Available online at www.pelagiaresearchlibrary.com

Pelagia Research Library

Advances in Applied Science Research, 2011, 2 (3): 567-573

Pelagia Research Library
ISSN: 0976-8610
CODEN (USA): AASRFC

Performance Analysis of Montgomery Multiplication Algorithm for Multi-core Systems Using Concurrent Java

Lavanya P and M Rajashekhara Babu

School of Computing Science and Engineering, VIT University Vellore, TamilNadu, India

---

ABSTRACT

The Requirement of information security on topology network has become more important. Cryptography is a method to provide information confidentiality, authenticity and integrity. There are so many challenges to implement a public key cryptography algorithm such as execution time, time consuming, time cost, integrated methods, memory Requirement in public key cryptography. In parallel computation to solve the formal algorithm in parallel by partitioning method can scale over different number of cores given .A parallel computation can be stable perform analysis of public key cryptography using concurrent java core communication such as available time, request time, allowance time . In this paper, public key cryptography algorithm can be implemented a parallel RSA with threads and to improve the performance of the algorithm by execution time. Various public key cryptography algorithms can be implemented using execution time.

Key Words: Cryptography, concurrent Java, RSA, DSA, ECC.

INTRODUCTION

One of the most important requirements of these networks is to provide secure transmission of information from one place to another. One of the techniques ensuring privacy of files and communications is Cryptography. Cryptography is the sciences of writing in secret code and is an ancient art; with applications ranging from war time battle plans. Any application to application communication there is some specific security requirements including: Authentication, Privacy/Confidentiality, and Integrity, on-repudiation.

In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into cipher text, which will in turn be decrypted into usable plaintext.

Modular multiplication is a fundamental operation in many popular Public Key Cryptography algorithms such as RSA and ECC. As the division operation in modular reduction is time-consuming, Montgomery proposed a new algorithm where division is avoided.

An integer X is represented as X · R mod M, where M is the modulo and R = 2r is a radix which is cop rime to M. This representation is called Montgomery residue. Multiplication is performed in this residue, and division by M is replaced with division by R.

Most public key algorithms are based on modular arithmetic, e.g. RSA, DSA, and ECC. Public key encryption and decryption are computationally heavy because a lot of modular multiplications with very large numbers are needed to perform these tasks. The security of the RSA cryptosystem is based on two mathematical problems: the problem of factoring large numbers and the RSA problem. In cryptography, the RSA problem summarizes the task of performing an RSA private-key operation given only the public key. Full decryption of an RSA cipher text is thought to be infeasible on the assumption that both of these problems are hard, i.e., no efficient algorithm exists for solving them1. Providing security against partial decryption may require the addition of a secure padding.

In this paper, we investigate how to parallelize the public key cryptography. In a practice, the world length of such key sizes larger than the text .In a RSA, ECC, DSA algorithm we use only encrypt and decrypt a text.Several implementation of a sequential have been presented before, through we believe that none of these meet all requirements we enumerated above.

We have to build in threads in java i.e. executor time, process time, synchronization, block queue. This paper is the extension of the previous work [1], Compared to the earlier work, the paper provides additional motivation on the execution time used for Asymmetric algorithms, it provides a analysis and demonstrate in designed.

II Related work

Types of Asymmetric algorithms (public key Cryptography) are available for hiding the information are RSA algorithm, Elliptic Curve Cryptography, Digital Signature Algorithm, ElGamal , ECDSA, Diffie-Hellman .These algorithms have many performance limitation such as memory requirement, time consuming, time cost and integrated methods. All of the above cryptographic algorithms perform arithmetic operations like byte, bit, modular multiplication, addition, subtraction. All these algorithms require more execution time. The efficient implementation of this long-word length modular multiplication is crucial for the performance of public-key cryptography. [1]

Modular multiplication and RSA From [2] we can find that the work of modular arithmetic is very old. Original works on modular arithmetic are very old. The Chinese Remainder Theorem was first proposed around the fifth century by Sun Tsu [3] But the use of this arithmetic to represent numbers was introduced only in 1959 by H.L. Garner [4].

Many others have built on this algorithm to produce a smaller and faster way of doing modular multiplication. A Residue Number System (RNS) represents a large integer using a set of smaller

integers, so that computation may be performed more efficiently. It relies on the Chinese remainder theorem of modular arithmetic.

Thus, the overall the computational time is to be investigated in detail. A single chip, 1024-bit RSA implementation is shown in. The multiplication part is implemented as an array multiplier. It is noted that this approach for multiplication requires multiple clock cycles to complete. An implementation of a 12x12 bits modular multiplier based on Montgomery multiplication algorithm is presented in the time and area results presented in this thesis work will be compared with the results presented. However, the comparison can be only an approximation of the technologies, architectures and bit sizes used in both case.

These algorithm have many performance limitations such as memory requirement , execution time and consumption power. Memory requirement for some of the algorithms are as follows in bits.

