Data Encryption Standard Algorithm Using Java Remote Method Invocation & OpenMP

Lakshay Nanda, SCOPE, VIT, Vellore. E-mail: lakshaynanda@ieee.org

Keshika Tank, SCOPE, VIT, Vellore. E-mail: keshika.tank2016@vitstudent.ac.in

V. Santhi, SCOPE, VIT, Vellore. E-mail: vsanthi@vit.ac.in

Abstract--- Cryptography is a field of study of mathematical algorithms which are related to specific aspects of information security such as confidentiality, integrity of content, authentication (entity wise), and data origin authentication. Not only is encryption important but the speed at which the encryption occurs is also a matter of major concern. Java RMI is used for the purpose of parallel encryption of text. The same can also be implemented using OpenMP.

Keywords--- DES, OpenMP, RMI, Parallel, Speed.

I. Introduction

The Data Encryption Standard (DES) [3] is a symmetric cipher introduced and found by the National Institute of Standards and Technology (NIST). DES is an execution of Feistel Cipher. It utilises 16 rounds Feistel Structure. The block size is 64-bit.

However, key length is 64-bit, DES has ultimately a key length of 56 bits, since 8 of the 64 bits of the key are not utlised by the encryption calculation (work as check bits only). Most cryptography algorithms work better in hardware than software, systems implemented on hardware have significant drawbacks: they fail to give responses to flaws discovered in the algorithm implemented or to the changes in standards. Alternatively, it is feasible to implement cryptographic algorithms in software executing on multiple processors. In this project we have implemented DES algorithm in serial and parallel using RMI[4] and OpenMP. RMI is the stepping stone of distributed object-oriented programming in Java.[6] It explains how Java components can interact with one another in a multiple JVM environment.

RMI provides the required tools that are needed to create distributed Java applications (objects) that can invoke methods on other Java applications which are running on other Java virtual machines, such as remote hosts across the Internet.[5] OpenMP is the short for Open Multi-Processing. It is used for parallelization of program on shared memory systems.

This project implements DES algorithm and compares the time of execution, serially and parallelly using two platforms individually RMI and OpenMP.

II. Literature Review

A. Cloud Computing Security Model with Combination of Data Encryption Standard Algorithm (DES) and Least Significant Bit (LSB).

In this paper, how cloud computing security model used in file and message storage processes requires the DES cryptography algorithm and LSB steganography in the 16th round of bits to keep up the security and confidentiality of files and messages is explained.

If the key does not match then the file will not be downloadable. But it concluded in a way that cloud are more prone to security issues hence need more rigid encryption algorithm compared to DES.

B. Implementation Cryptography Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES) Method in Communication System Based Near Field Communication (NFC)

DES and 3DES text data cryptographic method can be implemented on application of ACOS3 smart card data writing process and data reading process in NFC – based systems.

Received: 11 Mar 2019/Accepted: 05 Apr 2019

The execution time of ACOS3 smart card data writing process and data reading process using DES cryptographic method is faster than DES cryptographic method.

C. Design and Simulation DES Algorithm of Encryption for Information Security

Algorithm consumes least encryption time and DES consumes maximum encryption time.

D. Parallelisation of DES algorithm

Using the memory with more latency (for example, using the interleaving technique) we can even increase the speed-up of the whole parallel program.

The hardware synthesis of the DES algorithm will depend on appropriate adjustment of the data transmission capacity and the computational power of hardware. The code containing, I/O functions is not parallelizable, because the access to memory is, by its very nature, sequential. The total speed-up received on PC computer is about 1.86.

III. Existing DES Approach

Considering improvement in performance of advanced encryption standard using Parallel Computing, here the problem faced was that faster frameworks could also have been used for implementation. Also, this method is vulnerable to network traffic, which can affect its performance.

While another pre-existing method is that of Parallelization of the Data Encryption Standard Algorithm. Here, memory with more latency is made use of so that there is an effective increase in the speed-up of the entire program (parallel) except the I/O function region because of its sequential nature.

Another method talks about how if large amount of data is input then the encryption/decryption time required is greatly reduced, provided the processing is parallel execution under distributed environment.[1]

Yet further a study talks about how application of all standard algorithms (RSA, DES, 3DES, AES) [2] when applied to a piece of information then the security parameter is enhanced by greater percent.

