TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
CENTRAL DEPARTMENT OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
KIRTIPUR, KATHMANDU
NEPAL

SYMMETRIC ENCRYPTION ALGORITHM USING CODE REUSE TECHNIQUE FOR AUTHENTICATION BASED ON NEEDHAM SCHROEDERS' PROTOCOL

By

THANESHOR PRASAD PANERU

A dissertation submitted to the Central Department of Computer Science and Information Technology in partial fulfillment of the requirements for the Master's Degree in Computer Science and Information Technology

Date: Feb 12 - 2011

TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
CENTRAL DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KIRTIPUR, KATHMANDU
NEPAL

CERTIFICATION

Mr. Thaneshor Prasad Paneru has carried out this research work entitled "SYMMETRIC ENCRYPTION ALGORITHM USING CODE REUSE TECHNIQUE FOR AUTHENTICATION BASED ON NEEDHAM SCHROEDERS' PROTOCOL" under my supervision and guidance. In my best knowledge this is an original work in computer science. I, therefore, recommend for further evaluation.

………………………..

Dr. Tanka Nath Dhamala

Head
Central Department of Computer Science
and Information Technology
Tribhuvan University
Kirtipur, Kathmandu
Nepal

TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
CENTRAL DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KIRTIPUR, KATHMANDU
NEPAL

LETTER OF APPROVAL

We certify that we have read this research work and in our opinion, it is satisfactory in the scope and quality as dissertation in the partial fulfillment for the requirement of the Master's Degree in Computer Science and Information Technology.

Evaluation Committee

………………………..	………………………..
Dr. Tanka Nath Dhamala	**Prof. Dr. Jivanjyoti Nakarmi**
(Supervisor)	(Act. Head)
Head of the Department	CDCSIT
CSCSIT	Tribhuvan University
Tribhuvan University	

………………………..	………………………..
Internal Examiner	**External Examiner**

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Dr. T.N. Dhamala, for his guidance, encouragement, and optimism. His patience, support, and confidence have been the driving force of this thesis work. My gratitude also to my Professors Suvarna Sakya, Sashidhar Ram Joshi, Sudarshan Karanjeet and my teachers Min Bahadur Khati, Samujjwal Bhandari for extremely insightful knowledge discussions on various topics. Furthermore, I had the pleasure of working very closely with Dr. A Antonysamy S J, Vishnu K. Rana, Jitendra Manandhar, Kamal Raj Sharma, Rajan Karmacharya and Rajeev Nakarmi; from whom I received extensive assistance.

I am thankful to all members of evaluation committees for going through such a long document and giving me valuable feedback. I would also like to acknowledge the support and friendship I received from so many friends over the years in school and college: Shiv Raj Badu, Kriti Bistha, Ganesh Chataut, Narendra Bohara, Prakash Saud, Thaneshwar Paneru, Padam Bhandari, Kamala Paneru and many others that I forgot to mention here. Special thanks go to my former roommate Padam Shahi and my sister Khageshori Paneru for assisting me at the kitchen.

Finally, for all the support, love, and understanding they have given me throughout the years, I wish to thank my parents and my family members. I wish to thank my parents and my teachers for their encouragement, wonderful and loving support, to have faith in me for the preparation of this work in every step of its way.

Thaneshor Prasad Paneru

Kathmandu, Nepal

December, 2010

ABSTRACT

Secure data transmission is a significant problem in human history. Following the problem, security is one of the flourishing areas in the field of computer science and information technology which deals with prevention and protection of assets, both logical (data) and physical (hardware), from unauthorized access, use, alteration, degradation, destruction and other threats.

Many encryption algorithms have come and gone as cryptography, cryptanalysis, and technology have progressed. Today’s communication and computer technologies need cryptography to truly secure data in many applications. The demands on the cryptography needed for some commercial applications will exceed the security offered by the National Bureau of Standards Data Encryption Standard (DES) in the near future due to advances in technology, advances in cryptanalysis, and the increasing rewards for breaking such a heavily used algorithm. To meet part of this need, a new block encryption algorithm is proposed. This algorithm is implemented using C-programming language. One way to further increase security of encrypted data, as well as to achieve storage and/or transmission economy, is by redundancy reduction prior to encryption. A linguistic approach to redundancy reduction, together with an example computer program to implement it, is given for this purpose.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Secure data communication is a significant problem. The increasing proliferation of digital communication and computer data base storage has brought the increase in difficulty of maintaining the privacy and security of that data [7]. There is only one effective way to protect the privacy of communications using encryption [5]. It is impossible to deny unauthorized access by a determined and knowledgeable interceptor to the communications, but it is possible to render the communications totally unintelligible to all but the intended receivers [32]. A lot of schemes have been proposed and are in use. Many of these can be broken with few resources by someone else. Cryptology is the study of method of hiding secret in which trying to figure out the secrets that someone else has hidden is known as cryptanalysis. History reveals many examples of cryptology that worked, and that did not [13].

Successful cryptanalysis depends on taking advantage of as many of the following as are available to the cryptanalyst [33]

1. Taking advantage of the redundancy in any natural language to determine the validity of assumptions.
2. Clues gained from corresponding plain and cipher text.
3. Information that might be known about the algorithms used.
4. The general expected content of the cryptograms.
5. All of the cipher text that is available in the same system and key.
6. Compromised keys.
7. As much computational and analytical power as can be obtained.
8. Mistakes made in the users of the cryptographic system.

The cryptographer can make life as difficult as possible for the cryptanalyst by depriving him of some of these things [23, 36].

1. Using redundancy reduction before encryption.
2. Using an algorithm that is resistant to the known plain text attack.
3. Using a strong enough algorithm that these clues are not really useful.
4. Changing keys often and selecting them properly.
5. Guarding keys as closely as the data they protect justifies.
6. Ensuring that there are not enough computers in the world to do a brute force attack on the algorithm.
7. Making sure users of the system understand how to properly use it.

1.1 Secrecy system

A secrecy system is defined abstractly as a set of transformations of one space (the set of possible messages) into a second space (the set of possible cryptograms)[4]. Each particular transformation of the set corresponds to enciphering with a particular key. The transformations are supposed reversible so that unique deciphering is possible when the key is known. A secrecy system can be represented by following diagram [4].

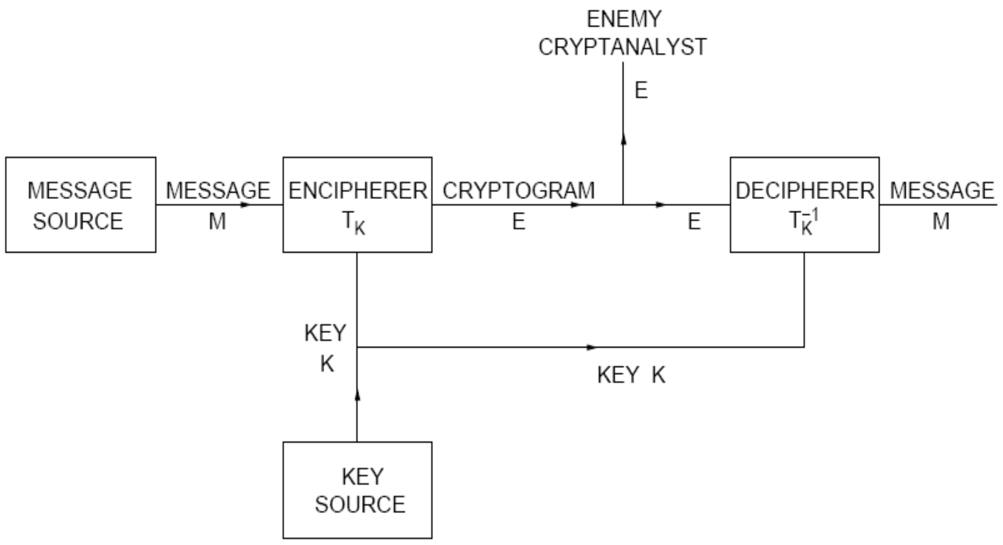

Figure 1: Schematics of Simple Secrecy System

If M is the message, K the key, and E the enciphered message, we have

$$E= f\,(M, K)$$

1.2 Valuation of Secrecy System

There are a number of different criteria that should be applied in estimating the value of a proposed secrecy system [20]. The most important of these are:

Amount of Secrecy

There are some systems that are perfect; the enemy is no better off after intercepting any amount of material than before [31]. Other systems, although giving him some information, do not yield a unique "solution" to intercepted cryptograms.

Among the uniquely solvable systems, there are wide variations in the amount of labor required to affect this solution and in the amount of material that must be intercepted to make the solution unique.

Size of Key

Size of key is very important factor for providing secrecy. The secrecy system may be attacked by using brute force so having large size key system will be strong against brute force. On the other hand the key should be transmitted through secure channel. Maintaining security is easier if the size of data is small; hence we have to come to some equilibrium point to select size of key.

DES is in use since 1977 with key size 64 bits. Recently we have i-series processor systems with average speed of 15 GHz. Trying brute force with such system will be faster. Therefore we have proposed some larger size key system in this work.

Complexity of Enciphering and Deciphering Operations

Enciphering and deciphering should be as simple as possible. If they are done manually, complexity leads to loss of time, errors, etc. If done mechanically, complexity leads to large expensive machines.

Propagation of Errors

In certain types of ciphers, an error of one letter in enciphering or transmission leads to a large number of errors in the deciphered text [44]. The errors are spread out by the deciphering operation, causing the loss of much information and frequent need for repetition of the cryptogram. It is naturally desirable to minimize this error expansion.

Expansion of Message

In some types of secrecy systems the size of the message is increased by the enciphering process [33]. This undesirable effect may be seen in systems where one attempts to swamp out message statistics by the addition of many nulls, or where multiple substitutes are used. It also occurs in many “concealment” types of systems.

1.3 Motivation

DES is the mostly used symmetric key cipher and a lot of researches have been completed on the study of DES. Most of it has been favorable to DES [7], but there are a few indicates that it would be better to search for some supplement of the DES.

DES is in use for last 34 years. During this time, it is possible that someone has discovered a computationally feasible method for breaking the cipher [8]. Under such circumstances, it is highly unlikely that such a discovery would be made known.

One of the closest thing of breaking DES is a story of the FBI successfully decrypting a file of drug transaction records that were encrypted on a PC using a DES board [46]. The DES board that the criminal used has an algorithm to generate a key from a word or phrase. By an exhaustive search of an English dictionary and key names from the criminal’s family and friends using a supercomputer, the file was solved. This indicates some weakness in DES.

DES is subject to attacks that require pre-computation that could tie up a supercomputer for a few years, after which it would take only a few days to solve a DES cryptogram. This is becoming less of a barrier as the price of computers drops and the speed and storage capacity of computers increase.

There are other alternatives to DES now, but none of them in the public domain are even as good for general cryptography [20]. Some better algorithms in terms of security are very costly in terms of execution. Asymmetric encryption techniques, in which different keys are used for encryption and decryption, are almost thousands times slower than Symmetric techniques using same key for encryption and decryption, because they require more computational processing power [9]. A study was conducted for different popular secret key algorithms such as Data Encryption Standard (DES), 3DES and Advanced Encryption Standard (AES)[9]. The algorithms were tested on two different hardware platforms; two different machines: P-II 266 MHz and P-4 2.4 GHz., to compare their performance. It showed that AES had a better performance than 3DES and DES. It also shows that 3DES has almost 1/3 throughput of DES. Some faster algorithms in terms of execution cannot provide the high level of security [9]. RSA (Named by R.L. **R**ivest, A. **S**hamir and L. **A**dleman) encryption has great advantages in the authentication of digital signatures, but the complications of selecting good keys and the fact that the security of RSA relies heavily on the failure of the state of the art in mathematics to progress makes it at least inconvenient to use and at most insecure. Similarly, Cryptosystems based on elliptic curve are faster for execution but to break the system, it is sufficient to factor its modulus [9, 25].

Because of the above considerations, it is our interest to suggest a better algorithm for general use in the private sector.

1.4 Approach

To design the supplement encryption algorithm of DES, it will be better to study weaknesses of the DES and various attacks that can happen on DES. The design criteria chosen for the algorithm are discussed under proposed solution topic. To avoid repetition of one or more of the many mistakes that have been made throughout history of cryptography, it is, of course, necessary to analyze the newly designed algorithm. The analysis of the proposed algorithm is done under testing and analysis topic. The ideas collected after the study of certain ciphers along with knowledge of current technology together with a bit of creativity are applied to design the algorithm that required a lot of hard work.

2. LITERATURE REVIEW

2.1 History of Cryptography

Cryptography is in use to maintain data security since ancient age. In “The Code Breakers”, David Kahn discussed about cryptography from prehistory to World War II. The first codes and ciphers were in written form, and used to protect the privacy of communications sent by courier or mail through hostile or unknown territory [13]. Some of these were reasonably good, but most were not difficult to break using manual methods, provided that the interceptor had sufficient cipher text and perhaps some probable text. The use of radio, especially by the military, increased the need for cryptography, as well as increasing the rewards for those who could break the encryption schemes in use. Kahn has written about the efforts of those who broke some of the very complex encryption schemes, like the German Enigma and the Japanese Purple Ciphers, lend great insight to the kind of process cryptanalysis really is. Kahn points out the kinds of mistakes the inventors and users of cryptographic algorithms tend to make that reduce the security of their communications. For example, German users of Enigma tended to choose a three-letter indicator for their messages that consisted of three consecutive letters on the keyboard. This substantially reduced the number of keys that had to be searched to determine the one that they were using. While the designer of an algorithm may calculate the great number of combinations of keys that there are, the cryptanalyst looks at ways to isolate parts of the key so that the difficulty of a solution is much less than the size of the key space indicates. The difference in mind set between the concealer of secrets and the one who pry into them has caused many an inventor of an encryption algorithm to be overconfident.

The job of the cryptanalyst is a tedious one. He tries all kinds of things to try to unscramble the cipher text in front of him. Sometimes the search is fruitless. Sometimes the search yields something that looks like a meaningful language. It is this ability to recognize a meaningful message when it comes out of the various operations that the cryptanalyst tries that makes the whole process possible. It is also helpful for the cryptanalyst to know some probable plain text that is contained in a message. This is almost always the case. For example, military messages even now have a very stereotyped format, with the from, to, and date indicators in the same places in the message. The cryptanalyst almost always knows what language to expect a message to be written in, and this is a great help. Natural languages contain a great deal of redundancy. A message that is only 90% recovered is usually readable [22]. Natural languages also have very consistent statistical properties that are very useful in cryptanalysis, especially when the cryptanalysis is automated. The only time that these things don't help the cryptanalyst is in the ''one-time pad.''

