CHAPTER - 1

Introduction

Contents

1.1 Introduction

Communication is the process of exchanging information. The most widely used method of communicating over distance is by the use of electrical signal, either over cables or through free space using radio waves. The requirements of information security within an organization have undergone a major change in last two decades. Before the widespread use of data processing equipment, the security of information used to be provided primarily by physical and administrative means. With the introduction of computers, the need of automated tools for protecting files and other information stored in the computer became evident [1, 2, and 3].

With the remarkable advancement of technology and availability of facilities, the interest of people to connect computers to form networks is increasing rapidly. Message can be intercepted by someone during the process of transmission which may cause problem.

In general, there exist following types of problems associated with such data transmission.

- A huge amount of data is to be handled.
- Much of the data is very sensitive to errors.
- The security of data transmitted from source to destination over communication links via different nodes is the most important matter to be worried.

Message can be intercepted by someone during the process of transmission which may cause problem. Hence data security and communication privacy have become a fundamental requirement for such systems.

Encoding a message prior to its transmission is the process of **Data Encryption**. The corresponding **Data Decryption** technique is used to decode the encrypted message. All the research activities of this researcher for the last few years were based on the field of cryptography, involving the planning, developing, designing and analyzing of some bit-level encryption/decryption techniques. Representation of this entire activity is the basic objective of this dissertation.

Section 1.2 of this chapter discusses cryptosystem with its different aspects. The historic background of the field of cryptography is represented in section 1.3. A brief

description of some of the current popular cryptographic techniques is represented in section 1.4. Section 1.5 represents an overview of all the proposed techniques created during this entire research activity. Section 1.6 points out merits of these proposed techniques followed by the concluding remark.

1.2 Cryptosystems

The fundamental objective of cryptography is to enable the transmission of message from one source point to the corresponding destination point over a transmission media in such a manner that the message during its transmission cannot be intercepted by someone [4,5,6,7,18,27,47,48,93,94,95].

An original message is termed as the **plaintext**. The coded message that is to be transmitted is referred to as the **ciphertext**. The process of converting from the plaintext to the ciphertext is known as **enciphering** or **encryption**. Restoring the plaintext from the ciphertext is termed as **deciphering** or **decryption**. The specialized area of study involving many schemes used for enciphering is known as **cryptography**. Such a scheme is known as **cryptographic system** or **cryptosystem** or **cipher**. Techniques used for deciphering a message without any knowledge of enciphering details constitute the area of **cryptanalysis**. The areas of cryptography and cryptanalysis together are called **cryptology** [14, 15, 16, 20, 21, 40, 41, 71, and 92].

The concept of a cryptosystem can be formally defined by the mathematical notation presented in section 1.2.1.

1.2.1 Definition of cryptosystem

A cryptosystem is a 5-tuple (Π,X,K,E,Δ), where the following conditions are satisfied:

1. Π is a finite set of possible plaintexts.
2. X is a finite set of possible ciphertexts.
3. K, the keyspace, is a finite set of possible keys.
4. For each $K \in K$, there exists an encryption rule $e_K \in E$ and a corresponding decryption rule $d_K \in \Delta$. Each $e_K: \Pi \rightarrow X$ and $d_K: X \rightarrow \Pi$ are functions such that $d_K(e_K(x)) = x$ for every plaintext $x \in \Pi$.

The main property is property 4. It says that if a plaintext x is encrypted using e_K, and the resulting ciphertext is subsequently decrypted using d_K, then the original plaintext x results.

Basically there are two types of cryptosystems:

1. **Secret key cryptosystem**
2. **Public key cryptosystem**

Section 1.2.2 discusses the secret key cryptosystem with its different aspects and the different aspects of the public key cryptosystem are described in section 1.2.3.

1.2.2 Secret key cryptosystem

The secret key cryptosystem has the following five ingredients:

- **Plaintext:** This is the original intelligible message or data that is fed into the encryption algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent to the plaintext. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitution and transformation performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and it is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

1.2.2.1 Requirements for secret key cryptosystem

There are two requirements for the secured use of secret key cryptosystem [31, 32, 33, 34, 35, 36, 55, 56, 57, and 75,102].

1. A strong encryption algorithm is needed. At a minimum, the algorithm should be such that an opponent who knows the algorithm and has access to one or more ciphertexts will be unable to decipher the ciphertext or figure out the key.
2. The sender and the receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key will be readable.

In case of the secret key cryptosystem, it is assumed that it is impractical to decrypt a message on the basis of the ciphertext and the knowledge of the encryption/decryption algorithm. Therefore, generally, it is not needed to keep the algorithm secret. One needs to keep only the key secret.

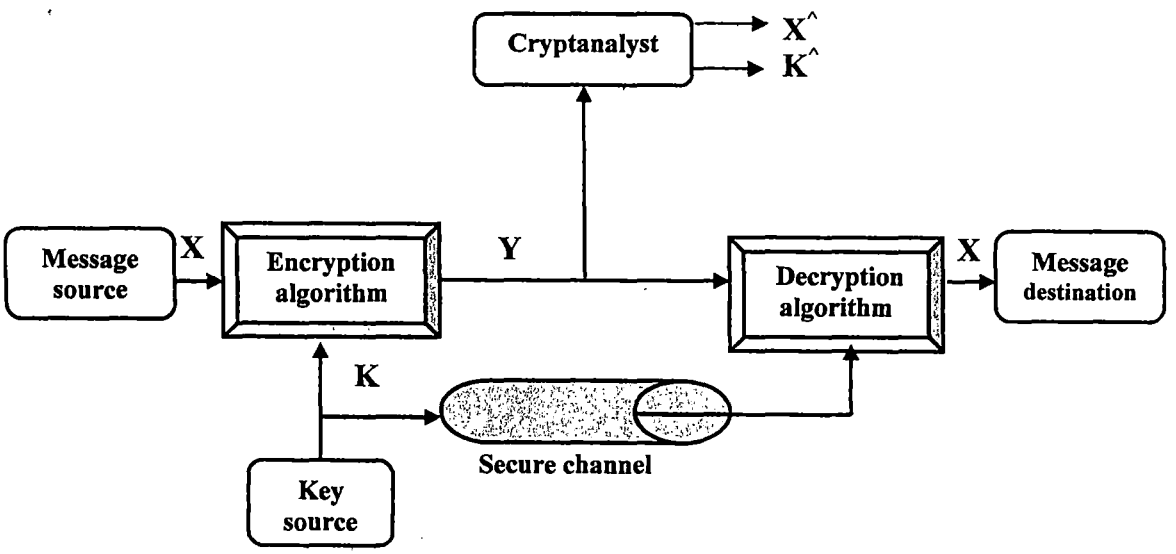

Figure 1.1
Model of secret key cryptosystem

Figure 1.1 shows the typical model of the secret key cryptosystem. A source produces a message in plaintext, $X = [X_1, X_2 \ldots X_M]$. The M elements of X are letters in some finite alphabet. It can be a stream of bits also. For the purpose of encryption, a key of the form $K = [K_1, K_2 \ldots K_j]$ is generated. If the key is generated at the source point, then it must also be provided to the destination point by means of some secure channel.

Alternatively, a third party can generate the key and securely deliver it to both source and destination.

With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext $Y = [Y_1, Y_2, Y_3 \ldots Y_N]$. We can write this as

$$\mathbf{Y = E_K(X)}$$

This notation indicates that Y is produced by using the encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.

The intended receiver, in possession of the key, is able to invert the transformation:

$$\mathbf{X = D_K(Y)}$$

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or the both X and K. It is assumed that the opponent knows the encryption algorithm (E) and the decryption algorithm (D). If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate X^.

1.2.2.2 Evaluating a cryptographic system

A cryptographic system is generally characterized by the following three independent dimensions [8,9,10,43,44,63,72,74,88,114,115].

