An Application to ensure Security through Bit-level Encryption

Mrinmoy Ghosh
Dr. B. C. Roy Engineering College
Durgapur - 713206
West Bengal, INDIA

Prof. Pranam Paul
Dr. B. C. Roy Engineering College
Durgapur - 713206
West Bengal, INDIA

Abstract
This paper presents a process by which we can secure any kind of file. The newly developed encryption algorithm is presented in this paper. With help of generated distinct blocks N-level, again some distinct blocks are regenerated for (N + 1)-level. Thus, source stream or plain text, target stream or encrypted text will be generated and decrypted text will be gotten on the applying the reverse process. The algorithm can be implemented on any kind file as it is implemented in bit-level. The strength of the technique is analyzed in this paper.
***Key Words*:**
cryptography, encryption, decryption, private key

1. Introduction

Cryptography is an essential tool of data security through mathematical manipulation of data with an incomprehensible format for unauthorized person.
Here a newly developed technique named, "Replacement of Cyclic Regeneration of Distinct Block (RCRDB)" is discussed. At first, plain text is decomposed into some blocks, having equal length. With help of all distinct blocks, some distinct blocks have been regenerated. The process will be continued up to a finite level of regeneration. Finally by rearrangement and replacement of all blocks in source stream or plain text, target stream or encrypted text will be generated. Hence getting source stream and the privet key, decrypted text will be gotten on the applying the reverse process.
Section 2 presents the scheme followed in the encryption technique. Section 3 describes an implementation of the technique. Section 4 presents results of executing the technique on some real files. Section 5 is an analytical discussion on the technique with a conclusion.

2. The Scheme

At first, plain text is decomposed into some blocks, having equal length. Then we take each distinct block and all the distinct blocks according to their sequence of appearance are kept in private key.
Source file is converted into binary form i.e. source bit-stream which is decomposed into N number of blocks of equal length; say L bits where L is an integer. It may be happened that after decomposition of total source-bit-stream into some L-bit blocks, a blocks, less than L bits is left at last, say U_B (means length of $U_B < L$) which is kept unchanged during encryption. So here N should be less or equal to 2^L ($N \leq 2^L$). For the encryption we need to generate replaced code, named Identification Marks for each distinct block.
Now section 2.1 and section 2.2 respectively describes generation of Identification Marks and the algorithm for encryption. The way of key generation is discussed in section 2.3, whereas section 2.4 defines the process of decryption to get back the plain text.

2.1 Generation of Identification Marks

Step 1: Set individual identification marks for each distinct L-bit decomposed block. So, maximum 2^L number of identification marks is needed.
Step 2: Collecting all possible permutation and combination of every two identification marks, again some identification marks for some blocks are regenerated.
Step 3: The regeneration of process identification marks, defined in step 2 for some blocks will be continued up to a certain finite level, say D defined by the sender that is kept in to the key.

2.2 Process of Encryption

Step 1: Release each decomposed blocks with recently generated corresponding identification marks.
Step 2: Considering the two consecutive identification marks, it is replaced with identification marks which is generated on and onwards 2^{nd} level regeneration process, defined in step 2 of section 2.1.
Step 3: The replacement process will be continued up to D level.

Manuscript received November 5, 2009
Manuscript revised November 20, 2009

Now U_B is appended at beginning with the output of step 3. Hence the encrypted text will be generated.

2.3 Key generation

Length of U_B, number of appeared distinct block (N), decomposed block length (L) and maximum level of generation of identification process (D) are kept into the key along with all distinct blocks with their sequence of appearance. Structure of key is shown in table 2.2.1.

Table 2.2.1 Structure of Key

Segment	Description
1	Length of unchanged block, U_B
2	Number of Appeared Distinct Block, N
3	Length of Decomposed Block, L
4	Maximum Level of Generation of Identification Process, D
5	All Distinct Decomposed in its proper sequence of Appearance

2.4 Decryption

Step 1: From the 5th segment of the private key, collecting all distinct blocks, identification marks for each block is assigned. This identification mark is same as first level of identification mark, defined in section 2.1. From the beginning of the encrypted text, unchanged block (U_B) is collected, length of which is defined in to the key.