2.2 Cryptography Model

There have been lots of techniques existing for producing the secure cryptogram [48]. Some of the techniques are tested and are in use; some new techniques are proposed and are under the testing phase for future use. On the basis of input of the plain text to the encryption algorithm, we have two different classes of the cryptographic approach.

Block Cipher: If the plain text is divided into different blocks of fixed size and each block are inputted to the encryption algorithm to produce cipher text then such approach is comes under Block Cipher.

Stream Cipher: The technique in which, whole plain text is taken as input to the encryption algorithm as stream is Stream Cipher [16].

However cryptography models can be classified as -

2.2.1 Symmetric Model

A symmetric cryptosystem € is a set of cryptographic transformations

$€ = \{E_k \mid k \in K\}$

The index set K is called the key space, with its elements k keys.

E_k is one to one mapping and its inverse must exist and let it be D_k. This D_k is used to recover the plain text from the scrambled one. Provided that k is known. This k is transferred through secure channel to the receiver.

A symmetric encryption scheme has five ingredients [23].

Plain text: - This is the original intelligible message or data that is fed into the algorithm as input.

Encryption algorithm: - The algorithm performs various substitution and transformations on the plain text

Secret key: - The secret key is also input to the encryption algorithm. It is value independent of the plain text, and of the algorithm. The algorithm produces a different output depending on the specific key.

Cipher text: - This is the scrambled text produced as output.

Decryption algorithm: - It is the algorithm that runs in reverse of the encryption algorithm.

2.2.1.1 Substitution Technique

A substitution technique [13,2] is one in which the letters of plain text are replaced by other letters or by numbers or symbols. If the plain text is viewed as a sequence of bits then substitution involves replacing plain text bit patterns with cipher text bit pattern. In this cipher each letter of the message is replaced by a fixed substitute, usually also a letter. Thus the message,

$M = m_1m_2m_3m_4\ldots$ (where, m_1, m_2 … are the successive letters)

becomes:

$$E = e_1e_2e_3e_4\ldots$$
$$= f(m_1)f(m_2)f(m_3)f(m_4)\ldots$$

where, the function f (m) is a function with an inverse. The key is a permutation of the alphabets.

The simple substitution cryptogram and a variation by Julius Caesar 'Caesar cipher' were popular classical cryptograms [3, 13]. The main weakness of substitution technique is that one can choose fixed permutation of the alphabet space of the plain text and if the frequency of occurrence of the characters is known, it is easy to recover the plain text with very less effort.

2.2.1.1.1 Vigenère, and Variations

In the Vigenère cipher the key consists of a series of letters [17]. These are written repeatedly below the message and the two added modulo 26 considering the alphabet numbered from A = 0 to Z = 25. Thus

$$e_i = m_i + k_i \pmod{26}$$

Where k_i is of period d, in the index i.

0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
2	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
4	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
5	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
6	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
7	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
8	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
9	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
10	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
11	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
12	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
13	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
14	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
15	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
16	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
17	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
18	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
19	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
20	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
21	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
22	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
23	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
24	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
25	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Figure 2: Vigenère's Table

The simplest form of Vigenère cryptosystem is the Caesar ciphers [2,23]. It has only one period. It is a simple substitution in which each letter of M is advanced a fixed amount in the alphabet. This amount is the key, which may be any number from 0 to 25. The Beaufort and Variant Beaufort are similar to the Vigenère. The equation for Beaufort is

$$e_i = k_i - m_i \pmod{26}$$

and for Variant Beaufort is

$$e_i = m_i - k_i \pmod{26}.$$

The Beaufort of period one is the reversed Caesar cipher. The application of two or more Vigenère in sequence is known as compound Vigenère. It has the equation

$$e_i = m_i + k_i + l_i + \ldots + s_i \pmod{26},$$

where k_i, l_i, …,s_i in general have different periods. The period of their sum, $k_i + l_i + \ldots + s_i$ as in compound transposition is the least common multiple of the individual periods.

Lemma[23]

Let C be the cipher text, which is the result of Vigenère encryption of plain text m of length n with key k of length r. further let all the letters in m are generated independently of each other, all with the frequency distribution p(m) given by Vigenère's table and let the letter k_i in the key are chosen with independent and uniform distribution from {a, b, ……, z}; then for each $1 \le i < j \le m$,

$$\Pr[c_i = c_j] = \begin{cases} \sum_m p(m)^2 \approx 0.06875, & if\ r\quad divides\quad j-i, \\ 1/26 \approx 0.03846, & if\quad r\quad doesn't\quad divide\quad j-i. \end{cases}$$

Noted point 1: Though, above lemma showed that the probability of recovering the plain text after knowing the part of plain text is very less, the cipher text produced by Vigenère cryptosystem can be cryptanalyzed by "method of probable word" or by Kasiski's method.

.

2.2.1.2 Transposition Technique

It refers to the changing of character position in the plain text to generate some cipher text [48]. A very different kind of mapping is achieved by performing some sort of permutation on the plain text letters [18]. The message M is divided into groups of length d and a permutation applied to the first group, the same permutation to the second group, etc. The permutation is the key and can be represented by a permutation of the first d integers. Thus for d = 5, we might have 2 3 1 5 4 as the permutation. This means that:

$m_1\ m_2\ m_3\ m_4\ m_5\ m_6\ m_7\ m_8\ m_9\ m_{10}$ …….

Becomes

$m_2\ m_3\ m_1\ m_5\ m_4\ m_7\ m_8\ m_6\ m_{10}\ m_9$ …….

Sequential application of two or more transpositions will be called compound transposition [4]. A pure transposition cipher is easily recognized, because it has the same letter frequency as the original plain text.

2.2.1.2.1 Vernam, Playfair, Rotor Machines

The one-time pad, also called the Vernam cipher [4,37], is a Vigenère cipher with key length equal to the length of the plaintext. Also, the key must be chosen in a completely random way and can only be used once. Such system is unconditionally secure, as is intuitively clear. The major drawback of this system is the length of the key, which makes this system impractical for most applications.

By Mono-alphabetic ciphers, it seems that making the key large is not sufficient to make an encryption secure[48,33]. One approach to improving security was to encrypt multiple letters. The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into cipher text digrams[2]. The Play-fair algorithm is based on the use of a 5x5 matrix of letters constructed using a keyword. The rules for filling in this 5x5 matrix are: Left to Right, top to bottom, first with keyword after duplicate letters have been removed, and then with the remaining letters, with I/J used as a single letter.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Figure 3: Matrix of Playfair cipher

Play-fair encryption was invented by Charles Wheatstone in 1854, but named after his friend Baron Play-fair, who championed the cipher at the British foreign office [37].

Plaintext is encrypted two letters at a time, according to the rules as shown below.

1. If a pair is a repeated letter, insert a filler like 'X', e.g. "balloon" encrypts as "ba lx lo on"
2. If both letters fall in the same row, replace each with letter to right (wrapping back to start from end), e.g. “ar" encrypts as "RM"
3. If both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom), e.g. “mu" encrypts to "CM"

4. Otherwise each letter is replaced by the one in its row in the column of the other letter of the pair, e.g. “hs" encrypts to "BP", and “ea" to "IM" or "JM" (as desired)

Decryption of the cipher text is done in the reverse order.

The Playfair cipher is a great advance over simple mono-alphabetic ciphers, since there are 26*26=676 digrams (vs 26 letters), so that identification of individual digrams is more difficult. Also, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis much more difficult.

The Playfair cipher was for a long time considered unbreakable. It was used as the standard field system by the British Army in World War I and still enjoyed considerable use by the U.S.Army and other Allied forces during World War II [8].

Despite this level of confidence in its security, the Playfair cipher is relatively easy to break because it still leaves much of the structure of the plaintext language.

Rotor machine consists of a set of independently rotating cylinders through which electrical pulses can flow [27, 49]. These cylinders provide substitution of the characters. The substitution will be done by each cylinder so it is the scheme using multiple stages of encryption. Different variations of rotor machine and similar other machines for encryption were found in use like Enigma, Purple, Typex, Hagelin etc.

Noted point 2: Multi-stage transposition ciphers are more secure and security can be enhanced using poly alphabetic technique.

2.2.2 Asymmetric model

Asymmetric cryptography uses different keys for encryption and decryption [20]. The ingredients of the asymmetric model are as follows -

Plain text: - This is the original intelligible message or data that is fed into the algorithm as input.

Encryption algorithm: - The algorithm performs various substitution and transformations on the plain text.

Encryption key: - The encryption key is also input to the encryption algorithm. It may be value dependent of the plain text, and of the algorithm. The algorithm produces a different output depending on the specific key.

Cipher text: - This is the scrambled text produced as output after the encryption process.

Decryption algorithm: - It is the algorithm that runs in reverse of the encryption algorithm.

Decryption key: - The decryption key is input to the decryption algorithm along with the cipher text. The algorithm recovers the plain text from the cipher.

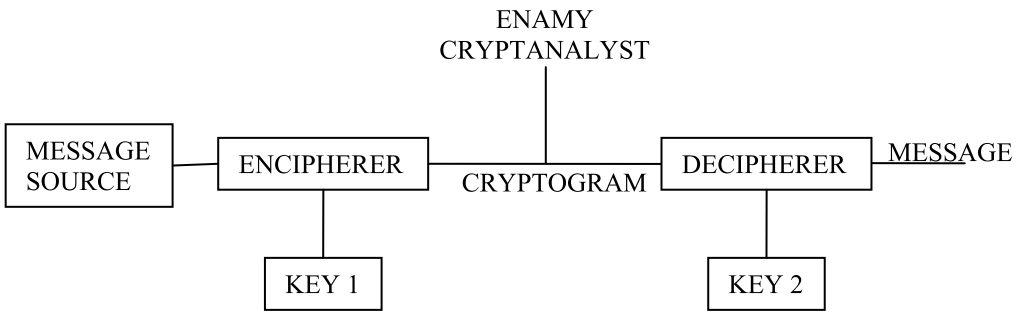

Figure 4: Asymmetric Cryptogram

Since many encryption keys can be used to encrypt the plain text and a single decryption key is used to recover the plain text; this model some time known as public key cryptography and the encryption and decryption keys are respectively known as public and private keys[48].

2.2.2.1The RSA based system

RSA system is a public key cryptogram introduced by R.L. Rivest, A. Shamir and L. Adleman in 1978 [26,42]. It is block cipher in which plain text and cipher text are integers between 0 and n-1 for some n. It makes use of the following three facts:

1) Exponentiation modulo a composite number n, i.e. computing c from $c \equiv m^e (\bmod n)$ for given m and e, is a relatively simple operation.

2) The opposite problem of taking roots modulo a large, composite number n, i.e. computing m from $c \equiv m^e (\bmod n)$ for given c and e, is, in general, believed to be intractable.

3) If the prime factorization of n is known, the problem of taking roots modulo *n* is feasible [6].

2.2.2.2 Galois Field based systems

A field {F, +, ×} is a set of elements with two binary operators, addition and multiplication such that F is an integral domain and multiplicative inverse of the elements must exist in F [14]. A field is said to be finite if there exist bijection between F and subset of natural number set N for some $n \in N$. A finite field with order as power of prime p^n, where n is positive integer and p is prime, is denoted by GF (p^n) and is known as Galois Field [11,40]. Based on the Galois field some cryptosystems like Hill cryptosystem are designed. Matrix operations and modulus arithmetic operations are performed in the Galois field to obtain the scrambled text [35,45].

2.2.2.3 Elliptic curve based systems

An elliptic curve ε over GF (p) is defined as the set of points (x, y) satisfying the relation

$$y^2 + uxy + vy = x^3 + ax^2 + bx + c,$$

together with a single element *O (u, v)*, called the point at infinity, where a, b and c are the constants[1]. It is possible to define some Cryptosystems over elliptic curves [21]. However, to break the system it is sufficient to factor its modulus. Since the original RSA system had the same security restriction and is faster in its calculations, there seems to be little reason to use this generalization of RSA to elliptic curves [6].

2.2.2.4 Coding Technique

Based on the algebraic coding theory, some cryptosystems are designed [29]. Using error correction code, McEliece in 1978 proposed a cryptosystem. Some special coding technique known as Goppa Code may lead to the very fast and efficient cryptosystems.

2.2.2.5 Quantum Encryption

G Brassard and C Bennett introduced BB84 protocol, based on the idea of using quantum mechanics to solve key distribution problem [10]. It uses light particles to communicate instead of bits. A light particle 'photon' can have one of the four orientations, Horizontal, Vertical, 45^0 diagonal and -45^0 diagonal. Each of these represent a bit; '-' and '/' represents a logic 0 and '|' & '\' represents logic 1. Each bit in the plain text is converted randomly in one of the two orientations connected with that bit and is transferred via fiber optic cable [35].

2.2.2.6 Noise addition

Adding some unnecessary data to the plain text or to the resulting cipher text makes the cipher text more difficult against cryptanalysis. This process is known as noise addition [24]. The unnecessary data is known as noise. Such noise may be added on the plain text or it can also be added to the cipher text. One of the popular noise additions is PN sequencing in which the Shift Register Sequencing technique results some noise on cipher text to change the period of the space.

Such noise may be in the form of electrical signal, some sound waves or some color code.

2.2.3 Steganography

Steganography is the scheme in which the actual message is covered by some object and transmitted to the receiver [32]. The basic model of steganography consists of Carrier, message, embedding algorithm and stego key.

The carrier is the object in which the message is embedded by using embedding algorithm along with stego key. The covered object with secretly embedded message is known as stego object [32].

2.3 Threat Models

Though cryptographic schemes transfers plain text to some scrambled text, the intruders try to recover the plain text from the scrambled text. The attacking techniques may vary in different schemes; but, by the study of historical background of cryptography, the possible attacks can be broadly classified into following models [33]:

Black-Box Model: In the traditional black-box model, the attacker is restricted to observe input and output of the algorithm, without any side-channels of information. A secret key of a cryptographic algorithm is hidden in the black-box and is never exposed. The security depends on the strength of the cryptographic algorithm.

