure out the private key

Attacks on RSA

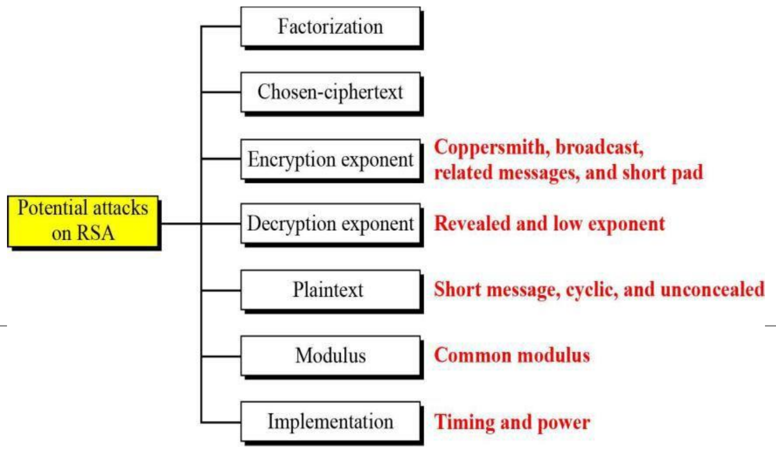
Factoring attacks

This is **the attack that attempts to find the key through the solving of the very large prime number factor problem**.

If attacker will able to know P and Q using N, then he could find out value of private key.

This can be failed when N contains atleast 300 longer digits in decimal terms, attacker will not able to find.

Attacks on RSA



Chosen cipher attack:

Alice creates ciphertext $C = P^e \bmod n$ and sends C to Bob. Bob will decrypt for eve

Eve intercept C and uses following steps to find P.

a. Eve chooses a random integer X in Z_n^*.

b. Eve calculates $Y = C \times X^e \bmod n$.

c. Eve sends Y to Bob for decryption and get $Z = Y^d \bmod n$; This step is an instance of a chosen-ciphertext attack.

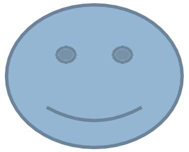
d. Eve can easily find P because

$$Z = Y^d \bmod n = (C \times X^e)^d \bmod n = (C^d \times X^{ed}) \bmod n = (C^d \times X) \bmod n = (P \times X) \bmod n$$
$$Z = (P \times X) \bmod n \quad \rightarrow \quad P = Z \times X^{-1} \bmod n$$

Attacks on RSA

C

Sender → Receiver



Eve Intercept C
Choose X random integer
Construct New Cipher $Y = C\,X^{e} \bmod n$

Receiver

$Z = Y^{d} \bmod n$

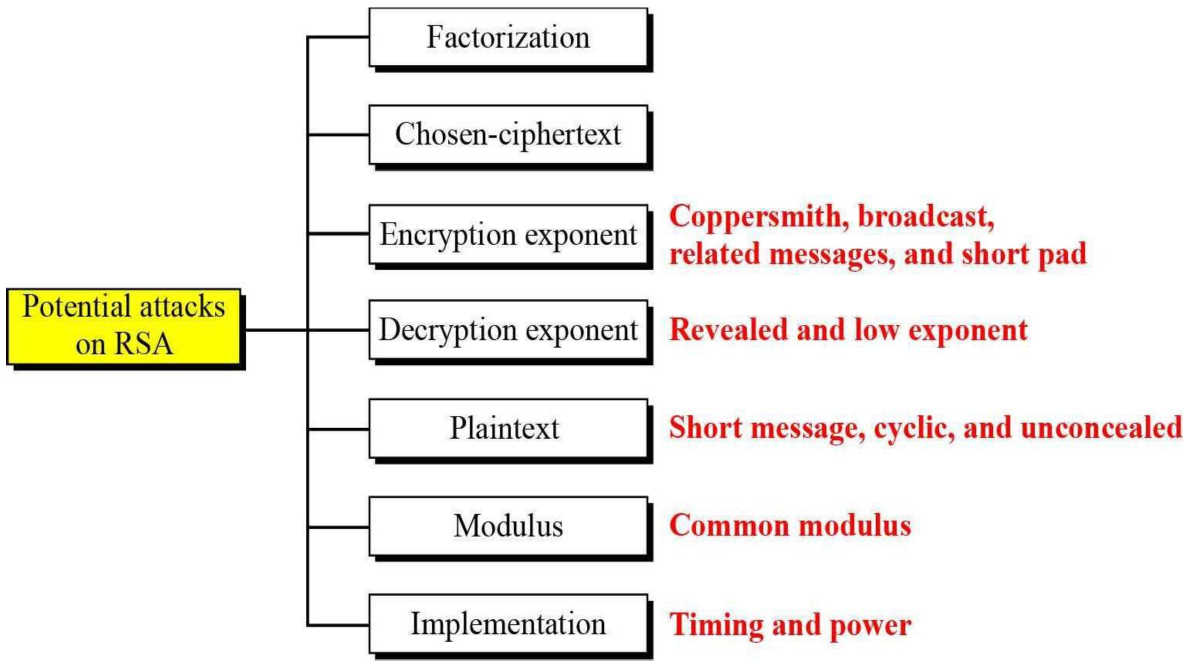
$$
\begin{aligned}
Z &= Y^{d} \bmod N \\
&= (C\,X^{e})^{d} \bmod n \\
&= C^{d}\,X^{ed} \bmod n \\
&= C^{d}\,X \bmod n \\
Z &= P\,X \bmod n \\
P &= Z\,X^{-1} \bmod n
\end{aligned}
$$

Attacks on RSA

Encryption exponent

Common attack occur when e is low, so use $e= 2^{16} +1 = 65537$.

