IJCTA, 9(21), 2016, pp. 225-231

Performance Analysis of Modified RC4 Cryptographic Algorithm using number of cores in Parallel Execution

Anandu Jayan* Akash Nair Bhargavi R. Upadhyay, Supriya M*****

Abstract **:** In this network-based computing era,software applications are playing a major role. With this role, the cryptography of these algorithm is also of a much concern. Cryptographic algorithms are complex and can consume lot of time and hence, it can benefit from parallelism. But parallelism is not that simple, as sometimes it is not possible to convert the whole algorithm into a parallel one. So in this paper we are proposing a modified RC4 algorithm which can be made fully parallel. This modified algorithm is compared with other cryptographic algorithms *i.e*., AES, DES and RSA for its speed and time complexity.This algorithm is also implemented in Cuda, MPI and OpenMP and the results are compared.

Keywords : Encryption, cryptography, parallelism, OpenMP, MPI, Cuda, algorithms, RC4.

1. INTRODUCTION

Nowadays, as the data access is going on increasing, we need to increase the level of security to prevent any unauthorised access. For this purpose different algorithms are used, like RC4, DES, 3-DES, AES, etc.

RC4 is a stream cipher algorithm and operates on individualbits to secure the algorithm. The efficiency of an encryption algorithmis very important. If the algorithm used is complex and time consuming then it will affect the application.

In such cases the effectiveness of algorithm can be increased by converting it from sequential to parallel execution.Nowadays with the increasing use of parallel computing this has become much easier using parallel processors in computing. The symmetric multiprocessors (SMP) which are readily available from companies such as Intel can be used in conjunction with parallel programming APIs such as OpenMP to make the algorithms parallel and faster.

In this paper we are presenting a parallel approach which is a modified version of RC4. In RC4 algorithm it was difficult to make it completely parallel. In the proposed algorithm the complete process of encryption is made parallel including the key scheduling algorithm. The main purpose of the research is to improve the efficiency of encryption which RC4 algorithm lacked, which will result in a much faster execution of the encryption algorithm.

When we compare the earlier code and the modified code sequentially, there is a possibility that the modified code is take more time but this time will be very less *i.e*., negligible. But when the original code is compared with the modified code in parallel, we can see that the original code cannot be made fully parallel and when the speed of the original and the modified code is compared, the difference will be very high. In section 3, the proposed algorithm is discussed in details and the algorithm is mentioned.

* Department of Computer Science and Engineering Amrita School of Engineering, Bengaluru Amrita Vishwa Vidyapeetham Amrita University India Email- ananthujayan@gmail.com,

** Department of Computer Science and Engineering Amrita School of Engineering, Bengaluru Amrita Vishwa Vidyapeetham Amrita University India Email- akash_nair92@rediffmail.com,

*** Department of Computer Science and Engineering Amrita School of Engineering, Bengaluru Amrita Vishwa Vidyapeetham Amrita University India Email- bhargavi. upadhyay @gmail. com,m_supriya@blr.amrita.edu

2. RELATED WORK

RC4 algorithm is implemented many times in many areas. Earlier a parallel key search engine is implemented by K.H Tsoi [2]. RC4 key is used to make a search engine. As this can be done in parallel, at the logical level. On board or on chip memories are used so as to get more memory bandwidth. As this is made parallel, the search speed is increased and it is compared with an ordinary *pc*.

RC4 algorithm can be done in two steps which are key generating and PRGA. This can be modified to make more strength by calculating the j value twice [5] [7]. This is to make the RC4 algorithm stronger. But by making this it cannot be make parallel. But it is efficient in sequential mode.

In RC4 cryptographic algorithm random key is generated in sequential manner is later converted to parallel to attain speed-up [1].The aim to make it parallel is to reduce barriers and time as it is readily used in important areas.For increasing the security of RC4 algorithm, a lot of research is been done. In the mention reference [3], a small change is done for increasing the security and speed. They have adopted a strategy in which they have modified the key scheduling algorithm *i.e*., from a single step to a double step key scheduling.