Table 1 Code size for different algorithms

Symmetric key size in bits	RSA/DSA key size of n in bits	ECC size of n in bits
56	112	512
80	160	1024
112	224	2048
128	256	3072
92	384	7680

One of the solutions to reduce the execution time for the time for these cryptography algorithms is by using parallel computation. It is a method in which servile computation can be carried out microprocessors. Concurrent Java is a application programming interface which provides various packages and library routines for parallel implementation. So by making use of Concurrent java we can parallelize the execution of any specified cryptography algorithm in order to reduce the time execution time of the algorithm. [8]

This paper aims at proving the performance of Multi-core which gives better scalability, performance than the single core.

The paper is organized in 5 sections. Section 3 gives the Design and Implementation details. Section 4 discusses the Results obtained.

III .Implementation

The important part of the project is the implementation where the design gets converted into corresponding coding. Screenshots will clearly bring a picture about how to parallelize execution of program between cores to improve the execution time of the code.

In the RSA algorithm are modular exponentiation, and arithmetic operations .DSA algorithm signature must be a bit pattern and it should depend on the message begin signed.ECC is the difference in the way to grouped both the numbers in the set and the arithmetic operations used more rapid increase in security a key length increases. This algorithm is directly supported by JAVA. Java application programming interface provide various directives, runtime routines and

environment variables which provide capability to parallelize a serial program. So by using Java Application we can parallelize execution of program to improve the execution time of the code. In an algorithm implementation we can use generate key, encrypt and decrypt a text message. In input message size will given in different KB.

The implementation language the algorithms should not matter in theory. In practice does a matter. That is because some algorithms are more compilers friendly. This means that some algorithms optimized during compilations time, others do not. Therefore those with that optimization will be faster and we should consider this during our algorithm study. In a implementation we use sequential and parallel implementation.

3.1 Sequential Schemes

In conventional cryptography, a sender and a recipient insecure communication share the sane key. The sender uses the key to encrypt a pieace of information into an illegible form and transmit the cipher text into the over public network .Only the recipient, who shares the same key is able to read the message by decrypt the cipher text.[7]

Algorithm 1 Rivest Adi Shamir Aldeman

Require: An p ,q are the two prime numbers (Private , chosen) with n =p.q (public , calculated) with e is chosen s.t gcd(0(n),e) = 1; 1< e< (n) are (public, chosen).d=e^1 mod (n)

1. Generate a key-pair expansion function for both encryption and decryption algorithm has been test for 1024 bit key size.
2. Store string NumDigits to prevent race conditions.
3. Retrieve the stored strNumDigits and call the original method. Processing is now done in the background.
4. Encrypt function has been checked for various sizes, this function generate properly encrypted.

Encrypt <input filename> <public key= e> <public key = n>

5. Decrypt function has been checked for encrypt, this function generate the original text after the decryption.

Decrypt <ciphertext filename> <private key = d> <public key = n>

6. The Sequential and parallel implementation Time required for RSA algorithm process has been checked its generating correct results.

After profiling algorithm 1, we implement the DSA Algorithm similar to RSA algorithm. In DSA algorithm we can use the message authentication protocol protect the two parties who exchange the message. But it does not protect each other. The digital signature can be implemented by encrypt the message with sender's private key , and encrypt the hash code of the message with the sender's private key.It can be encrpt the message + signature with the receiver's public key.Disadvantages: If the sender wants to deny sending a particular message the sender its private key was lost. And $\leq$ T.

ECC is the difference in the way to grouped both the numbers in the set and the arithmetic operations used more rapid increase in security a key length increases.[8] In ECC algorithm we can use a key mode ,parameters, i.e x ,y size. It will use Boolean operations [10]

In [7] the author analyzed several sequential implementation schemes of asymmetric key. The differences among those methods comes from the sequence to calculate the execution time.

3.2 Parallel Schemes

In this paper I describe an efficient implementation of public key cryptography algorithms for a highly parallel to concurrency Java.

The Public-key cryptography are implemented with parallel as less execution time .A fully decryption therefore requires about 15000 cycles with 100 MHz clock. So these algorithms can be implemented using java language .Concurrent Java application programming interface provides various compiler directives ,runtime routines, executor time , processing time, blocking queue and environment variables which provide capability to parallelize a serial program. Its same as sequential implementation but in parallelize its show less execution time.