- Coppersmith attack
- Broadcast attack
- Related Message attack
- Short pad attack



Attacks on RSA

Coppersmith attack :

Theorem states that in a modulo n polynomial f (x) of degree e, one can use an algorithm of the complexity log n to find the roots if one of the roots is smaller than n $^{1/e}$

Broadcast attack

Suppose Alice wishes to send same message to three receipents with the same public key exponent e and the moduli n1,n2,n3

$C_1 = P^3 \bmod n_1$ $C_2 = P^3 \bmod n_2$ $C_3 = P^3 \bmod n_3$

$C' = P^3 \bmod n_1 n_2 n_3.$ $P^3 < n_1 n_2 n_3.$

$C' = P^{1/3}.$

Attacks on RSA

Related Message attack

- If Alice encrypt two P1 and P2 with e =3 and send C1 and C2 to Bob.
- If P1 and P2 is related by a linear function, then eve can recover P1 and p2 in a feasible computation time.

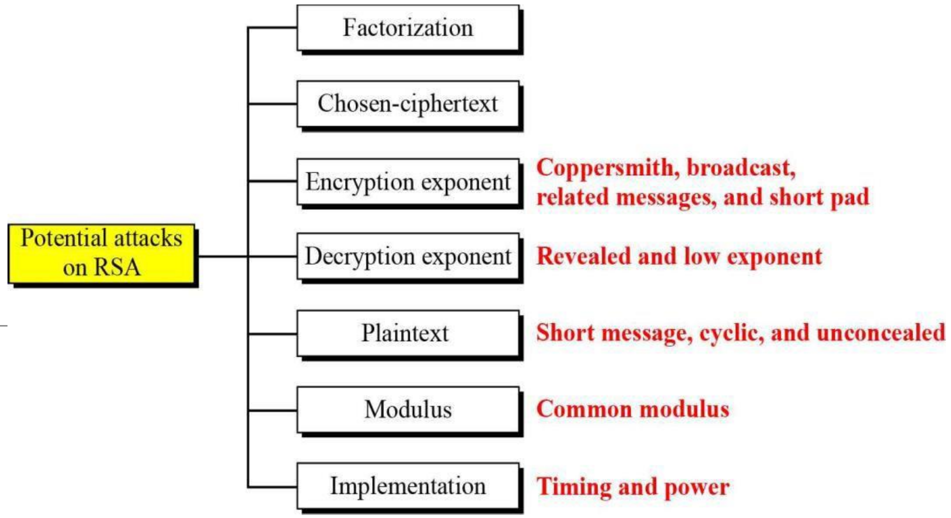
Attacks on RSA

Short pad attack

- Alice has a message M to send to Bob. She pads the message with r1, encrypt and send C1 to Bob.
- Eve intercept C1 and drops it
- Bob inform Alice that he has not received the message, so Alice pads the message again with r2, encrypt and send to Bob.
- Eve also will intercept the message.
- Eve now has C1 and C2, knows both belong to same plaintext .
- If r1 and r2 are short, eve may be able to recover M

C 1	**M**	**r1(padding)**
C 2	**M**	**r2(padding)**

Attacks on RSA



Attacks on Decryption key:

Revealed decryption exponent attack:

If attacker somehow guess decryption key d, cipher text generated by encryption key is in danger, and even future messages are also in danger.

So, it is advised to take fresh values of two prime numbers (i.e; P and Q), N and E.

Low decryption exponent attack:

If we take smaller value of d in RSA this may occur, so to avoid take value of

d = 2^{16+1}(atleast).

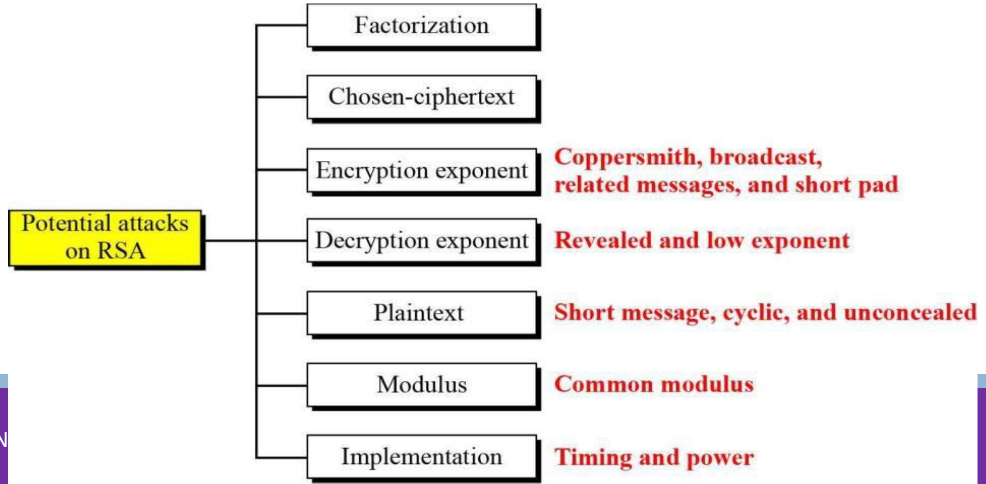
Attacks on RSA

Plain text attacks: It is classified into 3 subcategories:-

Short message attack:

Attacker knows some blocks of plain text. If this assumption is true, the attackers can try encrypting each plain-text block to view if it results into the known cipher-text.

Therefore, it can avoid this short-message attack, it is suggested that it can pad the plain text before encrypting it.



Attacks on RSA

Cycling attack:

Attacker will think that plain text is converted into cipher text using permutation.

Continuous encryption of ciphertext will eventually result in plain text. But attacker does not know the plain text. Hence will keep doing it until gets the ciphertext, goes back one step find the plain text

Intercepted ciphertext: C

$C_1 = C^e \bmod n$

$C_2 = C_1^e \bmod n$

...

