APTIKOM Journal on Computer Science and Information Technologies
Vol. 2, No. 3, 2017, pp. 137~144
ISSN: 2528-2417
DOI: 10.11591/APTIKOM.J.CSIT.75

RK algorithm: stochastic parallel methodology for symmetric key cryptography

Yegireddi Ramesh*[1], Kiran Kumar Reddi[2]
[1]Department of CSE,
Aditya Institute of Technology and Management, Tekkali 532201, Srikakulam, Andhra Pradesh, India
[2]Department of CS, Krishna University,
Andhra Pradesh, India
*Corresponding author, e-mail: rameshyegireddi@gmail.com

Abstract

With the enormous growth in the Internet and network, data security has become an inevitable concern for any organization. From antecedent security has attracted considerable attention from network researchers. In this perspective many possible fields of endeavour come to mind with many cryptographic algorithms in a broader way, each is highly worthy and lengthy. As society is moving towards digital information age we necessitate highly standard algorithms which compute faster when data size is of wide range or scope. On survey, numerous sequential approaches carried out by symmetric key algorithms on 128 bits as block size are ascertained to be highly in securable and resulting at a low speed. As in the course the commodities are immensely parallelized on multi core processors to solve computational problems, in accordance with, propound parallel symmetric key based algorithms to encrypt/decrypt large data for secure conveyance. The algorithm is aimed to prevail by considering 64 character (512 bits) plain text data, processed 16 characters separately by applying parallelism and finally combine each 16-character cipher data to form 64-character cipher text. The round function employed in the algorithm is very complex, on which improves efficacy.

Keywords*: ciphertext, decryption, encryption, parallel, plaintext, secret key*

1. Introduction

With all of the vital personal and business data being shared on computer networks every day, security has become one of the most essential aspects of networking. No one recipe to fully safeguard networks against intruders exists. As the security technology improves and evolves over time the methods for both attack and defence grow more sophisticated. Internet has become more and more widespread, authorization to access information being compromised. The data when conveyed in the network must be adopted with some provisions and policies such as data confidentiality, data integrity, authentication, and non-repudiation in order to block from adversaries. With all this aspect network has to be designed with secure technology in prevention from enormous threats. Well-known technology to guarantee data confidentiality and fine-grained data access control is cryptography, which is important for network security. The development of cryptography has been paralleled by the development of cryptanalysis, the "breaking" of codes and ciphers. Techniques used for decrypting a message without any knowledge of the encryption details fall into the area of cryptanalysis. Right away the challenging problem is how to effectively share encrypted data. In this panorama untold cryptographic algorithms came into evolution for storing, processing and communicating sensitive or valuable information. Cryptosystems that are used to encrypt or decrypt data are taxonomy of two classes, Symmetric and Asymmetric also called as secret key cryptography and public key cryptography [1].

1.1. Symmetric Encryption

If there is occurrence for symmetric encryption, cryptosystem for enciphering and deciphering keys are either identical or simply related, between sender and recipient. Keys must be kept secret on transferring, if either is compromised secure communication is impossible in further. It was also known as conventional encryption.

Received July 17, 2017; Revised August 20, 2017; Accepted September 8, 2017

1.2. Asymmetric Encryption

A public key cryptographic system that uses pair of keys – public key which is widely disseminated and private key which are only known to the owner. At this juncture unless the private key is disclosed message can't be unscrambled by any individual. This is an endeavour to guarantee privacy. The network to indulge the above said, propound numerous conventional encryption algorithms processed sequentially named underneath DES, AES, TDEA, IDEA, Blowfish, RC2, RC4, RC5, CAST 128, [2]-[6] etc. On comparison among these algorithms on data size of wide range and their ability to secure and protect data against attacks ascertained to be highly in securable which resulting at low speed. Due to continuing advancements in communications and eavesdropping technologies, business organizations and private individuals, algorithms with high degree of complexity has to be formulated for secure transmission. As transmission of data size is increasing in higher orders the process for encrypting has to be speed up on each blocks of data. In this course the sequential encryption algorithms are more protracted. In accordance, initiative towards parallelism has come forth for quick responsive. As here and now the machines are equipped with multi core processor, process for parallelism became plain sailing. Being the case proposed parallel symmetric key cryptographic technique named RK algorithm employed with round function which computes at high speed. This paper is structured as follows: Reviewed the related work on proposed algorithm for encryption, key generation and decryption in section 2, detailed construction of parallel architecture in section 3, Experimental outcomes in section 4 and potency of the algorithm in section5.

