Optimized DES Algorithm Using X-nor Operand Upto 4 Round on Spartan3

[1] PoojaRathore, [2] Jaikarn Singh, [3] MukeshTiwari, [4]Sanjay Rathore
[1, 2, 3, 4] Dept. of ECE, SSSIST
Sehore, (MP) – INDIA.
Department of Electronics and Communication Engineering, SSSIST, Sehore, (MP)

Abstract —
In this paper, linear cryptanalysis is a known-plaintext attack that uses a linear relation between input-bits, output-bits, and key-bits of an encryption algorithm that holds with a certain probability. If enough plaintext-ciphertext pairs are provided, this approximation can be used to assign probabilities to the possible keys and to locate the most probable one. Along with the society relies on more and more greatly to the computer, people also attach more and more importance to the security problem in the application. The cryptography is continuously safeguarding the safe effectively protective screen of system. Owing to the fact that to break the key using mathematics technology is very difficult, people put forward the side-channel attack method in recent years.

Keywords — Encryption; Key; Modalism; S-boxes.

Introduction

Data Security is an important parameter for the industries.It can be achieved by Encryption algorithms which are used to prevent unauthorized access of data. Cryptography is the science of keeping data transfer secure, so that eavesdroppers(or attackers) cannot decipher the transmitted Message. In this paper the DES algorithm is optimized upto 4 round using Xilinx software and implemented on Spartan 3 Modelsim. The paper deals with various parameters such as variable key length, key generation mechanism, etc. used in order to provide optimized results.

The DES Algorithm IllustratebyJ. Orlin Grabbe

The DES (Data Encryption Standard) algorithm is the most widely used encryption algorithm in the world. For many years, and among many people, "secret code making" and DES have been synonymous. And despite the recent coup by the Electronic Frontier Foundation in creating a $220,000 machine to crack DES-encrypted messages, DES will live on in government and banking for years to come through a life- extending version called "triple-DES." How does DES work? This article explains the various steps involved in DES-encryption, illustrating each step by means of a simple example. Since the creation of DES, many other algorithms (recipes for changing data) have emerged which are based on design principles similar to DES. Once you understand the basic transformations that take place in DES, you will find it easy to follow the steps involved in these more recent algorithms.But first a bit of history of how DES came about is appropriate, as well as a look toward the future.

The National Bureau of Standards Coaxes the Genie from the BottleOn May 15, 1973, during the reign of Richard Nixon, the National Bureau of Standards (NBS) published a notice in the Federal Register soliciting proposals for cryptographic algorithms to protect data during transmission and storage. The notice explained why encryption was an important issue.Over the last decade, there has been an accelerating increase in the accumulations and communication of digital data by government, industry and by other organizations in the private sector. The contents of these communicated and stored data often have very significant value and/or sensitivity. It is now common to find data transmissions which constitute funds transfers of several million dollars, purchase or sale of securities, warrants for arrests or arrest and conviction recordsbeing communicated between law enforcement agencies, airline reservations and ticketing representing investment and value both to the airline and passengers, and health and patient care records transmitted among physicians and treatment centers.

The increasing volume, value and confidentiality of these records regularly transmitted and stored by commercial and government agencies has led to heightened recognition and concern over their exposures to unauthorized access and use. This misuse can be in the form of theft or defalcations of data records representing money, malicious modification of business inventories or the interception and misuse of confidential information about people. The need for protection is then apparent and urgent.

It is recognized that encryption (otherwise known as scrambling, enciphering or privacy transformation) represents the only means of protecting such data during transmission and a useful means of protecting the content of data stored on various media, providing encryption of adequate strength can be devised and validated and is inherently integrable into system architecture. The National Bureau of Standards solicits proposed techniques and algorithms for computer data encryption. The Bureau also solicits recommended techniques for implementing the cryptographic function: for generating, evaluating, and protecting cryptographic keys; for maintaining files encoded under expiring keys; for making partial updates to encrypted files; and mixed clear and encrypted data to permit labelling, polling, routing, etc. The Bureau in its role for establishing standards and aiding government and industry in assessing technology, will arrange for the evaluation of protection methods in order to prepare guidelines.