$C_k = C_{k-1}^e \bmod n \rightarrow$ If $C_k = C$, stop: the plaintext is $P = C_{k-1}$

Attacks on RSA

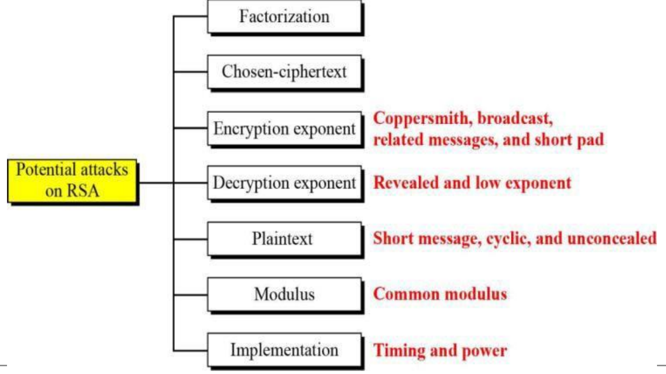
Unconcealed Message attack:

For some plain-text messages, encryption provides cipher-text which is the equal as the original plain-text.

If this appears, the original plain-text message cannot be secret.

Therefore, this attack is known as unconcealed message attack.

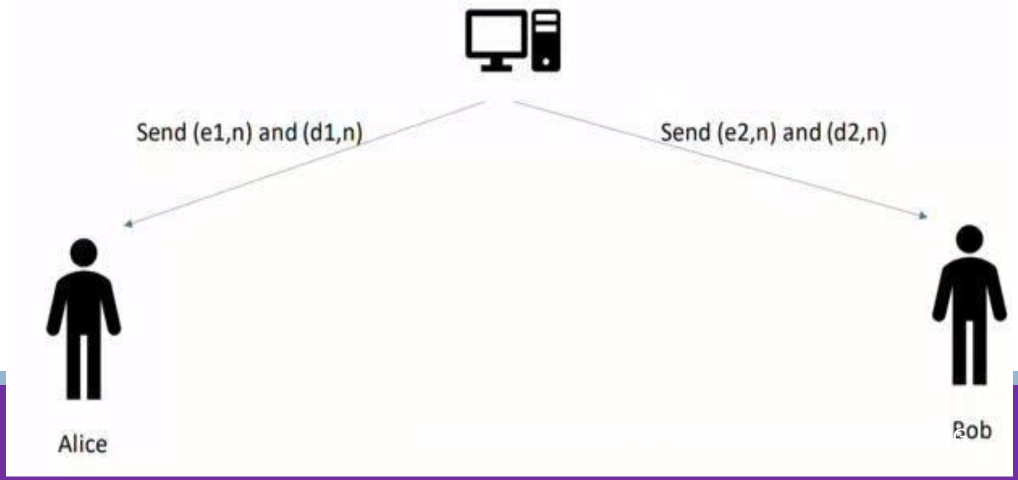
Attacks on RSA



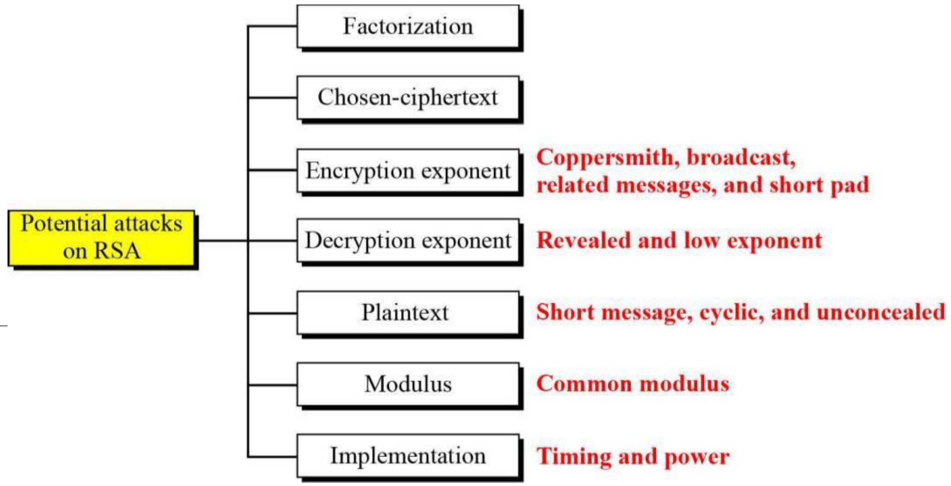
Attacks on the Modulus – Common modulus attack

If a community uses a common modulus n, select p and q, calculate n and Φ(n), and create a pair of exponents(e_i, d_i) for each entity.

The problem is eve can also decrypt the message, if he is a member of the community and assigned a pair of exponent (e_e, d_e)



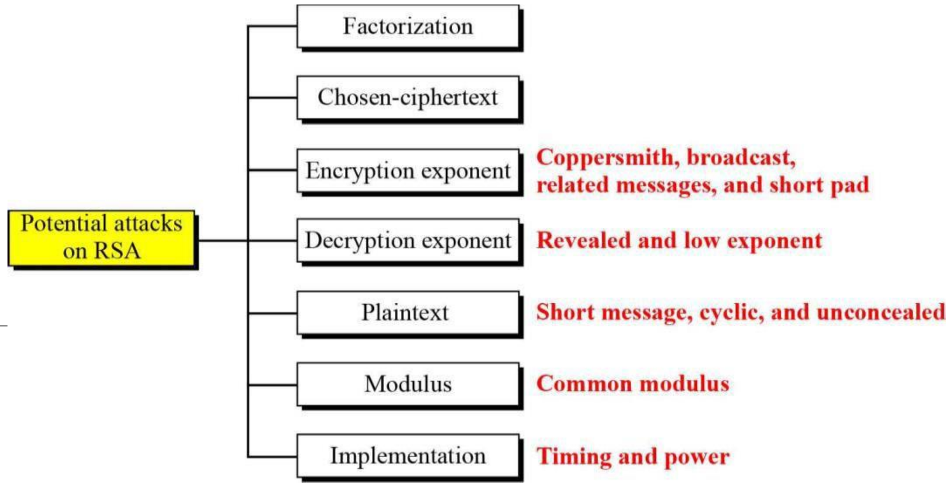
Attacks on RSA



Implementation –Timing attack

- Eve intercept a large number of ciphertext C1,C2… to Cm.
- Eve observe how long it takes for the underlying hardware to calculate a multiplication operation from t1 to tm(t is time required to calculate the multiplication operation)
- The timing difference allows Eve to find the value of bits in d, one by one