For applications like banking system, the DES algorithms effectiveness is being questioned due to the result of certain analytical results which demonstrate the weakness in the cipher.[7]

A Modified Simplified Data Encryption Standard Algorithm, a random key generation was done so that the attacker finds it difficult to guess the key and hence the vulnerability of communication data to be transmitted is reduced.

A. RMI Implementation

DES module: For the core DES cipher functionality. It will be executed in Client and server nodes. Key gen function will be included to generate random symmetric keys. Key extract function will be included to extract the symmetric keys. Encryption function and decryption function will be included in the module as well.

Decrypter module: Abstract class illustrating functionalities of the application. The DES module of decrypt and encrypt will be called remotely - (thus this module would need the use of library java.rmi.*). This classrom both Client and Server module.

Decrypter Interfaced (remotely) Implementation of the Decrypter interface remotely by both client and server modules.

Server Module (RMI): It will act as the server instance which will provide Decrypter Interfaced (remotely) objects to its clients. This module/class will be executed by the server node only.

Client Module (RMI): Simply a client, accessing the application via the objects available to it by the server. This module/class will be executed by the client node only.

Received: 11 Mar 2019/Accepted: 05 Apr 2019

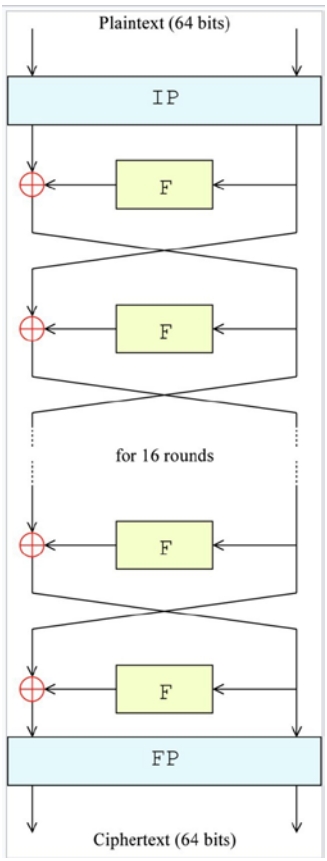


Fig. 1: The overall fiestal structure of DES

B. OpenMP Implementation

The simple serial code will be parallelised using OpenMP constructs wherever possible. omp get wtime construct will be used to get value of time elapsed in the execution.