2. Research Method

In the proposed technique, the original text is cleaving into 64 characters' block, if exceed 64 characters, otherwise padding is done for proper block division. As symmetric encryption technique is followed by means of secret key, here the used key holds with length of 64 bits' value.

2.1. Encryption

Encryption is the most effective way to achieve data security. To have encryption on data or plain text we need secret key or password to obtain cipher text.

2.2. Algorithm

Consider first 64 characters and place these characters into P_{8x8} matrix as shown in Figure 1. The 64-character plaintext is divided into four sub-blocks B1, B2, B3 and B4 for parallel execution. The 16 characters from each block is converted into equivalent binary, so each block contains 128 bits. Each sub-block is divided into two parts (64 bits each) and passed to Round Function (RF). The RF contains two shift registers. The first four bits from first shift register (SR1) is concatenated with the first four bits of second shift register (SR2). The next four bits from first shift register (SR1) is concatenated with the next four bits of second shift register (SR2). The two concatenated values are ex-ored with first round key. Two circular shift operations are perfomed to generate the input for next round. The same procedure is repeated for 8 times (8 rounds of operation). The 8^{th} round output is the required ciphertext. The results of all sub-blocks B1, B2, B3 and B4 are combined to generate the final output.

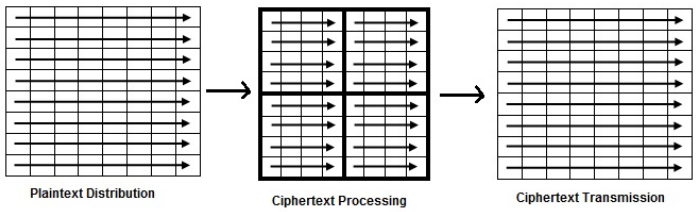

Figure 1. Distribution and parallel Processing of Plain text into Cipher text

Algorithm: RKEncryption

1. *// P[1:8][1:8] is a plaintext block and used to store (8X8) 64 plaintext characters*
2. *//P is divided into four sub-blocks B1, B2, B3 and B4 for parallel execution*
3. *Begin*

```
4.    {
5.      Let m=8 and n=8
6.      for i :=1 to m do
7.      begin
8.       for j :=1 to n do
9.       begin
10.          if i<m/2 then
11.           begin
12.           if i<n/2 then
13.          B1[i][j] :=P[i][j]
14.           else
15.          B2[i][j-n/2] :=P[i][j]
16.          endif
17.        else
18.           begin
19.           if i<n/2 then
20.           B3[i-m/2][j] :=P[i][j]
21.           else
22.           B4[i-m/2][j-n/2] :=P[i][j]
23.           endif
24.       endif
25.      endfor
26.     endfor
27.    //for parallel execution
28.  parallelExe(B1)
29.  parallelExe(B2)
30.  parallelExe(B3)
31.  parallelExe(B4)
32.  //C[1:8][1:8] is a ciphertext block and used to store (8X8) 64 ciphertext characters
33.     for i :=1 to m do
34.     begin
35.      for j :=1 to n do
36.      begin
37.          if i<m/2 then
38.            begin
39.            if i<n/2 then
40.            C[i][j] :=B1[i][j]
41.            else
42.            C[i][j] :=B2[i][j-n/2]
43.            endif
44.        else
45.            begin
46.             if i<n/2 then
47.            C[i][j] :=B3[i-m/2][j]
48.            else
49.            C[i][j] :=B4[i-m/2][j-n/2]
50.           endif
51.         endif
52.      endfor
53.     endfor
54.  end
```