Attacks on RSA

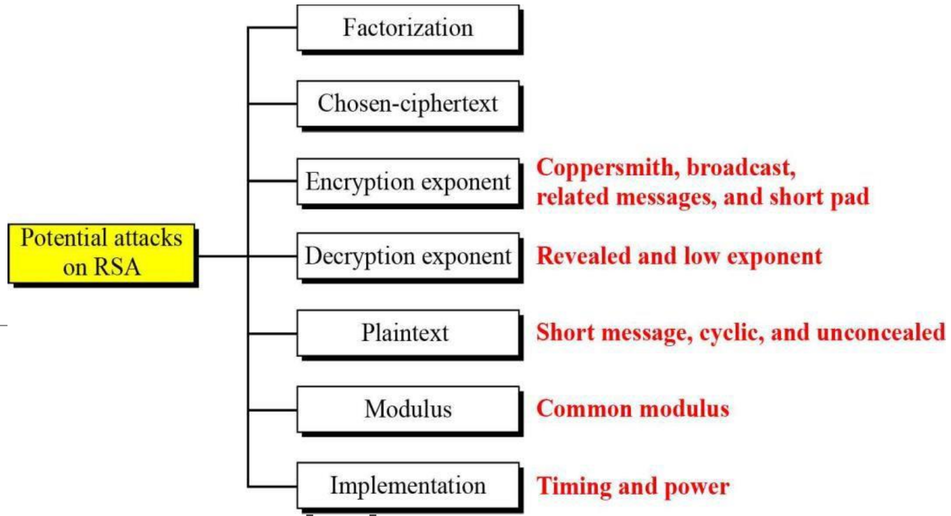


Implementation –Timing attack

There are two methods to thwart timing attack:

1. Add random delays to the exponentiations to make each exponentiation take the same amount of time.
2. Rivest recommended **blinding.** The idea is to multiply the ciphertext by a random number before decryption. The procedure is as follows:
   a. Select a secret random number r between 1 and $(n - 1)$.
   b. Calculate $C_1 = C \times r^e \bmod n$.
   c. Calculate $P_1 = C_1^{\,d} \bmod n$.
   d. Calculate $P = P_1 \times r^{-1} \bmod n$.

Attacks on RSA



Implementation – Power attack

Eve can precisely measure the power consumed during decryption, can launch power attack.

Multiplication and squaring consumes more power.

RSA Cryptosystem

- Introduction
- Procedure
- Attacks on RSA
- Optimal Asymmetric Encryption Padding (OAEP)
- Applications

Padding in RSA

- RSA without padding is also called Textbook RSA.
- RSA without padding is insecure.
- With RSA the padding is essential for its core function.
- RSA has a lot of mathematical structure, which leads to weaknesses. Using correct padding prevents those weaknesses.

