Parallelized Multi-Key Multi-Stage Cryptography

Ahmed Alsheikhy, and Badar Almarri

***Abstract*—**The idea of this project stems from the fact that the technology of this millennium is growing very fast, especially in computers. It focuses on the topic of information security by implementing an Encryption and Decryption program. It will be built on a way such that parallelization approach is applied. The most effective encryption algorithms being used herein are the ElGamal cryptosystem algorithm and RSA algorithm. This system is intended to satisfy user's requirements, emphasize one time pad only and ensure the high strength of encryption. The algorithm used herein uses a multiple of unpredictable operations in the encryption process, also employs a key that is a sequence of unknown operations.

***Keywords*—**Cryptography, Parallelization, MPI.

I. Introduction

WHEN we use Internet, we are not always just clicking around and passively taking in information, such as reading news articles or blog posts, a great deal of our time online involves sending others our own information. Ordering something over Internet, whether it is a book, CD or anything else from an online vendor, or signing up for an online account, requires entering in a good deal of sensitive personal information. A typical transaction might include not only our names, e-mail addresses, physical address and phone number, but also passwords and personal identification numbers (PINs). The incredible growth of the Internet has excited business owners and consumers alike with its promise of changing the way we live and work. It is extremely easy to buy and sell goods all over the world while sitting in front of a laptop. But security is a major concern on the Internet, especially when people are using it to send sensitive information between parties. There is a lot of information that we do not want other people to see, such as: credit card information, social security numbers, etc.

Information security is provided on computers and over the Internet by a variety of methods [1]. Nobody can deny that the networks are penetrating our lives more and more on daily basis in a way that was never foreseen. Sensitive and critical information are exchanged instantly between individuals or even between huge companies and governmental organizations, especially on the Internet.

Data encryption has become a necessity for responsible data managers [2]. Information security has many aspects such as authentication, privacy, integrity and non-repudiation. Cryptography can be loosely defined as the scrambling of data so that only an intended authorized user can unscramble it. Cryptography allows secure transfer of information over time (transmission over a communication channel) or over space (storage within a computer memory). Cryptography can be strong or weak, Cryptographic strength is measured in the time and resources it would require to recover the plaintext. The result of strong cryptography is cipher text that is very difficult to decipher without possession of the appropriate decoding tool. How difficult? Given all of today's computing power and available time even a billion computers doing a billion checks a second it is not possible to decipher the result of strong cryptography before the end of the universe. One would think, then, that strong cryptography would hold up rather well against even an extremely determined cryptanalyst. Who is really to say? No one has proven that the strongest encryption obtainable today will hold up under tomorrow's computing power [4]. There are several ways of classifying cryptographic algorithms, according to their purposes, applications and number of keys that are employed for encryption an decryption, three types of algorithm will be classified as [4],[5]: 1) secret key cryptography (SKC): where a single key is used for both encryption and decryption; 2) public Key cryptography (PKC) where a key is used for encryption and another key is used for decryption, and 3) hash function where a mathematical operation is used to encrypt data. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is not a surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet. Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into ciphertext, which will in turn (usually) be decrypted into usable plaintext [1],[2]. Since their first appearance to the world in ancient ages, encryption and decryption have been being developed. The serious need of having data, messages, computers, servers, and networks secured and private has increased the number of methods and

Ahmed Alsheikhy is with the Northern Borders University, Arar, KSA. He is also a PhD candidate in the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06268 USA (e-mail: ahmed.alsheikhy@uconn.edu).

Badar Almarri is with King Faisal University, Alhasa, KSA. He is also a PhD candidate in the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06268 USA (e-mail: badar@engr.uconn.edu).

http://dx.doi.org/10.15242/IIE.E1014083

techniques of encrypting/decrypting contents. In 1882, Frank Miller was the first person to describe the One-Time-Pad technique in Telegraph's system. In 1919, Gilbert Vernam from AT&T invented and described the first electrical version of O.T.P. later; Joseph Mauborgne, the captain in the USA army, realized that the key sequence could be a completely random which led the cryptanalysis to be more difficult and complicated. The last phase of invention was done by Claude Shannon, a mathematician scientist, in the 1940s. He proved the theoretical significance of the O.T.P. system and published the result in 1945. The goal of this project is to implement a multi-key multi-stage software package for cryptography. The package employs a one-time pad technique, wherein a key is used only once. The package combines a multitude of strong cryptographic algorithms, secret (symmetric) key ones together with public (asymmetric) key ones such as including the RSA and El-Gamal algorithms, whose cryptanalysis shows their relative unbreakability in front of strong attacks. Depending on the number of algorithms used, low, medium and high levels of security are achieved. As the speed matters, the project's concern is also to improve the time cost by applying the parallelization on the multi-key multi-stage encryption/decryption which will decrease the execution time. Correctness of the algorithms is demonstrated via some working examples to show how they work. Although several algorithms have been used in this project beside RSA and ElGamal algorithms, a brief explanation about the latter will be given.

