Secure and Cost Effective Cryptosystem Design Based on Random Multiple Key Streams

Penchalaiah Padugupati, Ramesh Reddy K
Vikrama Simhapuri University
Department of Computer Science
India
penchal.caliber@gmail.com, drkrreddy05@gmail.com

ABSTRACT: *There are many cryptographic systems that use complex operations involving substitutions and permutations to produce resistant ciphertext, even if the level of the security of these cryptosystems are good, there should be tradeoff between security level and operational cost and the ever increasing virtual infrastructure and mobile, cloud computing technologies creating much more complexities and demanding cost effective and secure cryptographic algorithms.*

In order to make a cipher more difficult to break, one could use various keys in cryptosystem. This would help to cover any perceptible patterns in the ciphertext. In fact, one can produce unbreakable ciphertext by supplying randomly generated key on each bit of data that is mathematically infeasible to break. Since different random bits or keys would not lead to any repeating patterns.

For the first time, in this paper, we present a construction method to generate multiple random keys from a core-key with highest possible immunity to crack. We are with a particular emphasis on novel technique to secure user data, we have designed a secure and cost effective new cryptosystem called Rbits (Random bits) cipher. In different directions we identify that Rbits having highest immunity to crack and presenting various analysis tests in support from this viewpoint.

Keywords: Cryptography, Rbits, Random bits, Cryptanalysis, Encryption, Decryption

Received: 30 August 2015, Received 2 October 2015, Accepted 8 October 2015

1. Introduction

The basic principle that all cryptosystem designers must kept in mind is that a cryptosystem should be secure even if the whole thing about the system is public knowledge except the encryption key. A cryptographic system is said to be secure if the ciphertext does not contain adequate details to find out the matching plaintext. There are many cryptographic systems that use complex operations involving substitutions and permutations to produce resistant ciphertext, even if the level of the security of these cryptosystems are good, there should be tradeoff between security level and operational cost and the ever increasing virtual infrastructure and mobile, cloud computing technologies creating much more complexities and demanding cost effective and secure cryptographic algorithms. The security level is measured in terms of various statistical tests and cost is measured based on the type of the operations used that transforms the plaintext to ciphertext.

The cryptographic system algorithm will not be kept secret. The general assumption is that it is impractical to decipher a

message by simply knowing the ciphertext plus knowledge of the algorithm. Only the key is kept secret. This quality of encryption is what makes algorithms feasible for widespread use.' The thing that is available to the unauthorized party is the ciphertext only'. Many techniques have been developed in recent years. The objective is to use the Rbits for secure communication and protecting personal data with a tradeoff between security and cost. The basic rule of Rbits cipher is that the ciphertext should not contain adequate details to find out the corresponding plaintext and time required to break the secret code should take more than the useful lifetime of the content.

2. Methodology

To develop secured and cost effective cryptosystem the services of OTP [Sharad Patil, Ajay Kumar, (2010)] and BBS [Sidorenko .A and Schoenmakers .B,(2005)] are used.

2.1 One Time Padding

This algorithm is called as unconditionally secure algorithm [Shannon, Claude,(1949)]. In this cryptosystem the key with no repetition patterns that is equal in length to the message to be encrypted is used. Each character of the message is bitwise XOR with the key, to produce ciphertext [Sharad Patil, Ajay Kumar, (2010)].

$$C=P \oplus K$$

Where P = Plaintext, K = Key, and C = Cipher text

Even though OTP well known for its perfect secrecy, it's key size limits its practicality by causing tremendous overheads. Rbits uses OTP behavior in different methodology and free up from overheads.

