Performance Modeling of Scalable Encryption Algorithm using Parallel Computation

M. Nagendra
Department of Computer Science & Technology
Sri Krishnadevaraya University
Anantapuramu, India

M. Chandra Sekhar
Department of Computer Science & Technology
Sri Krishnadevaraya University
Anantapuramu, India

Abstract — **The requirement of information security on network has become more and more important. Cryptography is a method to provide information confidentiality, authenticity and integrity. There are so many challenges to implement cryptography algorithm such as execution time, memory requirement, and computation power. Parallel computation is a promising technique to improve the performance of cryptography algorithm. Mainly divide-and-conquer strategy is used in parallel computation to solve the algorithms in parallel by partitioning and allocating, number of given subtask to available processing units. Parallel computation can be performed using multicore processors by parallelizing the execution of algorithm in multiple cores. In this paper we explore the implementation of Scalable Encryption Algorithm (SEA) cryptography algorithm on dual core processor by using OpenMP API to reduce the execution time**

Keywords - Cryptography, parallel computation, dual-core processor, OpenMP, SEA

I. INTRODUCTION

Currently computer networks are becoming more important for exchanging information. One of the most important requirement of these networks is to provide secure transmission of information from one place to another. Cryptography is one of the technique which provides most secure way to transfer the sensitive information from sender to intended receiver. Its main purpose is to make sensitive information unreadable to all other except the intended receiver [2]. So scalable encryption algorithm (SEA) is one of the most important cryptography algorithm for hiding the sensitive information. But SEA algorithm has many performance limitations such as memory requirement and execution time [1].

So one of the solution to reduce the execution time of SEA algorithm is by using parallel computation. Parallel computation is a method in which several computations can be carried out simultaneously on two or more microprocessors. Parallel computation can be performed by using multicore and multiprocessor computers having multiple processing elements within a single machine. OpenMP (Open multiprocessing) is one of the application programming interface which is supported by multicore architectures to provide multithreaded shared memory parallelism. OpenMP uses fork join model for the parallel execution of code. Execution of OpenMP programs begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered and then it creates a team of parallel threads. So, statements in the program that are enclosed within the parallel region construct are then executed in parallel among the various threads. So by using multicore architectures we can parallelize the execution of SEA algorithm among different cores to reduce the execution time of the algorithm.

II. SCALABLE ENCRYPTION ALGORITHM

SEA is a block cipher in which key and plaintext sizes can be a multiplier of 6. It works on 48, 96,.. …192-bit block size using 48, 96,.. …192-bit key respectively [3], [4].

Parameters used in the algorithm

- n: plaintext size, key size.
- b: processor (or word) size
- n_b=n/2b number of words per Feistel branch.
- n_r: Total number of block cipher rounds.
- Bit Representation x_b= x((n/2)-1)……x(0).
- Word Representation $x_w = x_{nb-1}\, x_{nb-2\ldots}\, x_1\, x_0$

Basic Operation:

1. Bitwise XOR – The . Bitwise XOR can be defined on n/2-bit vectors as:

$$Z(i) = X(i) \oplus Y(i);$$

2. Substitution Box S **:** SEA uses the following 3-bit substitution table:

$$S_T=\{0, 5, 6, 7, 4, 3, 1, 2\}$$

$$X_{3i} = (X_{3i+2} \wedge X_{3i+1}) \oplus X_{3i},$$

$$X_{3i+1} = (X_{3i+2} \wedge X_{3i}) \oplus X_{3i+1},$$

$$X_{3i+2} = (X_{3i} \wedge X_{3i+1}) \oplus X_{3i+2},$$

3. Word Rotation R : The word rotation is defined on n_b-word vectors:

$$R(x) \leftrightarrow Y_{i+1} = X_i,\ Y_0 = X_{nb-i}\ ,$$
$$\text{Where } 0 \leq i \leq n_b - 2\ ,$$

4. Bit Rotation r : The Bit rotation is defined on n_b-word vectors:

$$r(x) \leftrightarrow Y_{3i} = X_{3i} >>>1$$
$$Y_{3i+1} = X_{3i+1},\ Y_{3i+2} = X_{3i+2} <<<1$$