Padding in RSA

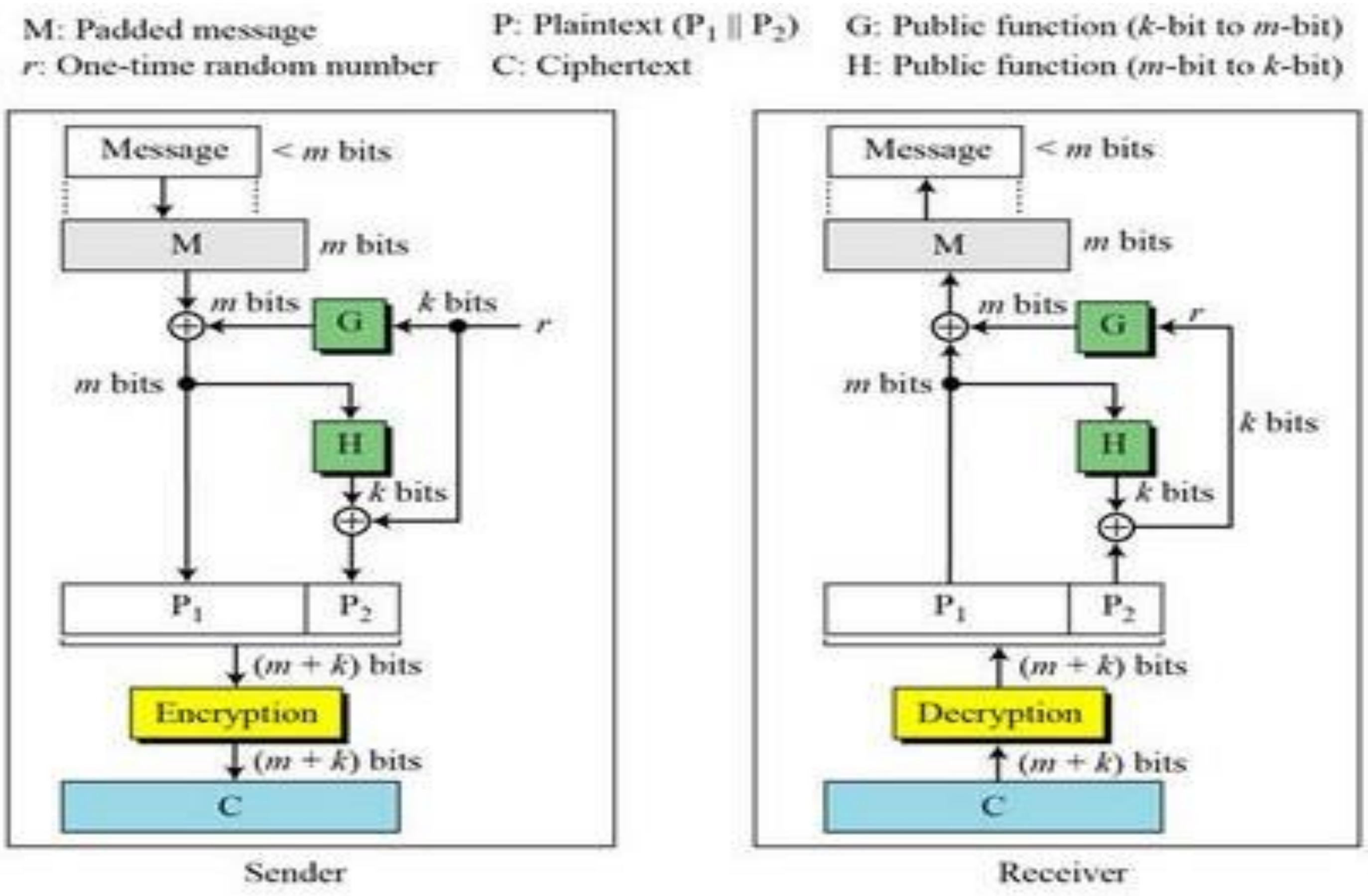
Padding schemes

- PKCS#1 (Public-Key Cryptography Standards)
- Optimal Asymmetric Encryption Padding (OAEP)

Optimal Asymmetric Encryption Padding (OAEP)

- Short message makes ciphertext vulnerable to short message attacks.
- Adding bogus data(padding) to the message make Eve's job harder, but with additional efforts can still attack the ciphertext.
- The solution is apply a procedure called OAEP.
- A 2048 bit RSA key allows for 256 bytes(2048*8) of which the OAEP padding takes 42 bytes, leaving around 214 bytes for encrypted data.

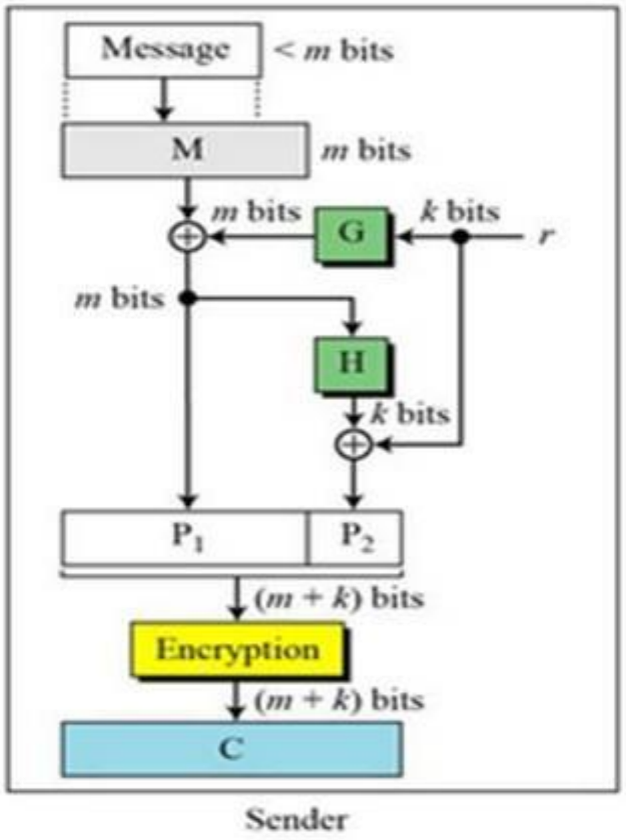
Optimal Asymmetric Encryption Padding (OAEP)



Optimal Asymmetric Encryption Padding (OAEP)

Encryption The following shows the encryption process:

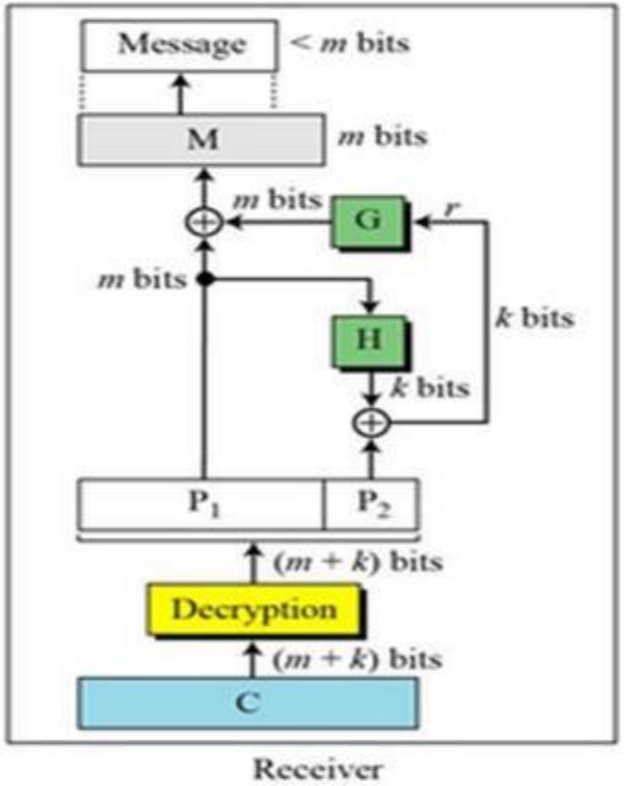
1. Alice pads the message to make an m-bit message, which we call M.
2. Alice chooses a random number r of k bits. Note that r is used only once and is then destroyed.
3. Alice uses a public one-way function, G, that takes an r-bit integer and creates an m-bit integer (m is the size of M, and $r < m$). This is the mask.
4. Alice applies the mask $G(r)$ to create the first part of the plaintext $P_1 = M \oplus G(r)$. P_1 is the masked message.
5. Alice creates the second part of the plaintext as $P_2 = H(P_1) \oplus r$. The function H is another public function that takes an m-bit input and creates an k-bit output. This function can be a *cryptographic hash function* (see Chapter 12). P_2 is used to allow Bob to recreate the mask after decryption.
6. Alice creates $C = P^e = (P_1 \| P_2)^e$ and sends C to Bob.