Algorithm: ParallelExe(B)

```
1.      //IC[16]is used to store equivalent ASCII values of each sub-block
2.      begin
3.      for i :=1 to 16 do
4.      IC[i] :=toAscii(B[i])
5.      endfor
```

```
6.  //BIC[128]is used to store binary equivalent of each  ASCII values
7.  i=1
8.  for j :=1 to 16 do
9.  begin
10. BIC[i]=toBin(IC[j])
11. i:=i+8
12. endfor
13. for i :=1 to 64 do
14. SB1[i]:=BIC[i]
15. j:=1
16. fori i :=65 to128 do
17. begin
18. SB2[j]:=BIC[i]
19. j:=j+1
20. endfor
21. RoundFunction(SB1, SB2)
22. End
```

Algorithm: RoundFunction

```
1.  SRs are shift registers used to calculate circular shift operation
2.  SB1 and SB2 values are copied into SR1 and SR2
3.  begin
4.  for x :=1 to 8 do
5.  k=1
6.  for j :=1 to 4 do
7.  begin
8.  SR3[k] :=SR1[j]
9.  k:=k+1
10. endfor
11. for j :=1 to 4 do
12. begin
13. SR3[k] :=SR2[j]
14. k:=k+1
15. endfor
16. k=1
17. for j :=5 to 8 do
18. begin
19. SR4[k] :=SR1[j]
20. k:=k+1
21. endfor
22. for j :=5 to 8 do
23. begin
24. SR4[k] :=SR2[j]
25. k:=k+1
26. endfor
27. Key contains 64 bit key value
28. Key :=~Key
29. for i :=1 to 8 do
30. begin
31. SR1[i] :=SR3[i]@Key[i]
32. SR2[i] :=SR4[i]@Key[i]
33. endfor
34. SR1<<<8
35. SR2<<<8
36. Key<<<8
37. endfor
38. end
```

2.3. Key Generation

Choose any random value as Secret key. The size of key is 64 bits. The Secret key is used to generate 8 sub keys, which are used in 8 different rounds of Round Function. Each sub-key contains a complement and circular shift operation. Each sub-key is generated from the previous sub-key. Figure 2 shows block diagram of RK algorithm encryption.