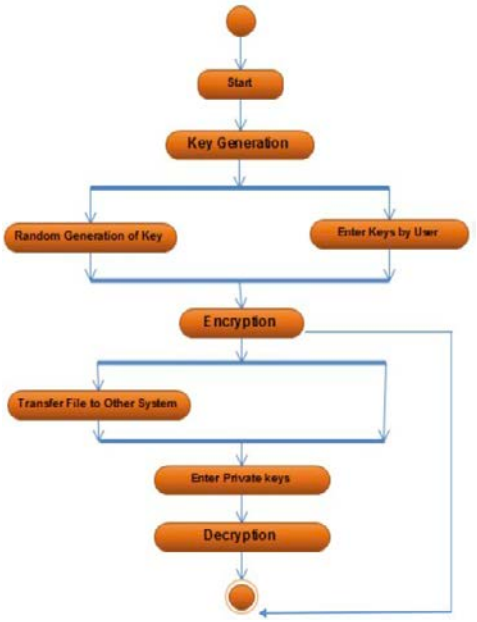


Fig. 2: Activity Diagram for OpenMP

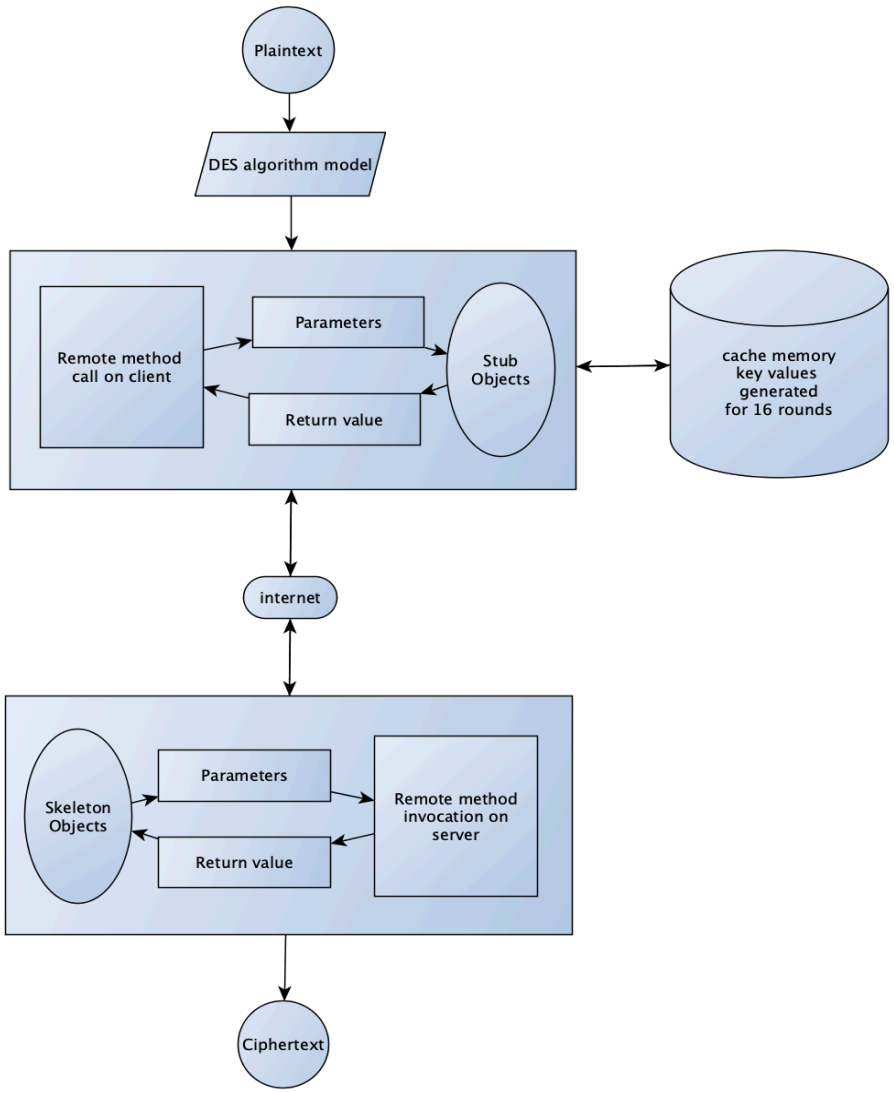
Received: 11 Mar 2019/Accepted: 05 Apr 2019

IV. Modified Approach

Our model uses two ways to implement DES algorithm, first method being using Java RMI to implement the algorithm parallelly in a client-server structure to reduce the execution time. Using Java RMI model to encrypt the DES algorithm is not restricted by the number of cores in a processor, unlike traditional parallel techniques which are bound by the number of processing elements, because Java RMI is an API that executes model in client and server stub and skeleton respectively which are created virtually by JVM. Furthermore, the time taken to run the model on a set of text data to encrypt them reduces as we run them later compared to the previous time they were executed. Hence, the entire code does not get executed instead only the encryption part is executed again.

The second way we implement DES is by using OpenMP. OpenMP is a traditional shared memory multiprocessing programming API that can be used on any platform. The run-time after executing any model in OpenMP depends on environment variables, compilers and library routines. In this paper we compare and contrast both the methods based on the time complexities.

