World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:1, No:11, 2007

GridNtru: High Performance PKCS

Narasimham Challa, and Jayaram Pradhan

***Abstract*—**Cryptographic algorithms play a crucial role in the information society by providing protection from unauthorized access to sensitive data. It is clear that information technology will become increasingly pervasive, Hence we can expect the emergence of ubiquitous or pervasive computing, ambient intelligence. These new environments and applications will present new security challenges, and there is no doubt that cryptographic algorithms and protocols will form a part of the solution. The efficiency of a public key cryptosystem is mainly measured in computational overheads, key size and bandwidth. In particular the RSA algorithm is used in many applications for providing the security. Although the security of RSA is beyond doubt, the evolution in computing power has caused a growth in the necessary key length. The fact that most chips on smart cards can't process key extending 1024 bit shows that there is need for alternative. NTRU is such an alternative and it is a collection of mathematical algorithm based on manipulating lists of very small integers and polynomials. This allows NTRU to high speeds with the use of minimal computing power. NTRU (Nth degree Truncated Polynomial Ring Unit) is the first secure public key cryptosystem not based on factorization or discrete logarithm problem. This means that given sufficient computational resources and time, an adversary, should not be able to break the key. The multi-party communication and requirement of optimal resource utilization necessitated the need for the present day demand of applications that need security enforcement technique .and can be enhanced with high-end computing. This has promoted us to develop high-performance NTRU schemes using approaches such as the use of high-end computing hardware. Peer-to-peer (P2P) or enterprise grids are proven as one of the approaches for developing high-end computing systems. By utilizing them one can improve the performance of NTRU through parallel execution. In this paper we propose and develop an application for NTRU using enterprise grid middleware called Alchemi. An analysis and comparison of its performance for various text files is presented.

***Keywords*—**Alchemi, GridNtru, Ntru, PKCS.

I. Introduction

NTRU is latest in the line of PKCS[2]. It is relatively new and was conceived by Jeffrey Hoffstein, Jill Pipher and Joseph Silvermann. NTRU uses polynomial algebra combined with clustering principle based on elementary mathematical theory. The security of NTRU comes from the interaction of polynomial mixing system with the independents of reduction modulo of two relatively prime numbers. NTRUEncrypt [4] employs certain rings of polynomials with convolution multiplication. It relies on the presumed difficulty of factoring certain polynomials in such rings into a quotient of two polynomials having very small coefficients. The Basic Collection of objects used by the NTRU PKCS is the ring R that consists of all truncated polynomials of degree N-1 having integer coefficients like a = $a_0 + a_1X + a_2X^2 + a_3X^3 + . . . + a_{N-2}X^{N-2} + a_{N-1}X^{N-1}$. Polynomials are added in the usual way, they are also multiplied more or less as usual. A full implementation of NTRU PKCS is specified by a number of parameters. N the polynomials in the truncated polynomial ring having degree N-1. q - large modulus: usually, the coefficients of the truncated polynomials will be reduced mod q, except the power X^N should be replaced by 1, the power X^{N+1} should be replaced by X, the power X^{N+2} should be replaced by X^2, and so on. p-small modulus as the final step in decryption, the coefficients of the message are reduced mod p. A grid [7] can be viewed as an aggregation of multiple machines each with one or more CPUs are abstracted to behave as one virtual machine with multiple CPUs. Security, uniform access, resource allocation, scheduling and network management are the challenges before grid architecture. Grid can be realized by the integration of individual software and hardware components into a combined network resource. However grid implementations differ in the way they implement the abstraction. A grid is a collection of machines sometimes referred to as nodes, resources, members, donors, clients, hosts, engines and many other such terms. They all contribute any combination of resources to the grid as a whole. Some resources may be used by all users of the grid while others may have specific restrictions. The common resources used in a grid are computing cycles provided by the processors, storage, and communications. Traditional grid implementations have only offered a high-level abstraction of the virtual machine where the smallest unit of parallel execution is a process. The specification of a job to be executed on the grid at the most basic level consists of input files, output files and an executable files. In this scenario, writing software to run on a grid involves dealing with files, an approach that can be complicated and inflexible. On the other hand, the primary programming model supported by Alchemi[6][7] an enterprise grid offers a more low-level (hence powerful) abstraction of the underlying grid by providing multithreaded programming. The smallest unit of parallel execution in this case is a grid thread that is

Manuscript received on September 30, 2007. This work is supported by The Department of Computer Science, Berhampur University as a part of research study.