For securing data transmission over wireless LAN, research is done in [6]. In this paper they have redesigned the RC4 algorithm for getting more security. As the security may be less while transmitting data over wireless LAN, it is necessary to use more secure algorithm, so that the data should be safe.By redesigning RC4 algorithm they were able to secure the data.

In paper [7], they have modified the RC4 algorithm to increase the security, but this algorithm is effective sequentially and cannot be made fully parallel. In case of our proposed algorithm, it can be made fully parallel and hence increasing the speed of it.

3. PROPOSED ALGORITHM

RC4 generates key stream that is random stream of bits. The key stream is combined with the plaintext using bit-wise XOR to generate the encrypted text. The algorithm has two main parts: the key scheduling algorithm (KSA) and the pseudo random generation algorithm (PRGA).

for **i** from 1 to 255
S[*i*] = *i*
end for
j = 0
for *i* from 0 to 255
j = (number of threads + *s*[*i*]) % 256
swap *s*[*i*] and *s*[*j*]
end for

From the algorithm above, it can be seen that S is an array which is having numbers from 0 to 255 initially. Then *j* is calculated using the values in S array and the number of threads which is used as a key in this proposed algorithm. The S array is recalculated by swapping the values of S[*i*] and S[*j*]. This modified S array is used to cypher the input text.

To get the cypher text which is depending on the length of input text, *j* value and *i* value is updated first and then swap the values of S[*i*] and S[*j*]. The generation of pseudo-random generation algorithm is givenbelow:

i = 0
j = 0
while GenerationOutput :
i = (number of threads) %256
j = (number of threads + 1 + *s*[*i*]) % 256
swap values of *s*[*i*] and *s*[*j*]
end while

In the earlier RC4 algorithm, encryption was done using the input plain text and the key, but in this algorithm it was not able to make it fully parallel. In the proposed algorithm, the key was combined with number of threads and then it was combined with the input plain text as shown in figure 1, in this approach it is possible to make the algorithm completely parallel. And hence the algorithm is more efficient and provides more security.

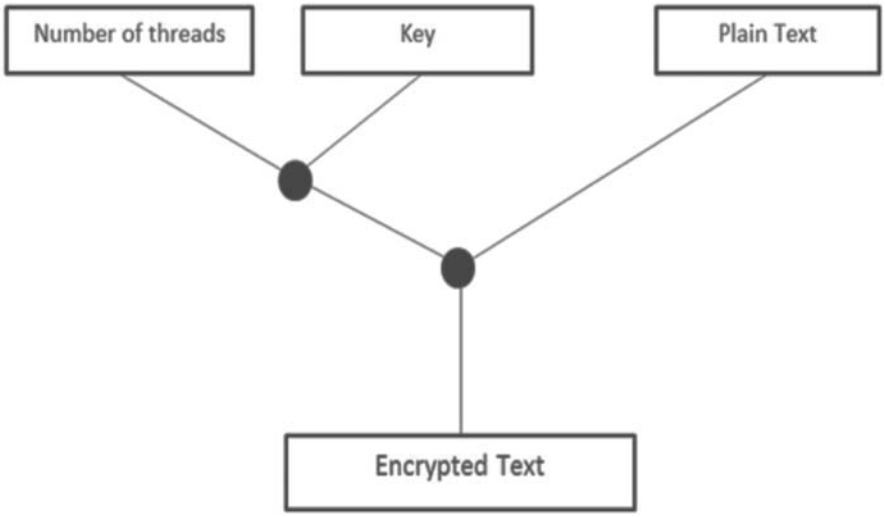

Fig. 1. Pseudo random generation algorithm

4. METHODOLOGY

For making the RC4 algorithm more efficiency it is made parallel. To make an algorithm parallel first we have to find which regions in the algorithm can be made parallel. More the number of parallel region, more is the efficiency and speed up. In this existing RC4 algorithm only the message splitting was made parallel, but the Key Scheduling Algorithm was not made parallel. In the proposed algorithm the Key Scheduling Algorithm is also made parallel due to which the new algorithm is more efficient and the security also has increased.