2.2 About Rbits

Rbits [Penchalaiah and Ramesh Reddy (2013)] is a security mechanism which is symmetric key block steam cipher as shown on Fig 1. The advantage of the stream cipher is that it is faster in encryption and decryption process. In brief Rbits generates random bits at the both ends and these random bits are used to encrypt and decrypt. Random bits are generated by Blum Blum Shub (BBS) algorithm; both the sender and receiver uses BBS algorithm and common shared key say core-key (CK), parameter 'P' which is input to BBS for random bit generation. The key principal of this algorithm is that generating randomly changing multiple keys (K_i)[Mihir Bellare et al (2011)], for encryption using a single key CK, P which are common to both sender and receiver[Wenjun Gu et al (2011)].

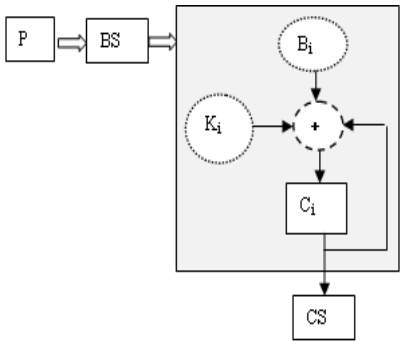

Figure 1. Encryption Process

Where P - Plaintext
BS - Binary Stream
B_i - i^{th} Plain Block Stream
\+ - XOR operation
K_i - i^{h} Key
C_i - Cipher Block Stream
C - Cipher Stream

2.3 BBS

The security of the BBS generator depends on the difficulty of factoring 'n'. Blum, Blum and Shub proved that the 'x^2 mod n' generator is unpredictable in polynomial time (for proofs [Sidorenko .A and Schoenmakers .B, (2005)]). The BBS generator is unpredictable to left and unpredictable to right. This means that, given a sequence generated by the BBS generator, the cryptanalysts cannot predict the next bit or the previous bit of the sequence. An attractive property of this generator is that we can directly compute any of the x values.

$$x_i = (x_0^{(2^i (mod((P-1)*(Q-1))))})(modN)$$

In the applications where multiple keys are required in this manner, it is not compulsory to store them all. All key can be effectively computed and recovered from previous values and the initial x and N [Junod .P, (1999)], [Blum L *et al*, (1986)].

2.4 Deriving Core-Key (CK) and Parameter (P)

The core key CK is derived from the initial steps of BBS as follows [Blum L *et al* ,(1986)].

1. Select two large prime numbers, *x* and *y*.
$x \equiv y \equiv r$ (mod m). Where $r = 3$ and $m = 4$

2. Computer $n = x * y$.

3. Choose a random number 's', relatively prime to '*n*'

4. Compute $Z_0 = s^2 \ mod \ n$.

$CK = Z_0$

Here 'Z_0' is the common core key '*CK*' and 'n' is parameter '*P*'.

2.5. Generating Multiple Keys

Multiple keys are generated by using BBS algorithm which is as follows.

```
loop  i = 1 to msg_len
  ck_i = (ck_(i-1))^2 mod n
  b_i =  ck_i  mod 2
  kj = kj || bi
 if (i % key_size) = 0 then
  j = j+1
```

msg_len → message length in binary mode.
key_size → key length

Enough multiple keys K_i of size key_size are generated to encrypt a message with msg_len in stream or binary mode.

2.6 Operation

To make the algorithm simple and faster, the proposed operation is XOR. XORing has computational complexity of "*order b*" which is written O(b) where *b* is the no of bits. XOR operation is very simple leads to cost effective.

2.7 Adding CBC

Even if this mechanism over comes the problem of producing the same ciphertext for a plaintext that appear more than once in the input (the corresponding ciphertext block will also appear more than once in the output). To make this mechanism hardest to cryptanalysis we still adding Cipher Block Chaining (CBC).

In CBC, the output of the encryption of the previous block streams is feedback into the encryption of the present block

stream. That is, each resultant is used to transform the encryption of the next block stream. Therefore every block of cipher text is reliant on the subsequent current input plaintext block, as well as all the previous plaintext blocks.

As we stated earlier in introduction "The thing that is available to the unauthorized entities is the ciphertext only", many techniques of cryptanalysis use statistical properties of the available ciphertext. So with the added CBC operation the statistical characteristics of the plaintext are masked to such an extent that any type of cryptanalysis is infeasible [Mihir Bellare *et al* (2011)].