Grey-Box Model: Another model is the grey-box model where an attacker is also able to monitor side effects of the program execution. For example, an attacker can monitor the execution time, power consumption, and electromagnetic radiation.

White-Box Model: In the white-box model [30], the attacker also has total visibility into software implementation and execution. To prevent an attacker from finding the key, the key needs to be hidden in the implementation.

2.4 Cryptanalytic Attacks

Various types of attacks can happen on the cipher text [48]. The main goal of the attacker is to obtain the key used in the scheme

Cipher text only attack: This type of attack involves the cipher text and the producing algorithm only. The attacker tries to obtain the key by Hit and Trial.

Known plain text attack: The attacker knows about the one or more plain text cipher text pairs and the encryption algorithm.

Chosen plain text attack: The attacker can choose one or more plain texts and can have the cipher text using the encryption algorithm.

Chosen cipher text attack: The attacker will have cipher text and also can choose one or more cipher texts and can decrypt the chosen cipher text to get the plain text.

Chosen text attack: The attacker can choose plain text to generate cipher text as well as can choose the cipher text to get plain text.

2.5 Brute Force Attack

The attacker tries every possible key on a piece of cipher text to recover the part of plain text. It is very costly and on average, half of all possible key must be tried to achieve the success. To prevent the encryption scheme from brute force attack, the possible no of key should be huge. For it, the key space should be sufficiently large.

2.6 Encryption Timeline

Modern Encryption Techniques

2010	The maser key of HDCP and private signing key for the Sony PlayStation 3 game console are recovered and published.
2007	Users swamp digg.com with copies of 128 bit key to the AACS system
2004	The hash MD5 was shown to be vulnerable to practical collision attack.
2003	First commercial use of Quantum Encryption

2000	Advanced Encryption Standard (AES) Developed
1991	First Quantum Encryption System developed
1984	BB84 Protocol proposing Quantum Encryption published
1978	RSA published
1977	Data Encryption Standard (DES) created
1976	Public Key Encryption proposed by Hellman and Diffie
1970	Lucifer Algorithm developed, later evolved into Triple – DES.

The Computer era

1943-1945	First computers created

Traditional Encryption

1942	Navajo Windtalkers used in World War II
1923	Arthur Scerbius builds the German Enigma Machine
1917	Vernam Cipher invented
1854	Charles Babbage reinvents the Wheel Cipher
1790	Thomas Jefferson invented the Wheel Cipher
1585	Blaise De Vigenere writes a book on Ciphers
1553	Password idea introduced by Giovan Belaso

The Dark Age of Encryption

50-60 BC	Caesar Cipher introduced by Julius Caesar
486 BC	Greek Skytale presumably used
500-600 BC	Hebrew ATBASH Cipher used in writing the book of Jeremiah
1500 BC	Mesopotamian tablet with encrypted recipe for Pottery Glaze
1900 BC	First Documented Cryptography in Egypt

2.7 Authentication

Authentication is the process of verifying that the user involved in the communication is the user supposed to involve in the communication even if it refuses the participation [41].

2.7.1 The Needham Schroeder Authentication Protocol

This protocol aims to establish mutual authentication between initiator A and a responder B [19]. The supposition behind the protocol is “There is minimal reliance on network wide services; in particular, there is no reliance on a single network clock or a single network name management authority”[41].

The protocol uses nonces: random numbers generated with the purpose of being used in a single run of the protocol. The nonces can be denoted by Na and Nb generated by A and B respectively.

The functions of the protocols is discussed in following three points[41]-

A. Establishment of authenticated interactive communication between two principals on different machines.
B. Authenticated one way communication, where it is impossible to require protocol exchanges between sender and the recipient.
C. Signed communication, in which the origin of a communication and the integrity of the content can be authenticated to a third party.

2.7.1.1 Symmetric Key Protocol

If a conventional algorithm is used then each principal has a secret key that is known only to itself and to authentication server (AS).

The protocol can be described as -

- The protocol opens with A communicating to AS his own claimed identity and the identity of the desired correspondent, B together with nonce of A

  $A \rightarrow AS\text{: } A, B, Na$

- Up on receiving the message, authentication server looks up the secret identifying keys of both parties and also computes a new key CK that will be key for convention.
- AS sends the information to A as

  $AS \rightarrow A\text{: } \{Na, B, CK, \{CK, A\}^{KB}\}^{KA}$

- A can decrypt above message and sends the message to B

  $A \rightarrow B : \{CK, A\}^{KB}$

- B can decrypt above message and understands that CK is the conventional key for communicating with A.

2.7.1.2 Public Key Protocol

Each agent A possesses a public key PK_a, and secret key SK_a. There will be key pair of AS as PK_{as} and SK_{as}, public and private keys. The exchange opens with A consulting the AS to find B's public key.

The protocol can be described as -

- A communicates with AS for public key of B

  $A \rightarrow AS$: A, B

- AS responds to A

  $AS \rightarrow A$: $\{PK_b, B\}^{SKas}$

- Now A can communicate to B

2.8 Cryptographic Principles

Designing the cryptographic scheme is notably difficult. There are various principles, regarding the design of the cryptographic scheme.

2.8.1 Kerkhoff's Principle

A fundamental assumption in cryptanalysis was first stated by A. Kerkhoff in the nineteenth century [23]. It states that the adversary knows all the details of the cryptosystem, including algorithms and their implementations i.e. the security of a cryptosystem must be entirely based on the secret keys.

2.8.2 Unconditionally Secure Scheme

An encryption scheme is said to be unconditionally secure if the cipher text generated by the scheme does not contain enough information to determine the plain text uniquely [48].

2.8.3 Computationally Secure Scheme

An encryption scheme is said to be computationally secure if at least one of the following holds-

- The cost of breaking the cipher text exceeds the value of encrypted information.
- The time required to break the cipher text exceeds the useful life time of the information [48].

3. ANALYSIS OF ALGORITHMS

3.1 Feistel Cipher Structure

Horst Feistel described the structure of the symmetric ciphers in 1973[48]. A block of plain text of size 2w bits is divided into two half blocks L_i and R_i each of size w bits and processed. The process requires sub key K_i generated from the original key K. It can be described as

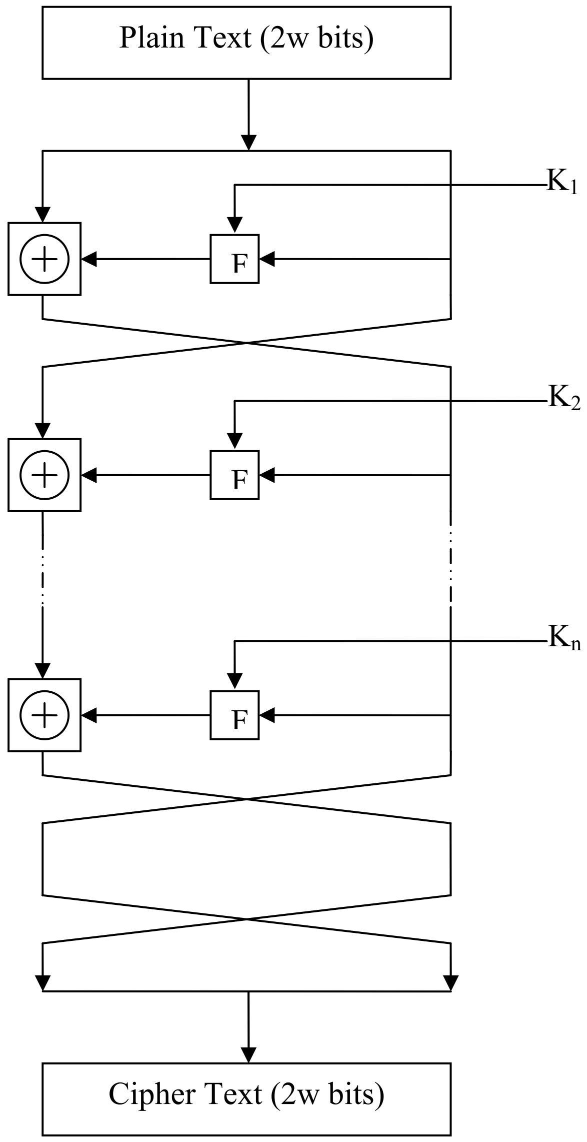

Figure 5: Fiestel Cipher Structure

The two halves of the data pass through n rounds of processing and then combine to produce the cipher text block. Each round i has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a sub key K_i, derived from the original key K. A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR (XOR) of the output of that function and the left half of the data. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

3.2 Data Encryption Standard

DES was published in the Federal Information Processing Standards Publication Number 46, dated January 15, 1977 by the National Bureau of Standards [34]. DES is based on Fiestel Cipher Structure where Encryption consists of an initial permutation, sixteen rounds of encryption, and then an inverse of the initial permutation. Each of the sixteen rounds of encryption consist of taking the right half of the input block (32 of the 64 bits) and running it through a nonlinear function of the 32 bits and an internal key, then adding this result to the left half of the input modulo two. This 32 bit answer becomes the next round's right half block. The next round's left half becomes the right half block without modification. The nonlinear function used consists of a bit selection E that selects 48 bits from the input of 32 (several of the bits are repeated). These 48 bits are added modulo 2 to the round key of 48 bits. The results of that operation are then fed six bits each into eight substitution boxes. Each of the eight substitution boxes is different, but the same sets of eight boxes are used for each round. Each substitution box gives an output of 4 bits. The outputs of these boxes are fed into a permutation P that rearranges the output in a fixed manner. The sixteen internal keys are generated from the 56 bit input key by feeding the input key into a fixed permutation that rearranges the order of the key bits. The key is then split into left and right halves called C and D. Each half is shifted left one or two times (according to a fixed table) before generating each internal key. Each of the sixteen internal keys is generated by taking the two halves of the key as shifted and permuting them in a fixed manner. The key and the resulting internal keys are the only things that vary in this algorithm. The initial and final permutations and the contents of each of the substitution boxes are constant. The two permutations used in generating the internal keys are constant. The bit selection and permutation used within the nonlinear function are constant. The strengths of the DES is that its cryptographic strength depends only on the key, that the algorithm is easy to implement in a single IC, that it has been well tested an no one has publicly announced a solution, that hardware and software that uses it is readily available, and that the algorithm places very few restrictions on key generation so that random numbers may be generated by the users for use as keys.

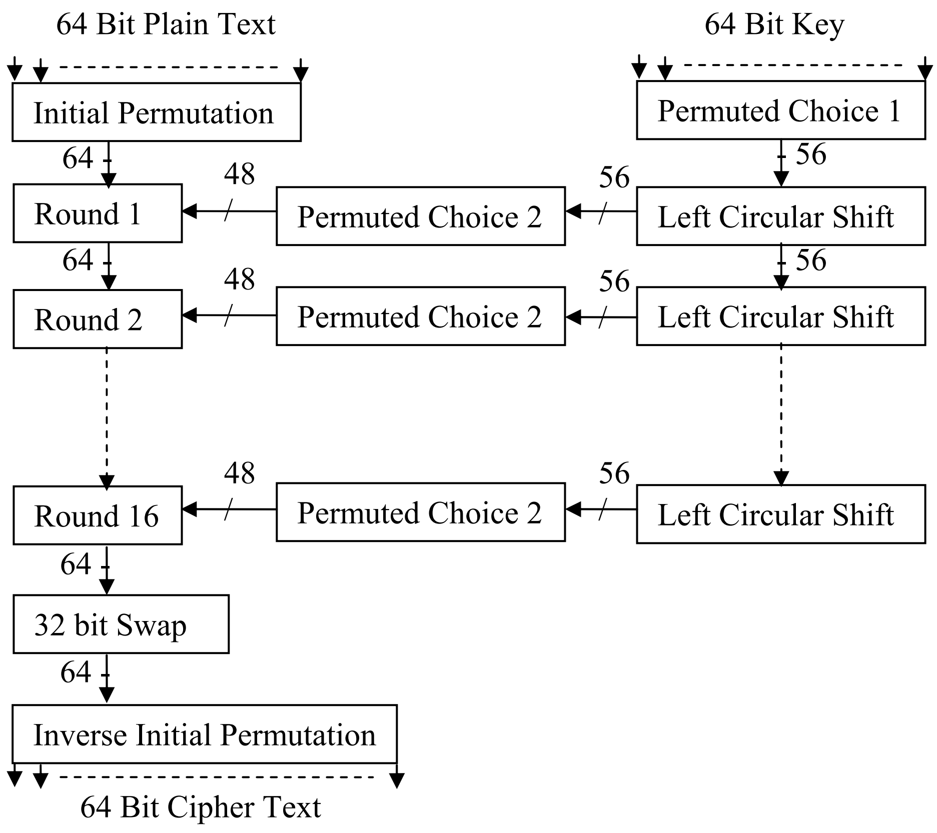

Figure 6: General Description of DES

The weaknesses of the DES are that the key is too short for security in the face of anticipated increases of computing power that it is old enough and likely that someone has broken it. Hardware implementation of the DES is too slow for some applications, and that it limits itself to be simpler than is really necessary with current technology.

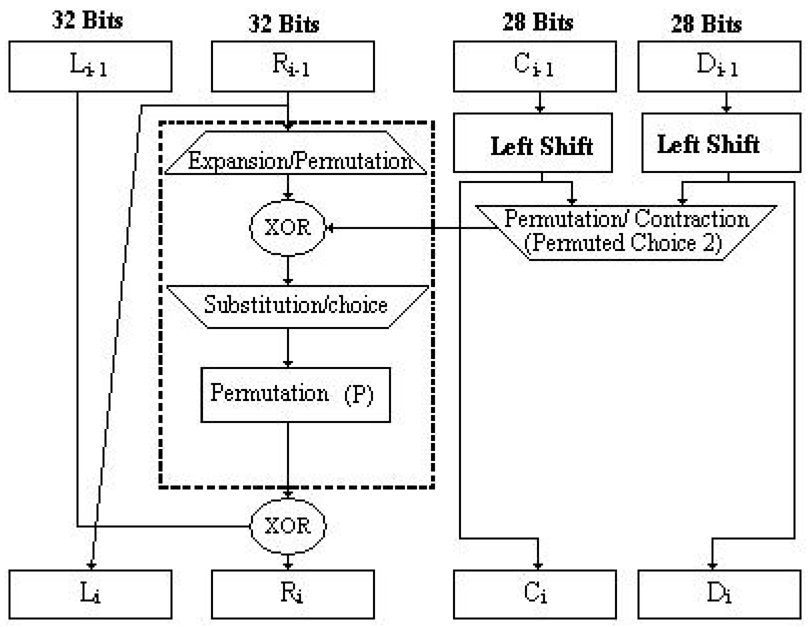

Figure 7: Single Step of DES

Cryptanalysis Practice

Differential cryptanalysis

Differential cryptanalysis was started after 1990 for the cryptanalysis of block cipher called FEAL by Murphy [50]. Differential cryptanalysis is the first published attack on DES capable to break DES in less than 2^{55} complexities. Biham and Shamir proposed one scheme that can successfully cryptanalyze DES with an effort of order of 2^{47} complexities[33].

