Vandana Publications

Volume-4, Issue-1, February-2014, ISSN No.: 2250-0758
International Journal of Engineering and Management Research
Available at: www.ijemr.net
Page Number: 121-127

Secured Efficiency Decryption Based on Random bits (Rbits)

P.Penchalaiah[1], K. Ramesh Reddy[2]
[1]Research Scholar, Department Of Computer Science, V.S University, Nellore, A.P, INDIA
[2]Senior Scale Assistant Professor, Department Of Computer Science, V.S University, Nellore, A.P, INDIA

ABSTRACT

This paper aimed on strengthening secured communication by introducing a new cipher Rbits (Random bits). Rbits cipher uses two algorithms, a *Key Exchanging Algorithm* and a *Random Bit Generation algorithm*.

In brief the Rbits cipher is more complex in nature of attackers view. This algorithm generates random bits at the both ends these random bits are used to encrypt and decrypt, these random bits are not carried along with message, only the seed is exchanged which used to generate random bits at both ends. By this way Rbits minimizes the unbearable communication overheads to zero. Here we are presenting the encryption/decryption process and inner workings of Rbits Cipher, which was not covered in my early papers. For implementation of this cipher we are used RSA, BBS algorithms, (RSA as Key Exchanging and BBS as Random bits) and computer programming language Java.

Keywords: Cryptography, Rbits Cipher [1], One-Time Pad, Key Exchange, Random Bits, RSA, BBS, Encryption, Decryption

I. INTRODUCTION

One well-known realization of perfect secrecy is One-Time Pad (OTP). Here we aimed on strengthening secured communication by introducing a new cipher Rbits (Random bits) and here we are presenting the development of One-Time Pad encryption with optimal communication overheads while transmission of message from/to source/destination..

We can only talk about OTP if four important rules are followed. If these rules are applied correctly, the one-time pad can be proven to be unbreakable.

The Four Rules are... [11]

- ***Rule 1****: The key is as long as the plaintext.*
- ***Rule 2:*** *The key is truly random*
- ***Rule 3****: There should only be two copies of the key: one for the sender and one for the receiver*
- ***Rule 4****: The keys are used only once, and both sender and receiver must destroy their key after* use.

Fig1 shows unbearable Key overheads. (In One hand Message and other hand Key of same length)

Requirements for new Cipher

- The services of OTP is limited to short message only, since of its key size.
- The major drawback of OTP to apply on lengthy message is Rule 1, which says, "*The key is as long as the plaintext"*, one of the four rules listed above.

- ✓ The Rule 1 causes unbearable communication overheads while transmission of message from/to sender/receiver.

A. Rbits Cipher

Rbits Cipher makes OTP free from Rule 1 by using a Key Exchanging Algorithm (KE) and a Random Bit Generation algorithm (RG) and follows all four rules that are listed above.

In Rbits Cipher Random bits are generated at the both ends, but not carried along with message unlike legacy One Time Pad (OTP), which avoids transmission overheads. Seed 'S' is only transmitted instead of long key (random bits). This feature frees up the communication parties from carrying large key along with the cipher text and causes a reduced traffic in the channel.

B. Design Parameters of Rbits Cipher

Block size: Larger block sizes mean greater security. But reduced encryption/decryption speed is Indirect proportional to Block size. Block size of 64 bits has been considered a reasonable trade-off and was nearly universal in block cipher design

Key size: Larger key size means greater security but may decrease encryption/decryption speed. Greater resistance to brute-force attacks and greater confusion achieves the greater security. Key sizes of 64 bits or less are now widely considered being inadequate and 128 bits has become a common size.

Complexity: As the Random bits are unique for each chunk the generated cipher stream contains no statically relationship with the plain text. So Rbits Cipher does not have an easily analyzed functionality, which makes more difficult to cryptanalysis.

Speed: There are no heavy computational requirements to encrypt/decrypt (like RSA, DES, and AES where major part of the algorithms are involved in generation of 'n' keys for 'n' rounds and requires heavy exponential computations which reduces the performance by consuming more time)

Ease of analysis: The entire strength of this Rbits cipher depends up on the KE and RG.

II. REQUIREMENTS OF Rbits

Rbits Cipher has two basic selection requirements for its implementation.