5. Addition mod 2^b : The mod 2^b addition is defined on n_b- word vectors:

$$Z(i) = X(i) + Y(i)$$

The complete cipher

```
C= SEA (P,K)
{
  initialization:
  $L_0$ and $R_0$ =P;  $KL_0$ and $KR_0$ =K;
  key scheduling:
  for i=1 to $\lfloor n_r/2 \rfloor$
      $[KL_i,KR_i]= F_K (KL_{i-1},KR_{i-1},C(i))$;
  switch $KL_{nr/2}$, $KR_{nr/2}$ ;
  for i= $n_r/2$ to $n_r$ -1
      $[KL_i,KR_i]= F_K (KL_{i-1},KR_{i-1},C(r-i))$;
  Encryption:
  for i=1 to $n_r/2$
      $[L_i,R_i]= F_E (L_{i-1},R_{i-1},KR_{i-1})$;
  for i= $n_r/2$ +1 to $n_r$
      $[L_i,R_i]= F_E (L_{i-1},R_{i-1},KL_{i-1})$;
  Final:
      C= $R_{nr}$ &$L_{nr}$
  Switch $KL_{nr-1}$ and $KR_{nr-1}$ ;
}
```

III. PARALLEL IMPLEMENTATION OF SEA

Parallel implementation of SEA cryptography algorithm can be performed on multicore architectures by making the use of openmp API.

A. Multicore Architectures

A multicore system is a system which consists of two of more cores within a single processor. Here, core is nothing but a processing or execution unit. Multi-core architecture consists of two or more processing cores on the same chip. So, it is also referred as Chip Multiprocessor. In multi-core architecture design, each core has its own execution pipeline and each core has the resources required to run without blocking the resources needed by the other software threads.

Fig.1 shows the architecture of multi-core systems it has n number of processing cores integrated onto a single Chip. Each processing cores has its own private L1 cache and share a common L2 cache. The bandwidth between the L2 cache and main memory is shared by all the processing cores.

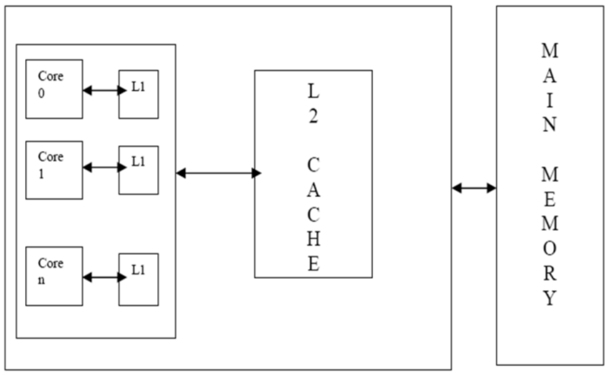

Figure 1. Architecture of multicore system

The dual core processor contains two-cores, quad-core processor contains four cores and so on. Multi-core processor supports multiprocessing into a single physical package. Different cores in a multicore system can be coupled together loosely or tightly. Some of the common network topologies to interconnect cores are ring, bus, 2-dimensional mesh, crossbar etc. Multicore system supports the concept of simultaneous multithreading. Fig.2 shows the multicore system with simultaneous multithreading. It permits several independent threads to execute simultaneously on the same core. So, in multicore systems no. of threads, can execute multiple numbers of tasks simultaneously.

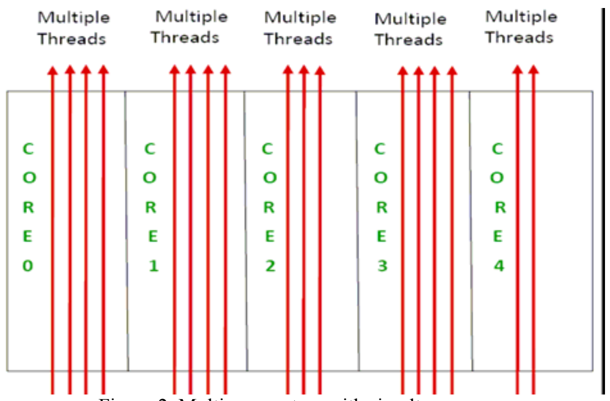

Figure 2. Multicore system with simultaneous multithreading

B. OpenMP

OpenMP (Open Multi-processing) is an application programming interface, it is jointly defined by a group of computer software and hardware vendors. OpenMP provides a scalable and portable model for developers of shared memory parallel applications. OpenMP supports programming languages C, C++ and FORTRAN on several architectures including Microsoft Windows and Unix platforms. Open MP uses fork-join model for the execution of any program or application. Fig3 shows the architecture of Fork-join model.