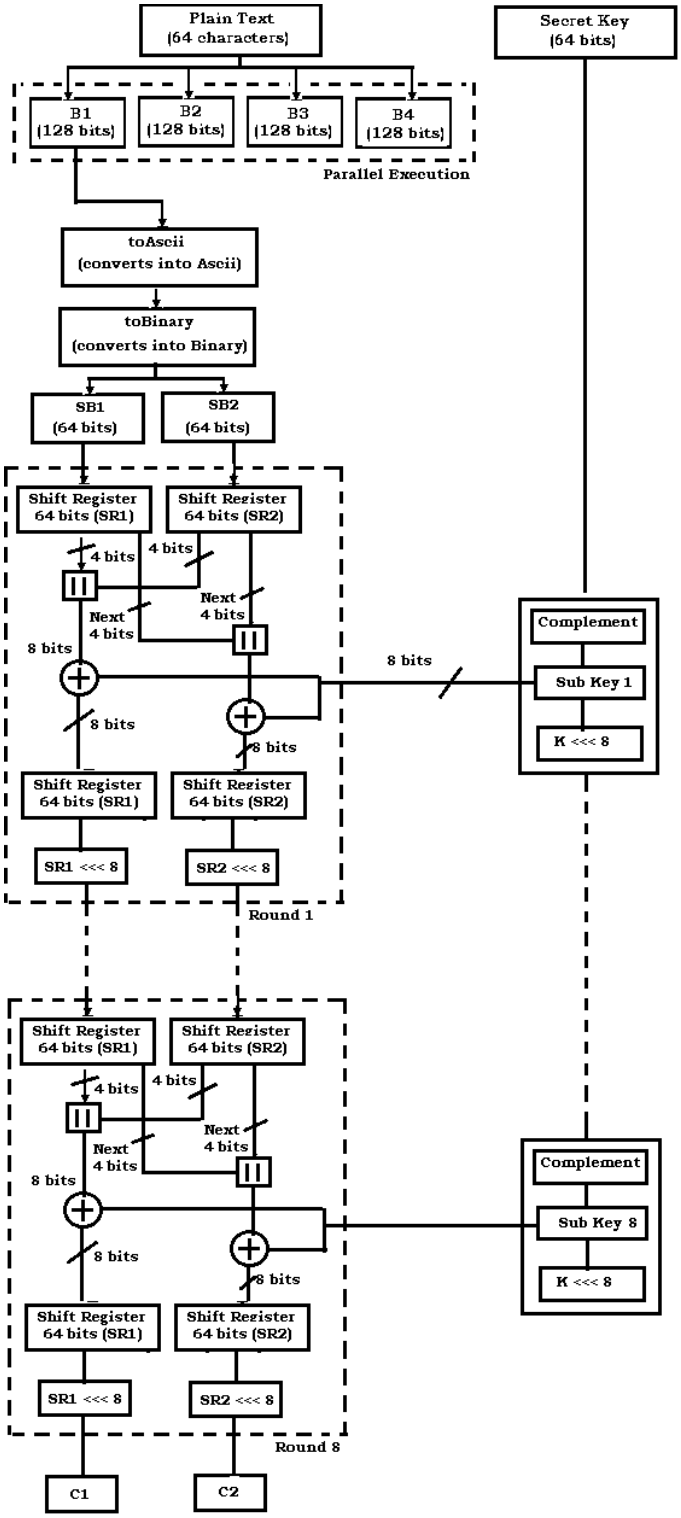

Figure 2. Block diagram of RK algorithm encryption

2.4. Decryption

Decryption is the process of taking encoded or encrypted text or other data and converting it back into text that you or the computer can read and understand.

3. Parallel Architecture

Data before transmitting from sender to recipient has to be encrypted by means of secret key in order to obtain cipher text. As data stream is collection of characters every single character has to be converted to binary form. During encryption technique, can either follow stream cipher or block cipher for transmission on data stream. Stream cipher encrypts by means of cryptographic key and algorithm on each binary digit of data stream, one bit at a time. With this stream cipher the key should be longer than the plain text. Most of the algorithms implemented by means of stream cipher carried out sequentially, makes computation too longer. In view of, this technique is not highly used in modern cryptography. As an alternative, block cipher technique applied to blocks on data stream as a group rather than one bit a time. The same key is shared between both ends for encryption and decryption. Here in the case the block cipher technique implemented parallel rather than sequentially, makes computation faster on each block and allow the process run concurrently [7].

3.1. Experimental setup:

Object-oriented programming is a method of programming based on a hierarchy of classes, and well-defined and cooperating objects. An instance is an executable copy of a class. Another name for instance is object. There can be any number of objects of a given class in memory at any one time. Thus makes the methods on the class to be called on different objects at same instances. The computation can contain sequential statements executed by a single thread, interspersed with parallel statements executed by multiple threads. Running on a multi-core computer, the Java Virtual Machine (JVM) schedules each thread on a separate core, thus executing the program in parallel [8][9]. The java.nio.file package defines interfaces and classes for the Java virtual machine to access files, file attributes, and file systems. This API may be used to overcome many of the limitations of the java.io.File class. The toPath method may be used to obtain a Path that uses the abstract path represented by a File object to locate a file. The resulting Path may be used with the Files class to provide more efficient and extensive access to additional file operations, file attributes, and I/O exceptions to help diagnose errors when an operation on a file fails.

```
public abstract class RKAlgoUtils
public static byte[] getInputText()
/**method reads all the input text with the Files class to operate on files, directories, and other types of files. For example, suppose we want a BufferedReader to read text from a file "text300.txt". getInputText() method gets invoked on RKAlgoUtils object from main class RKAlagorithm.*/
public void instantiateBlocks() on RAlagorithm
/** object reads the text from inputText and arrange the text into 4 blocks where each block arranged in 2D array form on each block object setCell2DArray() on instantiateBlocks().*/
```

Now the process of *encryption* takes a clear-text *document* and applies a key and a mathematical *algorithm* to it, converting it into crypto-text. The key generated by choosing any 8 digit Random value on

```
// public static int getRandomNumber()
// which returns the Random number on which keys K1 to K8 are generated by considering complement and shift operations.
```