Step 2: Then every identification marks is replaced into identification marks. In that process we find two different identification marks against each distinct block.

Step 3: Now we do Step 2 up to D level in reverse manner.

Step 4: After doing Step 3, identification marks at will be backed.

Step 5: Replace the all identification marks into its binary form with the help of key.

Step 6: Now we collected the entire bit-stream-blocks are merge together. After this merging, U_B is attached at last of the recently generated decrypted bit of stream.

3. Implementation

We consider the plaintext P as "**Encrypt**". The stream of bits, S, representing P is as follows: 01000101011011100110001101110010011110010111000001110100.

Now S is decomposed into some block with 2 bits length (L = 2). These are 01, 00, 01, 01, 01, 10, 11, 10, 01, 10, 00, 11, 01, 11, 00, 10, 01, 11, 10, 01, 01, 11, 00, 00, 01, 11 and 01, 00. So here is no unchanged block, (U_B = NULL). Number of distinct block is 4 (N = 4). Here we use maximum 2nd level to generate Identification Code. So D = 2.

Section 3.1 shows the way to generate identification marks for each block, while section 3.2 discussed about the structure of key. Section 3.3 and 3.4 respectively explain encryption and decryption process.

3.1 Generation of Identification Marks

Here S has 4 distinct blocks, according to the order they are 01, 00, 10, 11. So we put according to key generation technique 01=a, 00=b, 10=c, 11=d that is 1st level identification marks. For the generation of 2nd level identification marks, again the two bit representation of a, b, c & d is aa, ab, ac, ad, bb, bc, bd, cc, cd, dd, ba, ca, da, cb, db, dc. Now we put aa=e, ab=f, ac=g, ad=h, bb=i, bc=j, bd=k, cc=l, cd=m, dd=n, ba=o, ca=p, da=q, cb=r, db=s, dc=t.

3.2 Generation of Identification Marks

All information about encryption is kept into the key, shown in figure 3.2.1.

Table 3.2.1 Structure of Key

Segment	Description	Form of Keeping in the Key
1	Length of unchanged block, U_B	0
2	Number of Appeared Distinct Block, N	1000
3	Length of Decomposed Block, L	10
4	Maximum Level of Generation of Identification Process, D	10
5	All Distinct Decomposed in its proper sequence of Appearance	01 00 10 11

3.3 Encryption

In key generation we put 01=a, 00=b, 10=c, 11=d according to the algorithm. Now we encrypt S. Here S is decomposed into 01, 00, 01, 01, 01, 10, 11, 10, 01, 10, 00, 11, 01, 11, 00, 10, 01, 11, 10, 01, 01, 11, 00, 00, 01, 11, 01 and 00. After putting those values encrypted S is "abaaacdcacbdadbcadcaadbbad ab". Again the two bit representation of a, b, c & d are aa, ab, ac, ad, bb, bc, bd, cc, cd, dd, ba, ca, da, cb, db, dc. Now we put aa=e, ab=f, ac=g, ad=h, bb=i, bc=j, bd=k, cc=l, cd=m, dd=n, ba=o,

ca=p, da=q, cb=r, db=s, dc=t. Hence encrypted text is "fegtgkhjhphihf"