The idea behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher. Let the two message of block size m and m', where m_0 and m_1 are the two halves of m and so on then the new blocks for m can be generated as

$m_{i+1} = m_{i-1} \oplus f(m_i, K_i)$, i = 1,2,… …, 16

Now the XOR difference $\Delta m_i = m_i \oplus m'_i$, is calculated. It results the

$\Delta m_{i+1} = \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K'_i)]$

Now by using probability theory the cryptanalysis of the DES encryption can be done.

Linear cryptanalysis

Linear cryptanalysis is recent development than differential cryptanalysis. It can identify the encryption key from 2^{43} known plain texts. It performs XOR on plain text and cipher text to form a linear equation. The linear equation can be used to find the unique key with the probability of 0.5. It can break one round at a time.

3.3 Simplified DES

The Simplified DES [15] encryption algorithm takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8-bit block of cipher text as output. The decryption algorithm takes an 8-bit block of cipher text and the same 10-bit key used as input to produce the original 8-bit block of plaintext. The encryption algorithm involves five functions; an initial permutation (IP), a complex function called f_K which involves both permutation and substitution operations and depends on a key input; a simple permutation function that switches (SW) the two halves of the data; the function f_K again, and a permutation function that is the inverse of the initial permutation (IP^{-1}). The function f_K takes as input the data passing through the encryption algorithm and an 8-bit key. Consider a 10-bit key from which two 8-bit sub keys are generated. In this case, the key is first subjected to a permutation P10= [3 5 2 7 4 10 1 9 8 6], then a shift operation is performed. The numbers in the array represent the value of that bit in the original 10-bit key. The output of the shift operation then passes through a permutation function that produces an 8-bit output P8 = [6 3 7 4 8 5 10 9] for the first sub key (K_1). The output of the shift operation also feeds into another shift and another instance of P8 to produce sub key K2. In the second all bit strings, the leftmost position corresponds to the first bit. The block schematic of the SDES algorithm is as follow

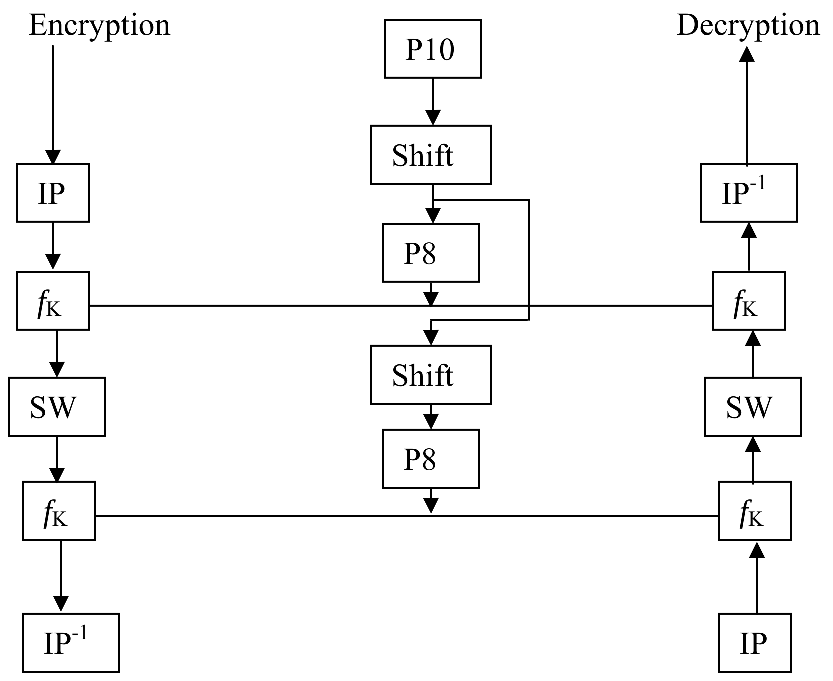

Figure 8: Simplified Data Encryption Standard

The definitions of the involved functions are as follow

1. Initial and final permutation (IP): The input to the algorithm is an 8-bit block of plaintext. It permuted using the IP function IP= [2 6 3 1 4 8 5 7]. It results all 8-bits of the plaintext but mixes them up. At the end, the inverse permutation is applied; by applying, IP^{-1} = [4 1 3 5 7 2 8 6] where we have IP−1 (IP(X)) =X.

2. The function f_K: it consists of a combination of permutations and substitution functions. The functions are as follows. Let L, R be the left 4-bits and right 4-bits of the input, then, f_K (L, R) = (L XOR *f*(R, key), R) where XOR is the exclusive-OR operation and key is a sub - key. Computation of *f*(R, key) is done as follows.

a. Apply expansion/permutation E/P= [4 1 2 3 2 3 4 1] to input 4-bits.

b. Add the 8-bit key (XOR).

c. Pass the left 4-bits through S-Box S_0 and the right 4-bits through S-Box S_1.

d. Apply permutation P4 = [2 4 3 1].

The two S-boxes are defined as follows:

$$S_0 = \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix} \qquad S_1 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix}$$

The S-boxes operate as follows: The first and fourth input bits are treated as 2-bit numbers that specify a row of the S-box and the second and third input bits specify a column of the S-box. The entry in that row and column in base 2 is the 2-bit output.

3. Since the function f_K allows only the leftmost 4-bits of the input, the switch function (SW) interchanges the left and right 4-bits so that the second instance of f_K operates on different 4- bits. In this second instance, the E/P, S_0, S_1 and P4 functions are the same as above but the key input is K2.

Cryptanalysis Practice

Memetic Algorithm:

The memetic algorithms can be viewed as a marriage between a population-based global technique and a local search made by each of the individuals [39]. Memetic Algorithms are a population-based approach. Such algorithms are orders of magnitude and faster than traditional genetic Algorithms for some problem domains. In a memetic algorithm the population is initialized at random or using a heuristic. Then, each individual makes local search to improve its fitness. To form a new population for the next generation, higher quality individuals are selected. Once two parents have been selected, their chromosomes are combined and the classical operators of crossover are applied to generate new individuals. The latter are enhanced using a local search technique. The role of local search in memetic algorithms is to locate the local optimum solution.

Genetic Algorithm:

The genetic algorithm is based upon Darwinian evolution theory [39]. The genetic algorithm is modeled on a relatively simple interpretation of the evolutionary process; however, it has proven to a reliable and powerful optimization technique in a wide variety of applications. J. Holland, in 1975 was first proposed the use of genetic algorithms for problem solving. D. E. Goldberg was also pioneer in the area of applying genetic processes to optimization [39]. Over the past twenty years numerous application and adaptation of genetic algorithms have appeared in the literature. During each iteration in the algorithm; the processes of selection, reproduction and mutation take place in order to produce the next generation of solution. Genetic Algorithm begins with a randomly selected population of chromosomes represented by strings. It uses the current population of strings to create a new population such that the strings in the new generation are on average better than those in current population. The selection process determines which string in the current will be used to create the next generation. The crossover process determines the actual form of the string in the next generation. Here two of the selected parents are paired. A fixed small mutation probability is set at the start of the algorithm. This crossover and mutation processes ensures that the GA can explore new features that may not be in the population yet. It makes the entire search space reachable, despite the finite population size.

A study was conducted, to compare the Memetic and Genetic algorithms for cryptanalysis of simplified data encryption standard and following results were found.

Amount of cipher text (characters)	**Memetic algorithm**			**Genetic algorithm**		
	Time (sec)	**St. Dev**	No of bits matched in the key (10)	**Time** (sec)	**St. Dev**	No of bits matched in the key(10)
100	5.1	4.70	8	2.62	4.82	6
200	14	3.40	6	4.5	6.13	6
300	15.3	2.72	5	2.13	6.01	4
400	12.5	2.27	7	2.35	4.61	6
500	10	2.16	6	2.52	4.61	6
600	5.5	1.86	8	2.07	4.37	7
700	3.05	1.73	7	4.07	4.42	6
800	2.85	1.59	8	2.4	3.39	8
900	2.24	1.56	9	2.53	2.23	6
1000	2.14	1.49	9.17	2.17	2.20	8

1. Both Memetic and Genetic Algorithms can be used for cryptanalysis
2. Memetic algorithm has less variance in result than genetic algorithm
3. Memetic algorithm is more accurate then the genetic algorithm for cryptanalysis of simplified data encryption standard.

3.4 3DES

When it became clear that DES could no longer be used to protect sensitive data, a modification was introduced, called Triple DES or 3DES[9]. It consists of three DES implementations in a row, except that the middle one is orientated the other technique.

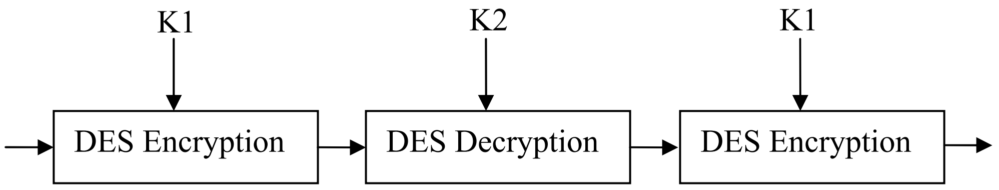

Figure 9: 3DES Encryption

In 3DES there are three steps first and third are the encryptions using DES with the same key but second step is the decryption. The second key used is different than the encryption key used.

Cryptanalysis Practice

There is no practical cryptanalysis scheme for triple DES but some known – plain text attacks are outlined for 3DES[47].

It was found that 3DES produce 1/3 throughput of DES, that’s why 3DES is far more inefficient for commercial use [12].

By studying above algorithms, we found sufficiently strong cryptographic algorithms for providing confidentiality but we do not have any cryptographic algorithm to provide confidentiality together with authentication. During this study it is claimed that there exists one cryptographic approach providing both confidentiality and authentication but it uses traditional hash function SHA 256 (Source: Enhanced Security Encryption for Data Storage Using Key Reuse.By Gladdman)

4. PROBLEM DEFINITION

Maintaining the information security against the attacks is significant problem. The primary goal of cryptography is to keep the plaintext secret from eaves-droppers trying to get some information about the plaintext. Various types of attacks may happen against information security.

In a network, having large number of computers communicating; there may not be central machine that contains authoritative description of the connected computers as explained in the Needham Schroeder’s Protocol (Authentication Server). There may not be organized data of the purpose of use and of the individuals using system.

The major component of Needham Schroeder’s authentication protocol using symmetric encryption is the pseudorandom number used to identify each communication, known as nonce. If the size of key used in encryption is small as in DES than it is quite easy to identify the nonce by clipping it from the cipher text. It may lead to reply attack.

By understanding the problems of secrecy, problem in authentication and the various attacking techniques, it is worthy and very important to introduce new schemes to preserve the confidentiality of the information and provide authentication in communication. The main goal of this research is to design a computationally secure symmetric scheme for maintaining the secrecy in communication that can provide authentication using Needham Schroeder’s protocol.

5. PROPOSED SOLUTION

In this proposed system, the plain text is divided into sixteen byte blocks and each block is individually converted into cipher text with the help of 128 bit key. The key of size 128 bit is selected randomly and further processed for applying to plain text.

5.1 Design Criteria

A. Strength Based on Key

The strength of the system must rely on the security of the key only. It cannot depend on the algorithm being kept secret, because the algorithm will be published. Even if the algorithm were not published, it would probably be reverse engineered from software implementations of the algorithm. The algorithm must be constructed in such a way that there is no computationally feasible way to derive the key from samples of corresponding plain text and cipher text.

B. Usability of Random Keys

The key selection should be as easy as the random selection of a number in a given range. Selecting a very secure key should be no more difficult than flipping a coin once for each bit of the key, or generating keys using a pseudorandom sequence combined with random events such as timing of keystrokes on a computer. A one bit change in the key should provide a drastically different transformation, so that a potential cryptanalyst has no idea when a key that he guesses might be close to the right one.

C. Key Length & Block Size

The key length should be significantly longer than the DES 56 bit size. A key size of 96 bits (three 32 bit blocks) was chosen as being very manageable, yet highly secure even when attacked by multiple array supercomputers.

The block size is also chosen as 128 bits (twice the size of DES) to provide a significant increase in complexity of the encryption. The first 32 bit of first block of plain text can be used as the nonce while using for authentication based on Needham Schroeder's protocol and first 32 bits of following blocks can be used as block pointer so that the cipher text can also be obtained in manageable packet form.

D. Effort Required to Break

The effort required to break the algorithm by any method should be so great as to make such a task unfeasible even if significant advances are made in computer technology. This requirement is intimately linked to the choice for key size and block size.

E. Computational Efficiency

The encryption algorithm must be computationally efficient enough to be implemented in software on a standard IBM PC or compatible (or on an Apple computer of comparable power), and fast enough to handle at least 10 megabits per second when implemented in dedicated hardware. Note that this is less restrictive with respect to the hardware for DES, which was required to be simple enough to implement on a single chip using 1970s technology.

F. Communication Channel Efficiency

The encryption algorithm must not significantly increase the size of the plain text when encrypting it. This precludes the use of noise addition as a technique to be used.

G. No Back Doors or Spare Keys

While it may be impossible to guarantee that no ‘‘back doors’’ or ways to decipher a message without the key exist, the algorithm should be a sufficiently complex combination of simple, well-understood operations that no help is offered to the cryptanalyst from the structure of the algorithm. Spare keys (the situation where more than one key will decipher a message) are avoided by making the number of keys possible much less than the number of possible transformations that can be done on a set of blocks.