1. *Key Exchanging Algorithm (KE) and*
2. *Random Bits Generation Algorithm. (RG)*

A. Key Exchanging Algorithm (KE) [1]

Alice/Bob can use any Key Exchanging Algorithm (KE) (like RSA, Diffie-Hellman, etc) to send the seed "S" to Bob/Alice.

B. Random Bits Generation Algorithm (RG) [1]

Alice/Bob selects Random seed "S" which will be used for Random Bits Generation. The Random Bit Generation (RG) algorithm must be such way that it must generate same sequence of random bits for the same seed "S". Both Alice and Bob must follow same RG.

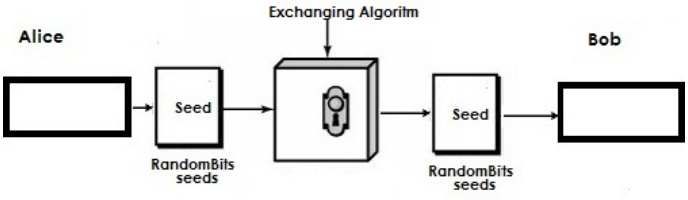


Figure 2

C. Rbits Algorithm

Alice (Sender) End

P_a -Plaintext

B_a -Binary Stream of P

S_a -large Random Seed

KE_a -Key Exchanging algorithm

RG_a -Random Bit Generation algorithm

$Rbits_a$ -Random Bits generated using RG_a

Bob (Receiver) End

KE_b -Key Exchanging algorithm (=KE_a)

S_b - Large Random Seed Received from Alice (=S_a)

RG_b -Random Bit Generation algorithm (=RG_a)

Encryption

$C= E (B_a, Rbits_a)$

Decryption

$P= D (C, Rbits_b)$

Let us assume Alice trying to initialize the communication with bob. Alice first selects a Random Key or Seed 'S', and exchanges the "S" with bob using RSA.

D. Encryption

Once the seed "S" has exchanged, Alice initiates encryption process.

Encryption Algorithm:

***Step 1*:** Alice converts plain text in to binary stream.

***Step 2*:** The binary stream is divided in to fixed size of chunks for effective system performance.

***Step 3*:** A chunk of binary stream is XOR with the same length of Random Bits, generated using seed "S" with a RG.

Step 4: Step 3 is continued until last chunk.

Encryption

$C= E (B_a, Rbits_a)$

Detailed explanation can be found in earlier paper [1].

E. Decryption

As Bob aware of the Random bit generation seed "S', Bob is also able to generate same random bits. Bob decrypts' each chunk of stream by XOR operation with the random bits by using the same seed "S'" and same RG algorithm. Bob continues until the last chunk of cipher stream. Once all chunks are decrypted the resultant binary stream again converted to plain text, which is in readable format.

Once the Bob receives cipher text "C", Bob initiates decryption process.

Decryption Algorithm:

***Step 1*:** Bob divides the cipher stream into fixed size of chunks.

***Step 2*:** Generates random bits using seed "S" with RG.

***Step 3*:** Each cipher stream chunk is XOR by the random bits.

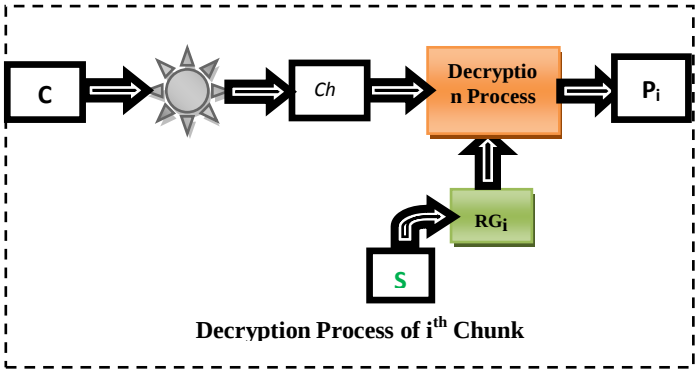
***Step 4*:** Steps 2 and 3 are continued until last chunk.