2.8 Algorithm

Prior to encryption and decryption process both the sender and receiver must satisfy the pre-requisites.

2.8.1. Pre-Requisites

a) Sender and receiver must have pre-established a security binding (SB). SB defines core-key exchanging, parameter agreement (parameter here means block size).

b) Both sender and receiver must use same stream block size.

c) Core-key, Parameter, block size must be confidential and should be infeasible to predict.

2.8.2. Encryption

Sender follows the following sequence of steps.

a) Sender converts message into stream of bits.

b) The binary stream is divided into specific size of block of bits called block stream (for easy of operations).

c) Fetch a block stream and XOR with key K_i , instantly generated random block of bits at sender end.

d) Apply CBC encryption operation.

e) Steps c, d is continued until last block stream of message and produces cipher stream.

2.8.3. Decryption

Receiver follows the following sequence of steps.

a) The cipher stream is divided into specific size of block of bits as sender.

b) Fetch a block stream and XOR with key K_i , instantly generated random block of bits at receiver end.

c) Apply CBC decryption operation.

d) Steps b, c is continued until last block stream of message.

e) Convert stream of bits in to plaintext.

3. Facts about Rbits

3.1. Unique Features of Rbits

1.Rbits is suitable for both short and long message communication.

2.Rbits is faster in both Encryption and Decryption Process.

3. Rbits is Simple and required no heavy computations to encrypt/decrypt data

4. Rbits is complex to cryptanalysis.

5.Unpredictable randomness of Key for each Block.

6.Added CBC mode of operation satisfies avalanche effect.

7.Rbits cipher is a symmetric cipher, both sender and receiver follows a common key.

8.Rbits is a block stream cipher.

9.Rbits takes variable length of block size as input; there is no fixed block size.

10.There are no rounds in Rbits.

11.For each block of stream new key is used. All keys are distinct.

12.Total Key length is as long as the length of the plaintext.

3.2. Where Can We Use

1. Secure communication services.

2. Client/server communication service.

3. Web applications.

4. Secure data store in cloud.

5. Secure backup and restore.

6.As an alternative to SSL

4. Testing and Simulation

We are testing the cipher in terms of security and cost of the system. The security level is measured in terms of various statistical tests and cost is measured based on the type of the operations. We simulated the security mechanism using java1.6, IDE Netbeans in a networked environment. The java.math package [Sun Microsystems] provides classes for performing very long integer arithmetic (BigInteger) is used. We adopted this mechanism as a communication service in a chat application. We simulated the Man-in-the-Middle-Attack by any third entity, between two entities and performed security test on ciphertext which is available at third entity. The experimental data are analyzed and reported using MS-Excel.

4.1 Security Level Analysis

4.1.1 For Plaintext without Repetitive Patterns

The following plaintext in Figure 2.a, 2.b, core-key and parameter as in Figure 2.c is used in simulation. Here we captured some instance of the simulation shown in Figure 3.a, 3.b, and 3.c.

We are Planning a 10cr Bio Chemical project @ AP

Figure 2. a. Data as Plaintext

10101110110010100100000011000010111001001100101001
00000010100000110110001100001011011100110111001101
00101101110011001110010000001100001001000000011000
10011000001000011011100100010000001000010011010010
11011110010000001000011011010000110010101101101011
01001011000110110000101101100001000000101000001110
01001101111011010100110010101100011011101000010000
00100000000100000010000010101000000001101000010 10