5.2 Recursive Odd Parity Operation (ROOP)

It is the recursive operation that operates on a bit string of size n and results bit string of the size n bits [38]. Let a_1a_2………a_n be a bit string. Then the Recursive Odd Parity Operation (ROOP) is a unary function on the bit string that maps the each bit of the string to the binary bit according to the following rule-

$$\begin{aligned} \text{ROOP}(a_i) &= a_i \text{ XOR } a_{i+1} \text{ for all } i < n \\ &= a_i \text{ for } i = n \end{aligned}$$

5.3 Design and implementation of the algorithm

5.3.1 Generation of sub key

This cryptographic scheme uses the pair of keys, KEY1 (32 bits) and KEY2 (64 bits). The KEY2 will be divided into two blocks KEY2B1 and KEY2B2. The KEY2B2 will be XORed with the 32 bit KEY1 to form TK2 while ROOP operation is performed on KEY2B1 to form TK1. The TK1, KEY1 and TK2 are processed (PROC1) to get INITIAL KEY (IK). On the other hand, the initial 32 bits of the plain text are XORed with the KEY1 to form PARTIAL KEY (PK). The PARTIAL KEY (PK) and INITIAL KEY (IK) are processed (PROC2) to get the EXTENDED KEY (EK).

The process regarding PROC1 can be described as

```
IK Procedure: PROC1 (TK1, KEY1, TK2)
{
        Int temp = TK1 % 3;
        Switch (temp)
        Case 0: return (concatenate (TK1, KEY1, TK2));
        Case 1: return (concatenate (TK2, KEY1, TK1));
        Case 2: return (concatenate (TK1, TK2, KEY1));
}
```

The process regarding PROC2 can be described as-

```
EK Procedure: PROC2 (PK, IK)
{
        Int temp = PK % 2;
        Switch (temp)
        Case 0: return (concatenate (XOR (PK [0-31], IK [0-31]), IK [32-95]));
        Case 1: return (concatenate (IK [0-63], XOR (PK [0-31], IK [64-95])) ;
}
```

The detailed processing method can be represented by using flow-chart as below.

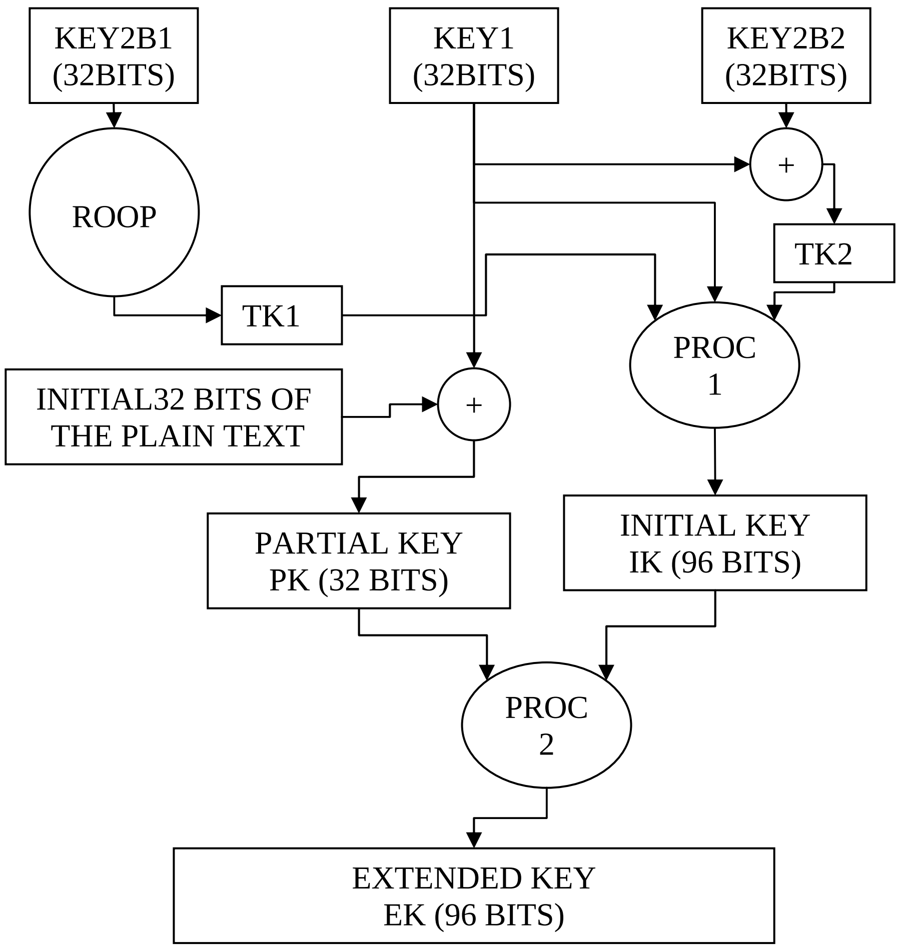

Figure 10: Key Space Modification

5.3.2 The Encryption Algorithm

The original message will be divided into 128 bit blocks for encryption. To use this algorithm for authentication, the block size should be 96 bit and the first block will be padded to 32 bit nonce. All blocks of plaintext other than first block will be padded to the 32 bit packet pointer. If there exist partial block then it will be made of size 128 by appending the initial text at the end.

The plain text block of size 128 bit will be further divided into four equal sub blocks PTs of size 32 bits. The processing method can be described as

```
CT PROCEDURE: ENCRYPTION (PT, KEY1, IK, EK)
{
    PT will be divided into PT1, PT2, PT3 and PT4;
    SB11 = PT1;
    SB21 = PT1 XOR PT2; SB31 = PT2 XOR PT3; SB41 = PT3 XOR PT4;
    SB12 = SB11 XOR KEY1;
    (SB22, SB32, SB42) = (SB21, SB31, SB41) XOR EK;
    (SB13, SB23, SB33) = (SB12, SB22, SB32) XOR IK;
    SB43 = ROOP (SB42);
    CT1 = SB13 XOR SB23; CT2 = SB23 XOR SB33; CT3 = SB33 XOR SB43;
    CT4 = SB43;
    RETURN CT = (CT1, CT2, CT3, CT4);
}
```

The pictorial representation of algorithm can also be shown as --

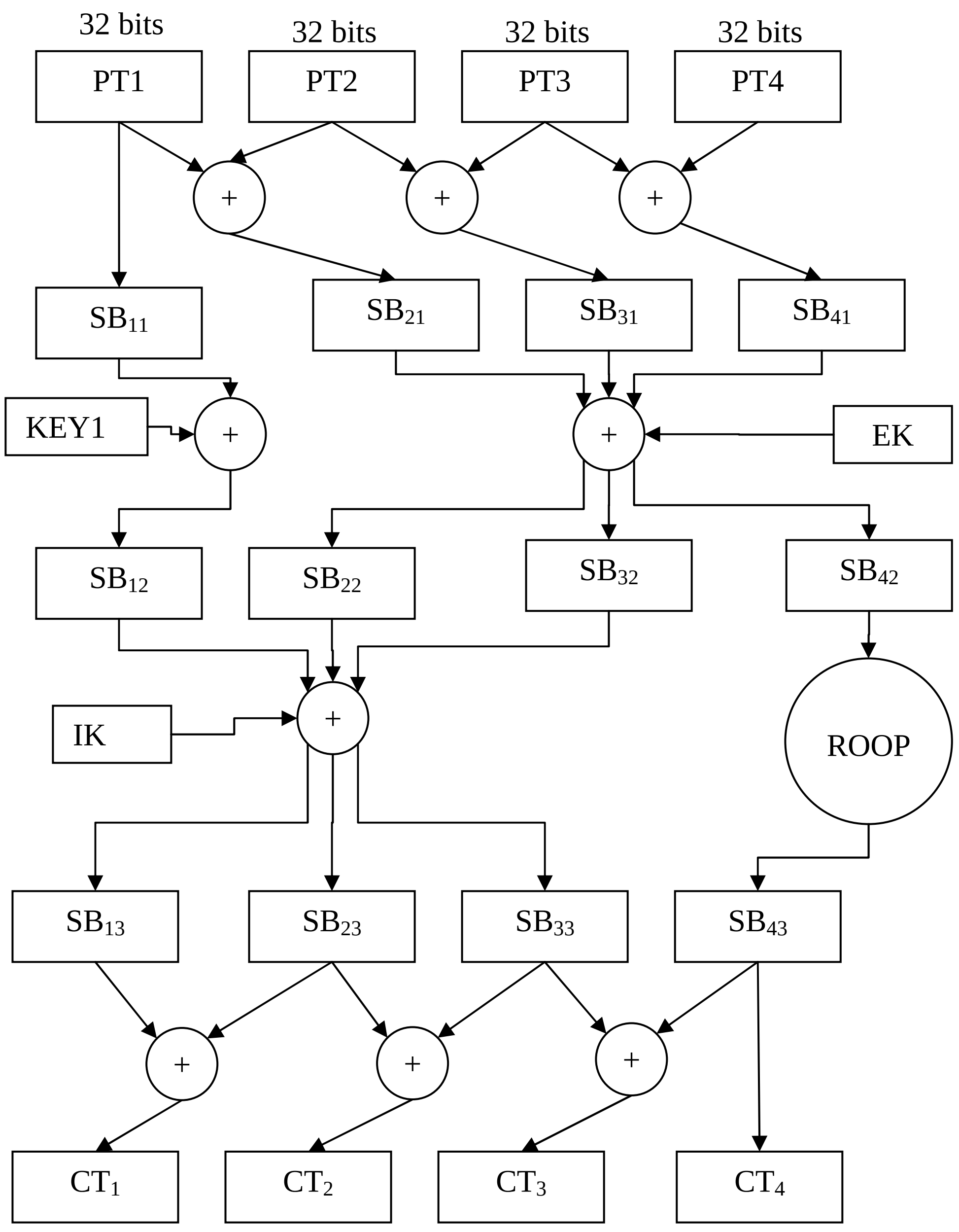

Figure 11: The Encryption Process

5.4 Implementation of the algorithm

5.4.1 Tools

5.4.1.1 Borland C++ 5.02

C++ is an Object oriented programming language developed by Strups Bazarne. Borland C++ 5.02 compiler is used to compile the codes written in C++. It provided interactive and easy environment to implement the algorithm.

6. TESTING AND ANALYSIS

SAMPLE INPUT

In symmetric encryption, there will be a secret key shared by the communicating parties. In this algorithm the secrete key will be stored into a file. The algorithm reads the key from the file key.dat.

KEY.DAT File

THIS IS MY KEY FOR ENCRYPTING THE FILE.

The file that will be encrypted is named myfile.dat. The content of the file is as follow-

MYFILE.DAT File

“1.0 Database Management System

Data are the raw facts that can be found after some experiment, observation or experience. Data itself do not provide any meaning but after processing it becomes information. The collection of related data organized in some specific manner is known as database. The database, its processing methods and the set of rules and conditions to be followed; collectively known as database management system (DBMS). Here, related data refers logically consistent facts of the real world. Random collection of data can not consider database. The primary goal of DBMS is to store and manage data both conveniently and efficiently. Database systems are generally designed to manage large volume of information. Management of data involves defining structure for storage of information and providing mechanisms for manipulation of information.

DBMS can also define as a general purpose software system that enables user to create, maintain and manipulate database. It provides fast and convenient access to information from data stored in database. DBMS interfaces with application programs so data contained in database can be accessed by multiple applications and users. Some popular DBMS software are: Oracle, SQL – Server, IBM-DB2, MySQL, MS Access, Sybase etc.

Some application areas of database system are:

- Banking: customer and their account info
- Airlines: reservations and schedules info
- Universities: student info, grades etc.
- Credit card transactions: for purchases on credit cards and generation of statements.
- Telecommunications: record of calls made
- Finance: for storing information about holding, sales and purchases etc.
- Sales: for customer, product and purchase information.
- Manufacturing: for management of supply chain.
- Human resources: for information about employee

1.1 Purpose of Database System

Traditionally, file processing system was used to manage information. It stores data in various files of different application programs to extract or insert data to appropriate file.

File processing system has several drawbacks due to which database management system is required. Database management system removes problems found in file processing system. Some major problems of file processing systems are:

1. Data redundancy and inconsistency

In file processing system, different programmer creates files and writes application programs to access it. After a long period of time files may exist with different formats and application programs may written in many different programming languages. Moreover, same information may be duplicated in several files. We have to pay for higher storage and access cost for such redundancy. It may leads database in inconsistent state because update made in one file may reflected in one file but it may not reflected in another files where same information exist in another files.

2. Difficulty in accessing data

In file processing system, we can not easily access required data stored in particular file. For each new task we have to write a new application program. File processing system can not allow data to be retrieve in convenient and efficient manner.

3. Data isolation

Since data are scatter in different files and data may stored in different format, so it is difficult to write program to retrieve appropriate data.

4. Integrity problem

In database, we required to enforce certain type consistency constraints to ensure the database correctness or to enforce certain business rules. It is in fact called integrity constraints (e.g. account balance > 0), integrity of database need not to be violated. In file processing system, integrity constraint becomes the part of application program. Programmer need to write appropriate code to enforce it. When new constraints are required to add or change existing one, it is difficult to change program to enforce it.

5. Atomicity problem

Failures may lead database in an inconsistent state with partial updates. For example, failure occurs while transferring fund from account A to B. There would be the case that certain amount from account A is retrieved and it is updated but failure occurs just before it is deposited to account B, such case may lead database in inconsistent state.

6. Concurrent access problem

Concurrent accessed increase the overall performance of system providing fast response time but uncontrolled concurrent accesses can lead inconsistencies in system. File processing system allow concurrent access but it is unable to coordinate different application programs so database may lead in inconsistent state. E.g. two people reading a balance and updating it at the same time

7. Security problems

Since file processing system consist large no. of application programs and it is added in ad hoc manner. So it is difficult to enforce security to each application to allow accessing only part of data/database for individual database users.