1. **The type of operations used for transforming plaintext into ciphertext:** All encryption algorithms are based on two general principles. One is **substitution,** in which each element in the plaintext (bit, letter, group of bits, or group of letters) is mapped into another element; and another is **transposition,** in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost, which means that all operations are to be reversible. Most of the cryptographic systems involve multiple stages of substitutions and transpositions.
2. **The number of keys used:** If both the sender and the receiver use the same key, the system is referred to as the **symmetric encryption,** or the **secret key encryption,** or the **conventional**

encryption or the **classical encryption.** If the sender and the receiver use different keys, the system is referred to as the **asymmetric encryption,** or the **two-key encryption,** or the **public key encryption.**

3. **The way in which the plaintext is processed:** The **block cipher** processes the input one block of elements at a time, producing an output block for each input block. A **stream cipher** processes the input elements continuously, producing output one element at a time.

1.2.2.3 Attacking a conventional encryption scheme

There are two general approaches to attack a conventional encryption scheme.

1. **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm plus some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used. If the attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.

2. **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

   For cryptanalytic attacks, the most difficult problem arises when all that is available is the ciphertext only. In some cases, not even the encryption algorithm is known. But, in general, it is assume that the opponent does know the encryption algorithm.

   One possible attack under these circumstances is the brute-force approach of trying all possible keys. If the key space is very large, this becomes impractical. Thus the opponent must rely on an analysis of the ciphertext itself, generally applying various

statistical tests to it. To use this approach, the opponent must have some general idea of the type of plaintext that is concealed.

1.2.2.4 The condition for an encryption scheme to be unconditionally secure

An encryption scheme is **unconditionally secure** if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available. This means, no matter how much time an opponent has, it is impossible to the opponent to decrypt the ciphertext, simply because of the reason that there is no required information [12, 13, 58, 59, and 60].

Except the encryption scheme, known as **one-time pad**, there is no encryption that is unconditionally secure.

In the research field of cryptography, there has been a continuous trend of developing encryption algorithms that are **computationally secure**.

1.2.2.5 The condition for an encryption scheme to be computationally secure

An encryption algorithm is said to be **computationally secure** if the following two criteria are met [10, 22, 23, 99, and 113].

- The cost of breaking the cipher exceeds the value of the encrypted information.
- The time required to break the cipher exceeds the useful lifetime of the information.

Table 1.1
Average time required for exhaustive key search

Key size (bits)	Number of alternative keys	Time required at 1 encryption / μs	Time required at 10^6 encryptions / μs
56	$2^{56} = 7.2 \text{ X } 10^{16}$	2^{55} μs =1142 years	10.01 hours
128	$2^{128} = 3.4 \text{ X } 10^{38}$	2^{127} μs = $5.4 \text{ X } 10^{24}$ years	$5.4 \text{ X } 10^{18}$ years
168	$2^{168} = 3.7 \text{ X } 10^{50}$	2^{167} μs = $5.9 \text{ X } 10^{36}$ years	$5.9 \text{ X } 10^{30}$ years
26 characters (Permutation)	$26! = 4 \text{ X } 10^{26}$	$2 \text{ X } 10^{26}$ μs = $6.4 \text{ X } 10^{12}$ years	$6.4 \text{ X } 10^{6}$ years

If we consider the time required to use a brute-force approach, which simply involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained. On the average, half of all possible keys must be tried to achieve success.

Table 1.1 shows how much time is involved for various key spaces used in some secret key encryption schemes, which will be discussed in brief in section 1.4. Four sample results have been shown in the table.

- The 56-bit key size is used with the DES (Data Encryption Standard) algorithm.
- The 128-bit key size is used with the AES (Advanced Encryption Standard) algorithm.
- The 168-bit key size is used with triple DES.

One result is also shown in table 1.1 for the substitution code that uses a 26-character key.

Now, it is assumed that 1 μs is required for a single decryption, which is a reasonable order of magnitude for today's machines. With the use of massively parallel organization of microprocessors, it may be possible to achieve processing rates many orders of magnitude greater. The final column of table 1.1 considers the results for a system that can process 1 million keys per microsecond (10^6 keys / μs). It is observed from the table that at this performance level, DES can no longer be considered computationally secure.

1.2.2.6 Basic building blocks in classical encryption techniques

In this section, a sampling has been considered what might be called classical encryption techniques. A study and presentation of these techniques enables this researcher to illustrate the basic approaches used in all the proposed techniques [11, 24, 25, 26, 37, 38, 39, 42, 82, 83, 84, 85, 86, 87, and 114,117].

The two basic building blocks of all encryption techniques are:

1. **Substitution techniques**
2. **Transposition techniques**

Section 1.2.1.6.1 discuses on principles of substitution techniques with examples and section 1.2.1.6.2 discusses on principles of transposition techniques, also with examples.

1.2.2.6.1 Principles of substitution techniques

If the plaintext is viewed as a sequence of English alphabets, the substitution technique is one in which the letters of a plaintext are replaced by other letters [17, 18, 19, 68, 70, 97].

In this section, some classical substitution techniques have been discussed from different perspectives clearly indicating the evolution in minimizing the chance of breaking ciphers. These include the following techniques:

- Caesar Cipher
- Monoalphabetic Substitution Cipher
- Homophonic Substitution Cipher
- Playfair Cipher
- Polyalphabetic Cipher (Vigenere Cipher)
- One-time Pad

1.2.2.6.1.1 Caesar cipher

The earliest and the simplest use of the substitution cipher was **Caesar Cipher**, in which each letter of the alphabet in the plaintext is replaced in the ciphertext with the letter standing three places further down the alphabet. The alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

Plain:	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
Cipher:	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
Plain:	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**
Cipher:	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**
Plain:	**u**	**v**	**w**	**x**	**y**	**z**				
Cipher:	**X**	**Y**	**Z**	**A**	**B**	**C**				

The algorithm for Caesar Cipher can be expressed as follows:

$$\mathbf{C = E(p) = (p+3) \bmod (26)}$$

Here p represents the numerical equivalent of a plaintext letter (0 is the numerical equivalent of a, 1 is the numerical equivalent of b, and so on) and C represents the numerical equivalent of a ciphertext letter.

Making the shift of any amount, we get the **General Caesar Algorithm** simply as follows:

$$C = E(p) = (p+k) \bmod (26)$$

Here k can be anything in the range of 1 to 25.

For this general Caesar algorithm, the decryption algorithm is simply as:

$$p = D(C) = (C-k) \bmod (26)$$

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed because there are only 25 keys to try.

1.2.2.6.1.2 Monoalphabetic substitution cipher

From Caesar cipher, a dramatic increase in the key space can be achieved by allowing an arbitrary substitution. This approach results in having as many as 26! = 4 X 10^{26} keys, as shown in table 1.1. Such an approach is referred to as a **monoalphabetic substitution cipher**.

But monoalphabetic ciphers are also easy to break, because they reflect the frequency data of the original alphabet. An enhancement is done using the **Homophonic Substitution Cipher**.

1.2.2.6.1.3 Homophonic substitution cipher

In homophonic substitution cipher, for a single letter multiple substitutes are provided. For example, the letter "e" may be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone used in rotation. If the number of symbols assigned to each letter is proportional to the relative frequency of that letter, then single-letter frequency information is completely obliterated.

However, even with homophones, each element of plaintext affects only one element of ciphertext, making cryptanalysis relatively straightforward.

1.2.2.6.1.4 Playfair cipher

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. Table 1.2 gives an example.

Table 1.2
An example of playfair cipher

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

In this example, the keyword is MONARCHY.

The rules for filling the matrix are as follows:

- The letters of the keyword (minus duplicates) are to be placed from left to right and from top to bottom of the matrix.
- The remainder part of the matrix is to be filled in with the remaining letters in alphabetical order. Any one of the letters I and J is to be used, not the both.

The plaintext is to be encrypted two letters at a time, according to the following set of rules:

- Repeating plaintext letters that would fall in the same pair are separated with a filler letter, such as x, so that "balloon" would be treated as "ba lx lo on".
- Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, "ar" is encrypted as "RM".
- Plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the row circularly following the last. For example, "mu" is encrypted as "CM".
- Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter in the pair. For example, "hs" becomes "BP", "ea" becomes "IM" or "JM".

The Playfair cipher is a great advance over simple monoalphabetic ciphers.