3.4 Decryption

From key, all information about encryption can be collected. From different segments of the key, it is come to know that there are 4 distinct blocks, found from the plain text; length of decomposed block is 2-bit and maximum 2nd level identification mark is generated. The 1st segment of the key says that length of unchanged block is 0 (zero). After encryption S, "fegtgkhjhphihf", no block is discarded from beginning as unchanged block. We know the two bit representation of a, b, c & d is aa, ab, ac, ad, bb, bc, bd, cc, cd, dd, ba, ca, da, cb, db, dc. Now we put aa=e, ab=f, ac=g, ad=h, bb=i, bc=j, bd=k, cc=l, cd=m, dd=n, ba=o, ca=p, da=q, cb=r, db=s, dc=t. . After putting those values decrypted S is "abaaacdcacbdadbcadcaadbbadab"In key generation we put 01=a, 00=b, 10=c, 11=d according to the algorithm. So here we put a=01, b=00, c=10, d=11. After putting those values decrypted S is 0100010101101110011000110111001001111001011100 0001110100 is same as source bits stream, so obviously decrypted text "**Encrypt**" is same as plane text.

4. Results

Here we have compared RCRDB with International Data Encryption Algorithm (IDEA) to establish a comparative analytical report that is helpful to understand strength and weakness of RCRDB. A brief idea on IDEA is discussed in section on 4.1 whereas section 4.2 describes comparative reports with IDEA on the basic of different parameters.

4.1 International Data Encryption Algorithm (IDEA)

International Data Encryption Algorithm is a 128-bit private key cipher, which is implemented in bit level with equally decomposed 64 bit blocks of plain text depending on modular arithmetic. There are regenerated 16-bit 52 sub keys from 128 bits key based on rotational replacement and some particular rules, say k_1, k_2, k_3 … k_{52}. A 64-bit block of plain text is decomposed into 4 16-bit sub blocks; say P_1, P_2, P_3 and P_4. In IDEA, there are 8 rounds, which have being followed same procedure. Each round takes output of previous blocks and 6 sub keys, according to the number of round, except 1st round, takes 4 sub blocks. Entire process of encryption through IDEA is shown in figure 4.1.1. After completing the total process we get a 64-bit block, which is an encrypted block of that 64 bit block of plain text. Same things will be done on each and every generated blocks of plain text and finally we get encrypted text, which is same in size of the plain text.

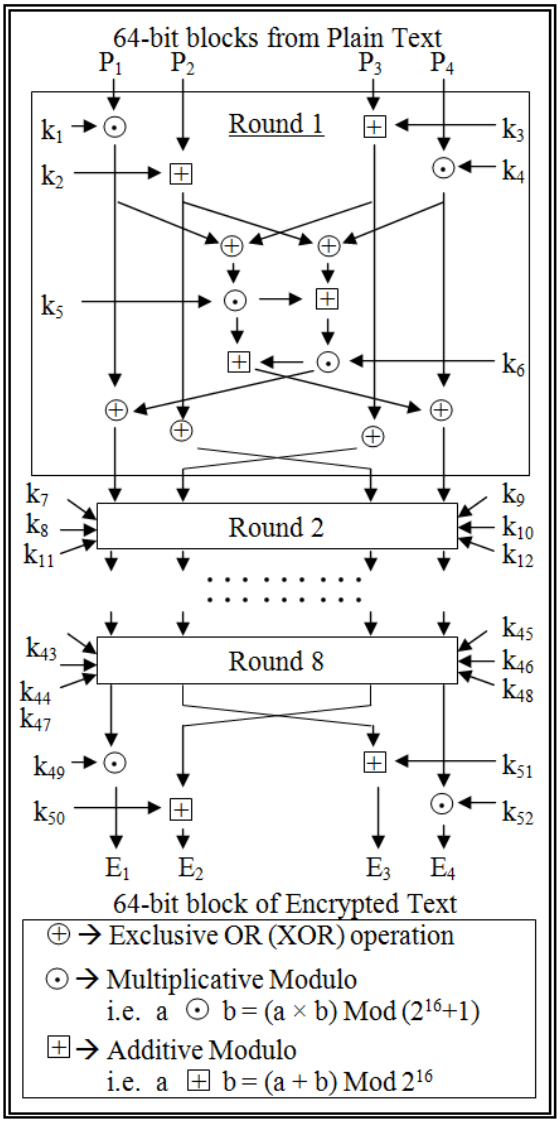

Figure 4.1.1
Process of Encryption through IDEA

4.2 Comparative Report

Both the proposed RCRDB protocol (with 128-bit block size in this paper) and exiting IDEA have been implemented on a number of data files varying types of content and sizes of a wide range, shown in table 4.2.1.