1.2 Data Abstraction

Data abstraction in database system is a mechanism to hide complexity of database. It allows database system to provide abstract view to database user. It hides how data are actually stored and maintain in database. Data abstraction simplifies users' interactions with the system. Three are three level of abstraction

Physical level

It is a lowest level of abstraction. It describes how data are actually stored in database. It describes complex low level data structures in detail.

Logical Level

This is a next highest level of abstraction. It describes what data are stored in database and what relationship exists among them. It describes entire database relatively in a simple structure. The user in logical level needs not to aware the complexity of physical level structure."

After the encryption the resultant encrypted text will be written in the file myfile.rp

<u>MYFILE.RP</u>

```
“!        /8S.´<xJÆt6   PAX--    S?8cÅWZ<‹PTNMS91EkKI   S      6    A
        ^O,- (F2   [ .    D   K    ,?T.uŠP   5Ê         -A]D$4E%b–
†L'ÃÏ   @ZXH"7 j•Òl'ÒÏ9ACX_cyd*ñÅÂOîÃ¬   F    HPm 7 O?    $ee "'-
  @COm4E*†H_AÒ    TKZS(+ ;(|Êh9'Å?@   IIm;E(Â   EtÑ       SJT90O
%¡ˆ   VîÛM       ]  O_90O%°GXŒü          ÒRHA   )8T*ˆ^S   Ì         PB
@#yS$ZÃïq   Ïõ$BDI    8N%Gu–     5Ò  HL   ,* /ZEÄ          Ã
  oD   m=A?—r@KÃ    E   IJ?6C.    Qp   [   $-   G    KR>yA%j    xS9    ?
EHUPm+U'Ï   ¬NÏ   ÿ   UB 90O%F”oW   Û-     E@".E/¡©JFüù    LMZU!
JbæR   :ù     HN   /8S.”\xJÆ   6   PAX->   S?xc»ã:   <õä9$$H  (+Eg   ×   ÷Ï
À   XMC,yR.   ÃZx  Ï  :   GGX!    (TkýB   9õ-S   J\.-SkW*©î   6õüYL     :
6R's…               ¡]5ÃöBX   NS!5E(}              QH(   -     F[   m:A%   ]Nr
    .   BBX?yD*_ÂCWÃ sE   S =+ I&Cru É3 0×   HN      m   œ
µ¬}B•TH   Em8N/  RDL       -QNX   m;O?È   FkØÆ   -S[BI!    *Ws   J   '
[KND!  k)|çÄx(îÃ   ZCE9<M8üR|iü  !:   ]MQ!   /F]Ó^   -Û   NB-,7A  ,M›-
SÒ   P@@              <           $R   ÞJ   ÏÛ   SSRQ#w            JÒÃ     ÀÃ
~E   C[m=A?ÃÀ_oÃ  Ì   .GT   A  (?I%  @E   ‹        ÒDHNT(yF$k   ec9  ?-
```

^ GRm0N-À,MZÑ- @HYm)R$®roRú+: L _#0
S&x™mY'Û? H TG!8T"SQ±E ü T RP,-
I$B½÷, î¥}udtS,5S$•]n Ã : \U m>E%Úc ZÏ? WODm*O-øXGlü
<B _Gm-H*û´SNüü FPE(+ ?\ìÙ^-öÅ
PW$7T* ¬S ü [B N85A?pžNh0Ô ?SL - ;#t½(M-
ú+VLI m8N/ NW] -GCS ,:C.fu›%60Å3] RP,-
I$B¯Cv ü 'ZQ >-O9|x–,:0Ñ-N OS><k% ‡Yi¥
• SAF.<SknKUÊ<
Ø @ZB90O%°cb_ü+' @ [Qm=A?ÄÀ]eÃÌ -TYL $7 /KmÎ
.Ã VZBS/< *VFÈ$ Ï6ET 85T"L].Œ 3Û\[XK $6N8 VFmÛ
N -6M.àu)•! ? `jl - yS$xÐtc_ü
G__Im R*Ð gßÅe9×T38L(V.(Ðå 9âútT7H7 s Xü£ ÿ•$d
{Cays2ÉÎ }ÒÃ`$Z'n_ < *0 îE'?Ô XG [?<A8¿] ´ö-
õRHNWm*Y8ê` °ü:<õ?3 +,?K".<ñ£ óÛXP_Um8N/ {HF ?
^Y G#- "Yi×ê 6Ñ¥/ aT * kzmÁÒ9 :ÛÏOQF Tm8N/ p^P
6 T_ V#?OAî—Hœ±ÊN¬NWXH(* kx[ÖÖ: ÑÌXH[ayG9ÆÊ
NM×Ì Z'6 +D/ G:é-~ Ï AHL >8C?}I\@. H_I? : H *W |„ -
ÃIC Y9y C*XÁ~˜
Ì0ÃRCC ?8T"f]¯U' üS _J(7T8 þÙ¸Ý¬ õcBH[87I(~gMI<.
H^ ?= $h…ÏA<ÌÅ UE^ GS –'Î ¥±ÃxQB S>-
O9o}L÷.. öOKNH$6NkW]Eø Ÿl HR*u 8UÀÏN ÀÏ
TYRE.1A8 }žh -Ã:+ 48) <Sqï MÐÔ $Ø DQE?u ; RÎÈI
ÅÌ YXIB8+C#s] "<
ØOKNH$6Neêû"¬¥±Ò¥TEH@?0N,Ÿ™KYØÛ H
CW<N?€RX¾×- ü KD @
%8I%ší½ŒÊ½½Òr VW>6U9y Dœ0-ØTGHG"+M*6~LE--]Y)L
$ps -ö<? ¯5 "*Ekkq'¯ ?Ï¬TOM - S?<~ÔgZ<‹`R@BT,5L2–
™Tt×Ô-0JB F>*I%i¡ma<õ .9 KMM m,S. ˜µR[Ìÿ
- BM $7F$qDC1- - nS]>-O9=|ƒ;: Ñ:\R ,+I$`WÛ
Q< Û KJ)0F- cL - P A@,-I$T¬bx ü+'V\ F"y E3{í r-ö -
HR(+Tk fSÃ 5 Ã LV ")R"HDr" 9Û +c_!< ;49êu"-Å?@ SD>-
E&µ8Seõ$" IMIm=R*èToXù 5 J^IC"yW#DÈ?—
Å!ØRHNWm4A%ÕZW]Ã- B_N (4 "x‹ìZ:×ö @Y 8T*M
-j BMP(7TkjxQâ:9üTH W(*;mrÀq9$Æ$WJO)yI%šÍUUÌÊ ^ \R>0

N,½ l{{õ.9< ~K (yM* êjœ Ñ!ÃH_KBm6FkÒH\ÏÊ -Ï]HO$7Gkp zaü.9<üPW _IGhkíûìŠ ÒðîÃ THV#: Yk^Rs ' SUCO9<N(Yï'T ±i PM":E8z An$- < ^] m=I-ÊJ` Ê - HSQ,4M.r Q}5× 9B [_!<Sk\ e 9 FI F=)L"tx-[53(II Q,4SkxL¨Â? õÃD ETcya-ç‡4‹îÏUÊ[L(+I$\« W ö

FW!<Sk^ B - E WI91 /s Ô]' Ê KHU8T8ÎO|k×.9 @ZB90O%°cb_ü+' @ E_4yW9n´Vy0 ü 9 AS4yD",ÏOy!Ê -ATR ?8M&{JUŽ- -×MH@Q>w vãÁ $äÏf ^Z (yI% ÊlU Ñ- [M]4yB.¨„ hëÌ9?LDN $7 8]kÂ (Ï \HFcyw. Ü!p ØZ5 \Em?O9TLMg ?OYT ,>EkF@@ FX U"*TkSl $ OO_87D*,Co‡- 'Ñ `DS!<A/tƒ^ !Ï SL[#yI%^Ü s Ñ 'SO E98T.²X.Oü 3 HJR,-Ek v_Ø ? Ï OBS+0L.—CW2Ê $ _OA(= "@…ÛG ÌÑ

] X8- "N¥ž @ ü × Y HU Y+5E(p] 6 ÅSI Z(+ - KwÒb "Ï? Z E,4Ek_E Ñ ÑOJF (!I8k·N'3è ' ABNm?I'ÏyŒçÏ?Éª ~.,L?p¤L`.ÿ . ZYY#> /NiÙì 5Ã²tJ =+O(rUg? $-C\ E u <X Ø{ Å:NHB]>0L2¦ Qeõ .C^K $+E/"eQrÛ:0Z _Cm0NkëKfë ð $ü\Y Y$5Ee‡( ÿÛ~-ä FU:yT*G•¯mÝü:G F"yW9Gñ(£ ü6öBHFF=5I(`,AT3(U K@ w ZÇÔÞ ÆÏªAIUY#> 8f`æs9<ü:WHD]9yA'f™|„(Ñ:Ô]R /< 9jð, 3ö-XP "7V .QA *H 5 DHM[+0C"S v‡ 9Û\[7~w ôú„±öîÃ¬\OOP#Ss"[ )Ž ZÛ HO Wm* C* ¨S Rbü 5 HA (+E%f¥OR3ö

YC^S)8T* Sf:Ì ?(^Y $7 /O[Ç Ê KHU 8TgÎy ×6- N S+?I(t cœ5-'ÅRN Am)R$ÏWz_É 5 YR $<V."DA{Û ?OF MDm =A?Þ"á,ÃÞ½À gO$-YkbSmÇ+ :ÆqC =,-A)|[‡ Ò Z] $+E /"UC ŽÛ-ÌC EJ .<R?CZ zû $ü RVX>0S?=Ke^:

0 _T\$7T8•|b´Û.-õCC R%< /U{ŠI 3Ã VT^ (:T %RtF 0

ÒT I+6R(XÖmT Ñ3 H JO>0N.I/—z .Ò9 eTm0Sk †ÍÛ

õÊVJ Q(= " O}ÊD 0Ï BRD>- R*)q^j..6 N .:O>G<×~ 9Ñ9VA mi g ö‰©ëÉèE V U+yD *pÒ]6Ã BH #6TkqB¥•. üÆ^ ^N(= k*^ Ít- ÊMVC (*S" \u w < õ.@ _#- E,s7? $' 3" _T\$7TkSO vÃ 9ÑME_S=8R? Sbûð <ü\[XK$6Nk cEÎ +' É yV *+A& So†-95× YC :+I?5¸W6ö

VS GW(yC$@ ÎŽe ÏÅ-] ^ ^(yI?Š `Ý B$TQI^"7S?y@4j' ..R^I@(
(U"z`/“9< 3ÛDCI]?yC#wÞ C5Ò YXT*y O%F ž Z Ñ
IF+0C>b†C-'Å Z M^m)R$ÜPi_É 5 R@ ?:Ekn|—b-
9Ý~ {.0T2†eZyÅ+:<bMW!,R. „{S!ÿ!L R,-A)TQ “ ÆS .6N
8G#` t-$.:N] Em .I?Ø§ } Øù+?J UB)8T.eØ´#9ÑüeC[F!< k^@Ç×-
ÏÔNYO ?* <gvÞg..Ô6 O]W ?+I%w¿Wu?ö -HC ,:C${
S~Í< 9Ñ5* 1E9sº4&SØ6'CA %< (Er † :ÏÅ CMU98I%üJILü
GTR-m8C(ÃgRdÑ< 9) _9+I.U#Eœ 9 Ï\R >yU;wßu9Ã ?DH
MG$5U9] Ðts Ô.3 W NS9yB.6ÓnE6Ñ0 R ()O8LG } -9 _E
87Tk ƒ¢é YÆüõB JR m4A2 GWg (D OS>< "tƒÐR'ÏÛ
^TSE#- 8}pá .?ü:Z . ,R9 S^¹- öFX
F?6B'lR³>9 ¸qITOH9yA(kÕ{n9Ï60_O _,*Ekdu^ ?- ZR@ =<R-
dxv !9! UM 4*T.~¨ | <õ 9RPG +8S?‘ rg Ê5: A SI <)t> a(3
9NYIQ!<Dk]A\Ú --ÅON W.:E8iU6´0 :õ[LF m0N(ÉMaJÑ
'-DUINm0Nkâxeüõ0'üwL
m)R$÷nT}Å? $N_] (4 * Y_Ë| Ñ: ^TO(7Tk^DvÎ
9ÏROIX9yI8©ã üÿ 6BK "6R/EK.d 50W HH(7Tk[bTÕ
"ÔTDF =+O,K`N (5 < Z R /8S.”\xsÆ 6$ E\m0N(Á i@Ñ
'-S _I,- e,O,ÒÃ SÝÛ HMA")L ….-N_Ñ @ V,5A%
^ËN Ã JH R$7Gk Ml†Ä .ÉÃE HNZ < ?P ÕÄ Ï, r$-Y
ksU Ç+ :Æ2~MN.< -#eÏ‘- ÏÊAIU Y#> 8f`æs9<ü:W CO9yL*DÒW,
É ÃU AG =)L"~m-[53(II Q,4SkGOd 9 N ^)<DkT €Ô ÌÃ
SHL]#7 E9™Ô gÀÒZ' R $?F"Jh y-5-' R@ ?:Ek@O@ã ÿ UR
(8C#Þ]p3Ï 5(LHE\m-OkÃ}LÌÃ(Ñ@ C $7GkTC@ë ë GGm
=A?ÃÑH~ÃÝ ?SL T"+ "Ke’} :Û!Q XS98 B*E ‰o Æ<
~yd*€¡ |ÉÃ¬ `WXIH"7* HkŠÒ .½¬ GKYS$6NkOE–Ë ÒÌTOM >
S?|~ª :<õ'J K ,7I8L‘dl Ò'$ CX)L.c3n}?+($ F[/8S.†‹
dÔÃN3 ^ [,-A)ZB “ ÆB\ S"y P9^ÿ2o
ú+:SH @,:TknFzø9-3ù]R /8S.”tNWÆ< RF s)<SkP ^ - HO
Wm8C?«HIjÿ (B _Cm8N/ OSD T ATm=A?Ù A @Ã
P 7, ;S?a[k 5 0 NNJ !0F"Pfˆn .Ê$+RN?<R* > N 93-
SO 91Ek@g~¯ 3:üeM <(yA9e ej ?.JHZ !yO-€ÎGoÔÃ ?TSG^G
H2¡:Sl‹K (N
< - "%¥ÆëNüÃÛKS-Z(/E' a

‡Ï+ ÔGKYS$6Ne”,j Ûj. BDXU>yH$Y WK RLIS.- U*Tq (ÃBX H#yD*qÏ [.Ã nYG (*C9l| F 3 GCFm5O< H]_ - PP >-R>vHS 0 _ 8I' ´+_î½o _ fW;<LA(f·f ‹ T 5- #~)ßI35Ø BJ Sm6Fk ÒFuãÃ ?üOFD m TkÑ Z˜Ìt ÅU \^,- /F Ý“ ÃÌFNC (= "_”ÊP ÌÌ SL S#= <CFµ¼ üùRDJ\>1I;¦^|fõ 3+FQU *yT#TÒ¼ ×îØxO E$;E8ƒPR[Û Y[X /8 S.”}tLÆ5: PPY $7 *ƒmË Û'Û WP^ .-U9N¦±> îÿjOH $7 'DXÀ Û PR Sm7E.Ýw†GÌ'Ï T P,+Eke4M ?. [B\J9 $z¿á\6ëð LW(/ E' Ÿ]?qÏ 3" SN & =)P:V] " “

7. CONCLUSION

The significance of data security scheme is tremendous now days. In such situation the evolution of the new scheme is beneficial to the data security concerns. The interest of security concerns is to obtain the computationally secure, fast executing, dual implemental (hardware/software) algorithm. This research work provides an efficient algorithm that can be implemented in hardware as well as in software. The software implementation is done in object oriented programming approach using C++. The implementation codes are in the appendix section of the document.