Optimal Asymmetric Encryption Padding (OAEP)

Decryption The following shows the decryption process:

1. Bob creates $P = C^d = (P_1 \parallel P_2)$.
2. Bob first recreates the value of r using $H(P_1) \oplus P_2 = H(P_1) \oplus H(P_1) \oplus r = r$.
3. Bob uses $G(r) \oplus P = G(r) \oplus G(r) \oplus M = M$ to recreate the value of the padded message.
4. After removing the padding from M, Bob finds the original message.



Applications of RSA

- RSA was used with Transport Layer Security (TLS) to secure communications among two individuals.
- Pretty Good Privacy algorithm use RSA
- Virtual Private Networks (VPNs), email services, web browsers
- Bluetooth
- MasterCard, VISA, e-banking
- e-commerce platform

RAMAIAH
Institute of Technology

RSA Cryptosystem

Disadvantages of RSA

- It may fail sometimes because for complete encryption both symmetric and asymmetric encryption is required and RSA uses symmetric encryption only.
- It has slow data transfer rate due to large numbers involved.
- It requires third party to verify the reliability of public keys sometimes.
- High processing is required at receiver’s end for decryption.

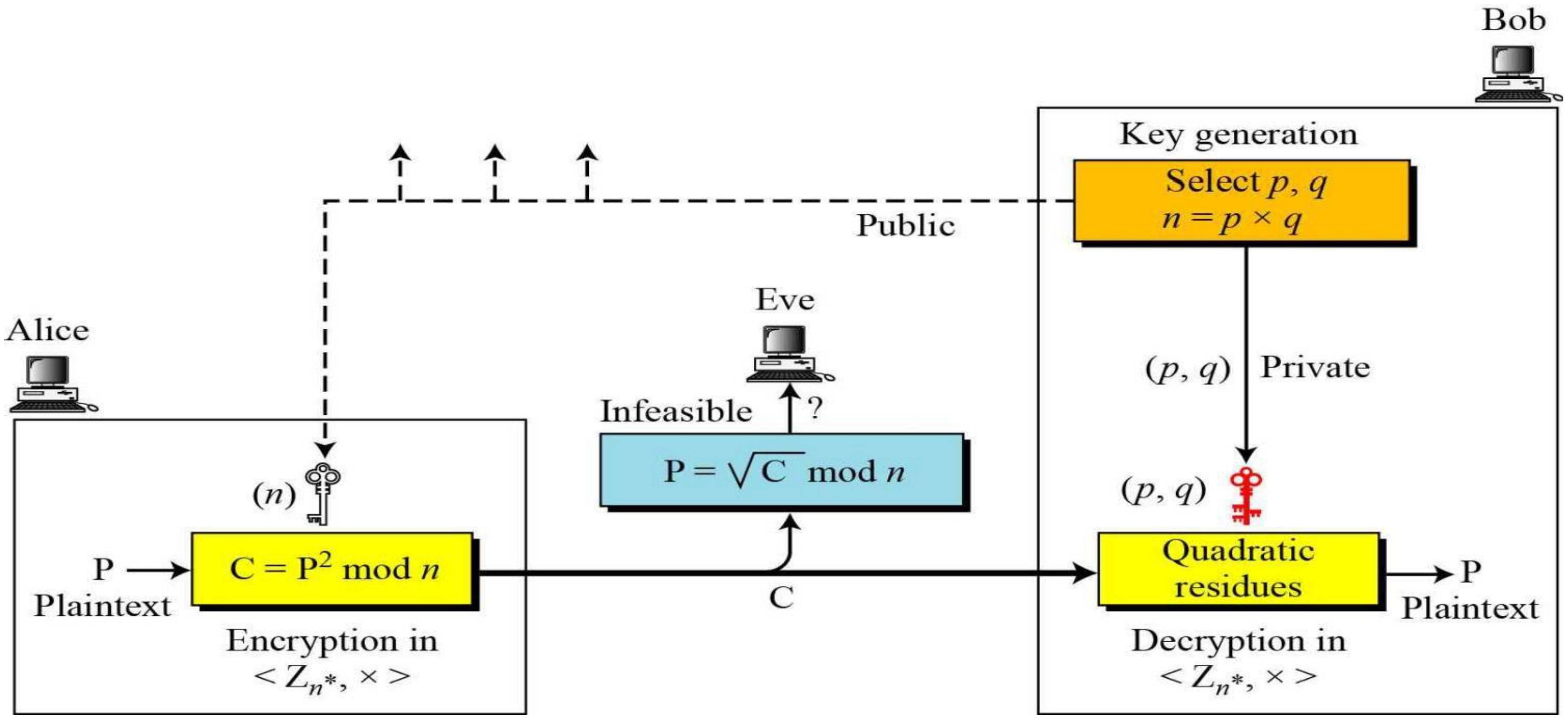
RABIN CRYPTOSYSTEM

- Rabin Cryptosystem is a public-key cryptosystem invented by Michael Rabin.
- In Rabin cryptosystem, value of e = 2 and d= 1/2 is fixed.
- Rabin is based on quadratic congruence

The encryption is $C \equiv P^2 \pmod{n}$ and the decryption is $P \equiv C^{1/2} \pmod{n}$.