Table 4.2.1 Set of Files with Size

Sl. No.	Source File Name	Original File Size
1	a.txt	14337
2	b.txt	73710
3	c.txt	131776
4	Des_56.cpp	14983
5	M.txt	48430
6	Calc.exe	114688
7	hh.exe	10752
8	Win.com	18432
9	Ansi.sys	9029
10	Watch.sys	14592
11	Blue.bmp	1272
12	ZAPO.BMP	9522

Accordingly the observations on the following points have been made:

1. **Encryption and Decryption Time* :** *To evaluate computational overhead*

In Table 4.2.3, a comparison on basis of encryption time with their file size between RCRDB and IDEA has been shown here on the same set of files, used in Table 4.2.1.

Table 4.2.2 Encryption Time for RCRDB and IDEA

Sl. No.	File Size	Encryption time for RCRDB	Encryption time for IDEA
1	14337	0.1098901099	2.4725274725
2	73710	0.8241758242	12.6923076923
3	131776	1.3736263736	22.5274725275
4	14983	0.1648351648	2.5824175824
5	48430	0.5494505495	8.3516483516
6	114688	1.2087912088	17.5274725275
7	10752	0.1098901099	1.6483516484
8	18432	0.2197802198	2.8021978022
9	9029	0.1098901099	1.3736263736
10	14592	0.1648351648	2.2527472527
11	1272	0.0549450549	0.2197802198
12	9522	0.1098901099	1.4835164835

A graphical representation for the table 4.2.2 is shown in figure 4.2.1 with continues line and dotted line for encryption time of RCRDB and IDEA, respectively. According to the graph, there is a tendency that encryption time for RCRDB and IDEA increases with file size. But required time for the encryption through RCRDB is much smaller than encryption time for IDEA.

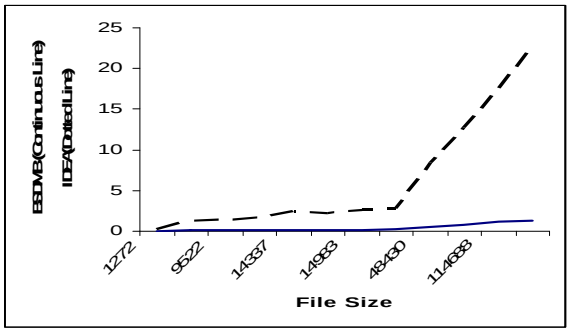

Figure 4.2.1 Relationship between Encryption Time of RCRDB and IDEA

**The observations were made using personal computer with specifications of 128 MB SD RAM, 1.5 GHz. processor and with Windows XP as the platform.*

2. **Pearsonian Chi-Square Value: T***o check the non-homogeneity of the source file and the corresponding encrypted file and also being termed as "Goodness-of-fit chi-square test", with the formula* $\lambda^2 = \Sigma\{(f_o - f_e)^2 / f_e\}$*, where* f_e *and* f_o *respectively being frequency of a character in source file and that of the same in the corresponding encrypted file*

Table 4.2.3 Chi-Square Value for RCRDB and IDEA

Sl. No.	Chi-Square Value for RCRDB	Chi-Square Value for IDEA
1	2703.404583	11954.841494
2	65300.567094	61455.579243
3	118951.809896	111446.333333
4	35151.146255	33284.270177
5	43693.152096	40947.847106
6	52617929.919497	21150819.758794
7	329746.244148	229908.574337
8	750661.149455	477534.450116
9	924586.280769	14823.865385
10	246543.852664	204617.010065
11	395.302800	1334.211477
12	219756.063093	15698.555711