1.2.2.6.1.5 Polyalphabetic cipher

The basic principle of the polyalphabetic cipher is to use different monoalphabetic substitutions as one proceeds through the plaintext message. All the polyalphabetic substitution cipher techniques have the following two things in common:

- A set of related monoalphabetic substitution rules is used.
- A key determines which particular rule is chosen for a given transformation.

The best known and one of the simplest of such algorithms is referred to as **Vigenere Cipher.**

1.2.2.6.1.6 Vigenere cipher

Vigenere cipher consists of a square matrix containing 26 Caesar alphabets. The first row, called row A, is ABCDEF-----XYZ. The next row called row B, is BCDEFGH------YZA. The last row, called row Z, is ZABCDEF----WXY. Using this matrix to encrypt a message, a key is needed, which is a repeating keyword. For example, if the keyword is "decsion", the message "we are discovered save yours" is encrypted as is shown in figure 1.2

The process of encryption is very simple: Given a key letter x and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled x and the column labeled y; in this case the ciphertext letter is V.

For the purpose of encryption, a key is needed that is long as the message. Usually the key is a repeating keyword, as is shown in figure 1.2.

Key

d	e	c	i	S	i	o	n	d	e	c	i	s	i	o	n	d	e	c	i	s	i	o	n

Plaintext

w	e	a	r	E	d	i	s	c	o	v	e	r	e	d	s	a	v	e	y	o	u	r	s

Cipher text

Z	E	C	Z	W	L	W	F	F	S	X	M	J	M	R	F	D	Z	G	J	G	C	F	F

Figure 1.2
Example of vigenere cipher

Decryption also is usually simple. The key letter identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column.

The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus the letter frequency information is obscured.

However, even this scheme is vulnerable to cryptanalysis. Since the key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied.

1.2.2.6.2 Some other substitution ciphers

Besides all that have been discussed, there are few more classical substitution ciphers including:

- **Hill Cipher**
- **Affine Cipher**

1.2.2.7 Principles of transposition techniques

The basic philosophy of the transposition cipher is to keep the plaintext characters unchanged, but to alter their positions by rearranging them.

In this section, one classical transposition cipher systems has been discussed briefly to illustrate the concept.

1.2.2.7.1 The rail fence technique

In this approach, the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

For example, we consider the following plaintext:

The rail fence technique

If it is encrypted with a rail fence of depth 2, we write the following:

t e a l e c t c n q e
h r i f n e e h i u

The resulting encrypted message is:

TEALECTCNQEHRIFNEEHIU

The rail fence technique is trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by

column, but permute the order of the columns. The order of the column then becomes the key of the algorithm.

Following this approach, the following plaintext is considered:

attack postponed until two am

The matrix is constructed as follows, with specifying the key:

Key:	**4**	**2**	**3**	**5**	**1**
Plaintext:	**a**	**t**	**t**	**a**	**c**
	k	**p**	**o**	**s**	**t**
	p	**o**	**n**	**e**	**d**
	u	**n**	**t**	**i**	**l**
	t	**w**	**o**	**a**	**m**

Ciphertext: CTDLMTPONWTONTOAKPUTASEIA

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext.

1.2.2.7.2 A cascaded approach

By performing more than one stage of transposition, this type of transposition technique can be made significantly more secured.

If the same technique is applied on the ciphertext of the previous stage, we get the following structure:

Key:	**4**	**2**	**3**	**5**	**1**
Plaintext:	**c**	**t**	**d**	**l**	**m**
	t	**p**	**o**	**n**	**w**
	t	**o**	**n**	**t**	**o**
	a	**k**	**p**	**u**	**t**
	a	**s**	**e**	**i**	**a**

Ciphertext: MWOTATPOKSDONPECTTAALNTUI

With having 25 letters in the source message, the original sequence of letters is:

01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
13	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
25											

After the first transposition, the following sequence is obtained:

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

It has a somewhat regular structure. But after the second transposition, the sequence obtained is as follows:

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

This is a much less structured permutation and is much more difficult to crypt analyze.

1.2.3 Public key cryptosystem

A public key encryption scheme has the following six ingredients:

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private key:** This is a pair of keys that have been selected so that if one is use for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different messages will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

1.2.3.1 The algorithm

The following essential steps are to be followed in this regard:

1. Each user generates a pair of keys to be used for the encryption and the decryption of messages.

2. Each user places one of the two keys in a public register or any other accessible file. This is the public key. The companion key is kept private. As given in figure 1.4, each user maintains a collection of public keys from others.
3. If the sender wants to send a confidential message to the receiver, he encrypts the message using the public key.
4. When the receiver receives the message, he decrypts it using his private key. No other recipient can decrypt the message because only the receiver known his private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a system controls its private key, its incoming communication is secured. At any time, a system can exchange its private key and publish the companion public key to replace its old public key.

Table 1.3 summarizes some of the important aspects of symmetric and public key encryption.

Table 1.3
Conventional and public key encryption

Conventional encryption	Public key encryption
For functional purpose	**For functional purpose**
1. The same algorithm with the same key is used for encryption and decryption.	**1.** One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.
2. The sender and the receiver must share the algorithm and the key.	**2.** The sender and the receiver must each have one of the matched pair of keys (not the same one)
For security purpose	**For security purpose**
1. The key must be kept secret.	**1.** One of the two keys must be kept secret.
2. It must be impossible or at least impractical to decipher a message if no other information is available.	**2.** It must be impossible or at least impractical to decipher a message if no other information is available.
3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	**3.** Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

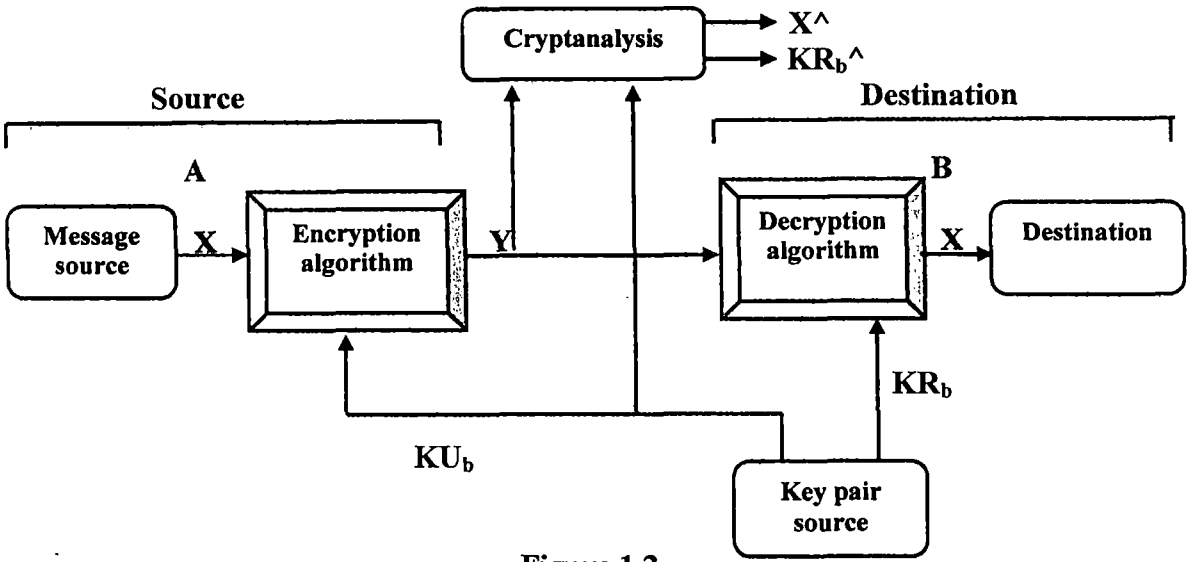

Figure 1.2
Public key cryptosystem

A closer look at the essential elements of a public key encryption scheme is shown in figure 1.2. There is a source A that produces a message in plaintext, say, $X=[X_1, X_2... X_M]$. The m elements of X are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, KU_b, and a private key, KR_b. KR_b is known only to B, whereas KU_b is publicly available and therefore accessible by A.