Algorithm



Received: 11 Mar 2019/Accepted: 05 Apr 2019

V. Implementation and Result Analysis

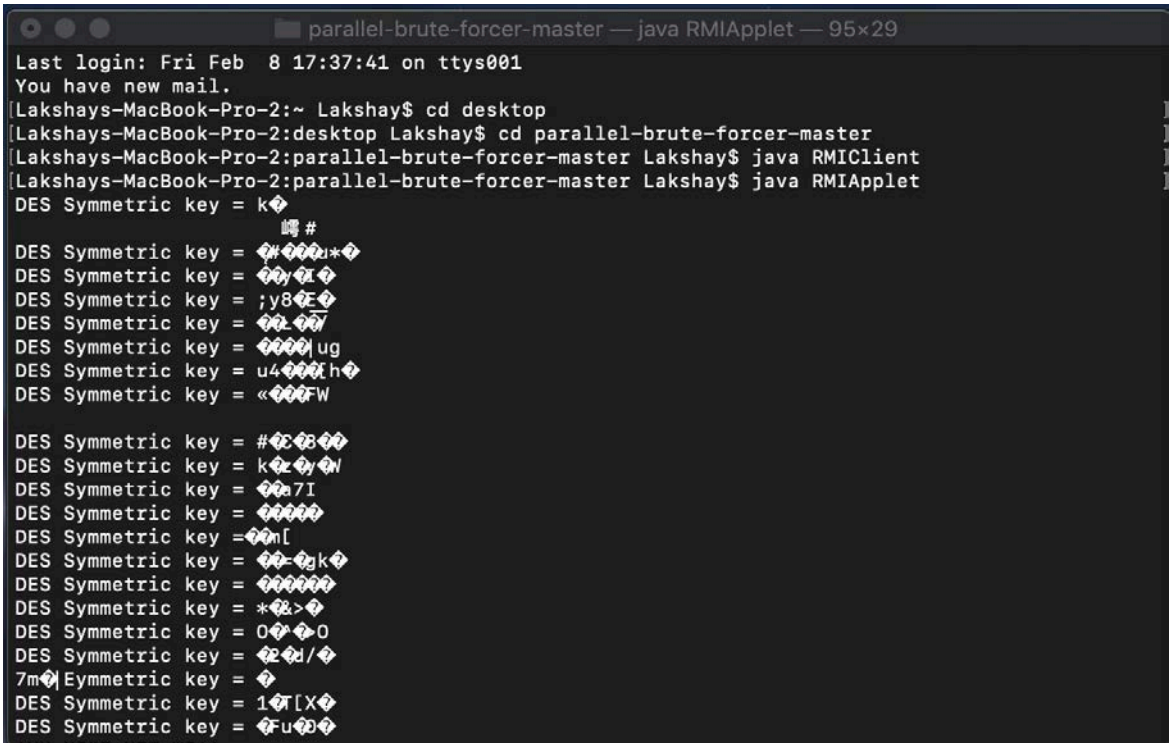


Fig. 3: RMI Client terminal

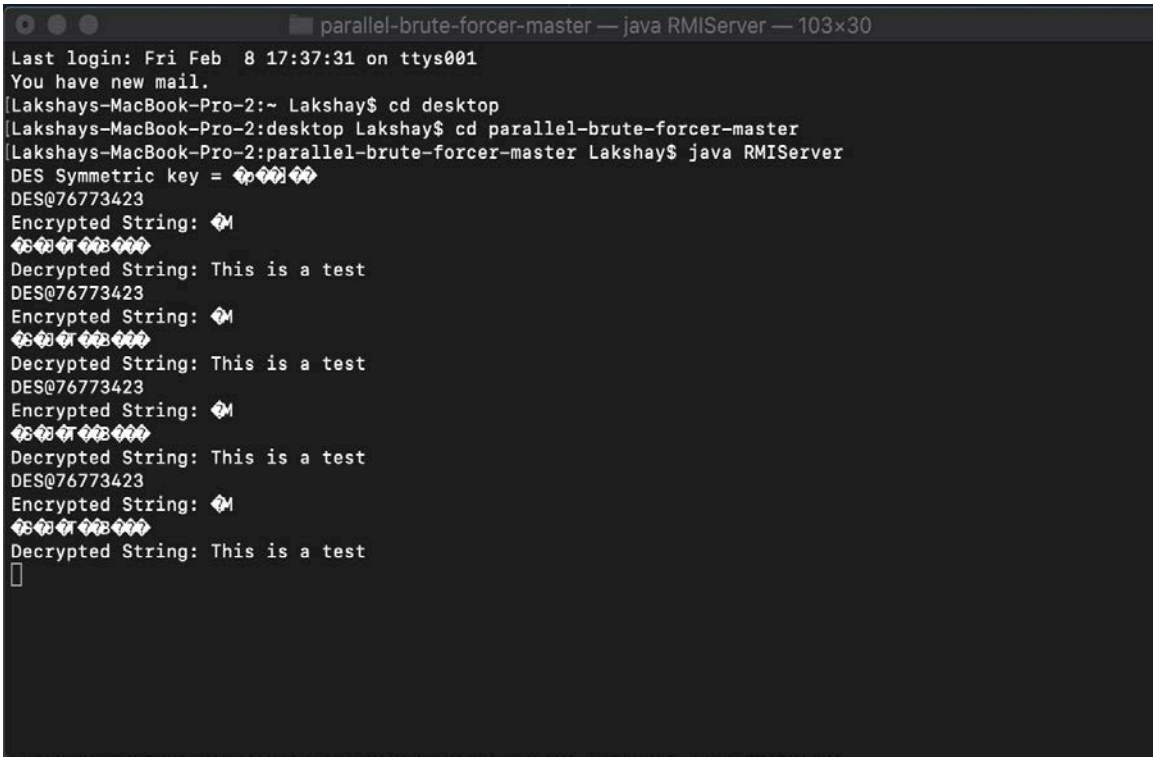
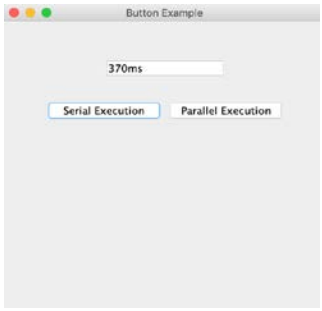