Narasimham Challa, Associate Professor, is with MVGR College of Engineering, A.P., India (e-mail: narasimham_c@yahoo.com)

Jayaram Pradhan, Professor, is with Berhampur University, India (e-mail: jayarampradhan@hotmail.com).

programmatically analogous to a normal thread without inter-thread communication.

With this knowledge, we consider the NTRU implementation in two different environments one is normal NTRU and the second one is GridNtru i.e., high performance NTRU with enterprise grid. There is no doubt that any of the methods implemented with grid computing will show better results. But, our study is mainly focused on the design and implementation of NTRU with high performance computing (GridNtru) ,comparing the performance characteristics with NTRU. In the later sections we described the design of GridNtru, NTRU key generation, encryption and decryption methods implemented in two environments.

II. GridNtru Design

The architecture of GridNtru is shown in Fig. 1. The .Net application interacts with Alchemi to enhance the NTRU performance. In this application we have developed three main classes. The first class GridNtruForm is the interface to control and monitor the progress of key creation, encryption, decryption and connection with Alchemi manager. The second class ConfigurationForm is the form that can be used to configure the number of threads to be submitted and specify the location of the Alchemi manager. The last class GridNtrutThread is the thread class that is run under Alchemi and it uses the algorithm classes.

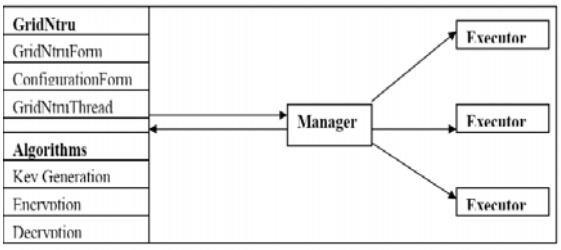

Fig. 1 GridNtru Architecture

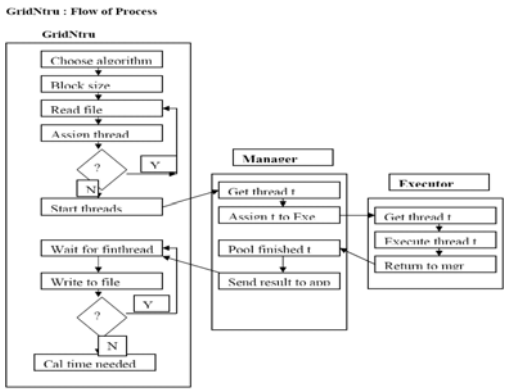

Fig. 2 GridNtru process flow diagram

The effect of this parallelization method is that we have to find a way to divide the files into several blocks so that the process can be done in parallel on each block of data. We divide the file into several blocks based on the configuration form. All the algorithms will be performed in a parallel manner. After the threads return with the result, finally GridNtru returns the result and time.submit your manuscript electronically for review as e-mail attachments.

III. Key Generation

Person B wants to create a public and private key pair for the NTRU Public Key Cryptosystem. B first randomly chooses two small polynomials f and g in the ring of truncated polynomials R. A small polynomial is a small relative to a random polynomial mod q. In a random polynomial, the coefficients will in general be randomly distributed mod q. In a small polynomial, the coefficients are much smaller than q. B must keep the values of the polynomials f and g private, since anyone who knows the value of either one of them will be able to decrypt messages sent to B. B's next step is to compute the inverse of f modulo q and the inverse of f modulo p. Thus B computes polynomials fq and fp with the property that f*fq = 1 (modulo q) and f*fp = 1 (modulo p). If by some chance these inverses do not exist, B will need to go back and choose another f. For information about computing inverses in the ring of truncated polynomials, B computes the product h = p* fq*g (modulo q). B's private key is the pair of polynomials f and fp. B's public key is the polynomial h. The CreateKey function is shown in the algorithm. Creating the inverse polynomial of the secret key modulo q i.e., fq. Creating the inverse polynomial of the secret key modulo p i.e., fp. Creating the Public Key h = p * ((fq)*g) mod q. Also the algorithm assumes $q = 2^w$ so the reduction is performed by extracting the lower w bits. The CreateKey function is implemented in two environments, normal NTRU and GridNtru, the performance results of these two implementations are summarized as follows:

TABLE I
Key Generation Timings

Text size	GridNtru(s)	NTRU without Grid(s)
128 b	0.00000	0.00001
256 b	0.00000	0.00001^2
512b	0.00001	0.00010
1K	0.00002	0.00021
2K	0.00005	0.00045
5K	0.00014	0.00102
10K	0.0002	0.0020

IV. Encryption

Person A wants to send a message to B using B's public key h. A first puts the message m in the form of a polynomial whose coefficients are chosen modulo p, say between -p/2 and p/2 (in other words, m is a small polynomial mod q). Next A randomly chooses another small polynomial r. This is the blinding value, which is used to obscure the message .A uses the message m, randomly chosen polynomial r, and B's public key h to compute the polynomial e = r*h + m (modulo q).The

polynomial e is the encrypted message which A sends to B. The Encode function describes the encrypted message e = (h * r) +m mod q. This is accomplished by 1.Performing the polynomial multiplication of h * r. 2. Adding the message m and again the modulo reduction will be performed by extracting the lower w bits. The encryption method of NTRU is implemented in normal environment and GridNtru environment, the computational running times are shown below.

TABLE II
ENCRYPTION TIMINGS

Text size	GridNtru(s)	NTRU without Grid(s)
128 b	0.000000	0.000001
256 b	0.000000	0.000001[2]
512b	0.00027	0.05494
1K	0.00050	0.1098
2K	0.00102	0.2747
5K	0.0032	0.6593
10K	0.0067	1.3186

V. DECRYPTION

The person B has received A's encrypted message e and wants to decrypt it. B begins by using the private polynomial f to compute the polynomial a = f * e (mod q). Since B is computing a mod q, B can choose the coefficients of a to lie in an interval of length q. The specific interval depends on the form of the small polynomials. It is very important that B does this before performing the next step. B then computes the polynomial b = a mod p. Finally B uses the other private polynomial fp to compute c = fp * b(mod p). The polynomial c will be the A's original message m. The decryption procedure is executed by the following three steps.Performing the polynomial multiplication of a = f * e (mod q). Shifting the coefficients of the range (-q/2, q/2). Performing the polynomial multiplication of c = a * fp (mod p). The decryption method of NTRU is implemented under normal environment and GridNtru environment, the computational running times are depicted as below:

TABLE III
DECRYPTION TIMINGS

Text size	GridNtru(s)	NTRU without Grid(s)
128 b	0.000000	0.000001
256 b	0.000001	0.0549[2]
512b	0.00005	0.0549
1K	0.00005	0.0549
2K	0.0001	0.0549
5K	0.0003	0.164
10K	0.001	0.361

VI. OBSERVATIONS

A. Key Generation

The key generation experiments were conducted on text files of various sizes like 128 bits, 256 bits, 512 bits, 1K, 2K, 5K and 10K under normal environment and GridNtru environment. The performance results of these experiments are shown in figure. According to the figure, we can understand that for small amount of data these results are not that much significant, where as for large amount of data it shows significant difference. One can understand that better performance is possible, if it is implemented under GridNtru environment.