- Public key is n
- Private key is tuple(p, q)
- Everyone can encrypt using n and only the receiver can decrypt using p and q

RABIN CRYPTOSYSTEM



RABIN CRYPTOSYSTEM

Algorithm 10.6 *Key generation for Rabin cryptosystem*

```
Rabin_Key_Generation
{
    Choose two large primes p and q in the form 4k + 3 and p ≠ q.
    n ← p × q
    Public_key ← n                         // To be announced publicly
    Private_key ← (q, n)                   // To be kept secret
    return Public_key and Private_key
}
```

Algorithm 10.7 *Encryption in Rabin cryptosystem*

```
Rabin_Encryption (n, P)          // n is the public key; P is the ciphertext from Zn*
{
    C ← P^2 mod n                // C is the ciphertext
    return C
}
```

RABIN CRYPTOSYSTEM

Algorithm 10.8 *Decryption in Rabin cryptosystem*

```
Rabin_Decryption (p, q, C)            // C is the ciphertext; p and q are private keys
{
    a1 ← +(C^((p+1)/4)) mod p
    a2 ← −(C^((p+1)/4)) mod p
    b1 ← +(C^((q+1)/4)) mod q
    b2 ← −(C^((q+1)/4)) mod q
    // The algorithm for the Chinese remainder algorithm is called four times.
    P1 ← Chinese_Remainder (a1, b1, p, q)
    P2 ← Chinese_Remainder (a1, b2, p, q)
    P3 ← Chinese_Remainder (a2, b1, p, q)
    P4 ← Chinese_Remainder (a2, b2, p, q)
    return P1, P2, P3, and P4
}
```

RABIN CRYPTOSYSTEM

1. **Bob selects** $p = 23$ **and** $q = 7$. **Note that both are congruent to** $3 \bmod 4$. p and q are in the form 4k+3 and p not equal to q
2. **Bob calculates** $n = p \times q = 161$.
3. **Bob announces** n **publicly; he keeps** p **and** q **private.**
4. **Alice wants to send the plaintext** P = 24.

Note that 161 **and** 24 **are relatively prime;** 24 **is in** Z_{161}^*.

She calculates $C = 24^2 = 93 \bmod 161$,

and sends the ciphertext 93 to Bob.

RABIN CRYPTOSYSTEM

5. **Bob receives 93 and calculates four values:**

$a_1 = +(93^{(23+1)/4}) \bmod 23 = +(93^6) \bmod 23 = 1 \bmod 23$

$a_2 = -(93^{(23+1)/4}) \bmod 23 = -(93^6) \bmod 23 = -1 \bmod 23 \rightarrow -1+23 \bmod 23 = 22 \bmod 23$

$b_1 = +(93^{(7+1)/4}) \bmod 7 = +(93^2) \bmod 7 = 4 \bmod 7$

$b_2 = -(93^{(7+1)/4}) \bmod 7 = -(93^2) \bmod 7 = -4 \bmod 7 \rightarrow -4+7 \bmod 7 = 3 \bmod 7$

6. **Bob takes four possible answers,** (a_1, b_1), (a_1, b_2), (a_2, b_1), **and** (a_2, b_2), **and uses the Chinese remainder theorem to find four possible plaintexts:** 116, 24, 137, **and** 45.
Note that only the second answer is Alice's plaintext.

RABIN CRYPTOSYSTEM

6. Bob takes four possible answers, (a_1, b_1) uses the Chinese remainder theorem

$a_1 = 1 \bmod 23$

$b_1 = 4 \bmod 7$

$X = [1 * 7\ (7^{-1} \bmod 23) + 4 * 23\ (23^{-1} \bmod 7)] \bmod 161$

$= [1 * 7\ (10) + 4 * 23\ (4)]$

$= [70 + 368] \bmod 161$

$= 116$

RABIN CRYPTOSYSTEM

6. **Bob takes four possible answers, (a_1, b_2) uses the Chinese remainder theorem**

$a_1 = 1 \bmod 23$

$b_1 = 3 \bmod 7$

$X = [1 * 7\ (7^{-1} \bmod 23) + 3 * 23\ (23^{-1} \bmod 7)] \bmod 161$

$= [1 * 7\ (10) + 3 * 23\ (4)]$

$= [70 + 276] \bmod 161$

$= 24$

RABIN CRYPTOSYSTEM

6. **Bob takes four possible answers,** (a_1, b_1), (a_1, b_2), (a_2, b_1), **and** (a_2, b_2), **and uses the Chinese remainder theorem to find four possible plaintexts:** 116, 24, 137, **and** 45.

Note that only the second answer is Alice's plaintext.

Security of the RABIN CRYPTOSYSTEM

- The Rabin system is secure as long as p and q are large numbers.
- The complexity of the Rabin system is at the same level as factoring a large number n into its two prime factors p and q. (Rabin system is as secure as RSA)

RABIN CRYPTOSYSTEM

- Suppose Alice wants to send message to Bob
- Bob(Receiver) chooses the prime numbers p = 43 and q = 47
  Note that $43 \equiv 47 \equiv 3 \bmod 4$
  - n → p* q = 2021
- To encrypt the message m = 741, Alice(Sender) computes
  $C = 741^2 \bmod 2021 = 549081 \bmod 2021 = 1390$
  sends c = 1390 to Bob