There is a popular concept that, the encryption scheme is not strong if it is dependent on the plain text. This works proves that the encryption can be strong even if it may be dependent on the plain text. The concept of encryption that uses the part of plain text to generate key is known as ‘Code Reusing technique’.

Basically this algorithm is designed considering the authentication problem. The first 32 bits of the plain text can be considered as the nonce for authentication hence can be mixed into Needam Schroeder’s authentication protocol.

The strong points regarding the algorithm are the generation of keys. The key is 96 bit long and provides strong resistance against the brut force attacks. The operation ROOP which is the XOR operation of the bits in a block, acts the role of confusion. The XOR operation of different blocks will act as the diffusion operation. The use of plain text bits in the generation of key provides the randomization so that it will provide strong resistance against the known cipher text attack. Since the operation do not use any substitution boxes so it is free from hidden weaknesses also. The encryption is done in bit level so it is quite easy to understand and can be implemented for any kind of application like text applications as well as graphics, sound etc. Since it uses the ASCII values for encryption, it is free from the burden of character set and the language of the message.

The use of data compression in conjunction with any encryption algorithm drastically increases the security of the encrypted data. For natural language text, one approach that yields improved compression over the compression of constant size blocks of data is the compression of linguistic units, such as words. I have achieved reversible compression of such files to less than 35% of their original size using linguistic parsing and a Huffman code. For binary data, such as computer programs, the use of existing techniques, such as those in PKARC, written by Phil Katz and available on most computer bulletin boards, are recommended. It is hoped that the encryption algorithm, as well as some of my ideas on data compression, will make a positive contribution towards data security, communications privacy, and efficiency of data storage and transmission.

8. REFERENCES

[1] A. Shamir, J. Patarin, N. Courtois and A. Klimov: *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations* , Eurocrypt'2000, LNCS 1807, Springer.

[2] Alan G. Konheim, *Cryptography: A Primer* , New York.

[3] C. B. Brookson and S. C. Serpell, *Security on the British Telecom Satstream Service* in International Conference on Secure Communication Systems, 22—23 February 1984, London.

[4] C. E. Shannon, *Communication Theory of Secrecy Systems*, Bell System Technical Journal, Volume 28, 1949.

[5] C. E. Shannon, *A Mathematical Theory of Communication*, Bell System Technical Journal, 1948.

[6] C. P. Schnorr, *Is the RSA Scheme Safe?* in Proceedings of the Workshop on Cryptography at Burg Feuerstein, Germany, 1982 Springer-Verlag, 1983.

[7] Carl H. Meyer & Stephen M. Matyas, *Cryptography: a New Dimension in Computer Data Security* — A Guide for the Design and Implementation of Secure Systems, 1982 New York.

[8] Cipher A. Deavers, David Kahn, Louis Kruh, Greg Mellen, and Brian Winkel, *Cryptology Yesterday, Today and Tomorrow* , 1987, Norwood.

[9] D Elminaam, M A Kader, M Hadhoud *Performance Evaluation of Symmetric Encryption Algorithms* , IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[10] D. G. Haenshke, D. A. Kettler, and E. Oberer, *Network Management and Congestion in the U. S. Telecommunications Network,* IEEE Transactions on Communications, volume COM-29, 1981.

[11] D. J. Bond, *Practical Primality Testing,* in International Conference on Secure Communication Systems, 1984, London.

[12] David B. Newman, Jr. and Raymond L. Pickholtz, *Cryptography in the Private Sector,* IEEE Communications Magazine, August 1986, Volume 24, Number 8, New York.

[13] David Kahn, *The Codebreakers — The Story of Secret Writing* , New York: The MacMillan Company, 1987.

[14] Dorothy Elisabeth Robling Denning, *Cryptography and Data Security* , Menlo Park, 1982.

[15] Eli Biham, *A Fast New DES Implementation in Software* , FSE'97, Springer, LNCS 1267.

[16] Fred Piper, *Stream Ciphers* ;Proceedings of the Workshop on Cryptography at Burg Feuerstein, Germany, 1982 : Springer-Verlag, 1983.

[17] Friedrich L. Bauer, *Cryptology — Methods and Maxims,* in Proceedings of the Workshop on Cryptography at Burg Feuerstein, Germany, 1982, Springer-Verlag, 1983.

[18] G. Goos and J. Hartmans, *Key of Computer Science: The Cryptology* Lecture notes, 2009.

[19] Gavin Lowe, *Breaking and fixing the Needham Schroeder public key protocol using FDR* , Oxford University Computing laboratory.

[20] H. J. Beker, *A Survey of Encryption Algorithms,* in International Conference on Secure Communication Systems, 1984, London.

[21] H. R. Chivers, *A Practical Fast Exponentiation Algorithm for Public Key,* in International Conference on Secure Communication Systems, 1984, London.

[22] Hardjono, *Security In Wireless LANS And MANS,* Artech House Publishers 2005.

[23] Henk C.A. van Tilborg, *Fundamentals of Cryptology* , Eindhoven University of Technology, Netherlands, 1990.

[24] J. A. Gordon and H. Retkin, *Are Big S-Boxes Best?* in Proceedings of the Workshop on Cryptography at Burg Feuerstein, Germany, 1982, New York: Springer-Verlag, 1983.

[25] J. Sattler and C. P. Schnorr, *Ein Effizienzvergleich Der Faktorisierungsverfahren von Morrison-Brillhart und Schroeppel,* Workshop on Cryptography at Burg Feuerstein, Germany, 1982, Springer-Verlag, 1983 (English Version).

[26] J. Vandewalle, R. Govaerts, W. De Becker, M. Decroos, & G. Speybrouk, *RSA–Based Implementation of Public Key Cryptographic Protection in Office Systems* International Carnahan Conference on Security Technology: Electronic Crime Countermeasures, 1986.

[27] Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators* , FSE'96, LNCS 1039, Springer.

[28] Leslie S. Chalmers, *An Analysis of the Differences Between the Computer Security Practices in the Military and Private Sectors,* IEEE 1986 Symposium on Securi ty and Privacy, Oakland, CA.

[29] Lester S. Hill, *Cryptography in an Algebraic Alphabet,* American Mathematical Monthly, June 1929.

[30] Marjanne Plasmans, *White-Box Cryptography for Digital Content Protection* , Master's Thesis, 2005.

[31] Markku-Juhani Olavi Saarinen: *A Time-Memory Tradeoff Attack Against LILI-128* , FSE 2002, LNCS 2365, Springer.

[32] Mehmet Kivanc Mihcak (2002), *Information Hiding Codes and Their Applications to Images and Audio* , PhD Thesis, Graduate College of the University of Illinois at Urbana-Champaign, US, 2002.

[33] Michael Paul Johnson, *The Diamond2 Block Cipher* , University of Colorado, 1995.

[34] National Bureau of Standards, Federal Information Processing Standards Publication Number 46 Dated 15 January 1977.

[35] Nicolas Courtois and Jacques Patarin, *About the XL Algorithm over GF(2)* , Cryptographers' Track RSA 2003, LNCS 2612, Springer 2003.

[36] Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback* , LNCS 2729, Springer 2003.

[37] Ole Immanuel Franksen, Mr. Babbage's Secret — *The Tale of a Cipher and APL,* Englewood Cliffs, NJ: Prentice-Hall, 1985.

[38] P.K. Jha, *A Bit Level Symmetric Encryption Technique Through Recursive Addition of Block (RAB)* , International Conference on Electronic Commerce in the 21st Century; Kathmandu, 2008.

[39] Poonam Garg, *A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm* IJNSA, Vol.1, No 1, 2009.

[40] R. H. Cooper, *Linear Transformations in Galois Fields and their Application to Cryptography,* Cryptologia Magazine, Volume 4, Number 3, 1980.

[41] R. M. Needham, M D Schroeder, *Using encryption for authentication in large networks of computers*, Communications of the ACM, Volume 21, Number 12,1978

[42] Ralph C. Merkle, *Secrecy, Authentication, and Public Key Systems,* Ann Arbor, MI: UMI Research Press,1982.

[43] S K Yadav, *Content Hiding: An Image Approach*. Project Work in Partial Fulfillment of The Requirements For Master Degree in Computer Science and Information Technology, TU, 2008.

[44] Samer Younes, *An improved quorum selection algorithm IQSA* Thesis submitted in partial fulfillment of the requirements for Master Degree of Science in Computer Science Lebanese American University, 2007.

[45] Tony Patti, *A Galois Field Cryptosystem, 1986.* Published electronically by Tony Patti, editor of Cryptosystems Journal, 9755 Oatley Lane, Burke, VA 22015.

[46] Vin McLellan, *Drugs and DES: A New Connection,* Digital Review, 1987.

[47] Willi Meier and Othmar Staffelbach: *Nonlinearity Criteria for Cryptographic Functions* , Eurocrypt'89, LNCS 434, Springer, 1990.

[48] William Stallings, *Cryptography and Network Security* , Fourth Edition, Prentice Hall of India; ISBN-978-81-203-3018-4.

[49] Wladyslaw Kozaczuk, *Enigma — How the German Machine Cipher was Broken and How it was Read by the Allies in World War Two* , University Publications of America, Inc., 1984.

[50] X. Lai, J. L. Massey, *Markov ciphers and Differential cryptanalysis* , Springer Verlag 1998.

9. BIBLIOGRAPHY

[51] A. M. Jackson, N. A. McEvoy, and B. B. Newman, *Project Universe Encryption Experiment* in International Conference on Secure Communication Systems, 22—23 February 1984, London,

[52] Alex Biryukov, Adi Shamir: *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers* , Asiacrypt 2000, LNCS 2248, Springer.

[53] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography* ,CRC Press.

[54] C. J. Mitchell, *A Comparison of the Cryptographic Requirements for Digital Secure Speech Systems Operating at Different Bit Rates,* in International Conference on Secure Communication Systems, 1984, London.

[55] Carl E. Landwehr, *The Best Available Technologies for Computer Security,* Computer, Vol. 16, No. 7, 1983.

[56] Christer Lindén, *Electronic Seal for Protection of Electronic Money in Sweden, Finland, and Norway* in Proceedings of 1986 International Carnahan Conference on Security Technology: Electronic Crime Countermeasures1986.

[57] Cipher A. Deavers and Louis Kruh, *Machine Cryptography and Modern Cryptanalysis* , 1985, Norwood.

[58] Dave Bursky, *Protect Your EEPROM Data and Gain More Flexibility,* Electronic Design, Waseca,1988.

[59] Douglas J. Albert and Stephen P. Morse, *Combating Software Piracy by Encryption and Key Management,* CA: IEEE Computer Society, Vol. 21, No. 5, 1984, Los Alamitos.

[60] Eric Filiol: *Decimation Attack of Stream Ciphers* , Indocrypt 2000, LNCS 1977, 2000.

[61] Frederik Armknecht, *A Linearization Attack on the Bluetooth Key Stream Generator* , Available on http://eprint.iacr.org/2002/191/.

[62] Frederik Armknecht, Matthias Krause, *Algebraic Atacks on Combiners with Memory* , Crypto 2003, LNCS 2729, Springer.

[63] Fredrik Jonsson, Thomas Johansson: *A Fast Correlation Attack on LILI-128* , Can be found at http://www.it.lth.se/thomas/papers/paper140.ps

[64] Gilbert Held and Thomas R. Marshall, *Data Compression* , Second Edition; 1987, New York.

[65] H. H. Ahmed, H M. Kalash, O S. Farag Allah, *Implementation of RC5 Block Cipher Algorithm for Image Cryptosystems* , International Journal of Information Technology Volume 3 Number 4

[66] H. J. Beker, J. M. K. Friend, and P. W. Halliden, *Simplifying Key Management in Electronic Fund Transfer Point of Sale Systems,* in International Conference on Secure Communication Systems, 1984, London.

[67] H. Retkin, *Multi-Level Knapsack Encryption,* in International Conference on Secure Communication Systems, 1984, London.

[68] Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme* Eurocrypt' 88, Springer, LNCS 963, 1995.

[69] James A. Storer, *Data Compression Methods and Theory* . Rockville, MD: Computer Science Press, 1988.

[70] Jovan Dj. Golic: *Fast low order approximation of cryptographic functions* , Eurocrypt'96, LNCS 1070, Springer.

[71] Jui-Cheng Yen and Jiun-In Guo, *A new image encryption algorithm and its VLSI architecture,* IEEE Work-shop Signal Processing Systems, 1999.

[72] Lester J. Fraim, *Scomp: A Solution to the Multilevel Security Problem,* Computer, Vol. 16, No. 7, Los Alamitos, CA: IEEE Computer Society,1983.

[73] Lester S. Hill, *Concerning Certain Linear Transformation Apparatus of Cryptography,* American Mathematical Society, 1931.