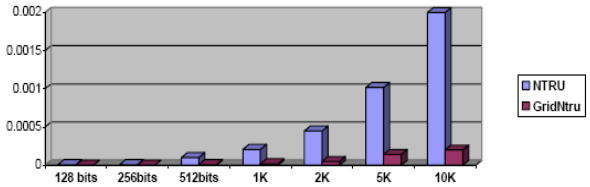

Fig. 3 Key generation performance comparison of NTRU and GridNtru

B. Encryption

The encryption experiments were conducted on text files of various sizes like 128 bits, 256 bits, 512 bits, 1K, 2K, 5K and 10K under normal and GridNtru environment. The performance results of these experiments are shown in figure. According to the figure, we can understand that for small amount of data these results are not that much significant, where as for large amount of data it shows significant difference. One can understand that better performance is possible, if it is implemented under GridNtru environment.

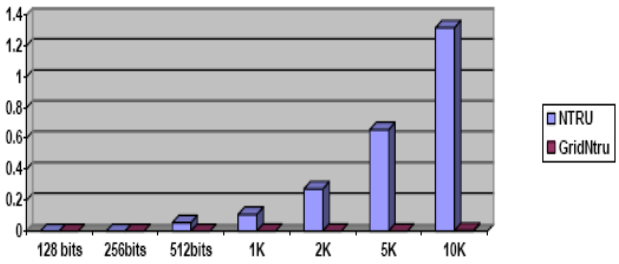

Fig. 4 Encryption performance comparison of NTRU and GridNtru

C. Decryption

The decode experiments were conducted on text files of various sizes like 128 bits , 256 bits, 512 bits, 1K, 2K, 5K and 10K under normal environment and GridNtru environment . The performance results of these experiments are shown in figure. According to the figure, we can understand that for small amount of data these results are not that much significant, where as for large amount of data it shows significant difference. One can understand that better performance is possible, if it is implemented under GridNtru environment.

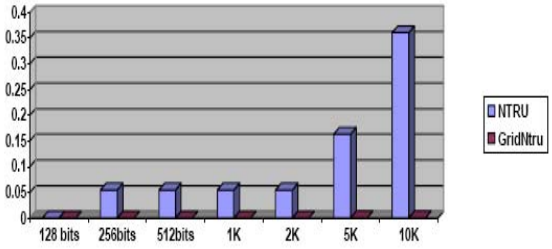

Fig. 5 Decryption performance comparison of NTRU and GridNtru

REFERENCES

[1] Whitefield Diffie, Martin E Hellman "New directions in Cryptography "IEEE Information theory, June 23-25, 1975.
[2] Joffrey Hoff stein, Jill Pipher, Joseph H Silverman "NTRU – A ring based public key cryptosystem".
[3] Joffrey Hoffstein, Joseph H Silverman "Optimizations for NTRU".
[4] Collen Marie O' Rourke " Efficient NTRU implementations".
[5] Wikipedia , the free encyclopedia " NTRU Cryptosystems Inc.,".
[6] Agus Setiawan, David A, Julius Liman, Akshay Luther, and Rajkumar Buyya "Grid Crypt: High performance symmetric key cryptography using enterprise grids".
[7] Akshay Luther, Rajkumar Buyya, Rajib Ranjan and Srikumar Venugopal " Alchemi : A .Net based desktop Grid computing framework".

Narasimham Challa was born in Andhra Pradesh of India in the year 1965. He received his Master of Computer Applications Degree from Andhra University College of Engineering, India in the year 1998, and received his M. Tech Degree in Information Technology in the year 2003 from Punjabi University, Patiala, India. Major field of study is network security and cryptography.

He has an experience of 14 years in teaching and about 18 months in software industry as software engineer. Presently, he is working as an Associate Professor in the Dept. of Computer Science and Engineering in MVGR College of Engineering, Vizianagaram, India. He has published a paper with an international journal IJCSNS and published & presented six papers in various national/international conferences.

Jayaram Pradhan was born in Orissa of India in the year 1959.He completed his Ph.D in Computer Science from Regional Engineering College (Currently it is familiar by the name of National Institute of Technology) Rourkela, India. His research areas include Network Security, Cryptography and Design and Analysis of Algorithms.

He is presently working as Professor in the Department of Computer Science, Berhampur University, Berhampur, India. He has an experience of about 22 years in teaching and 25 years in research. To his credit, he published many papers in national/international journals and conferences. He was the founder Head of the Department of Computer Science in Berhampur University.