4.1. Parallelism in KSA

Earlier in RC4 algorithm parallelism was done but not completely i.e., the Key Scheduling Algorithm was not made parallel as the swapping of elements was there. In the proposed method, the number of threads is also used, so the algorithm is in such a way that the swapping can be done by the number of threads as the swapping will be between near by elements. Due to this, the Key Scheduling Algorithm is also made parallel and the algorithm runs more efficiently. Therefore this can be make into parallel to get speed up.

4.2. Parallelism in input text

The input text is divided into different blocks and encrypted. Then the output is joined to get the complete cipher text. This can be done in any number of cores.

5. SPEEDUP

In parallel computing speedup is a very important aspect. speedup is a ratio used to find out how well the code is running parallely. It show how a code which is running serially, can execute faster when it is executed parallely using multiple processors. Its helps in improving the parallel code.

According to Amdahl's law, in a program if the parallel component is 90% of the total computation time then the speed will be 10 times the serial execution. According to this law speedup is the ratio of time take to execute the code serially to the time taken to execute in parallel.

$$S = T(1)/T(n)$$

In this paper we first calculate the speedup for the RC4 algorithm which is parallel partially. After that the proposed algorithm is tested which is fully parallel. We get the result as "1.2". Machine configuration used can seen from Table 1.

Table 1. Test Result

File size	*Sequential*	*2 cores*	*4 cores*	*6 cores*	*8 cores*
512kb	0.01037Ins	0.01015ms	0.00366ms	0.00261ms	0.00193ms
1mb	0.01457ns	0.00664ms	0.00641ms	0.00483ms	0.00196ms
2mb	0.02749ns	0.02559ms	0.01808ms	0.01098ms	0.00807ms
3mb	0.03957ns	0.03260ms	0.02261ms	0.01389ms	0.01074ms

6. OMPARISON BETWEEN DIFFERENT ALGORITHMS

RC4 is algorithm is compared with RSA, MD5, RC2, AES and DES. While comparing this RC4 with other algorithms, the encryption and decryption time for RC4 is found to be lesser than others. For the different file sizes this comparison is checked and time for RC4 is lesser for every check.

While checking the complexity for this modified parallel RC4, other algorithms are more complex and every other algorithm cannot be fully parallelized and only a certain portion of the code can make parallelized. This is because while making it parallel it is found that different threads are getting different values.

In security based comparison RC4 is better than some of the other encryption algorithm, but it is less secure than some of the algorithms.

7. COMPARISON BETWEEN DIFFERENT LANGUAGES

Now in this modern era of parallel computing, most common languages that support parallel computing are Compute Unified Device Architecture(CUDA), Open Multi-Processing(OpenMP), Message Passing Interface(MPI) and Hybrid(MPI and OpenMP). CUDA is for high performance computation as there are many numbers of cores. OpenMP is used in a shared system, while MPI is used in distributed system.

From the results generated from the proposed algorithm,CUDA takes the least time to execute asthere are many numbers of cores. After CUDA comes OpenMP. This is because the code in OpenMP is simple. There is a hybrid version i.e., MPI + OpenMP, due to the number of communications in MPI, it will take more time. MPI takes the most time as its full of communication, even if the computation is less.

8. WORKING ENVIRONMENT

For OpenMP and MPI the simulation was done in Ubuntu15.05. The version of GCC and G++ was 4.9. MPICH 3.1 isused for MPI coding.

For CUDA programming operating system is Ubuntu14.04. The versionof GCC and G++ was 4.7. CUDA 7.5 isused for simulation. Under these working environment wehave implemented the code and the results are obtained.

9. EXPERIMENTAL RESULTS

Figure 2 is a graph of ordinary RC4 program which isexecuted in OpenMP, Cuda and MPI.We can analyse that timetaken by cuda is lesser than MPI and OpenMP.