Fig. 4: RMI Server terminal

Received: 11 Mar 2019/Accepted: 05 Apr 2019

Serial Execution



Parallel Execution

Fig. 5: Java GUI for run-time comparison

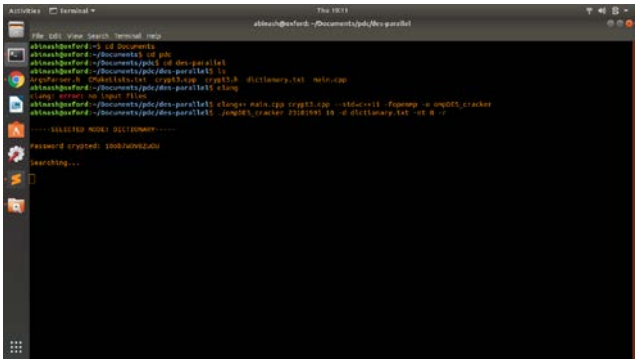


Fig. 6: OpenMP execution and runtime display

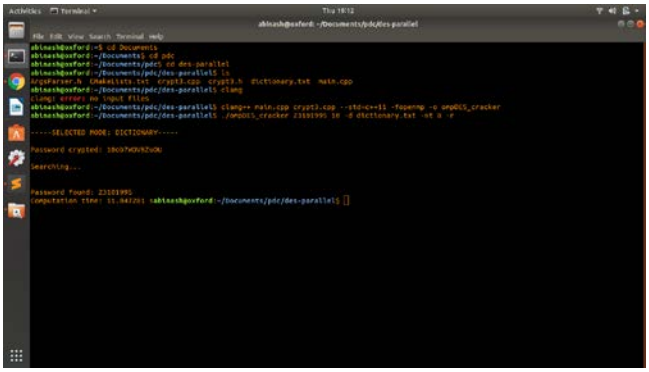


Fig. 7: Run Time Analysis of DES in Serial and Parallel

Received: 11 Mar 2019/Accepted: 05 Apr 2019

Number of times executed	serial run-time (in ms)	parallel run-time (in ms)
1	1377	650
2	1012	9
3	427	2
4	461	12
5	386	2
6	403	6
7	401	8
8	495	5
9	384	10
10	465	16