***Step 5*:** Bob converts binary stream into plain text.

Decryption

$P= D (chunk_i , Rbits_b)$

Decryption of i^{th} chunk



Decryption Process of i^{th} Chunk

III. IMPLEMENTATION

As described above, the implementation of Rbits Cipher starts with by selecting KE and RG. We used java1.6 for the following implementation.

Selection of KE and RG

In my previous paper[1] we described Rbits Cipher Encryption process, by selecting RSA as KE and BBS as RG. So for better understanding in this paper we are going to use same KE and RG, RSA as KE and BBS as RG.

Receiving BBS Seed set S={s, n}

Alice selects seeds 'S" for BBS random bit generation, i.e. S=***{s, n}.***

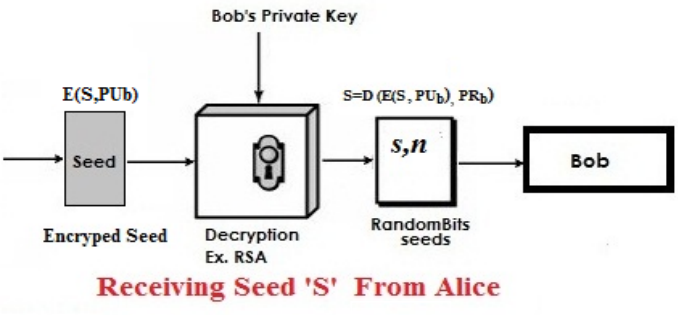


Fig 4

Let Alice selected 'S'.

Prior to exchanging of seed 'S' both Alice and Bob generates their own pair of keys.

Alice Keys= {PUa, PRa}

Bob Keys= {PU_b, PR_b}

Alice and Bob follow RSA encryption/Decryption process. Alice encrypts the seed set 'S' by accessing Bob's public key from Alice and sends the resultant key to Bob. Bob

$$S=D\ (E(S, PU_b), PR_b)$$

decrypts the encrypted seed set 'S" by his private key PR_b and retains it with him for BBS random bit generation.

Where

D- RSA Decryption Process

E- RSA Encryption Process

S-

Chunk #	**Stream**	**Chunk#**	**Stream**
Chunk1	10100000	Chunk2	10001010
Chunk3	10011100	Chunk4	10000110
Chunk5	10010000	Chunk6	10000010
Chunk7	10011000	Chunk8	10000010
Chunk9	10010010	Chunk10	10000010
Chunk11	1001000		

Table 1 List of chunks After sizing

Length Of Cipher Stream	*Len (c)*	87 bits
Block/Chunk Size	B_Z	8 bits
No Of Blocks/Chunks	*L*	Len(c) / 8 =10
Length Of Last Chunk	*Len(r)*	Len(c) MOD L =7bits

Table 2 Sizing Statistics

Seed set {s, n}

Now Both Alice and Bob having the seed set S={s, n}.

Decryption

Bob applies decryption process on cipher stream 'C' received from Alice.

The cipher stream 'C' undergoes to *sizing process*[1]. Block size of 64 bits has been considered a reasonable.

The ***sizing*** process for decryption is just reverse process of encryption sizing.

Bob divides the cipher stream "C" into L fixed size of chunks ($Chunk_1$, $Chunk_2$, $Chunk_3$…$Chunk_i$…$Chunk_n$).

C= {$Chunk_1$, $Chunk_2$, $Chunk_3$…$Chunk_i$…$Chunk_n$}.