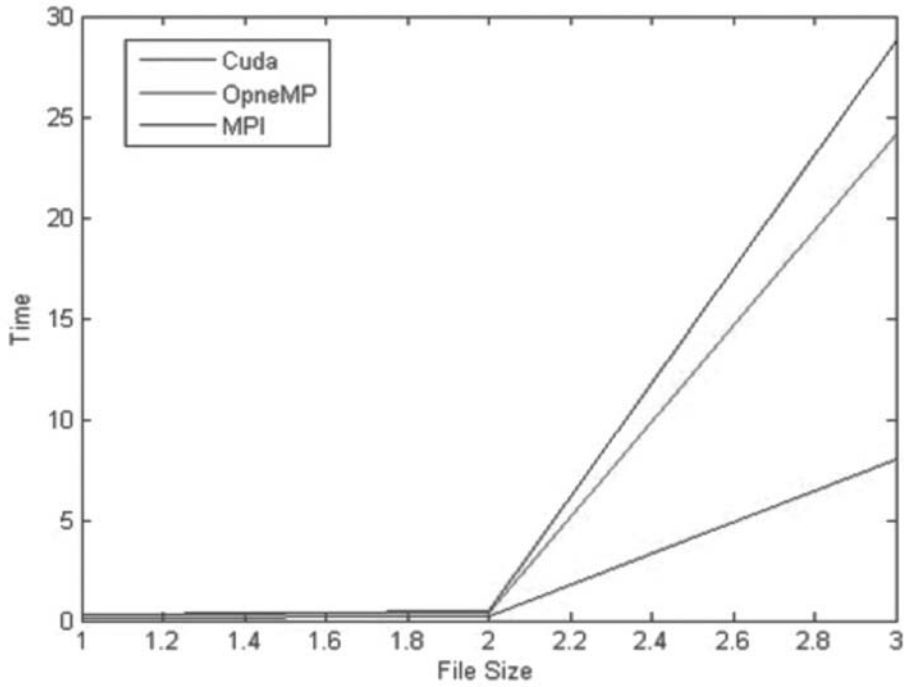

Fig. 2. Comparison between CUDA, OpenMP and MPI

Figure 3 shows the graph of execution of code parallely in OpenMP using sequential, 2cores, 4 cores, 6 cores and 8 cores.