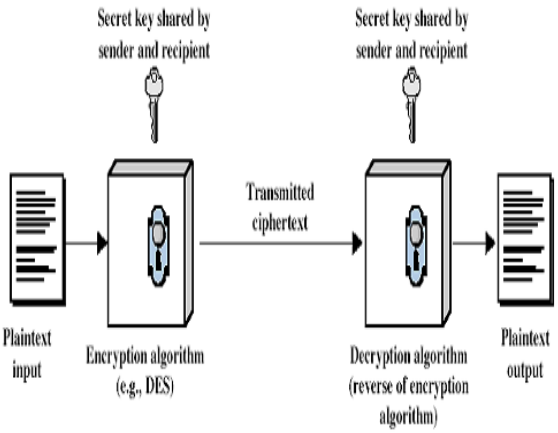
NBS waited for the responses to come in. It received none until August 6, 1974, three days before Nixon's resignation, when IBM submitted a candidate that it had developed internally under the name LUCIFER. After evaluating the algorithm with the help of the National Security Agency (NSA), the NBS adopted a modification of the LUCIFER algorithm as the new Data Encryption Standard (DES) on July 15, 1977.

DES was quickly adopted for non-digital media, such as voice-grade public telephone lines. Within a couple of years, for example, International Flavors and Fragrances was using DES to protect its valuable formulas transmitted over the phone ("With Data Encryption, Scents Are Safe at IFF," Computerworld 14, No. 21, 95 (1980).)

Meanwhile, the banking industry, which is the largest user of encryption outside government, adopted DES as a wholesale banking standard. Standards for the wholesale banking industry are set by the American National Standards Institute (ANSI). ANSI X3.92, adopted in 1980, specified the use of the DES algorithm.

Cryptography: Overview

An overview of the main goals behind using cryptography will be discussed in this section along with the common term used in this field.



Encrption / Decryption

Cryptography is usually referred to as "the study of secret", while nowadays is most attached to the definition of encryption. Encryption is the process of converting plain text "unhidded" to a cryptic text "hidded" to secure it against data thieves.This process has another part where cryptic text needs to be decrypted on the other end Some Preliminary Examples of DES works on bits, or binary numbers--the 0s and 1s common to digital computers. Each group of four bits makes up a hexadecimal, or base 16, number. Binary "0001" is equal to the hexadecimal number "1", binary "1000" is equal to the hexadecimal number "8", "1001" is equal to the hexadecimal number "9", "1010" is equal to the hexadecimal number "A", and "1111" is equal to the hexadecimal number "F".

DES works by encrypting groups of 64 message bits, which is the same as 16 hexadecimal numbers. To do the encryption, DES uses "keys" where are also apparently 16 hexadecimal numbers long, or apparently 64 bits long. However, every 8th key bit is ignored in the DES algorithm, so that the effective key size is 56 bits. But, in any case, 64 bits (16 hexadecimal digits) is the round number upon which DES is organized.

For example, if we take the plaintext message "8787878787878787", and encrypt it with the DES key "0E329232EA6D0D73", we end up with the ciphertext "0000000000000000". If the ciphertext is decrypted with the same secret DES key "0E329232EA6D0D73", the result is the original plaintext "8787878787878787".

This example is neat and orderly because our plaintext was exactly 64 bits long. The same would be true if the plaintext happened to be a multiple of 64 bits. But most messages will not fall into this category. They will not be an exact multiple of 64 bits (that is, an exact multiple of 16 hexadecimal numbers).

For example, take the message "Your lips are smoother than vaseline". This plaintext message is 38 bytes (76 hexadecimal digits) long. So this message must be padded with some extra bytes at the tail end for the encryption. Once the encrypted message has been decrypted, these extra bytes are thrown away. There are, of course, different padding schemes -- different ways to add extra bytes. Here we will just add 0s at the end, so that the total message is a multiple of 8 bytes (or 16 hexadecimal digits, or 64 bits).

The plaintext message "Your lips are smoother than vaseline" is, in hexadecimal, "596F7572206C6970 732061726520736D 6F6F746865722074 68616E2076617365 6C696E650D0A".
(Note here that the first 72 hexadecimal digits represent the English message, while "0D" is hexadecimal for Carriage Return, and "0A" is hexadecimal for Line Feed, showing that the message file has terminated.) We then pad this message with some 0s on the end, to get a total of 80 hexadecimal digits:
"596F7572206C6970 732061726520736D 6F6F746865722074 68616E2076617365 6C696E650D0A0000".