Figure 2. b. Plaintext in stream mode

Core-Key(CK)	4108308173
Parameter (P)	7162207672705334201

Figure 2. c. Core-key and parameter

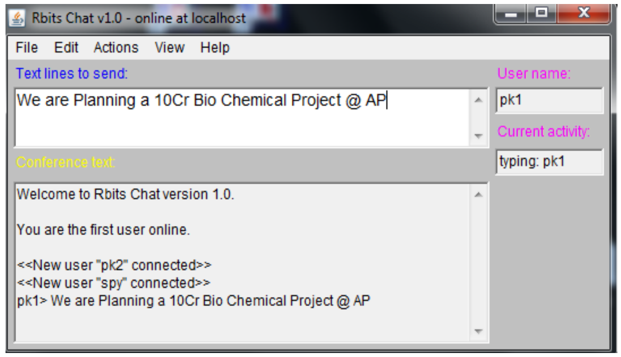

Figure 3. a. User interface at sender

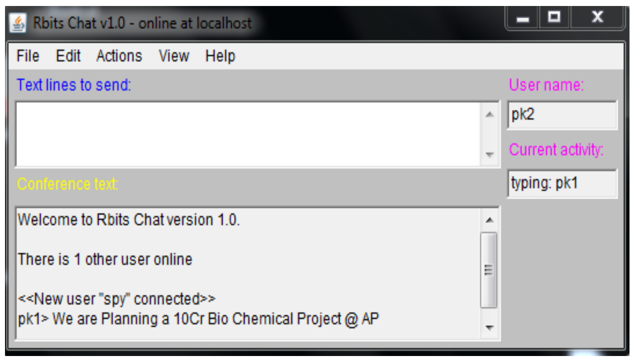

Figure 3. b. User interface at receiver

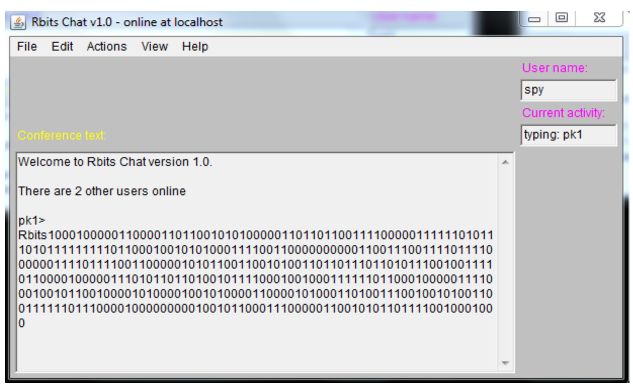

Figure 3. c. User interface at spy

The spy tries to intercept the message and captures the enciphered data shown in fig 4.a, which is in unintelligent format. The same available cipher text is shown in char mode figure 4.b

1000100000011000011011001010100000110110110011110000
0011111101011101011111111101100010010101000111110011
0000000000110011100111101111000000111101111100110000
0101011001100101001101101110110101110010011110110 0
0010000001110101101101001011110001001000111111101100
0100000111100010010110010000101000010010100001 1000
0101000110100111001001010011001111110111000010000 0
00001001011000111000001100101011011110010001000

Figure 4. a. Enciphered data in stream mode

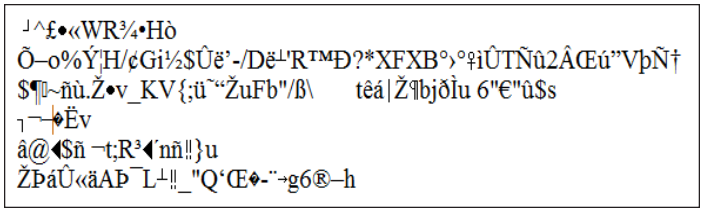

Figure 4. b. Enciphered data in char mode

We are P;anning a 10cr Bio Chemical project @AP

Figure 4. c. Decrypted data in char mode.

By comparing the original messages shown in figure 2.a, 2.b and their encrypted messages shown in figure 4.a, 4.b, the encrypted messages was very different than the original. Figures 2.a and 4.c prove that the algorithm is lossless and reversible.

The *histogram* shows the differences between original message and its corresponding encrypted messages are shown in figure 5, figure 6 shows the byte frequency, corresponding 8 bits blocks of original and encrypted message figure 5.a, figure 5.b and bit frequency, corresponding single bit figure 6.a, figure 6.b.