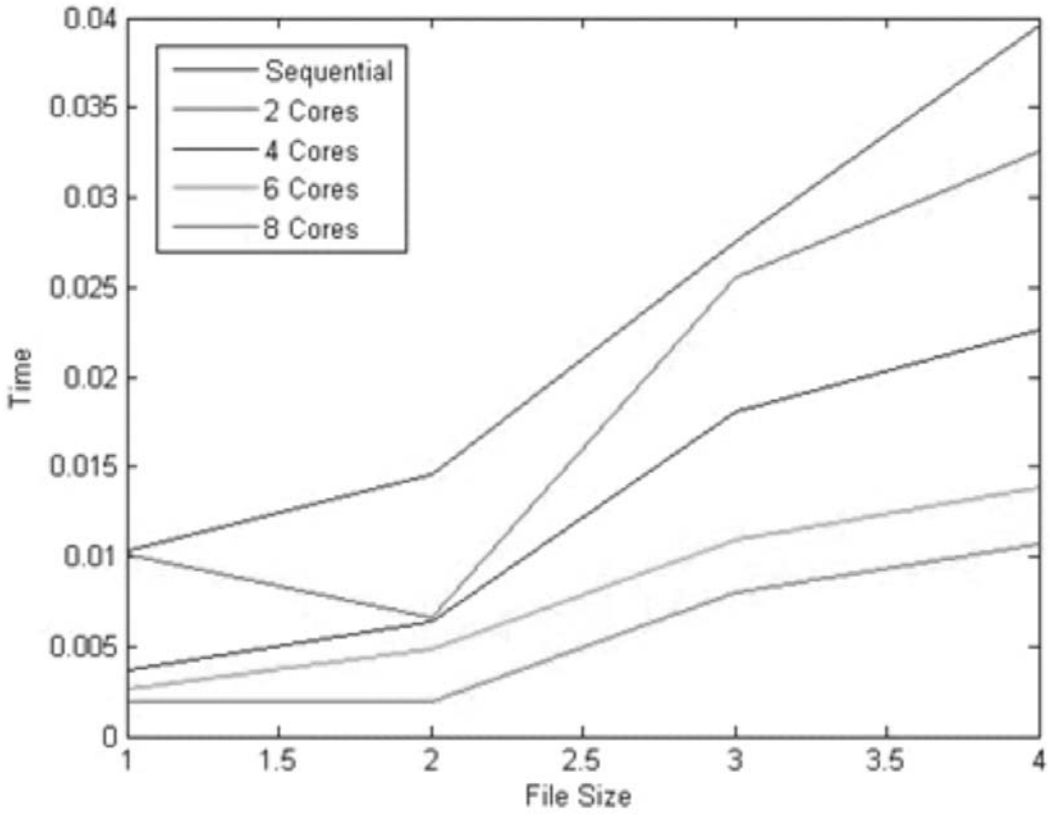

Fig. 3. Comparing with different cores in OpenMP

In figure 4 we compare CUDA, OpenMP and MPI usingsequential and multiple cores. And from the observation, CUDAtakes lesser time than OpenMP and MPI while, MPI take moretime than CUDA and OpenMP.