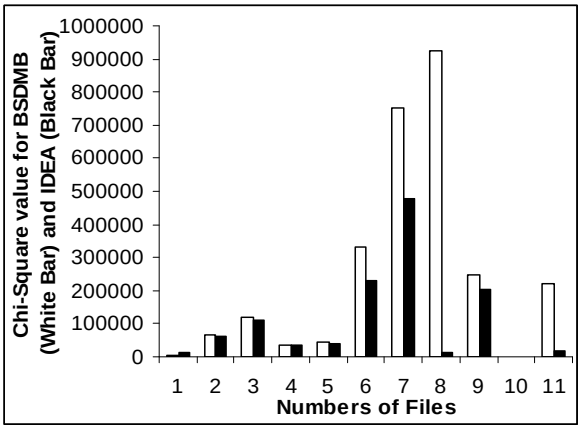

Figure 4.2.2 Comparison between Chi-Square value RCRDB (White Bar) and IDEA (Black Bar)

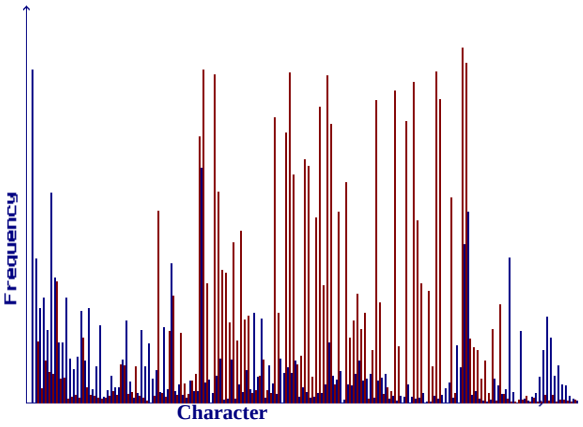

Figure 4.2.3 Frequency Distribution of Characters in "ansi.sys" and Corresponding Encrypted File

Chi-square value of RCRDB and IDEA for the set of same files which are used in table 4.2.1 with maintaining the Sl. No. for the respective files, have been compared in table 4.2.3. Among them figure 4.2.2 shows comparison of eleven files, except Calc.exe. As chi-square values of RCRDB and IDEA for Calc.exe are respectively 52617929.919497, 21150819.758794 which are very high with respect to other files, so for clear display of figure 4.2.2 comparisons on calc.exe file is not in the figure. White bar shows chi-square value of RCRDB of those eleven files, while black bar are been used for chi-square value of IDEA. That indicates chi-square value for the technique is generally higher than IDEA.

Apart form these there comparative observations; observation also on the following was made:

Graphical Test for Frequency Distribution: *To test the degree of security of the proposed protocol against the cryptanalytic attack, where the frequency of the all 256 characters in source file and their corresponding encrypted file are compared graphically, the observation being to show whether the exists any fixed relation ship between of the a character in both source and encrypted file*

To established this relationship between plain text and cipher text, figure 4.2.3 shows the distribution of the frequency (along Y axis) of the set of characters with their ASCII value (along X axis) of arbitrarily chosen a file from table 4.2.1, **ansi.sys** and its encrypted file. Blue pillar represents the frequency of characters appeared in source file or plain text whereas red pillar is measured the frequency of characters in encrypted file or cipher text. From the figure 4.2.3 it is clearly shown that frequency of characters in plain text and cipher text are well distributed.

5. Analysis and Conclusion

Using this RCRDB private key technique, we can encrypt any size of file, as well as any kind of file, as RCRDB protocol is implemented in bit level. As level of generation of identification marks for each block and length of decomposed block are chosen at run time as randomly, for it key is differed from each encryption to another. Not only that we are taken decomposed blocks in its sequence appearing for generating the identification marks for each block. So it is also varied every time along with its sequence of appearance. Due to the reason, key is differed for every time and almost unbreakable.