With the message X and the encryption key KU_b as input, A forms the ciphertext $Y= [Y_1, Y_2, Y_N]$. The intended receiver, in possession of the matching private key, is able to invert the transformation.

An opponent, observing Y and having access to KU_b, but not having access to KR_b or X, must attempt to recover X and/or KR_b. It is assumed that the opponent does have the knowledge of the encryption (E) and the decryption (D) algorithms. If the opponent is interested only in this particular message, then the focus of effort is to recover X, by generating a plaintext estimate X^.

Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover KR_b by generating an estimate KR_b^.

1.2.3.2 Characteristics of public key cryptography:

Following are the requirements to be satisfied for the public key cryptography:

- It should be computationally easy for a party B to generate a pair (public key KU_b and private key KR_b).
- It should be computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext C.
- It should be computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message.
- It should be computationally infeasible for an opponent, knowing the public key, KU_b, to determine the private key, KR_b.
- It should be computationally infeasible for an opponent, knowing the public key, KU_b, and a ciphertext C, to recover the original message, M.

One more point can be added in this regard, which, although useful, is not necessary for all public key applications:

- The encryption and the decryption functions can be applied in either order.

1.3 Evolution in the field of cryptography

The field of cryptography has a curious history. Until the First World War, important developments in this field appeared in a more or less timely fashion and the field moved forward in much the same way like other specialized disciplines [28, 29, 30, and 76].

In 1920, one of the most influential cryptanalytic papers of twentieth century, William F. Friedman's monograph "**The index of Coincidence and its applications in cryptography**", appeared as a first research report of the private Riverbank Laboratories.

In the same year, Edward H. Hebern of Oakland, California, filed the first patent for a rotor machine, the device destined to be a mainstay of military cryptography for nearly 50 years [50, 51, 56, 69, 73, 77, 79, 98, 100,101,105, 106].

After the First World War, things began to change. U.S. army and Navy organizations, working entirely in this field, began to make fundamental advances in cryptography.

During the thirties and forties, a few basic papers did appear in the open literature and several treatises on the subject were published, but the later were farther and farther behind the state of art.

By the end of the war, the transition was complete. With one notable exception, the public literature had died. The exception was Claude Shanon's paper, "**The Communication Theory of Secrecy Systems**", which appeared in the Bell System Technical Journal in 1949.

During the period of 1949 to 1967, the cryptographic literature was fruitless. During this period, a different sort of contribution appeared, which was David Kahn's history, "**The Codebreakers**". It did not contain any new technical ideas, but it did contain a remarkably complete history of what had gone before, including mention of something that the Govt. of USA still considers secret. The significance of the Codebreakers laid not just its remarkable scope, but also in the fact that it enjoyed good sales and made tens of thousands of people aware of cryptography, who had never given the matter a moment's thought.

At about the same time, Horst Fiestel, who had earlier worked on identification of friend or foe devices for the Air Force, took his life long passion for cryptography to the IBM Watson Laboratory in Yorktown Heights, New York. There, he began development of what was to become the US Data Encryption Standard. By early 1970s, several technical reports on this subject by Fiestel and his colleagues had been made public by IBM.

This was the situation when W. Diffie entered the field in late 1972.

In 1975, Hellman and W. Diffie proposed public Cryptography. One of the indirect aspects of their contribution was to introduce a problem that does not even appear easy to solve. This enthused a number of people in Cryptography, the number of meetings held, and number of books and papers published.

W. Diffie told the following statements to the audience in his acceptance speech for the Donald E. Fink award – given for the best expository paper, "**Privacy and Authentication: An Introduction to Cryptography**", to appear in an IEEE Journal – which he received jointly with Hellman in 1980:

"I had an experience that I suspected was rare even among the prominent scholars who populate the IEEE award Ceremony: I had written the paper I had wanted to study, but could not find, when I first became seriously interested in Cryptography. Had I been able to go to the Stanford Bookstore and picked up a modern cryptography text, I would probably have learned about the fields years earlier. But only things available in the fall of 1972 were a few classic papers and some obscure technical reports."

The contemporary researcher has no such problem. The problem now is choosing where to start among the thousands of papers and dozens of books. It has been necessary to spend long hours hunting out and then studying the research literature before being able to design the sort of cryptographic utilities glibly described in popular articles.

This gap has been filled by Bruce Schneier's "**Applied Cryptography**". Bruce Schneier has given a panoramic view of the fruits of 20 years (1976 – 1996) of public research. He has included an account of the world in which Cryptography is developed and applied, and discusses entities ranging from the International Association for Cryptologic Research (IACR) to the National Security Agency (NSA).

In early 80s, NSA made several attempts to control Cryptography and issued a letter to the IEEE mentioning that the publication of Cryptographic material is a violation of the International Traffic Arms Regulations (ITAR). This viewpoint turned out not even to be supported by the regulation of them – which contained an explicit exemption for published material.

1.4 Some of the existing techniques

While discussing cryptosystem and its different aspects in section 1.2, some classical encryption techniques have already been discussed. These include the following techniques:

- Caesar Cipher
- Monoalphabetic Substitution Cipher
- Homophonic Substitution Cipher
- Playfair Cipher
- Polyalphabetic Cipher (Vigenere Cipher)
- One-time Pad

- Rail-Fence Technique
- Rotor Machines

In this section, without going to the detailed discussion, the special characteristics of **The Data Encryption Standard (DES), The Triple Data Encryption Standard (TDES),** the most widely used private cryptosystem in the world, and **The RSA Technique**, the most successful public key cryptosystem, have been pointed out in section 1.4.1, section 1.4.2, and section 1.4.3 respectively.

1.4.1 The Data Encryption Standard (DES)

The DES was adopted in 1977 by the then National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard46 (FIPS PUB 6) [10, 52, 53, 54, 61, 62, 64, 65, 66, 67, 76, 78, 80, 81, 89, 90, 91, 96, 111].

1.4.1.1 Basic principles of DES

1. It is a symmetric block cipher.
2. Data are encrypted in 64-bit blocks using a 56-bit key.
3. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps with the same key are used to reverse the encryption.

1.4.1.2 The basic algorithm

The algorithm proceeds in the following three stages:

1. Given a plaintext x, a bitstring x_0 is constructed by permuting the bits of x according to a fixed initial permutation IP. It can be represented as $x_0 = IP(x) = L_0R_0$, where L_0 comprises the first 32 bits of x_0 and R_0 comprises the last 32 bits.
2. 16 iterations of a certain function are then computed. For $1 \leq i \leq 16$, L_iR_i is computed as follows:

   $L_i = R_{i-1}$

   $R_i = L_{i-1} \oplus f(R_{i-1},K_i)$

   Here $\oplus$ denotes the XOR operation of two bitstrings, f is a function, and K_1, K_2... K_{16} are each bitstring of length 48

computed as a function of the key K. In fact, each K_i is a permuted selection of bits from K. K_1, K_2, ..., K_{16} comprises the key schedule. One round of encryption is depicted in figure 1.5.

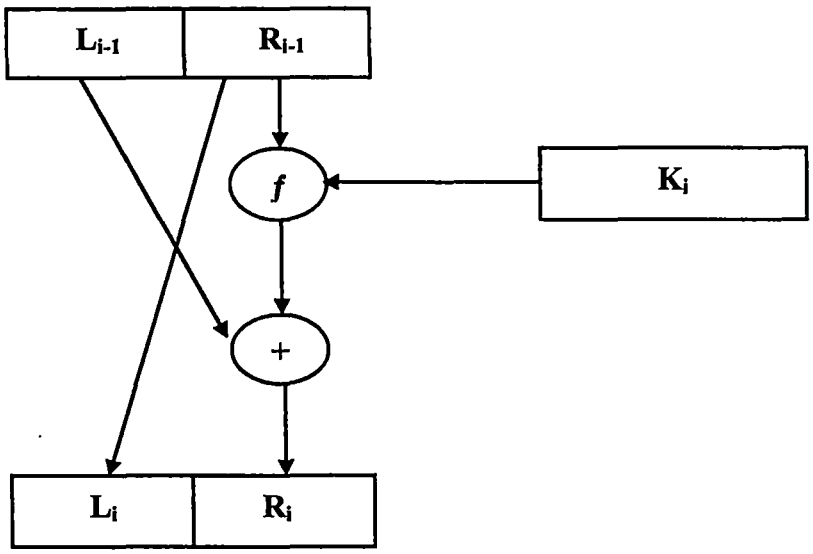

Figure 1.4
One round of DES encryption

3. Apply the inverse permutation IP^{-1} to the bitstring $R_{16}L_{16}$, obtaining the ciphertext y. That is, $y = IP^{-1}(R_{16}L_{16})$.