If we then encrypt this plaintext message 64 bits (16 hexadecimal digits) at a time, using the same DES key "0E329232EA6D0D73" as before, we get the ciphertext:
"C0999FDDE378D7ED 727DA00BCA5A84EE 47F269A4D6438190 9DD52F78F5358499 828AC9B453E0E653".

This is the secret code that can be transmitted or stored. Decrypting the ciphertext restores the original message "Your lips are smoother than vaseline". (Think how much better off Bill Clinton would be today, if Monica Lewinsky had used encryption on her Pentagon computer!)
DES is a block cipher--meaning it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size. Thus DES results in a permutationamong the 2^64 (read this as: "2 to the 64th power") possible arrangements of 64 bits, each of which may be either 0 or 1. Each block of 64 bits is divided into two blocks of 32 bits each, a left half block L and a right half R. (This division is only used in certain operations.)

Example: Let M be the plain text message M = 0123456789ABCDEF, where M is in hexadecimal (base 16) format. Rewriting M in binary format, we get the 64-bit block of text:

M =	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
L =	0000		0001		0010		0011		0100		0101		0110		0111	

R = 1000 1001 1010 1011 1100 1101 1110 1111

The first bit of M is "0". The last bit is "1". We read from left to right.
DES operates on the 64-bit blocks using key sizes of 56- bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64). However, we will nevertheless number the bits from 1 to 64, going left to right, in the following calculations. But, as you will see, the eight bits just mentioned get eliminated when we create subkeys.

Example: Let K be the hexadecimal key K = 133457799BBCDFF1. This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):
K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001
The DES algorithm uses the following steps:
Step 1: Create 4 sub-keys, each of which is 48- bits long. The 64-bit key is permuted according to the following table, PC-1. Since the first entry in the table is "57", this means that the 57th bit of the original key K becomes the first bit of the permuted key K+. The 49th bit of the original key becomesthesecond bit of the permuted key. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key. Example: From the original 64-bit key

K =11111111111111100000000000000001010101 01010101001010101010101 we get the 56-bit permutation
K+=00110011110000110011001111100001100111100001100110011
Next, split this key into left and right halves, C0 and D0, where each half has 28 bits. Example: From the permuted key K+, we get
C0 = 0011001111000011001100111100
D0 = 0011001111000011001100110011

Table 1: PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

With C0 and D0 defined, we now create sixteen blocks Cn andDn, 1<=n<=4. Each pair of blocks Cn and Dn is formed from the previous pair Cn-1 and Dn-1, respectively, for n = 1, 2, … 4, using the schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block. This means, for example, C3 and D3 are obtained from C2 and D2, respectively, by two left shifts, and C4 and D4 are obtained from C3 and D3, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1. Example: From original pair C0 and D0 we obtain:
C0 = 0011001111000011001100111100
D0 = 0011001111000011001100110011
C1 = 1110000110011001010101011111
D1 = 0110011110000110011001100110
We now form the keys Kn, for 1<=n<=4, by applying the following permutation table to each of the concatenated pairs CnDn. Each pair has 56 bits, but PC-2 only uses 48 of these.

Table 2: PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Therefore, the first bit of Kn is the 14th bit of CnDn, the second bit the 17th, and so on, ending with the 48th bit ofKn being the 32th bit of CnDn
Step 2: Encode each 64-bit block of data

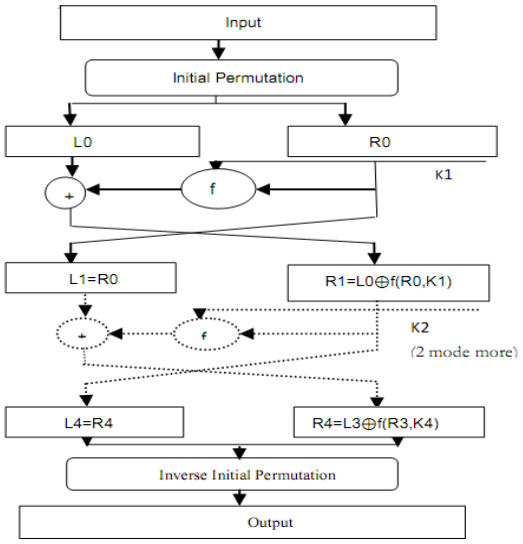
There is an initial permutation IP of the 64 bits of the message data M. This rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order.