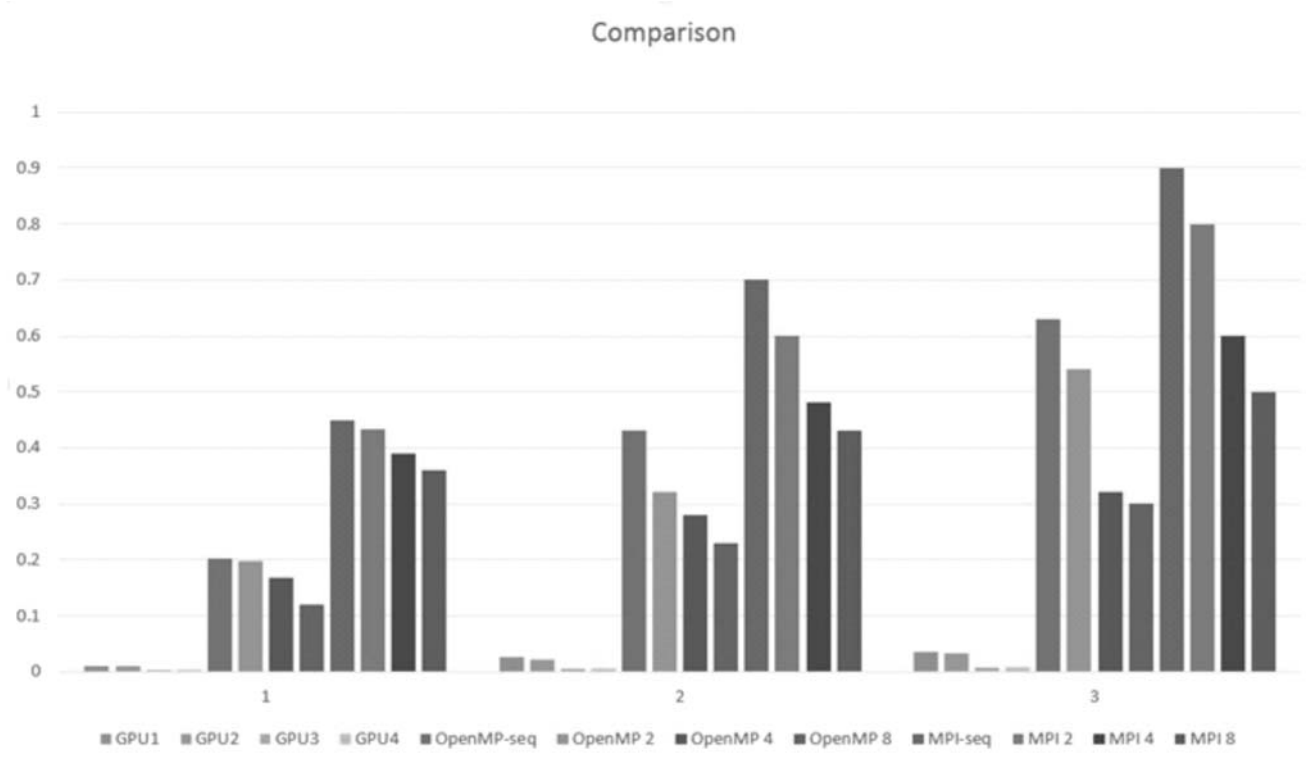

Fig. 4. Comparison between CUDA, OpenMP and MPI using different cores

In figure 5 many words are considered. But as you cansee, still the time in CUDA is lesser than execution in OpenMPusing multiple threads.

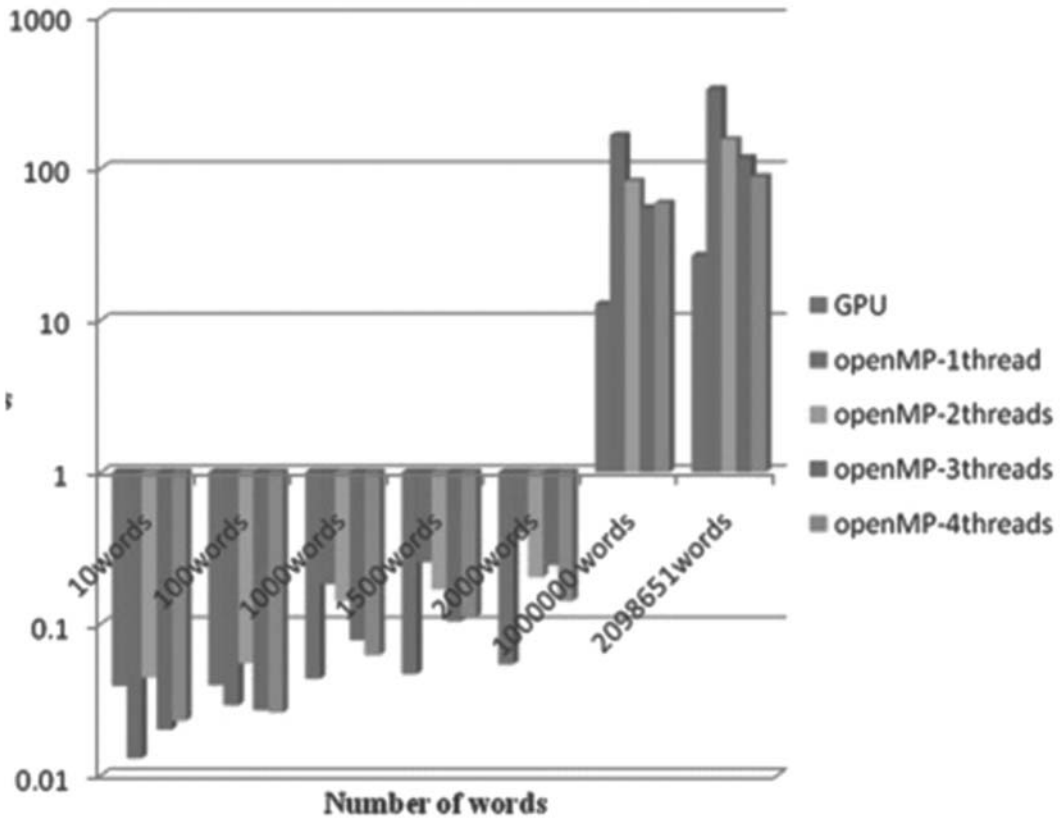

Fig. 5. Comparison between CUDA, OpenMP and MPI using different number of words

10. CONCLUSION

In this paper we have implemented an improved RC4algorithm using different parallel languages using which theentire algorithm could be made parallel.For more efficiency, we can try doing this in CUDA + MPI. As MPI can be used for move work to each node and CUDAcan be used for computations. Using this hybrid language, *i.e.*,CUDA + MPI, it is expected that we get more efficiency.