Figure 1.4 shows the one round encryption.

1.4.1.2.1 The Function *f*

The function *f* takes as input a first argument A, which is a bitstring of length 32, a second argument J that is a bitstring of length 48, and produces as output a bitstring of length 32. The following steps are executed:

1. The first argument A is "expanded" to a bitstring of length 48 according to a fixed expansion function E. E(A) consists of 32 bits from A, permuted in a certain way, with 16 of the bits appearing twice.
2. $E(A) \oplus J$ is computed and the result is to be written as the concatenation of eight 6-bit strings $B = B_1B_2B_3B_4B_5B_6B_7B_8$.

3. The next step uses eight S-boxes $S_1, S_2 \ldots S_8$. Each S_i is a fixed 4 x 16 array whose entries come from the integers 0 – 15.

   Given a bitstring of length six, say, $B_j = b_1b_2b_3b_4b_5b_6$; S_j (B_j) is to be computed as follows:

   The two bits b_1b_6 determine the binary representation of a row r of S_j $(0 \leq r \leq 3)$, and the four bits $b_2b_3b_4b_5$ determine the binary representation of a column c of S_j $(0 \leq c \leq 15)$. Then S_j (B_j) is defined to be the entry $S_j(r, c)$, written in binary as a bitstring of length four.

   Hence each S_j can be thought of as a function that accepts as input a bitstring of length two and one of length four, and produces as output a bitstring of length four.

   In this fashion, C_j can be calculated as $C_j = S_j (B_j)$, $1 \leq j \leq 8$.

4. The bitstring $C = C_1C_2C_3C_4C_5C_6C_7C_8$ of 32 is permuted according to a fixed permutation P. The resulting bitstring P(C) is defined to f (A, J).

1.4.1.3 Application

It can be implemented very easily, either in hardware or in software. One important application of DES is in banking transaction, using standards developed by the American Bankers Association. DES is used to encrypt personal identification numbers (PINs) and account transactions carried out by automated teller machines (ATMs). DES is also used by the Clearing House Interbank Payments System (CHIPS) to authenticate transactions involving over $\$1.5 \times 10^{12}$ per week.

1.4.1.3.1 Modes of operations

There are the following four modes of operations that have been developed for DES:

1. Electronic Codebook Mode (ECB)
2. Cipher Feedback Mode (CFB)
3. Cipher Block Chaining Mode (CBC)
4. Output Feedback Mode (OFB)

A detailed discussion on the scheme for each of these modes is excluded from this section.

1.4.1.4 The DES controversy

When DES was proposed as a standard, there was a considerable criticism. One objection to DES concerned the S-boxes. All computations in DES, with the exception of the S-boxes, are linear. The S-boxes, being the non-linear component of the cryptosystem, are vital to its security. However, the design criteria of the S-boxes were not completely known until 1976, when the National Security Agency (NSA) asserted a set of properties as the design criteria of the S-boxes.

The most pertinent criticism of DES is that the size of the keyspace, 2^{56}, is too small to be really secured [47, 48, and 55].

1.4.2 The TDES algorithm

After the DES criticism TDES was introduced. In Triple-DES, two or three keys are supplied. The plain text is encrypted with the first, "decrypted" with the second, and encrypted with the third or with the first again if no third key has been supplied. The second action is decryption rather than encryption so that a hardware implementation of Triple-DES can be made compatible with a hardware implementation of single DES simply by supplying the same key three times.

1.4.3 The RSA technique – The special characteristics

The first realization of a public key system came in 1977 by Rivest, Shamir, and Adleman, who invented the well-known RSA Cryptosystem [45, 46, 49, 103, 104, 107, 108, 109, 110, 112, 116].

1.4.3.1 The RSA algorithm

To create an RSA public/private key pair, following are the basic steps:

1. Choose two prime numbers, p and q. Calculate n = pq.
2. Select a third number, e, which is relatively prime to the product (p-1) (q-1). The number e is the public exponent.
3. Calculate an integer d from the quotient (ed-1)/ [(p-1) (q-1)]. The number d is the private exponent.

4. The public key is the number pair (n, e). Although these values are publicly known, it is computationally infeasible to determine d from n and e, if p and q are large enough.
5. To encrypt a message, M, with the public key, create the ciphertext, C, using the equation: $\mathbf{C = M^e \bmod n}$.
6. The receiver then decrypts the ciphertext with the private key using the equation: $\mathbf{M = C^d \bmod n}$.

Figure 1.5 depicts the key generation by a pictorial representation.

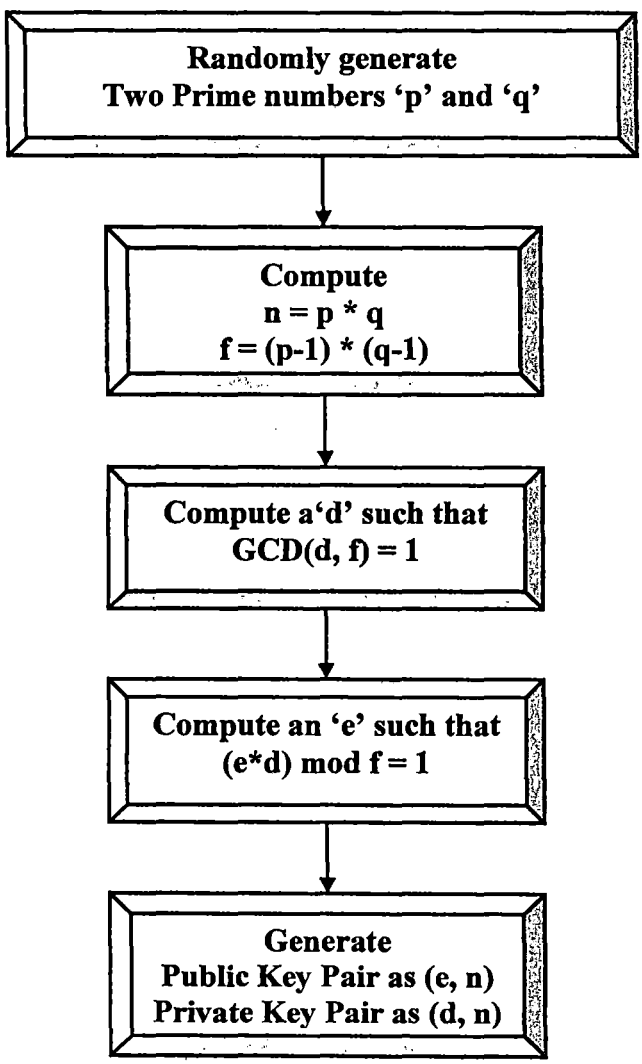

Figure 1.5
Key generation in RSA

Figure 1.6 and figure 1.7 respectively show the pictorial representations of the encryption and the decryption techniques in the RSA technique.