The process of encryption takes place in step by manner on

```
public void instantiateBlocks()
{
Block block1=new Block();
 int[][] cells1=new int[16][16];
 block1.setCell2DArray(cells1);
 inputBlockArray[0]=block1;
 StringBuilder[][] cell1=new StringBuilder[16][16];
  block1.setChiper(cell1);
  inputBlockArray[0]=block1;
}
/**This method creates four instances on Block class of input text dividing into 2Darray with 16 elements on each block calling on each Block object */
```

```
block1.setCell2DArray(cells1),
/**which generates 4 objects block1, block2, block3, block4 each with 2D array containing 16 elements. Each block is initialized into inputBlockArray , inputBlockArray[]=block1;*/

public static void initializeBlocks(byte[] inputText)
/** this method gets called on its class RAlagorithm.initializeBlocks(inputText) to initialize text from inputText on each block object with 16 element each. Each 2D block with 4*4 elements gets returned on int[][] cell2DArray=inputBlockArray[block].getCell2DArray()*/
StringBuilder[][] cell1=new StringBuilder[16][16];
block1.setChiper(cell1);
inputBlockArray[0]=block1;
/**Now each block is called on its object to perform the computation and generate cipher text on stringBuilder object*/
public void generateBinaryForAscii(boolean encryptMode);
/**generates binary value for the Ascii generated on intermediate cipher text called on toBinary() on Block object*/
 inputBlockArray[blocki].generateMergeBinary();
/** On the generated four 64 bit block we call the generateMergeBinary() on inputBlockArray of block1 object which obtain the first 4 bits of B1S1 is append with first 4 bits of B1S2, similarly the next 4 bits of B1S1 is merged with next four bits of B1S2, which finally generates 4 values of individual four 64 bits. The same process is repeated on all four blocks by calling on individual objects, generates 16 binary data to its equivalent decimal. */
```

4. Experimental Outcomes

Doing experimentation is not simple, have to prepare, conduct and analyze to draw general conclusions. To sketch the outcome, the experiment should be given proper setup to ensure successful experimentation. The relationship between the treatment and outcome is brought on Operating System; Windows 10 Pro, System Hardware: Processor: Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz 2.30 GHz, installed memory: (RAM): 4.00 GB, System type: 64-bit Operating System, x64-based processor, Development Platform: Java SE 8 (jdk-build 1.8.0_141-b15). Also the performance is measured and compared with stanadard algorithms like DES, Blowfish and AES. The plaintest block size of DES is 64 bits, Blowfish is 64 bits and AES is either 128 or 192 or 256 bits only, but the block size ofRK algorithm is 512 bits (64 characters X 8 bits). Even the block size is large RK algorithm shows best performance, because it is implemented using parallel approach. Table 1 shows comparison of algorithms. Figure 3 shows plaintext. Figure 4 shows ciphertext. The comparison analysis is shown in the Figure 5.

Table 1. Comparison of Algorithms

	DES	AES	Blowfish	RK
Algorithm Type	Symmetric, Block Cipher	Symmetric, Block Cipher	Symmetric, Block Cipher	Symmetric, Block Cipher
Block Size	64 bits	128/192/256 bits	64 bits	512 bits
Key Length	56 bits	128/192/256 bits	32 to 448 bits	64 bits
Rounds	16	10/12/14	16	8