[74] M. Mihaljevic, H. Imai: *Cryptanalysis of Toyocrypt-HS1 stream cipher* , IEICE Transactions on Fundamentals, vol. E85-A, pp. 66-73, Jan. 2002.

[75] N. Lodge, B. Flannaghan, and R. Morcom, *Vision Scrambling of C-MAC DBS Signals,* International Conference on Secure Communication Systems, 1984, London.

[76] Nicolas Courtois, *The security of Hidden Field Equations (HFE)* , Cryptographers' Track Rsa Conference 2001, LNCS 2020, Springer 2001.

[77] Nicolas Courtois, *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt* , ICISC 2002, Seoul, Korea, LNCS 2587, Springer 2002.

[78] Palash Sarkar, Subhamoy Maitra: *Nonlinearity Bounds and Constructions of Resilient Boolean Functions* , LNCS 1880, Springer,2000.

[79] Palash Sarkar: *The Filter-Combiner Model for Memoryless Synchronous Stream Ciphers* , LNCS 2442, Springer,2002.

[80] Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier, *On Correlation-immune Functions* , LNCS 576, Springer, 1991.

[81] Phil Katz, *PKARC FAST! Archive Create/Update Utility Version 3.5* , published electronically in PKX35A35.EXE.

[82] Rainer A. Rueppel: *Analysis and Design of Stream Ciphers* , Springer, 1986.

[83] Richard Doherty, *FBI Nabs Pirated VCII,* Electronic Engineering Times, Manhasset, N. Y.: CMP Publications, Inc., Issue 500, 1988.

[84] Ross Anderson, *Searching for the Optimum Correlation Attack* , FSE'94, LNCS 1008, Springer, 1994.

[85] Simpson, E. Dawson, J. Golic and W. Millan: *LILI Keystream Generator* Springer.

[86] Stanley R. Ames, Jr., Morrie Gasser, and Roger R. Schell, *Security Kernel Design and Implementation: An Introduction,* Computer, Vol. 16, No. 7, CA: IEEE Computer Society,1983.

[87] Steve Babbage, *Cryptanalysis of LILI-128* , Nessie project internal report, available at https://www.cosic.esat.kuleuven.ac.be/nessie/reports/.

[88] Tekla S. Perry and Paul Wallich, *Can Computer Crime be Stopped?,* IEEE Spectrum, 1984.

[89] Terry Costlow, *Global Computer Network Cracks Cryptographic Mathematics Barrier,* Electronic Engineering Times, Issue 508, 1988.

[90] Thomas Beth, *Introduction to Proceedings of the Workshop on Cryptography at Burg Feuerstein* , Germany, Springer-Verlag, 1983.

[91] Ultron Labs Corporation, *Crypto Module Processes TOP SECRET Data,* EE Product News, 1988.

[92] W. P. Lu and M. K. Sandareshan, *A Hierarchical Key Management Scheme for End-to-End Encryption in Internet Environments* in Proceedings of the IEEE 1986 Symposium on Security and Privacy, Oakland, CA, 1986.

[93] W. W. Rouse Ball & H. S. M. Coxeter, *Mathematical Recreations & Essays* . New York: The MacMillan Company, 1962.

[94] Willi Meier and Othmar Staffelbach: *Fast correlation attacks on certain stream ciphers* , Journal of Cryptology, Vol 1 No. 3, 1989.

APPENDIX

First 32 bit of plain text

```
//**********************
int bpt[blocksize];
//************************************************************
```

Function to concatenate the blocks

```
//*******************************
void concatenate(int k1[blocksize],int k2[blocksize],int k3[blocksize],int res[3*blocksize])
  {
        for(int i=0;i<blocksize;i++)
    {
        res[i] = k1[i];
      res[blocksize+i]=k2[i];
      res[(2*blocksize)+i] = k3[i];
    }
        }
//************************************************************
```

Function to convert char to binary

```
//*******************************
void bin(char blkchar[], int blkbin[])
{
        int i, count=0;
  for (i=0;i<(blocksize/8);i++)
  {
                int temp = blkchar[i];
    int bstr[8];
    int j;
    for(j = 8; j>0;j--)
    {
        int rem;
      rem = temp %2;
      temp = temp /2;
```

```
          bstr[j-1] = rem;
        }

        for(j=0;j<8;j++)
        blkbin[count++]=bstr[j];
            }
}
   int xor(int a, int b)
        {
            if(a!=b)
            return 1;
          return 0;
        }

            void roop(int first[blocksize])
        {
            for(int i=0;i<blocksize-1;i++)
            first[i]=xor(first[i],first[i+1]);
        }

//***********************************************************
```

Block wise XOR operation(BROOP)

```
//************************************
        void broop(int first[blocksize],int second[blocksize])
        {
            for(int i = 0;i<blocksize;i++)
            first[i] = xor(first[i],second[i]);
        }

//***********************************************************
```

The class to generate key

```
//*******************************************
class keygen
{
  int key1[blocksize];
  int key2b1[blocksize];
  int key2b2[blocksize];
  int ik[3*blocksize];
  int ek[3*blocksize];
  friend class encrypt;
  void proc1(int tk1[blocksize],int tk2[blocksize])
        {
                unsigned long int dec_tk1;
                int temp;
                dec_tk1 = bintodec(tk1);
                temp=dec_tk1%3;
                switch(temp)
                {
                        case 0: concatenate(tk1,key1,tk2,ik);
                                        break;
                        case 1: concatenate(tk2,key1,tk1,ik);
                                        break;
                        case 2: concatenate(tk1,tk2,key1,ik);
                                        break;
                }

        }
  //************************************
void proc2(int pk[blocksize])
        {
                unsigned long int dec_pk;
                int temp;
                int i;
                int ik1[blocksize],ik2[blocksize],ik3[blocksize];
                dec_pk = bintodec(pk);
                temp=dec_pk % 2;
                for(i=0;i<blocksize;i++)
                {
                        ik1[i]=ik[i];
                        ik2[i]=ik[i+blocksize];
                        ik3[i]=ik[i+(blocksize*2)];
                }
                switch(temp)
                {
                        case 0:  broop(pk,ik1);
                                        concatenate(pk,ik2,ik3,ek);
                                        break;
                        case 1: broop(pk,ik3);
                                        concatenate(ik1,ik2,pk,ek);
                                        break;
                }
        }

//*************************************************************
  unsigned long int bintodec(int tk1[])
        {
                int i;
                unsigned long int dec=0;
                cout<<endl;
                for(i=0;i<blocksize;i++)
                {
                        dec =dec +(pow(2,i)*tk1[i]);


                }
                return dec;
        }
```

```
//****************************************
   public:
   friend int key();
   //****************************************
void getdata(int k1[blocksize], int k2b1[blocksize],int k2b2[blocksize])
      {
           for(int i = 0;i<blocksize;i++)
         {
           key1[i] = k1[i];
            key2b1[i] = k2b1[i];
            key2b2[i] = k2b2[i];
         }
      }
   //****************************************
     void keyprocess()
     {
           int tk1[blocksize];
           int tk2[blocksize];
           int pk[blocksize];
         for(int i = 0;i<blocksize;i++)
         {
           tk1[i] = key2b1[i];
            tk2[i] = key2b2[i];
            pk[i] = key1[i];
         }
           roop(tk1);
           broop(tk2,key1);
           broop(pk, bpt);
         proc1(tk1,tk2);
         proc2(pk);
     }
//*******************************************
     void display(void)
      {
           cout<<endl<<"Key 1:"<<endl;
           for(int i = 0;i<blocksize;i++)
           cout<<key1[i];
         cout<<endl<<"Key2b1:"<<endl;
         for(int i = 0;i<blocksize;i++)
           cout<<key2b1[i];
         cout<<endl<<"Key2b2:"<<endl;
         for(int i = 0;i<blocksize;i++)
           cout<<key2b2[i];
         cout<<endl<<"Initial Key:"<<endl;
         for(int i = 0;i<(blocksize*3);i++)
           cout<<ik[i];
         cout<<endl<<"Extended Key:"<<endl;
         for(int i = 0;i<(blocksize*3);i++)
           cout<<ek[i];
      }
      friend void getkey(int key1[],int e1[],int e2[],int e3[],int i1[],int i2[],int i3[]);
      //*************************************************
int key()
{
           ifstream inkey;
   int k1[blocksize];
   int k2b1[blocksize];
   int k2b2[blocksize];
   char data[(blocksize/8)+1];
  //----------data for completing the block size--------
   inkey.open("key.dat");
   if(!inkey)               // file couldn't be opened
   {
      cerr << "Error: file could not be opened" << endl;
      getch();
      exit(1);
   }
   int i;
   for(i = 0;i<(blocksize/8)+1;i++)
           data[i]='a';
           for(int j = 0;j<3;j++)
      {
           for(i = 0; i<(blocksize/8);i++)
           {
           if(!inkey.eof())
            {
                       inkey.get(data[i]);
            }
            else
                                  data[i] = 'p';

         }
         data[i] = '\0';
  //*********************************************
         switch(j)
         {
         case 0: bin(data, k1);
           break;
         case 1: bin(data, k2b1);
           break;
         case 2: bin(data, k2b2);
           break;
         }
      }
      getdata(k1,k2b1,k2b2);
  //****************************************************
  inkey.close();
  keyprocess();
}
```

```
//************************
void getkey(int k1[blocksize],int e1[blocksize],int e2[blocksize],int
e3[blocksize],int i1[blocksize],int i2[blocksize],int i3[blocksize])
    {
        for(int i = 0;i<blocksize;i++)
      {
        k1[i] = key1[i];
        e1[i] = ek[i];
        e2[i] = ek[blocksize+i];
        e3[i] = ek[2*blocksize+i];
        i1[i] = ik[i];
        i2[i] = ik[blocksize+i];
        i3[i] = ik[2*blocksize+i];
      }
    }
};
//***********************************************************
```

Class to encrypt the file

```
//************************************
class encrypt
{
        int pt1[blocksize],pt2[blocksize],pt3[blocksize],pt4[blocksize];
  int ct1[blocksize],ct2[blocksize],ct3[blocksize],ct4[blocksize];

//**********************************************************
  void bin2char(int binblk[],char charblk[]) // The function to convert binary
string to ASCII
        {
                int b[8]={128,64,32,16,8,4,2,1};
                for(int i =0;i<blocksize/8;i++)
        {
                int val = 0;
                for(int j = 0;j<8;j++)
             {
                val = val+(binblk[i*8+j]*b[j]);
        }
             charblk[i] = val;
        }
        }
  //****************************************************
  public:
        void   getdata(int   p1[blocksize],int   p2[blocksize],int   p3[blocksize],   int
p4[blocksize])
      {
        for (int i = 0;i<blocksize;i++)
      {
        pt1[i] = p1[i];
        pt2[i] = p2[i];
        pt3[i] = p3[i];
        pt4[i] = p4[i];
        bpt[i] = p1[i];
      }
      }
      void proc()
      {
        int s1[blocksize],s2[blocksize],s3[blocksize],s4[blocksize];
      int
k1[blocksize],e1[blocksize],e2[blocksize],e3[blocksize],i1[blocksize],i2[blocksize
],i3[blocksize];
      keygen k;
      k.key();
      k.getkey(k1,e1,e2,e3,i1,i2,i3);
      for(int i=0;i<blocksize;i++)
      {
        s1[i] = pt1[i];
        s2[i] = pt2[i];
        s3[i] = pt3[i];
        s4[i] = pt4[i];
      }
      broop(s2, pt1);
      broop(s3, pt2);
      broop(s4, pt3);
      broop(s1,k1);
      broop(s2,e1);
      broop(s3,e2);
      broop(s3,e3);
      broop(s1,i1);
      broop(s2,i2);
      broop(s3,i3);
      roop(s4);
      broop(s1,s2);
      broop(s2,s3);
      broop(s3,s4);
      for(int i=0;i<blocksize;i++)
      {
        ct1[i] = s1[i];
        ct2[i] = s2[i];
        ct3[i] = s3[i];
        ct4[i] = s4[i];
      }
      }
      void bin_to_char(char cd[])
                {
        char cip[blocksize/8];
      int i= 0;
      bin2char(ct1,cip);
      for(i=0;i<blocksize/8;i++)
        cd[i] = cip[i];
      bin2char(ct2,cip);
      for(i;i<2*(blocksize/8);i++)
        cd[i] = cip[i-(blocksize/8)];
```

```
        bin2char(ct3,cip);
        for(i;i<3*(blocksize/8);i++)
          cd[i] = cip[i-(2*blocksize/8)];
        bin2char(ct4,cip);
        for(i;i<4*(blocksize/8);i++)
          cd[i] = cip[i-(3*blocksize/8)];
                    }
};
//*************************************************************
int main()
{
        ifstream indata;
  encrypt r;
  int b0[blocksize];
  int b1[blocksize];
  int b2[blocksize];
  int b3[blocksize];
        char data[blocksize+1];
  char cd[blocksize/2];
//   char d1[textsize];
//   char d2[textsize];
//   char d3[textsize];
//----------data for completing the block size--------
  indata.open("myfile.dat");
  if(!indata)            // file couldn't be opened
  {
    cerr << "Error: file could not be opened" << endl;
    getch();
    exit(1);
  }
  ofstream outdata;
  outdata.open("myfile.rp");
  int i;
  for(i = 0;i<(blocksize/8)+1;i++)
```

```
          data[i]='a';
 while(!indata.eof())
 {
          for(int j = 0;j<4;j++)
    {
          for(i = 0; i<blocksize/8;i++)
          {
          if(!indata.eof())
           {
                     indata.get(data[i]);
           }
           else
                               data[i] = 'p';

       }
       data[i] = '\0';
//*********************************************************
       switch(j)
       {
       case 0: bin(data, b0);
         cout<<data;
          getch();
         break;
       case 1: bin(data, b1);
         cout<<data;
          getch();
         break;
       case 2: bin(data, b2);
         cout<<data;
          getch();
         break;
       case 3: bin(data, b3);
         cout<<data;
          getch();
```

```
            break;
          }
        }
        r.getdata(b0,b1,b2,b3);
        r.proc();
    //****************************************************
            r.bin_to_char(cd);
        for(i=0;i<blocksize/2;i++)
          outdata.put(cd[i]);


     }
    indata.close();
    outdata.close();
     getch();
  }
```