In the figure 5.a, 5.b the original message character frequencies (with bin size is 25) and encrypted message frequencies are listed which are distinct in nature. In the figure 6.a, 6.b bit wise (0 and 1) of original and encrypted messages are drawn and from the analysis both the frequencies are has big variances.

The line chart deviations shown in figure 7.a, figure 7.b, variances indicates that there is big statically deviations in original and encrypted messages.

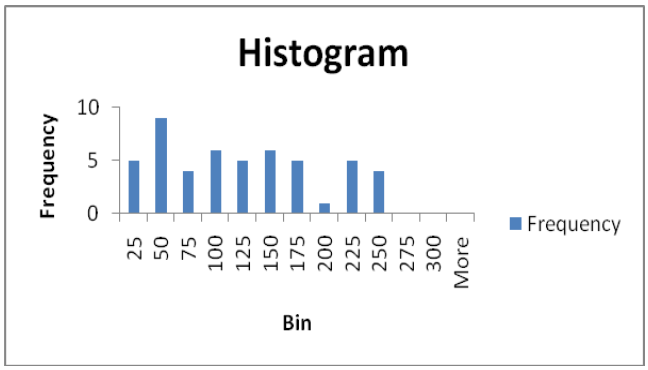

Figure 5.a

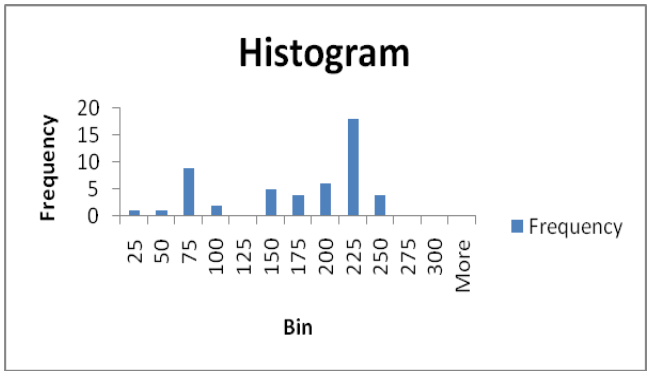

Figure 5.b

Figure 5. Byte-wise Histogram Analysis: (5.a) Frequency of original message, (5.b) Frequency of encrypted message

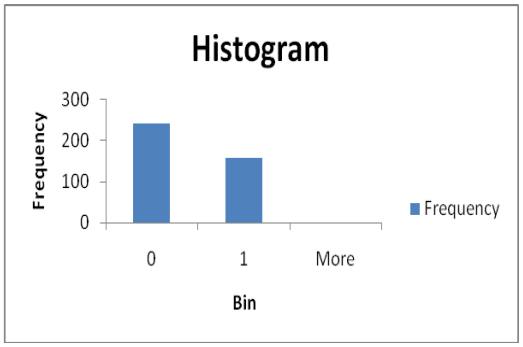

Figure 6.a

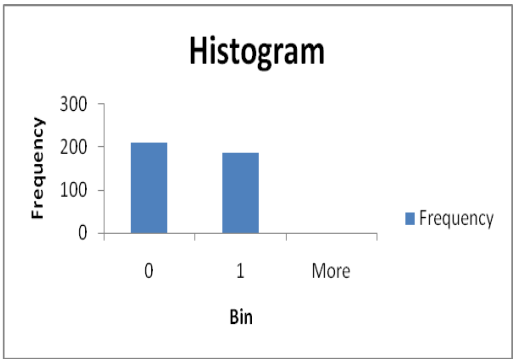

Figure 6 b

Figure 6. bit-wise Histogram Analysis: (6.a) Frequency of original message, (6.b) Frequency of encrypted message