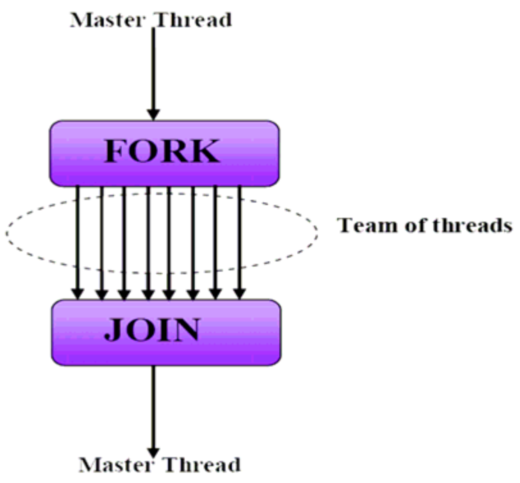

Figure 3. Fork-join model

All OpenMP program begins execution as a single thread of execution called the master thread. The master thread will executes in a single region until the first parallel construct is encountered. When a parallel construct is encountered then the master thread creates a team of parallel threads. The statements that are enclosed by the parallel region construct are then executed in parallel among the various team threads. After the execution of all the statements within the parallel region, team threads will terminate and leaving only the master thread.

OpenMP mainly comprised of three components: Compiler directives, Runtime library routines and Environment variables. All compiler directives are case sensitive. Some of the compiler directives are parallel region construct, section directive, single directive, barrier directive, master directive etc. Similarly some of the runtime library routines Omp_set_num_threads, Omp_get_num_threads, Omp_get_ thread_num ,Omp_get_num_procs , Omp_get_max_threads etc. OpenMP supports the various environment variables for controlling the execution of parallel code. Some of the environment variables are OMP_SCHEDULE, OMP_NUM_THREADS,OMP_DYNAMIC,OMP_NESTE Dand OMP_STACKSIZE.

C. *Implementation of SEA on Dual core system*

The SEA Cryptography algorithm has been implemented in dual core system by using OpenMP (Open multiprocessing) interface. Here we have parallelized the encryption and decryption process of SEA cryptography algorithm by making the use of openmp directives, between the two cores to reduce the execution time. Fig.4 shows the flow chart for parallel implementation of encryption and decryption algorithm.

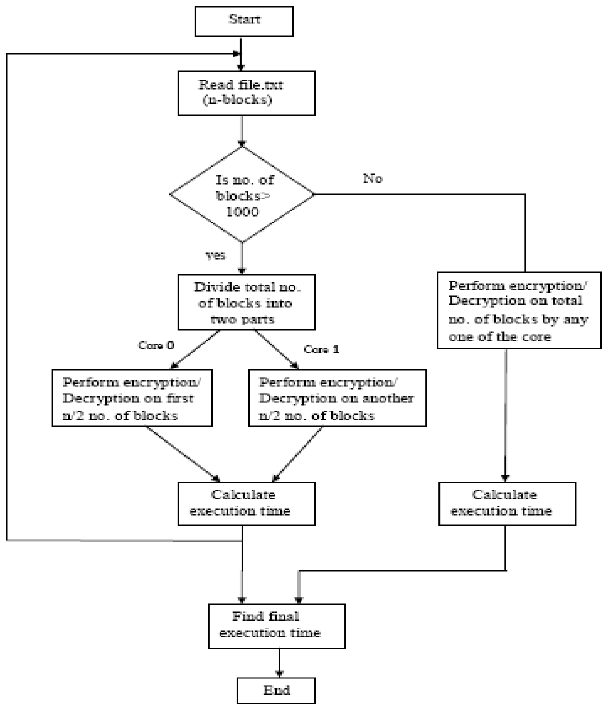
Start
Read file.txt (n-blocks)
Is no. of blocks> 1000
No
yes
Divide total no. of blocks into two parts
Core 0
Core 1
Perform encryption/ Decryption on first n/2 no. of blocks
Perform encryption/ Decryption on another n/2 no. of blocks
Perform encryption/ Decryption on total no. of blocks by any one of the core
Calculate execution time
Calculate execution time
Find final execution time
End