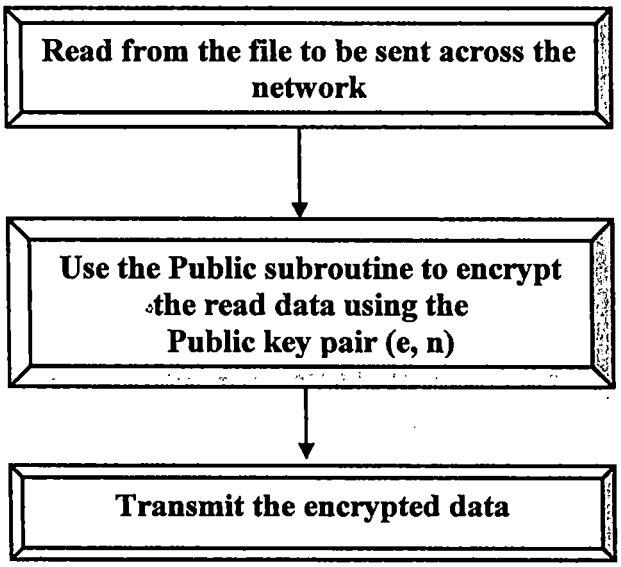

Figure 1.6
Encryption in RSA technique

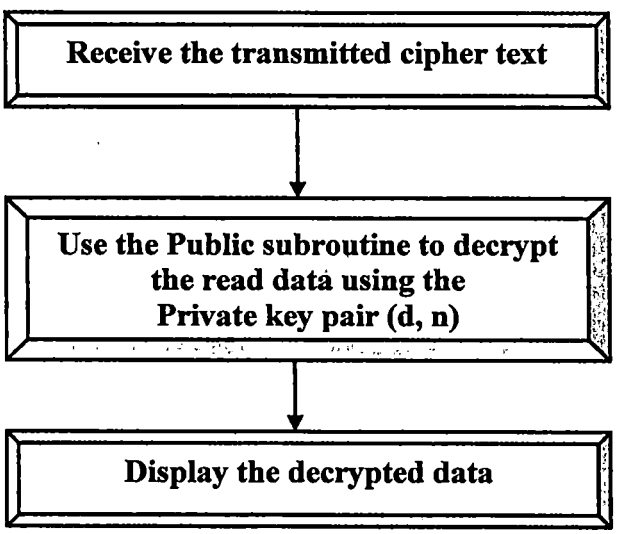

Figure 1.7
Decryption in RSA technique

1.4.3.2 The security of RSA

Possible approaches to defend against at least two possible attacks on the RSA algorithm are as follows:

1. **Brute force:** The defense against the brute force approach is the same for RSA as for other cryptosystems, namely, use a large key space. Therefore it is better to take as larger as possible number of bits in e and d. However, because the calculations involved, both in key generation and in encryption/decryption, are complex, the larger the size of the key, the slower the system will run.
2. **Mathematical attacks:** To avoid values of n that may be factored more easily, the inventors of the algorithm suggest the following constraints on p and q:
   - Lengths of p and q should differ by only a few digits. For example, for a 1024-bit key (309 decimal digits), both p and q should be on the order of magnitude of 10^{75} to 10^{100}.
   - Both (p-1) and (q-1) should contain a large prime factor.
   - The greatest common divisor (gcd) of (p-1) and (q-1) should be small.

1.5 An overview of proposed techniques

This section provides an overall idea on all the techniques proposed in the thesis. Following are the proposed names of the set of independent techniques developed during the research work:

1. **Recursive Carry Addition (RCA)**
2. **Recursive Key Rotation (RKR)**
3. **Recursive Session Key Arithmetic (RSKA)**
4. **Cascaded Arithmetic Operation on Pair of Bits of Streams (CAOPB)**
5. **Recursive Modulo-2 Operation of Paired Bits of Streams (RMOPB)**
6. **Cascaded Recursive Key Rotation of a session key with Transposition and Addition of Blocks (CRKRTAB)**

Before pointing out the overview of each technique separately, a combined overview of all the techniques has been summarized in table 1.4. The table points out schematic characteristics of the techniques in terms of the following attributes:

- Whether bit-level implementation or not
- Whether public key system or private key system
- Whether techniques of encryption and decryption exactly the same (symmetric)
- Whether block cipher or stream cipher
- Whether a substitution technique or a transposition technique
- Whether the basic operation is Boolean or Non-Boolean
- Whether there is data compression/expansion or no alteration in size
- Whether there is any formation of cycle or not
- Whether the technique cascaded with more than one techniques

Table 1.4
Schematic characteristics of proposed techniques

	RCA	RKR	RSKA	CAOPB	RMOPB	CRK RTAB
Implementation in bit-level	√	√	√	√	√	√
Implementation not in bit-level						
Public key system						
Private key system	√	√	√	√	√	√
Symmetric	√	√	√	√	√	√
Block cipher		√	√	√	√	√
Stream cipher						
Substitution technique	√	√	√	√	√	√
Transposition technique						√
Boolean as basic operation		√	√	√	√	√
Chance of alteration in size				√		
No alteration in size	√	√	√		√	√
Formation of cycle	√	√	√	√	√	√
No formation of cycle						
Cascaded with more than one technique						√

From table 1.4, the following attributes that are common to all the proposed techniques:

- **Implementation in bit-level**
- **Private key system**
- **Symmetric**
- **Block cipher**
- **Formation of cycle**

For all the techniques there is no chance of any alteration (increase) in size while encrypting the source file except CAOPB technique.

Since all the techniques are block ciphers and have to be implemented in bit-level, the source file to be encrypted is to be converted into a stream of bits and the whole stream is to be decomposed into a finite number of blocks before the actual scheme is to be implemented for each block.

Section 1.5.1 to section 1.5.6 gives an overview for all the proposed techniques. Section 1.5.7 presents a brief proposal on the key structures of different proposed techniques. Section 1.5.8 points out the factors considered in the thesis for each of the proposed techniques for the purpose of evaluation.

1.5.1 An overview of the RCA technique [118]

This technique considers the plaintext as block of bits with different size like 4/ 8 /16 /32 /64 /128 /256. The rules to be followed for generating a cycle are as follows:

1 Consider any source stream of a finite number (where $N=2^n$, n =2 to 8).

1. Add 1(one) to the left most bit of the string with binary addition method. If there is carry, add this carry to the second bit of the source string and continue up to the last bit to get the first intermediate block.
2. Again add 1(one) to the second bit of the source stream with the binary addition method, and get the second intermediate block with the same carry addition to its respective places, continued at the last.
3. This addition process is continued till the last bit of the given string. If there is a carry at the last bit then simply add that carry to the first place of the respective iteration and continued up to the last.

This process is repeated until the source stream is generated. After a finite number of iteration the source stream is regenerated. Any intermediate block may be considered as the encrypted block for the technique.

It is a kind of block cipher and symmetric in nature hence, decoding is done following the same procedure. A comparison of the proposed technique with existing and industrially accepted RSA/TDES has also been done in terms of frequency distribution and homogeneity of source and encrypted files.

Chapter 2 of the thesis discusses the RCA technique in detail from different perspectives.

1.5.2 An overview of the RKR technique [119]

This technique considers the plaintext as block of bits with different size like 4/ 8 /16 /32 /64 /128 /256. The rules to be followed for generating a cycle are as follows:

1 Consider any source stream of a finite number (where $N=2^n$, n =2 to 8) and divide it into two equal parts.

2 Consider any key value (key= 2^n, where n=1 to 7) depends upon the source stream that is, key value is the half of the source stream).

3 Make the modulo-2 addition (X-OR) with the key value to the first half of the source stream, to get the first intermediate block.

4 Make the modulo-2 addition (X-OR) with the key value (but now the key value is reversed) to the last half of the source stream to get the second intermediate block.

This process is repeated until the source stream is generated. After a finite number of iteration the source stream is regenerated. Any intermediate block may be considered as the encrypted block for the technique.

It is a kind of block cipher and symmetric in nature hence, decoding is done following the same procedure. A comparison of the proposed technique with existing and industrially accepted RSA/TDES has also been done in terms of frequency distribution and homogeneity of source and encrypted files.

Chapter 3 of the thesis discusses the RKR technique in detail from different perspectives.

1.5.3 An overview of the RSKA technique [120]

The technique considers the plaintext as block of bits with different size like 4/ 8 /16 /32 /64 /128 /256. The rules to be followed for generating a cycle are as follows:

Step 1: Apply step 3 and step 4 exactly L number of times, for the values of the variable P ranging from 0 to (L-1) increasing by 1 after each execution of the loop.

Step 2: Apply arithmetic operation on the first two bit of the source stream with the two bit fixed key to get the first intermediate block.

Step 3: Apply Arithmetic operation again on the second two bit of the source stream with the two bit fixed key to get the second intermediate block.