Table 3: IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

The 58th bit of M becomes the first bit of IP. The 50th bit of M becomes the second bit of IP. The 7th bit of M is the last bit of IP. Example: Applying the initial permutation to the block of text M, given previously, we get
M = 0000 00010010 00110100 01010110 01111000 10011010101111001101111011111
IP=1100110000000000110011001111111111110000 10101010111100001010101010
Here the 58th bit of M is "1", which becomes the first bit of IP. The 50th bit of M is "1", which becomes the second bit of IP. The 7th bit of M is "0", which becomes the last bit of IP. Next divide the permuted block IP into a left half L0 of 32 bits, and a right half R0 of 32 bits.

Example: From IP, we get L0 and R0
L0 = 11001100000000001100110011111111
R0 = 11110000101010101111000010101010
We now proceed through 4 iterations, for 1<=n<=4, using a function f which operates on two blocks--a data block of 32 bits and a key Kn of 48 bits--to produce a block of 32 bits. Let + denote XOR addition, (bit-by-bit addition modulo 2). Then for n going from 1 to 4 we calculate Ln = Rn-1 Rn = Ln-1 + f(Rn-1,Kn)n This results in a final block, for n = 4, of L4R4. That is, in each iteration, we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation f . Example: For n = 1, we have
K1=000110110000001011101111111111000111000001110010
L1 = R0 = 1111 0000 1010 1010 1111 0000 1010 1010
R1 = L0 + f(R0,K1) It remains to explain how the function f works. To calculate f, we first expand each block Rn-1 from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in Rn-1 We'll call the use of this selection table the function E. Thus E(Rn-1) has a 32 bit input block, and a 48 bit output block. Thus the first three bits of E(Rn-1) are the bits in positions 32, 1 and 2 of Rn-1 while the last 2 bits of E(Rn-1) are the bits in positions 32 and 1.Example: We calculate
E(R0) from R0 as follows:
R0 = 1111 0000101010101111000010101010
E(R0)=011110100001010101010101011110100001
010101010101
(Note that each block of 4 original bits has been expanded to a block of 6 output bits.) Next in the f calculation, we XOR the output E(Rn-1) with the key Kn: Kn + E(Rn-1). Example: For K1 , E(R0), we have
K1=000110110000001011101111111111100011 1000001110010
(R0)=011110100001010101010101011110100001 010101 010101
K1+E(R0)=100101010001100001010101011101101000010111000111
To this point we have expanded Rn-1 from 32 bits to 48 bits, using the selection table, and XORed the result withn the key Kn . We now have 48 bits, or eight groups of six bits. We now do something strange with each group of six bits: we use them as addresses in tables called "S boxes". Each group of six bits will give us an address in a different S box. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the S boxes) for 32 bits total. Write the previous result, which is 48 bits, in the form:
Kn + E(Rn-1) =B1B2B3B4B5B6B7B8, where each Bi is a group of six bits. We now calculateS1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6 (B6)S7(B7)S8(B8) where Si(Bi) refers to the output of the i-th S box. To repeat, each of the functions S1, S2,..., S8, takes a 6-bit block as input and yields a 4-bit block as output. The table to determine S1 is shown and explained below: If S1 is the function defined in this table and B is a block of 6 bits, then S1(B) is determined as follows: The first and last bits of B represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be i. The middle 4 bits of B represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be j. Look up in the table the number in the i-th row and j-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output S1(B) of S1 for the input B. For example, for input block B = 011101 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1110". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0011, so that the output is 0101. Hence S1(011101) = 0011.



DES Fiestel Diagram

Example: For the first round, we obtain as the output of the eight S boxes:
K1+E(R0)=100101010001100001010101011101101000010111000111
S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)=010111001000 00101011 010110010111
The final stage in the calculation of f is to do a permutation P of the S-box output to obtain the final value of f: f = P(S1(B1)S2(B2)...S8(B8)) P yields a 32-bit output from a 32-bit input by permuting the bits of the input block. Example: From the output of the eight Sboxes:
S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)
S8(B8) = 010111001000 00101011 01011001 0111
we get f = 0010 00110100101010100110111011
R1=L0+f(R0,K1)= 11001100000000000110011001
111 1111+ 00100011010010101010100110111011
= 11101111010010100110010101000100
In the next round, we will have L2 = R1, which is the block we just calculated, and then we must calculate R2 =L1 + f(R1, K2), and so on for 4 rounds. At the end of the sixteenth round we have the blocks L4 and R4. We then reverse the order of the two blocks into the 64-bit block R16L16 and apply a final permutation IP-1 as defined by the following table:

Table 4: IP^ (-1)

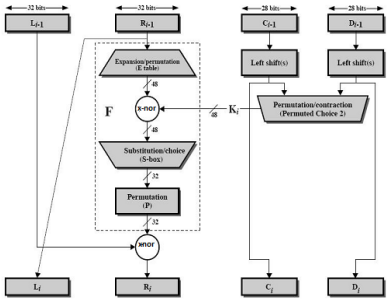
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

That is, the output of the algorithm has bit 40 of the pre output block as

How "FOUR-ROUND DES" Works

This section briefly gives an overview of Four-Rounded DES Algorithm. Four-Rounded DES is composed of substitutions and permutations which take place in four rounds. It is a symmetric cryptosystem which means that both the parties use the same key. Hence, the key must be kept secret.

In DES, 64-bit data is divided into left and right halves. In each round, a main function F is applied on right half of the data and a sub-key (Ki) of 48 bits. During this process eight S-boxes are used which convert each 6-bit block into 4-bit block generating 32-bit data. Finally, the left half of data is X-NORed with 32-bit output of the main function.



Single Round of DES Algorithm

DES Modes of Operation

The DES algorithm turns a 64-bit message block M into a 64-bit cipher block C. If each 64-bit block is encrypted individually, then the mode of encryption is called Electronic Code Book (ECB) mode. There are two other modes of DES encryption, namely Chain Block Coding (CBC) and Cipher Feedback (CFB), which make each cipher block dependent on all the previous messages blocks through an initial X-NOR operation. Single Round of DES Algorithm is as:

Cracking DES

Before DES was adopted as a national standard, during the period NBS was soliciting comments on the proposed algorithm, the creators of public key cryptography, Martin Hellman and Whitfield Diffie, registered some objections to the use of DES as an encryption algorithm. Hellman wrote: "Whit Diffie and I have become concerned that the proposed data encryption standard, while probably secure against commercial assault, may be extremely vulnerable to attack by an intelligence organization" (letter to NBS, October 22, 1975).

Diffie and Hellman then outlined a "brute force" attack on DES. (By "brute force" is meant that you try as many of the 2^56 possible keys as you have to before decrypting the ciphertext into a sensible plaintext message.) They proposed a special purpose "parallel computer using one million chips to try one million keys each" per second, and estimated the cost of such a machine at $20 million.

Fast forward to 1998. Under the direction of John Gilmore of the EFF, a team spent $220,000 and built a machine that can go through the entire 56-bit DES key space in an average of 4.5 days. On July 17, 1998, they announced they had cracked a 56-bit key in 56 hours. The computer, called Deep Crack, uses 27 boards each containing 64 chips, and is capable of testing 90 billion keys a second.

Despite this, as recently as June 8, 1998, Robert Litt, principal associate deputy attorney general at the Department of Justice, denied it was possible for the FBI to crack DES: "Let me put the technical problem in context: It took 14,000 Pentium computers working for four months to decrypt a single message . . . . We are not just talking FBI and NSA [needing massive computing power], we are talking about every police department."

Responded cryptograpy expert Bruce Schneier: " . . . the FBI is either incompetent or lying, or both." Schneier went on to say: "The only solution here is to pick an algorithm with a longer key; there isn't enough silicon in the galaxy or enough time before the sun burns out to brute- force triple-DES" (Crypto-Gram, Counterpane Systems, August 15, 1998).

Conclusion and Future Works

In this paper, for the cryptanalysis of Data Encryption Standard is presented. It shows that it is an effective approach for cryptanalysis of four-rounded DES using X-Nor. The cost function used in this paper is generic and can be used for the cryptanalysis of other block ciphers In the future, it also performed under different operands and even by altering them.

References

[1]. "Cryptographic Algorithms for Protection of Computer Data During Transmission and Dormant Storage," Federal Register 38, No. 93 (May 15, 1973).