Figure 4. flow chart for parallel implementation

File.txt is a text file for encryption/decryption, here its reading n-block of data at a time where n should be greater number such as 1000, 2000, 3000 etc to achieve the better performance. So that first n/2 blocks can be assigned to core-0 for encryption/decryption, while another n/2 bocks can be assigned to core-1 for performing encryption/decryption. In this case we are performing encryption/decryption on multiple blocks of data simultaneously by using the concept of simultaneous multithreading some of the bocks by core-0 and some of the blocks by core-1. This process will continue till the end of the file and after the last encryption/decryption step it will give the encryption/decryption time for entire file.

IV. RESULTS AND PERFORMANCE ANALYSES

The implementation results reported in this section makes the comparison between the sequential and parallel implementation of SEA block cipher. In this paper, SEA algorithm which is implemented by using the dual-core processor has been checked for correct execution time of encryption and decryption in both sequential and parallel implementation in the environment of Visual Studio 2005, intel C++ Compiler, Windows Xp, Core 2 Duo- 2.4GHz, and 1GB RAM.

TABLE I: EXECUTION TIME OF SEA ALGORITHM FOR ENCRYPTION

Input File Size In KB	Time required for Decryption In seconds	
	Sequential Implementation	Parallel Implementation
1000	0.627134	0.317329
2000	1.255218	0.635982
4000	2.518451	1.274926
6000	3.7652195	1.905742
8000	5.020543	2.57462
10000	6.275369	3.07438

Table I shows the execution time required by different size text files for encryption process. Here we reported two types of results. First of all, we show the execution time for different input plaintext size, in sequential implementation. Afterwards, we show the result of parallel implementation for same input plaintext size. Here it can be seen that implementation using parallel processing provides good performance. The mathematical formula to calculate the reduction in execution time (in %) for parallel implementation is as follows:

$$\text{Time}= \frac{\text{time in sequential imp. - time in parallel imp.}}{\text{time in sequential implementation}} * 100$$

Fig. 5 shows graphical representation of time for encryption process. In this graph blue line shows the encryption time for sequential implementation and the red line shows the encryption time for parallel implementation. Graph shows the difference in execution time for sequential and parallel implementation for encryption process. Here we can see the performance improvement in the parallel implementation. In this it can be seen that the performance is not fixed or constant for all file sizes. Here we reported that the performance of parallel implementation for small size file is less and it will increase as the file size will increase. But it will increase till a particular value and after that it will be a constant value. In our implementation value of reduction in time starts with 49% for very small size file and after that it will increase according to the file size but the maximum value for all the large files is 55% .

TABLE II: EXECUTION TIME OF SEA ALGORITHM FOR DECRYPTION

Input File Size In KB	Time required for Encryption In seconds	
	Sequential Implementation	Parallel Implementation
1000	0.626231	0.317429
2000	1.25368	.634832
4000	2.504424	1.26487
6000	3.75176	1.904376
8000	5.03856	2.539651
10000	6.76298	3.076578

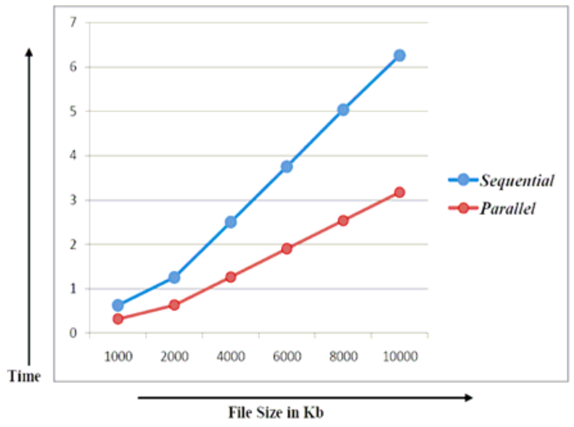

Figure 5. Execution time for encryption

Table II shows the execution time required by different size text files for decryption process. Here also we reported two types of results. First of all, we show the execution time for different input plaintext size, in sequential implementation. Afterwards, we show the result of parallel implementation for same input plaintext size. Here also it can be seen that implementation using parallel processing provides good performance. Above mentioned formulae can be used to calculate the reduction in time for parallel implementation.