Fig. 8: Run Time Analysis of DES in Serial and Parallel using java RMI

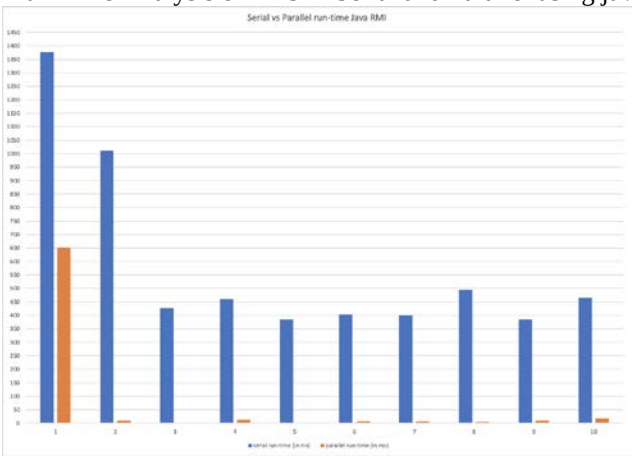


Fig. 9: Time comparison amongst the number of times the model is executed

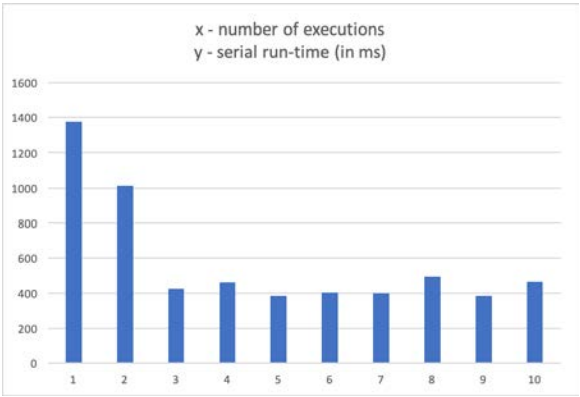


Fig. 10: Comparison of serial parallel-using java RMI and parallel-using OpenMP

Serial	**Parallel**	
	Java RMI	***OpenMP***
1012	11	9

VI. Performance Evaluation

Parallel execution time is less than serial execution time by almost half in RMI parallelization which is a significant difference in time. But the difference reduces further if the encryption happens in the second cycle because after each cycle the amount of code to be executed reduces as the basic code's output is stored in cache memory and only the encryption part of the code is executed in further cycles. Thus, reducing time of execution significantly.

Received: 11 Mar 2019/Accepted: 05 Apr 2019

VII. Conclusion

DES Algorithm is computationally expensive and when implemented in a single processor environment takes a few seconds to encrypt even a small phrase like This is a test phrase. When implemented in a client and server based scenario using Remote Method Invocation (RMI), time taken for encryption is nearly ten times faster. This not only shows the computational power that RMI gives to a programmer but also justifies the motive of the study of parallelising the code for faster results.

References

[1] Pallavi S. Shendekar; Vijay S. Gulhane. Task Parallelism using Distributed Computing for Encryption and Decryption. International Journal of Scientific and Research Publications, Volume 4, Issue 6, June 2014

[2] Gurpreet Singh; Supriya. A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security. International Journal of Computer Applications (0975 8887) Volume 67 No.19, April 2013

[3] Mohammed A. Hameed; Ahmed I. Jaber; dr Jamhoor M.Alobaidy; Alaa A. Hajer. Design and Simulation DES Algorithm of Encryption for Information Security. American Journal of Engineering Research (AJER), Volume-7, Issue-4, pp-13-22

[4] Alt M., Gorlatch S. (2003) Future-Based RMI: Optimizing Compositions of Remote Method Calls on the Grid. In: Kosch H., Bszrmnyi L., Hellwagner H. (eds) Euro-Par 2003 Parallel Processing. Euro-Par 2003. Lecture Notes in Computer Science, vol 2790. Springer, Berlin, Heidelberg

[5] Poo D., Kiong D., Ashok S. (2008) Object Serialization and Remote Method Invocation. In: Object-Oriented Programming and Java. Springer, London

[6] Hunt J., Loftus C. (2003) Java and Remote Method Invocation. In: Guide to J2EE: Enterprise Java. Springer Professional Computing. Springer, London

[7] Nirmaljeet Kaur; Sukhman Sodhi. Data Encryption Standard Algorithm (DES) for Secure Data Transmission. International Journal of Computer Applications (0975 8887)

Received: 11 Mar 2019/Accepted: 05 Apr 2019