When this process is continued, source block may be generated after a finite number of iteration. Any intermediate block may be considered as the encrypted block for the technique. It is a kind of block cipher and symmetric in nature hence, decoding is done following the same procedure.

A comparison of the proposed technique with existing and industrially accepted RSA/TDES has also been done in terms of frequency distribution and homogeneity of source and encrypted files.

Chapter 4 of the thesis discusses the RSKA technique in detail from different perspectives.

1.5.4 An overview of the CAOPB technique [121]

A stream of bits is considered as the plaintext. Like in other proposed techniques, the plaintext is divided into a finite number of blocks, each having a finite fixed number of bits like 8/ 16/ 32/ 64/ 128/ 256. The CAOPB technique is then applied for each of the blocks in the following way. The rules to be followed for generating cycles are as follows:

1. Consider any source stream of a finite number (where $N=2^n$, n =3 to 8) and divide it into two equal parts.
2. Make the source stream into paired form so that a pair can be used for the operation.

3. Make the modulo-2 addition (X-OR) between the first and second pair, second and third pair, third and fourth pair of the source stream, to get the first intermediate block.

This process is repeated until the source stream is generated. After a finite number of iteration the source stream is regenerated. Any intermediate block may be considered as the encrypted block for the technique. It is a kind of block cipher and symmetric in nature hence, decoding is done following the same procedure.

A comparison of the proposed technique with existing and industrially accepted RSA/TDES has also been done in terms of frequency distribution and homogeneity of source and encrypted files.

Chapter 5 of the thesis discusses the CAOPB technique in detail from different perspectives.

1.5.5 An overview of the RMOPB technique [122]

The technique, considers the plaintext as a stream of finite number of bits N, and is divided into a finite number of blocks, each also containing a finite number of bits n, where, $1 <= n <= N$.

Let $P = s^0_0\, s^0_1\, s^0_2\, s^0_3\, s^0_4 \ldots s^0_{n-1}$ is a block of size n in the plaintext. Then the first intermediate block $I_1 = s^1_0\, s^1_1\, s^1_2\, s^1_3\, s^1_4 \ldots s^1_{n-1}$ can be generated from P in the following way:

$$s^1_0\, s^1_1 = s^0_0\, s^0_1 \oplus s^0_2\, s^0_3$$

$$s^1_2\, s^1_3 = s^0_0\, s^0_1 \oplus s^0_4\, s^0_5$$

$s^1_i\, s^1_{j+1} = s^0_{i-j}\, s^0_{i-j+1} \oplus s^0_{i+j+2}\, s^0_{i+j+3}$, $0 <= i < (n-1)$, $0 <= j < (n-1)$; $\oplus$ stands for the exclusive-OR operation.

In the same way, the second intermediate block $I_2 = s^2_0\, s^2_1\, s^2_2\, s^2_3\, s^2_4 \ldots s^2_{n-1}$ of the same size (n) can be generated by:

$$s^2_0 s^2_1 = s^1_0\, s^1_1 \oplus s^1_2\, s^1_3$$

$$s^2_2\, s^2_3 = s^1_0\, s^1_1 \oplus s^1_4\, s^1_5$$

$s^2_i\, s^2_{j+1} = s^1_{i-j}\, s^1_{i-j+1} \oplus s^1_{i+j+2}\, s^1_{i+j+3}$, $0 <= i < (n-1)$, $1 <= j < (n-1)$;$\oplus$ stands for the exclusive-OR operation.

If this process continues for a finite number of iterations, the source block P is regenerated forming a cycle, which depends on the value of block size n.

Any intermediate block in the recursive process may term as intermediate encrypted block for that source block. The operation is repeated for the whole stream in the source.

The same operation is performed for whole stream number of time with a varying block sizes. k such iteration is done and the final intermediate stream after k iterations generates the encrypted stream. All of the different block sizes and k constitute the key for the session. This key may be considered as session key for that particular session.

1.5.6 An overview of the Cascaded Recursive Key Rotation of a session key with Transposition and Addition of Blocks (CRKRTAB) [123]

This technique operates in three phases:

a. First phase encrypt the plaintext using Recursive Key Rotation

The technique considers the plaintext as a stream of finite number of bits in the form of blocks with different size like 8/ 16/ 32/ 64/ 128/ 256. The rules to be followed for generating a cycle are as follows:

1. Consider any source stream of a finite number (where $N=2^n$, n =3 to 8) and divide it into two equal parts.
2. Consider any key value (key= 2^n, where n=1 to 7) depends upon the source stream that is, key value is the half of the source stream).
3. Make the modulo-2 addition (X-OR) with the key value to the first half of the source stream, to get the first intermediate block.
4. Make the modulo-2 addition with the key value (but now the key value is reversed) to the last half of the source stream to get the second intermediate block.

b. Second phase encrypt the output of the first phase by Recursive Transposition of Blocks

The technique, considers the encrypted message from the first phase as blocks of bits with different size like 8/ 16/ 32/ 64/ 128/ 256. The bit swapping can be applied to each block separately. The principle of bit transposition is discussed in following for different block size.

a. Swapping on 8 bit:

Considering a block (X) with eight binary bits, we have

X = a b c d e f g h

X 1 = c d a b g h e f after 2 bit transposition on X

X 2 = g h e f c d a b after 4 bit transposition on X_1

X 3 = e f g h a b c d after 2 bit transposition on X_2

X 4 = a b c d e f g h after 4 bit transposition on X_3

b. Swapping on 16 bit:

Considering a block (X) with eight binary bits, we have

X = a b c d e f g h i j k l m n o p

X 1 = c d a b g h e f k l i j o p m n after 2 bit transposition on X

X 2 = g h e f c d a b o p m n k l i j after 4 bit transposition on X_1

X 3 = o p m n k l i j g h e f c d a b after 8 bit transposition on X_2

X 4 = m n o p i j k l e f g h a b c d after 2 bit transposition on X_3

X 5 = i j k l m n o p a b c d e f g h after 4 bit transposition on X_4

X 6 = a b c d e f g h i j k l m n o p after 8 bit transposition on X_5

8 bit, 16 bit, transpositions are applied on the block and the same process is followed to get back the original block. Swapping on 32, 64, 128 and 256 follows the same principle as described above.

c. Third phase encrypt the output of the second phase by Recursive Addition of Blocks

The technique, considers the encrypted message from the second phase as a stream of finite number of bits N, and is divided into a finite number of blocks, each also containing a finite number of bits n, where, $1 <= n <= N$.

Let $P = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4 \ldots s^0_{n-1}$ is a block of size n in the plaintext. Then the first intermediate block $I_1 = s^1_0\ s^1_1\ s^1_2\ s^1_3\ s^1_4 \ldots s^1_{n-1}$ can be generated from P in the following way:

$s^1_0 = s^0_0 \oplus s^0_1$

$s^1_{n-1} = s^0_{n-1}$

$s^1_i = s^0_i \oplus s^0_{i+1}$, $1 < i < (n-1)$; $\oplus$ stands for the exclusive-OR operation.

In the same way, the second intermediate block $I_2 = s^2_0\ s^2_1\ s^2_2\ s^2_3\ s^2_4 \ldots s^2_{n-1}$ of the same size (n) can be generated by:

$s^2_0 = s^1_0 \oplus s^1_1$

$s^2_{n-1} = s^1_{n-1}$

$s^2_i = s^1_i \oplus s^1_{i+1}$, $1 < i < (n-1)$. $\oplus$ stands for the exclusive-OR operation.

Any intermediate block in the recursive process may term as intermediate encrypted block which can be used as cipher text for security for 256 bit block string. The same operation is performed for whole stream number of time with a varying block sizes. K such iteration is done and the final intermediate stream after k iterations generates the cascaded form of the encrypted stream. All of the different block sizes and k constitute the key for the session. This key may be considered as session key for that particular session. This process is repeated until the source stream is generated.

Chapter 7 of the thesis discusses the CRKRTAB technique in detail from different perspectives.

1.5.7 A proposal on key structures for proposed techniques

Before proposing key structures of different proposed techniques, all the proposed techniques are categorized in following three ways [37, 38, 50, 51, 61, 90, and 96].