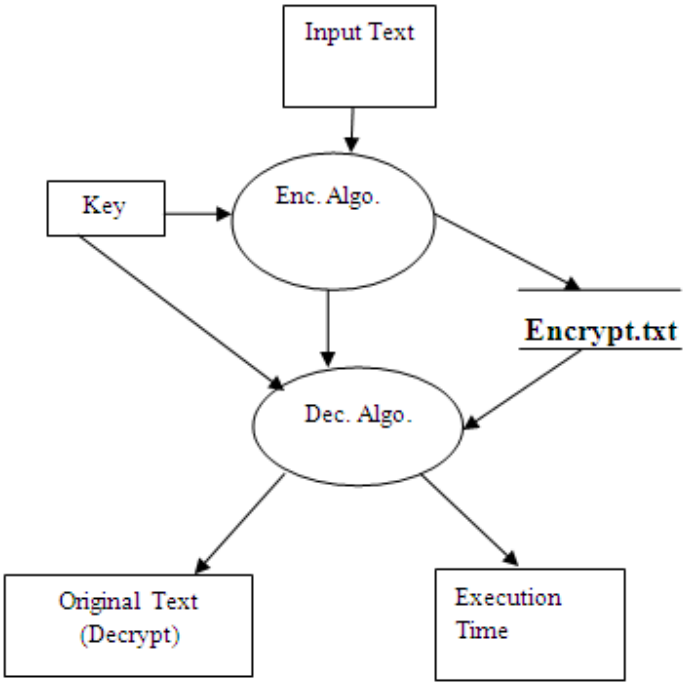

Figure 4.1 Data flow diagram

1V Analysis And Design

These sections analyze based on overall performances of RSA, DSA, ECC. In our practical work, we have measured the time consumption during the encryption and decryption .In this particular study we had to less repetitions because the algorithm spends much more time during the encryption / decryption. The main problem with DSA is the fixed subgroup size (the order of the generator element), which limits the security to around only 80 bits. Hardware attacks can be

menacing to some implementations of DSS. However, it is widely used and accepted as a good algorithm.

RESULTS

Results were built on a java platform , we implemented public key cryptography are connected in a ring network. The following algorithm takes file size of v kb as ainput and it will calculate the execution time as the output. Here input message will be construct into the encrypt and decrypt. If it's not excepted its output its failed. After this calculate the execution time if the text file is less than or equal to 99 kb. While its calculate the execution time for a file if the file size is greater than or equal to 100 kb.

In a table 2 the algorithms its depend on the process or speed. The execution time for different algorithms is as follows.

Table 2 Improvements of various algorithms.

Performance improvement of various algorithms	Reduction time in %
RSA	45
DSA	55
ECC	60

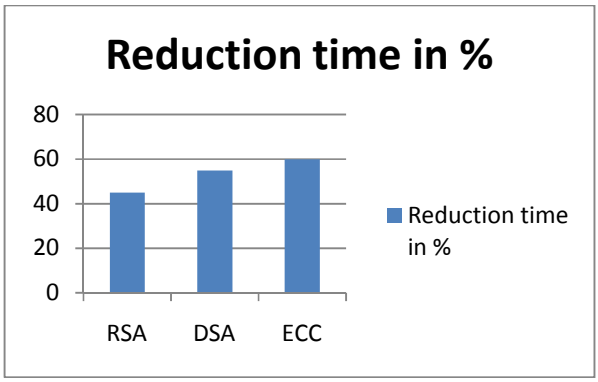

Figure 5.1 Performances improvements of various algorithms

In the fig 5.1 Results shows that the proposed system for implementation of various cryptography algorithms using java application programming interface has been reduced the execution time for algorithm from 40-60% for different algorithms.

From the implementation results above three algorithms. It is taking very less time for execution time process compared to other algorithms.

CONCLUSION

This project has described the concept of parallel programming by using concurrent java processor. Here it is shown how to efficiently and effectively implement the various cryptography algorithms by using java application, extraction as much parallelism as possible

from the algorithm in parallel implementation approach. It also includes the extensive quantitative evaluation of execution time for both sequential and parallel implementation. Implementation results shows that, parallel computation of public key cryptography algorithms provides highly efficient and reliable way to perform encryption and decryption. After evaluation of execution time for RSA.DSA,ECC algorithm it I reported that parallel implementation of processor takes very less time for performing the encryption and decryption than the other algorithms. Overall, it can be concluded processor provide efficient and reliable way to implement public key cryptography algorithm.

In future the project can be extended for multi-core processor machines. This project can also be executed to reduce the power consumption for each core.

Acknowledgement
Authors express sincere thanks to the Director of SCSE, VIT UNIVERSITY & VELLORE, for extending his support.

REFERENCES

[1] C.K. Koc¸,T. Acar, and B. S. Kaliski Jr., *IEEE Micro*, vol. 16, no. 3, pp. 26–33, **1996**.
[2] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C.Shantz, *LNCS*, vol. 3156,pp. 119–132, **2004**.
[3] R. Rivets, A. Shamir, and L. Adleman, *Communications of the ACM* vol. 21, pp. 120–126, **1978**.
[4] Wei Liu; Rong Luo;Yang; ”Cryptography Overhaed Evalutation and Analysis for Algorithms”, WRI International Conference on Mobile, Pages: 496-501.
[5] W.Diffie and M.E.Hellman. *IEEE Transactions on Information Theroy*, 22: 644-690,Nov **1989**.
[6] Andrew Matthew Lines, *IEEE Computer Society*, June **1998**.
[7] Parikh c.;Patel.p; *IEEE International Symposium on Electronics*,Pages:1-7.
[8] Behrouz Foruazan-Cryptography And Network Security 4th edition.
[9] Willa Stallings-Cryprography concepts in 3rd edition.
[10] http://java.org/wp.html