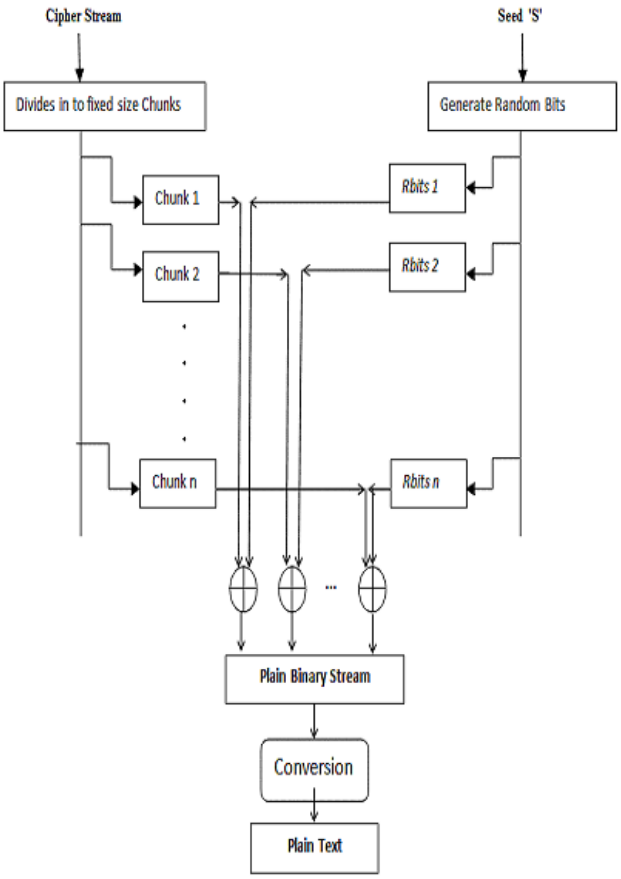


Figure 5

The following example shows the sizing process, here 'C' cipher stream, which was produced in Rbits encryption process.

C="10100000100010101001110010000110100100001000001010011000100000101001001010000001010010000".

Rbits cipher produced the above chunks and statistics (Table 1 and Table 2) using java.

In the example 'C' has the length of 87bits. So C will be sized in to '10' chunks (Table 1) and each chunk has 8bits (as Bz=8).'C' is not a multiple of '8', so it has $\mathbf{b_r}$ having 7 bits (table 2 chunk11).

By using BBS, random bits are generated $\mathbf{Rbits_i}$, as same length of a chunk in question.

Decryption function can be any decryption technique. For Simplicity, here we are using XOR operation as decryption

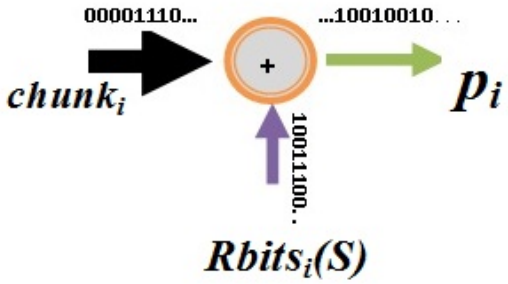


Figure 6

function since encryption process followed XOR operation[1].

Note : All *Rbitsi(S))* are unique in nature.

Rbits Cipher -Decryption Run Log.

We used Java 1.6 to develop Rbits cipher Decryption algorithm. Machine implementation of Rbits Cipher Decryption produced the following log during its execution under windows 2007 Operating System, Duel core processor.

```
--------------------LOG START-------------------
-----------------------------

BOB Is Ready....
Press Any Key To Receive BBS SEEDS....
STREAM SIZE Init..128
BOB :Accessing BBS SEEDS.
BOB :BBS SEEDS Decryption Started.
BOB :BBS S-SEEDS Decryption Started.
Accessing S....
96877480982164706622663028304734251722663028304734
25173433696840388535745053450982722315902434251751
43369684038853574505345098272231590243369684038853
57450534509827223159021403940076536623685226630283
04734251722663028304734251722663028304734251722663
02830473425172266302830473425172266302830473425172
2663028304734251722663028304734251722663028304734
2517
BOB :BBS S-SEEDS Decryption Done.
BOB :BBS N-SEEDS Decryption Started.
Accessing n....
50534509827223159025143369684038853574232922257313
54497761403940076536623685505345098272231590243556
48873290978884968774809821647066114655404381869617
41403940076536623685140394007653662368596118504573
25430501146554043818696174140394007653662368550534
50982722315902232922257313544977651433696840388535
74435564887329097888413215923209859021051433696840
38853574505345098272231590 2
BOB :BBS N-SEEDS Decryption Done.
BOB :Sending Request to BBS.....
```

```
The Value Of BBS N is 13401928005801439631
The Value Of BBS S is 2773107377
BOB :Accessing Cipher Stream Init..
Receiving Cipher Text....
BOB: Cipher Stream Received..
00101110010111010001111010100111110011011101010010
10011100011101101101110010110101111101100001110011
11110010101001100110001011010001010001010001100101
00111010111101011101011110011101010111110100101001
01110111111001011100100100101111001110101111101000
10110100001100011111010011010010011011110110010101
11001111111100011101101101110010111101011111011000
11101111011001011011000010001011010000010000011011

BOB :Rbits Cipher Decryption Started..
BOB :Generating Random Bits.....
10000110100011011101010001000011000001111001010000
11010111111011111101111010111100111101000000111110
00011111000001010001001101101100001101000110111010 1
00010000110000011110010100001101011111101111110111
10101111001111010000001111100001111100001010001001
10110110000110100011011101010001000011000001111001
01000011010111111011111101111010111100111101000000
11111000011111000010100010011011011000011010001101
11010100010000110000011110010100001101011111101111
11011110101111001111010000001111100001111100001010
001001101101
BOB :Rbits Cipher Decrypted Bin Stream
10101000110100001100101001000000100100101110011001
00000011000010010000001000010011011110110001001101
10100100000010010010110111000100000011101000110100
00110010100100000010000110110111110110110001101 1000
11001010110011101100101001011100010000001000101011
11000011100000110110001101111011001000110010101110
01100100000010000000011000100110000001011100011001
10011000000100000010000010100 1101
BOB :Rbits Cipher Decryption Finished..
BOB :Plain Text
There Is A Bomb In The College, Explodes @10.30 AM

----------LOG END----------
```