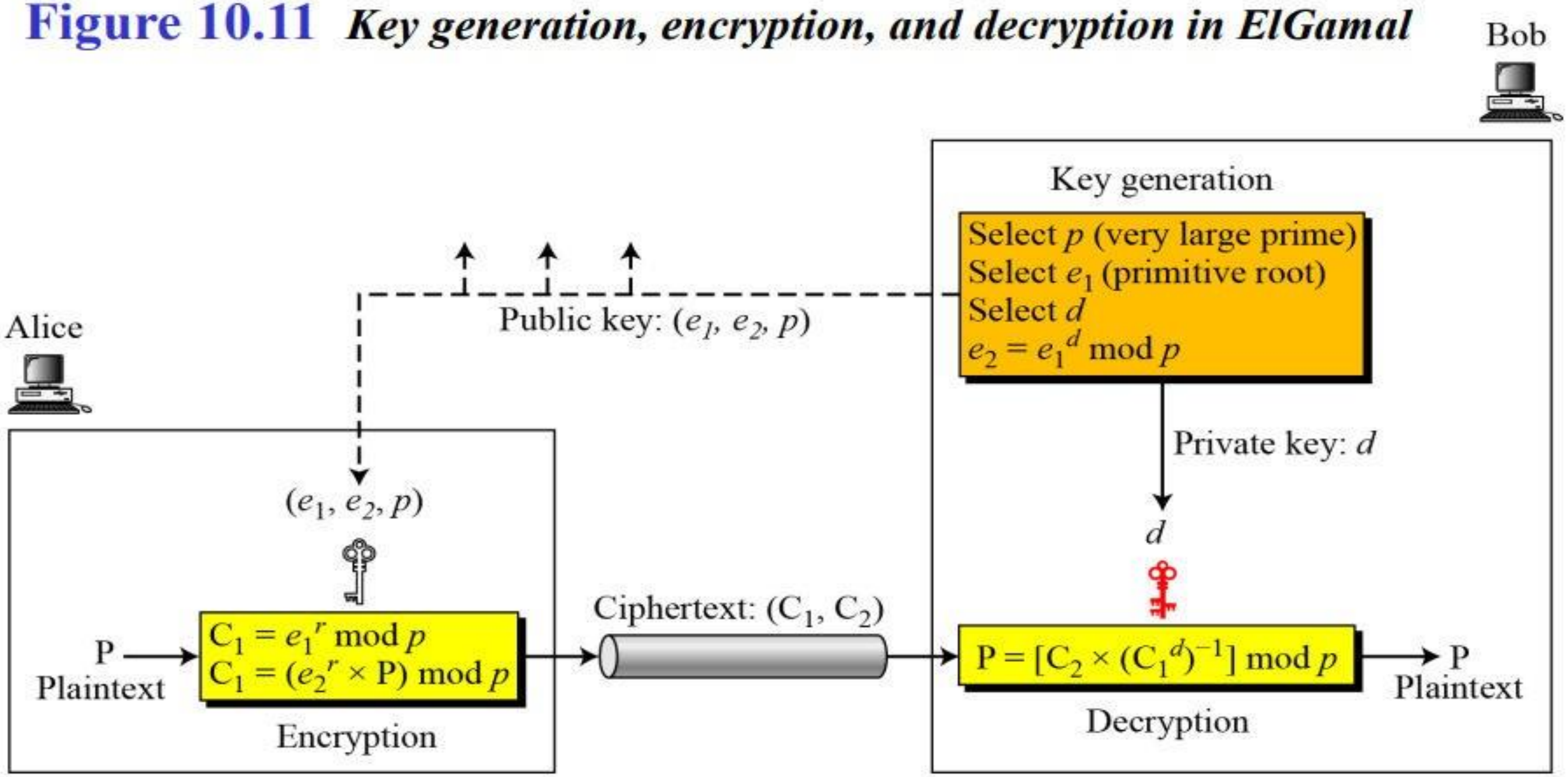
ELAGAMAL CRYPTOSYSTEM

- The ElGamal cryptosystem is a public key encryption algorithm invented by Taher Elgamal in 1985 that is based on the Diffie-Hellman key exchange.

- It can be considered the asymmetric algorithm where the encryption and decryption happen by using public and private keys.

ELAGAMAL CRYPTOSYSTEM

Figure 10.11 ***Key generation, encryption, and decryption in ElGamal***



ELAGAMAL CRYPTOSYSTEM

ElGamal_Key_Generation

{

Select a large prime p

Select d to be a member of the group $\mathbf{G} = <\mathbf{Z}_p{}^*, \times>$ such that $1 \leq d \leq p - 2$

Select e_1 to be a primitive root in the group $\mathbf{G} = <\mathbf{Z}_p{}^*, \times>$

$e_2 \leftarrow e_1{}^d \bmod p$

Public_key $\leftarrow (e_1, e_2, p)$ // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

ELAGAMAL CRYPTOSYSTEM

ElGamal_Encryption $(e_1, e_2, p, \mathrm{P})$ // P is the plaintext

{

Select a random integer r in the group $\mathbf{G} = <\mathbf{Z}_p^*, \times>$

$\mathrm{C}_1 \leftarrow e_1^r \bmod p$

$\mathrm{C}_2 \leftarrow (\mathrm{P} \times e_2^r) \bmod p$ // C_1 and C_2 are the ciphertexts

return C_1 and C_2

}

ELAGAMAL CRYPTOSYSTEM

```
ElGamal_Decryption (d, p, C1, C2)          // C1 and C2 are the ciphertexts
{
    P ← [C2 (C1^d) ^-1] mod p              // P is the plaintext
    return P
}
```

ELAGAMAL CRYPTOSYSTEM

- Bob(receiver) chooses p = 11 and e1 = 2, and d = 3 , e2 = e1 ^{d} = 8.
- So the public keys are (2, 8, 11) and the private key is 3.
- Alice(sender)chooses r = 4 and calculates C1 and C2 for the plaintext 7.

Plaintext: 7

$C_1 = e_1{}^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2{}^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

Ciphertext: (5, 6)

ELAGAMAL CRYPTOSYSTEM

Bob receives the ciphertexts (5 and 6) and calculates the plaintext.

$[C_2 \times (C_1{}^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$

Plaintext: 7

$= 6 * (125)^{-1} \bmod 11$

$= 6 * 3 \bmod 11$

$= 18 \bmod 11$

$= 7$

ELAGAMAL CRYPTOSYSTEM

- Bob(receiver) chooses p = 19 and e1 = 10, and d = 5 , e2 = ?
- Alice(sender)chooses r = 6 and calculates C1 and C2 for the plaintext 17

ELAGAMAL CRYPTOSYSTEM

Security of Elagamal Cryptosystem

Low Modulus attack

Value of p should be large enough(atleast 1024 bits)

ELAGAMAL CRYPTOSYSTEM

Security of Elagamal Cryptosystem

Known plaintext attack

- If Alice uses the same r to encrypt P and P'.
- Eve discover P' if she knows P.
- Assume $C_2 = P * e_2^r \bmod p$ and $C'_2 = P' * e_2^r \bmod p$
- Eve can find P' using the following steps

1. $(e_2^r) = C_2 \times P^{-1} \bmod p$
2. $P' = C'_2 \times (e_2^r)^{-1} \bmod p$

THANK YOU