[2]. Data Encryption Standard, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C. (January 1977).

[3]. Carl H. Meyer and Stephen M. Matyas, Cryptography: A New Dimension in Computer Data Security, John Wiley & Sons, New York, 1982.

[4]. Dorthy Elizabeth Robling Denning, Cryptography and Data Security, Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.

[5]. D.W. Davies and W.L. Price, Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronics Funds Transfer, Second Edition, John Wiley & Sons, New York, 1984, 1989.

[6]. Miles E. Smid and Dennis K. Branstad, "The Data Encryption Standard: Past and Future," in Gustavus J. Simmons, ed., Contemporary Cryptography: The Science of Information Integrity, IEEE Press, 1992.

[7]. Douglas R. Stinson, Cryptography: Theory and Practice, CRC Press, Boca Raton, 1995.

[8]. Bruce Schneier, Applied Cryptography, Second Edition, John Wiley & Sons, New York, 1996.

[9]. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, 1997.

[10]. Ruth M. Davis, "The Data Encryption Standard" Proceedings of Conference on Computer Security and the Data Encryption Standard, National Bureau of Standards, Gaithersburg, MD, Feb. 15, 1977, NBS Special Publication 500-27, pp. 5-9.

[11]. WhitfieldDiffie, "Cryptographic Technology: Fifteen Year Forecast" Reprinted by permission AAAS, 1982 from Secure Communications and Asymmetric Crypto Systems. AAAS Selecte8 Symposia. Editor: C.J. Simmons. Vol. 69, Westview Press, Boulder, Colorado, pp. 38-57.

[12]. C. Boyd. "Modern Data Encryption," Electronics & Communication Engineering Journal, October 1993, pp. 271-278.

[13]. Seung-Jo Han, "The Improved Data Encryption Standard (DES) Algorithm" 1996, pp. 1310-1314.

[14]. A.Kh. AI Jabri, "Secure progressive transmission of compressed images" IEEE Transactions on Consumer Electronics, Vol. 42, No. 3, AUGUST 1996, pp. 504-512 .

[15]. K. Wong, "A single-chip FPGA implementation of the data encryption standard (des) algorithm" IEEE 1998 pp. 827-832 .

[16]. Subbarao V. Wunnava, "Data Encryption Performance and Evaluation Schemes" Proceedings IEEE Southeastcon 2002, pp. 234-238

[17]. Xun Yi," Identity-Based Fault-Tolerant Conference Key Agreement" IEEE transactions on dependable and secure computing, vol. 1, no. 3, July-September 2004, pp. 170-178 .

[18]. M. Backes, "Relating Symbolic and Cryptographic Secrecy" IEEE transactions on dependable and secure computing, vol. 2, no. 2, April-June 2005, pp. 109-123 .

[19]. ElisaBertino, "An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting" IEEE transactions on dependable and secure computing, vol. 5, no. 2, April-June 2008, pp. 65-70.

[20]. Clark, A.,"Modern Optimization Algorithms for Cryptanalysis".Proceedings of Second IEEE Australian and New Zealand Conference on Intelligent Information Systems, pp.258-262, 1994.

[21]. Laskari, E. C., Meletiou, G. C., Stamation, Y. C., and Vrahatis, M. N.,"Evolutionary Computation based Crypt-analysis: A first study". Nonlinear Analysis, vol. 63, no.(5- 7), pp. 823-830, 2005.

[22]. R, Vimalathithan, and Valarmathi, M. L.,"Cryptanalysis of S-DES using Genetic Algorithm". International Journal of Recent Trends in Engineering, vol. 2, no. 4, pp.76-79, Nov.2009.

[23]. Shahzad, W., Siddiqui, A. B., and Khan, F. A., "Cryptanalysis of Four-Round DES using Binary Particle Swarm Optimization". Genetic and Evolutionary Computation Conference, pp. 1757-1758, July 8-12, 2009.

[24]. Song, J., Zhang, H., Meng, Q., and Wang, Z.,"Cryptanalysis of Four-Round DES Based on Genetic Algorithm". International Conference on Wireless Communications Networking and Mobile Computing, Issue 21-25, pp. 2326-2329, Sept. 2007.

[25]. Spillman, R., Janssen, M., Nelson, B., and Kepner, M.,"Use of A Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers". Cryptologia, vol.17, no.1, pp.