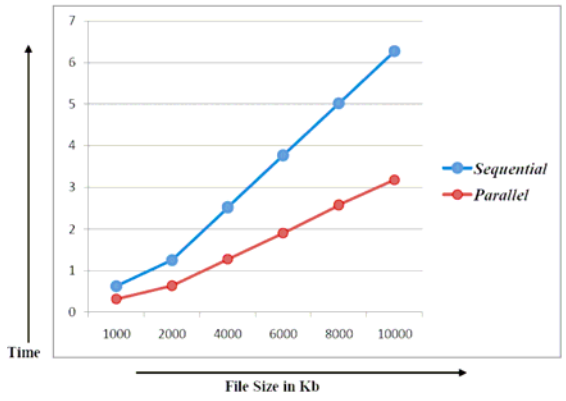

Figure 6. Execution time for decryption

Fig. 6 shows graphical representation of time for decryption process. In this graph, blue line shows the decryption time for sequential implementation and the red line shows the decryption time for parallel implementation. Graph shows the difference in execution time for sequential and parallel implementation for decryption process. Here we can see the performance improvement in the parallel implementation. In this also it can be seen that the performance is not fixed or constant for all file sizes. Here we reported that the performance of parallel implementation for small size file is less and it will increase as the file size will increase. But it will increase till a particular value and after that it will be a constant value. In our implementation value of reduction in time starts with 48% for very small size file and after that it will increase according to the file size but the maximum value for all the large files is 53% .

On the basis of results of implementation we have described algorithm to calculate the value of encryption and decryption process for both sequential and parallel implementation.

```
Calculate time ()
{
  Input: File size y in Kb
  Output: Execution time for y kb file size in seconds.
  Initialization:
     $x \leftarrow 1$, $i \leftarrow 1$, $count \leftarrow 0$;
  Repeat while y!=0
     $\delta_{i,0} \leftarrow y\%10$
     $\delta_{i,1} \leftarrow x$
     $y \leftarrow y/10$
     $x \leftarrow x*10$
     $count \leftarrow count+1$
     $i \leftarrow i+1$
  end while
  $ET = c.[(\delta_{2,0} * \delta_{2,1})+\delta_{1,0}]$     for $1 \leq y \leq 99$     (1)
  $ET = \sum_{n=3}^{count}\left(\sum_{z=1}^{\delta n,0} c.\delta n,1\right) + c.[(\delta_{2,0} * \delta_{2,1})+\delta_{1,0}]$     (2)
                                          Where $y \geq 100$
}
```

The following algorithm takes file size of y kb as a input and it will calculate the execution time as the output. Here $\delta_{i,0}$ is a variable which stores the digits of input y, for e.g. value of y is 45366 then the value of $\delta_{i,0}$'s will be $\delta_{1,0} =6$, $\delta_{2,0} =6$, $\delta_{3,0} =3$, $\delta_{4,0} =5$, $\delta_{6,0} =4$. $\delta_{i,1}$ is a variable which stores the numbers which are multiples of 10's. For $\delta_{1,1}$ it will store 1, similarly $\delta_{2,1}=10$, $\delta_{3,1}=100$, $\delta_{4,1} =1000$, $\delta_{5,1} =10000$ and so on. Count is a variable which counts the total number of digits in the input file size. For 45366 kb file size value of count is 5. After this equation (1) is given to calculate the execution time if the file size is less than or equal to 99 kb. While equation (2) can be used to calculate the execution time for a file if the file size is greater than or equal to 100 kb.