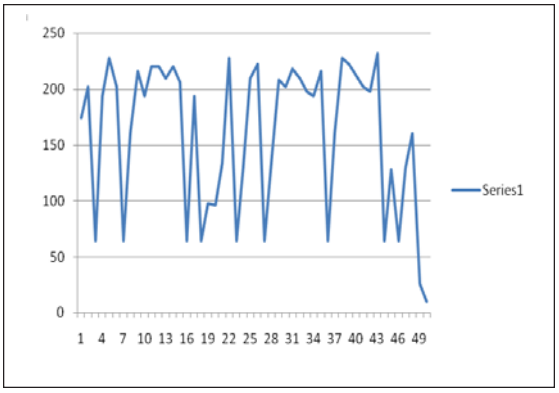

Figure 7 a

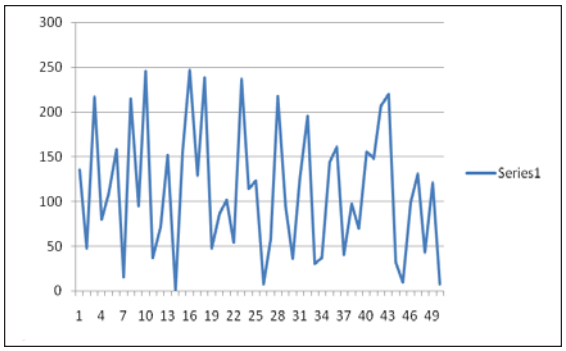

Figure 7.b

Figure 7. Byte-wise line chart Analysis: (7.a) Original message, (7.b) Encrypted message

Correlation analysis examines each pair of measurement variables (plaintext and ciphertext) to determine whether the two measurement variables tend to move together. The correlation analysis values are shown in table1 shows byte-wise, bit-wise original message and encrypted message co-relation values. It is observed that the corr_value in the table 1 are almost near to the value of zero, Which says that the original message and its encryption are totally different i.e. the encrypted message has no features and highly independent on the original message.

4.1.2. For Plaintext with repetitive patterns

We also tested the algorithm for the plaintext with repetitive patterns and also observed there is no statistical relationship even for case 2 between original and cipher text as shown in the histogram figure 8.a, 8.b, line chart figure 9.a, and 9.b and correlation analysis table 2. The plaintext for case 2 is as follows.

"CryptographyCryptographyCryptographyCryptographyCryptographyCryptographyCryptographyCryptographyCryptographyCryptographyCryptography"

Case	Corr_Value
Byte wise Data Correlation	-0.071130
Bit wise Data Correlation	0.020813

Table 1. Correlation Analysis

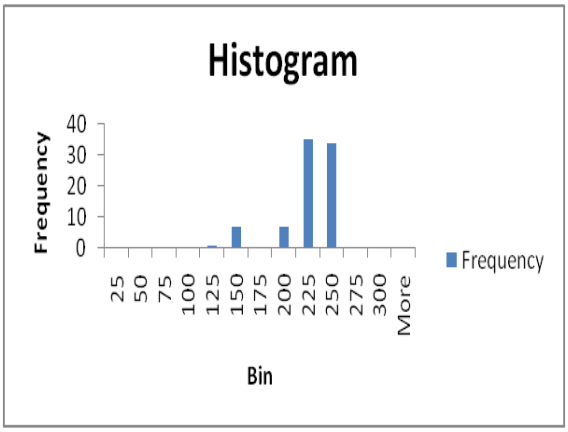

Figure 8.a

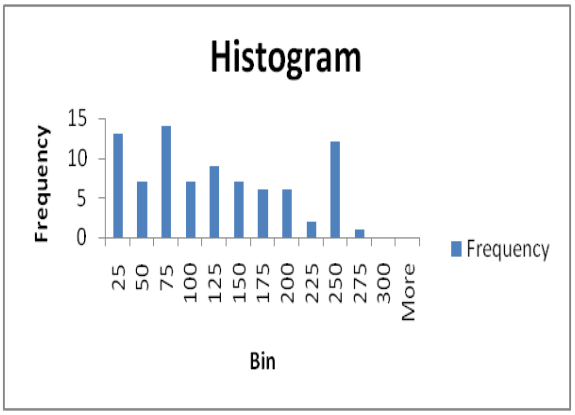

Figure 8.b

Figure 8. Byte-wise Histogram Analysis for repetitive patterns: (8.a) Frequency of original message, (8.b) Frequency of encrypted message.