Also we can implement OpenMP along with this hybrid(CUDA + MPI) so that the calculation part for the same instruction can be divided to several threads using OpenMP.Through this, CUDA + OpenMP + MPI, we can make thealgorithm, a fully parallelized and more efficient in terms oftime complexity.

Earlier RC4 algorithm cannot be parallelized fully. Usingthe proposed algorithm, i.e. RC4 algorithm, it is made fullyparallelized. Also the time required to encrypt and decrypt isreduced. In security aspects, it can be used in varies messengerto encrypt the messages and data. The proposed algorithm isfaster and efficient than the earlier version of RC4.

RC4 algorithm is widely used nowadays. The amount of data to be dealt is enormous due tp the rise in technology. Big data analytics using Hadoop can be incorporated with this work in future.

11. REFERENCES

1. Disha Handa, Bhanu Kapoor, PARC4: High Performance Implementation of RC4 Cryptographic Algorithm using Parallelism, 2014 International Conference on Reliability, Optimization and Information Technology - ICROIT 2014, India, Feb 6-8 2014.
2. K.H.Tsoi et aI, A massively parallel RC4 key search engine(WithFPGA), not published.
3. Hisham A. Kholidy, Khaled S. Alghathbar, Adapting and accelerating the stream cipher algorithm “RC4” using “ultra gridsec” and “HIMAN” and use it to secure “HIMAN” data, journal of information assurance and security 4(2009) 474-483.
4. B. Chapman,G. Jost and Ruud Pas, Using OpenMP: Portable Shared Memory Parallel Programming, MIT Press, 2008.
5. T.D.B Weerasinghe, An Effective RC4 Stream Cipher, 2013 IEEE 8th International Conference on Information Systems, ICIIS 2013, Aug 18-20, 2013, Sri Lanka.
6. Chugh, S. Kamal, Securing data transmission over wireless LAN (802.11) by redesigning RC4 Algorithm, Green Computing and Internet of Things (ICGCIoT), 2015 International Conference.
7. Jian Xie; Xiaozhong Pan, An improved RC4 stream cipher, Computer Application and System Modeling (ICCASM), 2010 International Con-ference.
8. Jindal, P.; Singh, B., Performance analysis of modified RC4 encryption algorithm, Recent Advances and Innovations in Engineering (ICRAIE), 2014.
9. Qian Yu; Zhang, C.N.; Orumiehchiha, M.A.; Hua Li, RC4-BHF: An Improved RC4-Based Hash Function, Computer and Information Technology (CIT), 2012 IEEE 12th International Conference.
10. Das, S.; Dey, H.; Ghosh, R., An approach to assess the optimality of refining RC4, Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference.
11. Yanling Xing; Yukui Pei; Ning Ge, LT code design based on RC4 sequential cipher, Communication Technology (ICCT), 2012 IEEE 14th .
12. Gupta, S.S.; Chattopadhyay, A.; Sinha, K.; Maitra, S.; Sinha, B.P.,High-Performance Hardware Implementation for RC4 Stream Cipher, Computers, IEEE Transactions in 2013.
13. Zhang, C.N.; Qian Yu; Xun Huang; Cungang Yang, An RC4-Based Lightweight Security Protocol for Resource-constrained Communications, Computational Science and Engineering Workshops, 2008. CSEWORK-SHOPS ’08. 11th IEEE International Conference.