In both the equations value of c is constant and it depends upon the processor speed. The value of c varies from processor to processor. Here we have given the value of c for Core 2 Duo- 2.4GHz processor. From the implementation we have reported that value of c for encryption process in sequential implementation is $6.26113*10^{-04}$ and for parallel implementation is $3.17402*10^{-04}$. Similarly value of c for decryption process in sequential implementation is $6.27532*10^{-04}$ and for parallel implementation is $3.17627*10^{-04}$.

Here results of parallel implementation shows that the proposed system for implementation of SEA cryptography algorithms using dual core architecture by using OpenMP application programming interface has been reduced the execution time for encryption and decryption processes. So by using parallel processing technique, we can improve the performance of the system.

V. CONCLUSION

In this paper, we have described concept of parallel programming by using multi-core processor. We have shown how to efficiently and effectively implement the SEA cryptography algorithm by using multicore systems and openmp API, extracting as much parallelism as possible from the algorithm in parallel implementation approach. We provided an extensive quantitative evaluation of execution time for both sequential and parallel implementation. After evaluation of execution time, we reported that parallel implementation of SEA block cipher using dual-core (Intel Core 2 Duo) processor takes 50-53% less time for performing the encryption and decryption than the sequential implementation. These experiments allow us to confirm that, for the SEA block cipher and similar algorithms, it is possible to efficiently use the multi-core processors for parallel implementation. Overall, we can conclude that multi-core processors provide an efficient and reliable way to implement SEA cryptography algorithm.

REFERENCES

[1] Wei Liu; Rong Luo; Huazhong Yang; "*Cryptography Overhead Evaluation and Analysis for Wireless Sensor*", Networks Communications and Mobile Computing, 2009. WRI International Conference on Volume 3, Page(s):496 – 501, 6-8 Jan.2009.

[2] Wei Liu; Beihua Ying;Huazhong Yang; Hui Wang; "*Accurate modeling for predicting cryptography overheads on awireless sensor nodes*", Advanced Communication Technology. ICACT 2009. 11th International Conference Volume 02, Page(s):997 – 1001, 15-18 Feb. 2009.

[3] F. Standaert , F.-X.; Quisquater, J.-J., "*FPGA Implementation(s) of a Scalable Encryption Algorithm*",Very Large Scale Integration (VLSI) Systems, IEEE Transactions on Volume 16, Issue 2, Page(s):212 – 216, Feb. 2008.

[4] Bandirmali, N.; Erturk, I.; Ceken, C., "*Securing Data Transfer in Delay-sensitive and Energy-aware WSNs Using the Scalable Encryption Algorithm*" Wireless Pervasive Computing, 2009. ISWPC 2009. 4th International Symposium on 11-13 Feb. 2009 Page(s):1 – 6.

[5] Gil-Ho Kim; Jong-Nam Kim; Gyeong-Yeon Cho; "*An improved RC6 algorithm with the same structure of encryption and*

decryption", Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference , Page(s): 1211 – 1215.

[6] Ahmed, H.E.H.; Kalash, H.M.; Allah, O.S.F.; "*Encryption Efficiency Analysis and Security Evaluation of RC6 Block Cipher for Digital Images*", Electrical Engineering, 2007. ICEE '07. International Conference, Page(s): 1 – 7.

[7] Sklavos, N.; Koufopavlou, O.; "*Asynchronous low power VLSI implementation of the International Data Encryption Algorithm*",Electronics, Circuits and Systems, 2001. ICECS 2001. The 8th IEEE International Conference, Page(s): 1425 - 1428 vol.3.

[8] Granado, J.M.; Vega, M.A.; Sanchez, J.M.; Gomez, J.A.; "*Implementing the IDEA Cryptographic Algorithm in Virtex-E and Virtex-II FPGAs*", Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean , Page(s): 109 – 112.

[9] Suying Yang; Hongyan Piao; Li Zhang; Xiaobing Zheng; "*An Improved IDEA Algorithm Based on USB Security Key*", Natural Computation, 2007. ICNC 2007. Third International Conference, Page(s): 184 – 188.

[10] Deshpande, A.M.; Deshpande, M.S.; Kayatanavar, D.N.; "*FPGA implementation of AES encryption and decryption*", Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009 International Conference, Page(s): 1 – 6.