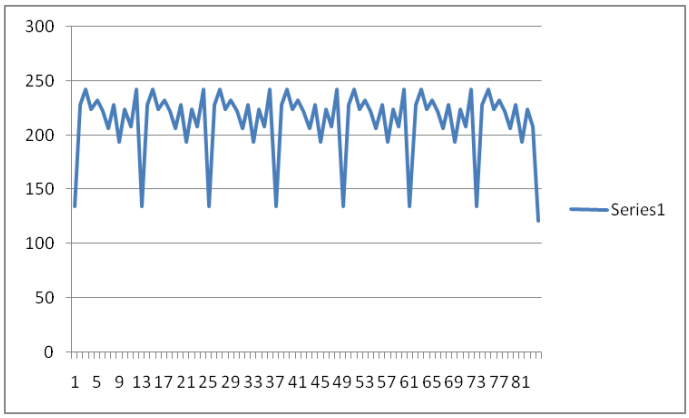

figure. 9.a

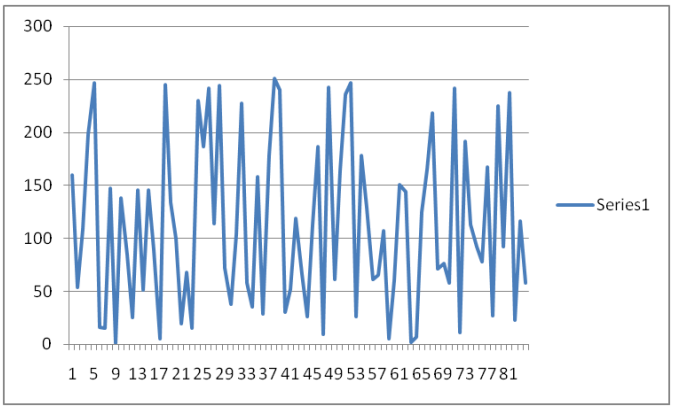

Figure. 9.b

Figure 9: Byte-wise Line Chart Analysis for repetitive patterns: (9.a) Original message with patterns, (9.b) Dissipated Encrypted message

Case	Corr_Value
Byte wise Data Correlation	-0.05811

Table 2. Correlation Analysis

4.2 Cost Level Analysis

The cost level is measured based on the type of the operations used to transform the plaintext to ciphertext. For cost analysis two things need to be considered in Rbits:

1). Cost of Random bit generation 2). Operations on plaintext

4.2.1. Cost of Random bit generation

The hardware implementation of BBS with module sizes of 160 and 512 bits with operating frequency at 100 KHz, in the Classical combinational multiplier, with Barrett's reduction method [Pedro Peris-Lopez *et a*l (2010)] shown below table 3.

No Of Bits	Time (µsec.) @ 100 KHz
160 bits	1,850
512 bits	5,270

Table 3. Hardware implementation of BBS

Another factor in cost analysis consideration is the complexity and proved that O (log log N) bits can be extracted on each iteration, where N is the modulus (a Blum integer) [Blum L et al ,(1986)].

4.2.2. Operations on plaintext

Basically there are two XORing operations, one is for key and plaintext and another is for CBC operations. XORing has computational complexity of *"order b"* which is written O(b) where b is the no of bits. XOR operation is very simple leads to cost effective.

We can derive a cost analysis equation by considering above points.