Additionally, if we use X-bit key, 2^X number of key may be generated, from which only one option is used for correct decryption. So, complexity of key breaking increases according to increased value of X. Refer to the references' number 11, a processor, capable of doing 10^6 encryptions per millisecond, requires 5.9×10^{30} years to break a 168-bit key. Encryption time is not also increasing with increased key size. So, generation of 168-bit or more than 168-bit key is recommended for correct implementation of the technique. Due to that large block size is recommended for increasing complexity and getting more efficient effect.

References

[1] J. K. Mandal, S. Dutta, "A 256-bit recursive pair parity encoder for encryption", Advances D -2004, Vol. 9 nº1, Association for the Advancement of Modelling and Simulation Techniques in Enterprises (AMSE, France), www. AMSE-Modeling.org, pp. 1-14

[2] Pranam Paul, Saurabh Dutta, "A Private-Key Storage-Efficient Ciphering Protocol for Information Communication Technology", National Seminar on Research Issues in Technical Education (RITE), March 08-09, 2006, National Institute of Technical Teachers' Training and Research, Kolkata, India

[3] Pranam Paul, Saurabh Dutta, "An Enhancement of Information Security Using Substitution of Bits Through Prime Detection in Blocks", Proceedings of National Conference on Recent Trends in Information Systems (ReTIS-06), July 14-15, 2006, Organized by IEEE Gold Affinity Group, IEEE Calcutta Section, Computer Science & Engineering Department, CMATER & SRUVM Project-Jadavpur University and Computer Jagat

[4] Dutta S. and Mandal J. K., "A Space-Efficient Universal Encoder for Secured Transmission", International Conference on Modelling and Simulation (MS' 2000 – Egypt, Cairo, April 11-14, 2000

[5] Mandal J. K., Mal S., Dutta S., A 256 Bit Recursive Pair Parity Encoder for Encryption, accepted for publication in AMSE Journal, France, 2003

[6] Dutta S., Mal S., "A Multiplexing Triangular Encryption Technique – A move towards enhancing security in E-Commerce", Proceedings of IT Conference (organized by Computer Association of Nepal), 26 and 27 January, 2002, BICC, Kathmandu

[7] William Stallings, Cryptography and Network security: Principles and practice (Second Edition), Pearson Education Asia, Sixth Indian Reprint 2002

[8] Atul Kahate (Manager, i-flex solution limited, Pune, India), Cryptography and Network security, Tata McGraw-Hill Publishing Company Limited

[9] Mark Nelson, Jean-Loup Gailly, The Data Compression Book. BPB Publication

[10] S Mal, J K Mandal and S Dutta, "A Microprocessor Based Encoder for Secured Transmission", Conference on Intelligent Computing on VLSI, Kalyani Govt. Engineering College, 1-17 Feb, 2001, pp 164-169

[11] Saurabh Dutta, "An Approch Towords Development of Efficient Encryption Technique", A thesis submitted to the university of North Bengal for the Degree of Ph.D., 2004

MRINMOY GHOSH has passed 10 level, Madhyamik Exam from Lal Gopal Higher Secondary School under West Bengal Board of Secondary Education and 10+2 level, Higher Secondary from Ranaghat Yusuf Institution under West Bengal Council of Higher Secondary Education in 2001 and 2003 respectively. He has completed my graduation with science from Ranaghat College under Kalyani University in 2007. Now he is continuing post-graduation in Master of Computer Application from Dr. B.C.Roy Engineering College, Durgapur, approved by AICTE and affiliated to West Bengal University of Technology. His area of interest to research is Cryptography and Network Security along with Mathematics. It is his 1st International Journal.

Prof. Pranam Paul is associated with department of MCA of Dr. B. C. Roy Engineering College, Durgapur, West Bengal, INDIA in the. He has already submitted his Ph.D. thesis in ECE department NIT, Durgapur in the field of Cryptography and Network Security. He has total 20 research publications.