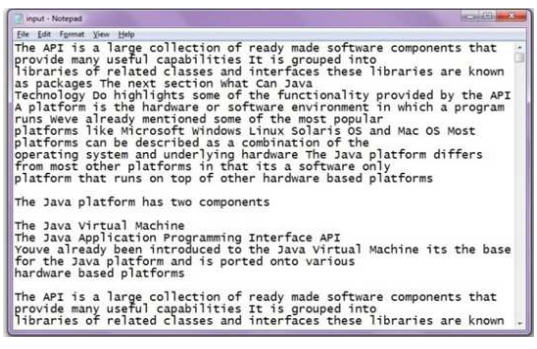

Figure 3. Plaintext

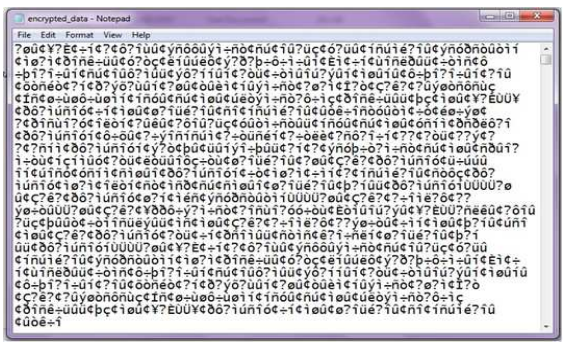

Figure 4. Ciphertext

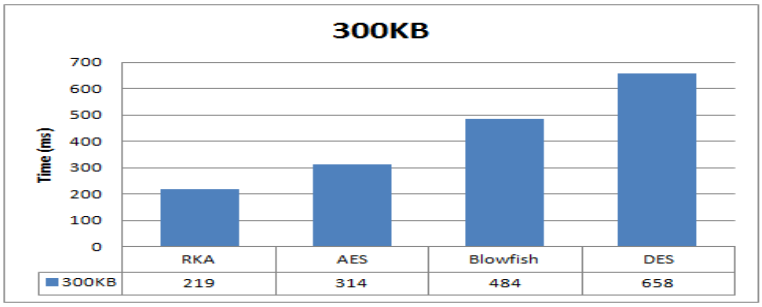

Figure 5. Comparsion of algorithms

5. Potency of the Algorithm

In summary, the experimental results show that our scheme incurs less computational cost on encryption and decryption of data stream which ensures secure transmission. In this cryptographic technique discrete cipher text values will be generated for the plain text, which complicates the cryptanalyst to unscramble. Process carried out by many steps like converting into ASCII and binary, concatenation of bits, shift operations. The cipher text has been calculated individually for each 4*4 matrix, but data is transmitted from first value to last sequentially. Over and above that the round function employed in this algorithm is very complex to break, on which improves the potency of the algorithm. The performance analysis of this algorithm is compared with AES, Blowfish and DES. The experimental result shows that this parallel algorithm has the best performance.

6. Conclusion

With highly advancement of technology, society moving toward digital information age requires secure transference in the network. In provision to this, network security has come up with numerous algorithms in means of transmuted readable to unreadable form. Together with network has adopted many provisions and polices in its infrastructure to protect the network from unauthorized user accessing sensitive data. On study, acquired knowledge on many symmetric cryptographic algorithms carried out sequentially using stream cipher for encryption, which leads to high computation and less securable. As the things are moving towards parallelism, contemporary strategy has designed guarantees high security. In this paper proposed symmetric encryption technique on block cipher executing the things parallel targeting speed and energy consumption running on symmetric multi processing machines. The attackers to crack the algorithm require profound knowledge. Hence the results proved to be more efficient on compared to sequential approach.

Acknowledgements

N. Preeti, Department of CSE, AITAM, Tekkali

References

[1] William Stallings. Cryptography and Network Security Principles and Practice, 5th Edition, 2011, Pearson Education.
[2] Ramesh Yegireddi, R Kiran Kumar. A survey on Conventional Encryption Algorithms of Cryptography, 978-1-5090-5515-9/16/ ©2016 *IEEE*.
[3] http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
[4] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[5] How-Shen Chang. International Data Encryption Algorithm. CS-627-1, Fall 2004.
[6] B Schneier. *Description of a New Variable Length Key, 64Bit Block Cipher (Blowfish)*. Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993), Springer Verlag, 1994: 191204.
[7] Peter Cappello, Bernd Christiansen, Mihai F. Ionescu Michael O. Neary, Klaus E. Schauser, and Daniel Wu, Javelin: Internet-Based Parallel Computing Using Java, 1996
[8] Hiromitsu Takagi, Satoshi Matsuoka, Hidemoto Nakada, Nin et: a Migratable Parallel Objects Framework using Jav.
[9] J E Moreira, S P Midkiff, M Gupta, P V Artigas, M Snir, R D Lawrence. Java programming for high-performance numerical computing. *IBM Systems Journal, IEEE*. 2000.