IV. CONCLUSION

This is in continuation to my earlier paper[1], Here we come to the point that Rbits Cipher scheme *is very strong and secure* and is more simple for user point of view but very difficult for attacker point of view. Here we are following four rules of OTP [11], if these rules are applied correctly, the one-time pad can be proven to be unbreakable. There are no heavy computational requirements to encrypt/decrypt, which reduces the performance by consuming more time. The entire strength of this Rbits cipher depends up on the selection of KE and RG.

V. FUTURE PAPERS

In my earlier and current papers we presented how Rbits cipher encrypts and decrypts the given messages and the way Rbits cipher minimizes the unbearable communication overheads to zero. In my forth coming papers I am going to present the performance analysis of Rbits cipher by comparing with well known and predominant ciphers like RSA, DES, AES etc. on the factors Running Time, speed, Accuracy, complexity, resource utilization, cryptanalysis, brut force attacks, etc.

.

REFERENCES

[1]. P.Penchalaiah , K. Ramesh Reddy “Efficient and Secure Encryption Schema based on Random bits (Rbits)” International Journal of Advanced Research in Computer Science and Software Engineering” Volume 3, Issue 10, October 2013 PP. 1026-1032.

[2]. Sharad Patil Ajay Kumar “Implemented Encryption Scheme (One Time Pad) using 9’S Complement” International Journal of Advanced Research in Computer Science Volume 1, No. 2, July-August 2010 PP 48-50

[3]. Md. Mizanur Rahman “Any File Encryption by Translating ASCII Value of Characters” International Journal of Advanced Research in Computer Science Volume 3, No. 2, March-April 2012 PP 41-43

[4]. Dr. S. Udaya Kumar , Ravindra Babu Kallam “An Enhanced RSA Public key Cryptographic Algorithm” International Journal of Advanced Research in Computer Science Volume 2, No. 5, Sept-Oct 2011 PP 497-499

[5]. Information Technology Journal 4(3): 204-221, 2005

[6]. Neal R. Wagner “The Laws of Cryptography: Perfect Cryptography: The One-Time Pad”.

[7]. Avanish Kumar Singh, Amit Kumar Pathak "Encryption Techniques as Security Tools: A Technical Review" International Journal of Advanced Research in Computer Science Volume 3, No. 7, Nov-Dec 2012 PP 269-2073

[8]. Ayushi, (2010), A Symmetric Key Cryptographic Algorithm, International Journal of Computer Applications (0975 - 8887) Volume 1. No. 15.

[9]. Bruce Schneier, "Beyond Fear", Springer-Verlag, New York, 2006

[10]. Saikat Ghosh Sukalyan Som A Survey of Traditional or Character Oriented Symmetric Key Cryptography Volume 2, No. 4, July-August 2011 International Journal of Advanced Research in Computer Science PP 147-151

[11]. Claude Shannon's "Communication Theory of Secrecy Systems