Ca = BBSc + 2XORc

Where Ca - Cost Analysis
BBSc -Cost of generation/bit
XORc -Cost of XOR/bit

Rbits Complexity = O (log log N) + 2n XOR

Cost analysis simulation is performed on windows 7 OS, 3.0GHz Dual core intel processor, using Java 1.6 and IDE Netbeans 6.3 .The values of results or time depends on the underlying operating system, and Hardware resources available at that time. The algorithm is applied on 1024 Bytes and 2048 Bytes of data in various test runs and the experimental results are shown in the table 4 and 5.

Test	Encryption Process (Time in ms)	Decryption Process (Time in ms)
Test 1	47	31
Test 2	32	31
Test 3	32	31
Test 4	31	31
Test 5	35	35
Average	35.4	31.8

Table 4. Speed Analysis for 1024 Bytes of Data

In each test run we changed the CK and MK_i values and the encryption and decryption process times values are recorded.

Test	Encryption Process (Time in ms)	Decryption Process (Time in ms)
Test 1	16	15
Test 2	32	16
Test 3	31	30
Test 4	46	46
Test 5	32	47
Average	31	31

Table 5. Speed Analysis 2048 Bytes of Data

5. Conclusion

In Rbits cipher random multiple keys are generated at the both sender and receiver, but not transmitted along with message and avoids transmission overheads. By supplying multiple keys to each block of data ensure highly secured ciphertext even for the plaintext with repetitive patters and with the added CBC operation the statistical characteristics of the plaintext are masked to such an extent that any type of cryptanalysis is infeasible.

6. Future Enhancements

We will extend the algorithm for secure store and retrieval of large data to and from cloud technology with the emphasis on keeping the security in the hand of client not in the hand of cloud service providers.

References

[1] Blum, L., Blum, M., Shub, M. (1986). A simple unpredictable pseudorandom number generator, *SIAM Journal on Computing*, 15 (2) 364–383.

[2] Chandra, S., Paira, S., Alam, S. S., Sanyal. (2014). A comparative survey of Symmetric and Asymmetric Key Cryptography *IEEE-Transactions- International Conference*. 83-93.

[3] Junod, P. (1999). Cryptographic Secure Pseudo-Random Bits Generation, The Blum-Blum-Shub Generator.

[4] Mihir Bellare, David Cash, Sriram Keelveedhi. (2011). Ciphers that Securely Encipher their own Keys, *In*: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), ACM.

[5] Pedro Peris-Lopez., Enrique San Millan., Jan, C. A. van der Lubbe., Luis A. Entrena. (2010) Internet Technology and Secured Transactions (ICITST), 2010 International Conference IEEE- Conference, 1- 6.

[6] Penchalaiah, P., Ramesh Reddy, K. (2013), Efficient and Secure Encryption Schema based on Random bits (Rbits). *International Journal of Advanced Research in Computer Science and Software Engineering*, 3 (11) 1026-1032.

[7] Safavi-Naini, R., Canetti eds, R. (2012), Multi-Instance Security and its Application to Password-Based Cryptography, Advances in Cryptology - Crypto, *In* : Proceedings, Lecture Notes in Computer Science ,7417, Springer,.

[8] Shannon, Claude. (1949), Communication Theory of Secrecy Systems, *Bell System Technical Journal*, 28 (4) 656-715, Sharad Patil, Ajay Kumar. (2010). Implemented Encryption Scheme (One Time Pad) using 9'S Complement, *International Journal of Advanced Research in Computer Science* 1 (2), July-August, 48-50.

[9] Sidorenko., A, Schoenmakers, B. (2005). Concrete Security of the Blum-Blum-Shub Pseudorandom Generator, Cryptography and Coding: 10th IMA International Conference, Lecture Notes in Computer Science 3796, 355-375. Springer-Verlag. Sun Microsystems, JavaTM 2 Platform, Standard Edition, 1.6.1 API Specification.

[10] Wenjun Gu., Neelanjana Dutta., Sriram Chellappan., Xiaole Bai. (2011). Providing End-to-End Secure Communications in Wireless Sensor Networks, 8, *IEEE-Transactions*, 3, 205-208.