- **Block cipher with direct block-to-block conversion:** Technique(s) in which the task of encryption is done block wise and for a block of length N, all the N bits of the corresponding encrypted block are placed contiguously, so that the one-to-one relationship between the source block and the encrypted block can be established.

  Example: The RCA, and the RSKA techniques

- **Block cipher with non-contiguous bit-allocation:** Technique(s) in which the task of encryption is done block wise but for a block of length N, different resultant bits are not placed contiguously, so that no one-to-one or one-to-many relationship between the source block and the corresponding encrypted block can be established.

Example: The RKR, and the CAOPB techniques

- **Block cipher with repeated block-to-block conversion:** Technique(s) in which the task of encryption is done block wise and for a block of length N, after a finite number of specific iterations, the source block is regenerated, so that a cycle is formed. The one-to-many relationship between the source block and the encrypted block can be established as any of the intermediate blocks can be considered as the corresponding encrypted block.

  Example: The RMOPB and the CRKRTAB techniques

1.5.7.1 Proposed key structure for block cipher with direct block-to-block conversion

To ensure a better security, varying lengths can be chosen for different blocks, and accordingly, the session key is to be structured, so that from the key, one gets the information regarding the lengths of different blocks. Once this information becomes available, the task of decryption can be performed easily to generate the exact source stream of bits. Naturally, this key is to be kept absolutely secret and hence the technique in this category is a **private key system**. Having different block lengths causes the key space to be reasonably large.

1.5.7.2 Proposed key structure for block cipher with non-contiguous bit-allocation

Here the scenario is the same as described in section 1.5.7.1. Although the techniques choosing varying block lengths will cause the process of encryption as well as decryption a bit more complicated, but, it can be handled properly, and this can produce the security of the highest level. Obviously, the key will consist of the information on the different blocks and, hence, it should be kept secret. As a result, these are **private key systems.**

1.5.7.3 Proposed key structure for block cipher with repeated block-to-block conversion

In this case, for each of the blocks its length as well as the identification value of the intermediate block considered as the encrypted block is to be placed in the key.

The key is to be made secret, so that the techniques falling under this category are **private key systems.**

More about the key structures of these different proposed techniques are discussed and analyzed in the respective chapters and also in chapter 8, which is especially based on the key distribution.

1.5.8 Factors considered for evaluating proposed techniques

Several factors have been considered to evaluate the proposed techniques. These include the following:

- **Frequency distribution test**
- **Chi square test**
- **Analysis of the key space**
- **Computation of the encryption/decryption time**
- **Comparison of performance in terms of Chi square values with the RSA/TDSA technique**

1.5.8.1 Frequency distribution test

Frequency distribution test is considered to asses the degree of security of the proposed techniques against the cryptanalytic attack. Through this test, performed simultaneously on the original as well as the encrypted files, the frequencies of all 256 characters in two files are compared graphically. These are shown in two different graphs. It is the objective of the frequency distribution test to show that there exists no fixed relationship between the frequency of a character in the source file and that of the same character in the corresponding encrypted file, and that the characters are well distributed in the encrypted file [31, 32, 33, and 39].

1.5.8.2 Chi square test

Through the chi square test performed between the original and the encrypted files, the non-homogeneity of the two files is tested [116].

The "**Pearsonian Chi-square test**" or the "**Goodness-of-fit Chi-square test**" has been performed here to decide whether the observations onto encrypted files are in good agreement with a hypothetical distribution, which means whether the sample of encrypted files may be supposed to have arisen from a specified population. In this case, the chi-

square distribution is being performed with (256-1)=255 degrees of freedom, 256 being the total number of classes of possible characters in the source as well as in the encrypted files. If the observed value of the statistic exceeds the tabulated value at a given level, the null hypothesis is rejected.

The "Pearsonian Chi-square" or the "Goodness-of-fit Chi-square" is defined as follows:

$$X^2 = \Sigma \{(f_0 - f_e)^2 / f_e\}$$

Here f_e and f_0 respectively stand for the frequency of a character in the source file and that of the same character in the corresponding encrypted file.

On the basis of this formula, the Chi-square values have been calculated for sample pairs of source and encrypted files.

1.5.8.3 Analysis of the key space

The key space plays an important role in attempting to tackle the Brute-force attack successfully. The key space of each technique has been attempted to enlarge reasonably to make the techniques computationally secure.

1.5.8.4 Computation of encryption/decryption time

The result of the encryption/decryption time plays an important role in assessing the efficiencies of the algorithms from the execution point of view. Here it has been attempted to establish a relationship between the size of the file being encrypted and the encryption/decryption time.

Now, the time consumed in encrypting and decrypting files is related to the code written for that purpose and the architecture of the machine where the code is being executed. All the results in this regard shown in different chapters are taken after compiling and executing C codes in a machine of the following configuration:

Mother Board : **810T**

CPU : **1.2 GHz Celeron**

RAM : **256 MB**

HDD : **40GB**

Now, the same code, if is executed on a machine of a different configuration, will require different amounts of time to encrypt and decrypt. Also, if the code is written in a different approach, it may offer different execution time even if it is run on the same

machine. But such changes do not produce any change in the relationship between execution time and the source file size. Still there will exist the "almost linear nature" of the relationship.

1.5.8.5 Comparison of performance with the RSA /TDES technique

By comparing the performance of the proposed techniques with the RSA and TDES technique, it is attempted to evaluate the proposed techniques with respect to the existing field of cryptography. The most popular public key system, the RSA technique and the private key system TDES, has been considered here as the model and the comparative analysis is done on the basis of frequency distribution and chi square distribution for the purpose of evaluation.

1.6 A note on merits of proposed techniques:

All the six independent techniques included in this thesis offer security of highly satisfactory level. The statistical results have been shown in the respective chapters using the frequency distribution test and the chi square test. For each technique, a reasonably large key space has been proposed, so that by the brute-force attack the chance of breaking the cipher by key estimation becomes computationally infeasible. Moreover, implementation of each of the algorithms is well tested with satisfactory performance. The execution time varies with the size of the file being encrypted. Only one out of the six proposed techniques causes alteration in size of file after being encrypted. But, in turn, this alteration in size helps in ensuring a better security and space efficiency also.

Following is the list of all the chapters in which all the proposed independent techniques have been presented:

- Chapter 2 entitled, ***Encryption Through Recursive Carry Addition (RCA)***, based on the RCA technique
- Chapter 3 entitled, ***Encryption Through Recursive Key Rotation (RKR) Technique*** , based on the RKR technique
- Chapter 4 entitled, ***Encryption Through Recursive Session Key Arithmetic (RSKA) Technique***, based on the RSKA technique

- Chapter 5 entitled, ***Encryption Through Cascaded Arithmetic Operation on Pair of Bits of (CAOPB)***, based on the CAOPB technique
- Chapter 6 entitled, ***Encryption Through Recursive Modulo-2 Operation of Paired Bits of Streams (RMOPB)***, based on the RMOPB technique
- Chapter 7 entitled, ***Cascaded Recursive Key Rotation of a session key with Transposition and Addition of Blocks (CRKRTAB,)*** based on CRKRTAB technique

The structure of the key plays the most important role, as all the proposed techniques are private key systems. The following chapter discusses this aspect:

- Chapter 8 entitled, ***Formation of Secret Key***, based on suitable key structure for the proposed techniques

Apart from introducing six independent cryptographic techniques, there is also one proposal to implement these techniques in cascaded manner. The following chapter shows how this cascading can be performed tactfully:

- Chapter 9 entitled, ***Encryption Through Cascaded Implementation of the Proposed Techniques***, based on the cascaded approach

Finally, a conclusive discussion on the entire development work, including the final assessment of all the proposed cryptographic systems, is done in the following chapter:

- Chapter 10 entitled, ***A Conclusive Discussion***, based on the conclusion on the entire work

The references are included in Appendix A, List of publications of the candidate are included in Appendix B, Source codes of proposed implemented techniques are included in Appendix C, List of tables of proposed implemented techniques are included in Appendix D, List of figures of proposed implemented techniques are included in Appendix E, List of graphs of proposed implemented techniques are included in Appendix F.