II. Background

A. RSA

RSA is the best known public key algorithm, named after its inventors: Rivest, Shamir and Adleman. RSA uses public and private keys that are functions of a pair of large prime numbers. Its security is based on the difficulty of factoring a composite integer that equals the product of two large primes. The RSA algorithm can be used for both public key encryption and digital signatures. The keys used for encryption and decryption in the RSA algorithm, are generated using random data. The key used for encryption is a public key and the key used for decryption is a private key. Public keys are stored anywhere publicly accessible. The sender of a message encrypts the data using the public key, and the receiver decrypts it using his/her own private key. That way, no one else can intercept the data except the receiver [8]. The RSA algorithm involves three steps: key generation, encryption and decryption. Key generation: RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way [6]: 1- Choose two distinct prime numbers p and q. In general, p and q are very large numbers, for security purposes, the integers p and q should be chosen uniformly at random and should be of similar bit-length. 2- Compute $n = p.q$, where n is used as the modulus for both the public and private keys. Then Compute $\varphi(pq) = (p - 1)(q - 1)$, where φ is Euler's totient function. 3- Choose an integer e such that $1 < e < \varphi(pq)$, and e and $\varphi(pq)$ share no divisors other than 1 (i.e. e and $\varphi(pq)$ are coprime or relatively prime). Where e is released as the public key exponent and choosing e having a short addition chain results in more efficient encryption. Small public exponents (such as $e = 3$) could potentially lead to greater security risks. 4- Determine d (using modular arithmetic) which satisfies the congruence relation $d*e \bmod \varphi(pq) = 1$. The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent which must be kept secret. Encryption: this step can be done with the following procedures: A) Obtain authentic public key (n,e). B) Represent the message as an integer m in the interval $[0, n-1]$. C) Compute $c = m^e \bmod n$ where the value of c represents the ciphertext. D) Send the ciphertext to the recipient. Decryption: to decrypt message m from ciphertext, use private key d as follows: $m = c^d \bmod n$.

B. El-Gamal encryption system

It is an asymmetric key encryption algorithm for public key cryptography which is based on the Diffie-Hellman key agreement. It was described by Taher El-Gamal in 1985. Essentially, it generates a shared secret and uses it as a one-time pad to encrypt one block of data. El-Gamal is the predecessor of DSS and is perfectly usable today, although no widely known standard has been created for it. El-Gamal consists of three components: the key generator, the encryption algorithm, and the decryption algorithm. KEY GENERATOR: this step can be done as follows:

1. Select large prime number p.
2. Select random number g in the interval $[1, p-1]$.
3. Select private key number a.
4. Compute value of $y = g^a \bmod p$, so the public key is (p,g,y) where the private key is a.

ENCRYPTION: to achieve this procedure, the sender must do the following:

1. Select random integer number k in the interval $[0, p-1]$.
2. Represent the message m as an integer in the range $[0, p-1]$.
3. Compute first part of Elgamal block $y1 = g^k \bmod p$.
4. Compute second part of elgamal block $y2 = m*y^k \bmod p$.
5. Combine both results to appear as one block which is the ciphertext and send it to the recipient.

DECRYPTION: to decrypt a desired message from ciphertext, do the following:

1. Obtain sender private key a.
2. Compute $m = (y2* (y1^a)^{-1}) \bmod p$, the value of m is the original message.

III. Parallelization Scheme

One time pad technique is a key in the implementation of the parallel paradigm. The most suitable method to parallelize

the work of this application is to start with the input size and divide it to be assigned to any number of processors we have in a rational way. It is meant by a rational way that the number of processors should not be greater than the keys as counted in the beginning of the program. This application deals with texts as input. Based on the cluster used and the number of processors, it is capable to divide the input text to the processors available. One processor will do count, divide, and assign. Each other processor receives the assigned portion of the input and performs the cryptographic operations as it has been selected in the beginning of the program.

The cryptographic operations performed on the input will be as follows: In the low level, there are five operations which are Addition, Circular Shift, Swapping, Insertion and RSA; whereas there are seven operations in the medium level which are Addition, Circular Shift, Swapping, Flipping, Exclusive OR, Insertion and RSA; and finally nine cryptographic operations used in the high level that are Exclusive OR, Circular Shift, Swapping, Addition, Flipping, Multiplication, El-Gamal, Insertion and RSA. These operations can be used for decryption as well as for encryption but reversely. For example, in the low level, we apply Addition, Circular Shift, Swapping, Insertion then RSA for encryption, whereas in decryption we apply RSA, Swapping, Circular Shift, and then Addition. Knowing that these operations are algorithms that have been invented and developed in the past separately, but the mixture of them can provide a solid cryptography.

This application is developed using MATLAB because of the extreme powerful built-in tools and the state of the art deals with matrices and different functions. The parallelization in MATLAB is available after that Mathworks Inc. has released the Parallel Computing Toolbox. This toolbox supports different functions in parallel computing programming. MPI functionalities that are needed to communicate cores with each other are available in this toolbox without the need to MPI. "Parallel Computing Toolbox provides several high-level programming constructs that let you convert your applications to take advantage of computers equipped with multicore processors and GPUs. Constructs such as parallel for-loops(parfor) and special array types for distributed processing and for GPU computing simplify parallel code development by abstracting away the complexity of managing computations and data between your MATLAB session and the computing resource you are using." To start a parallel work, a Matlabpool function is the principal to establish the environment. Workers (or processors) can be determined and configured using the Cluster Profile which can be either local or remote server. At the first step, we used the local cluster which basically represents the workers in the single machine (PC). There are only two workers in the used machines which can give a convenient perception about how parallelization has been applied. The pool size (the number of workers) can be determined by using matlabpool open pool size, but most clusters have a maximum number of processes that is twelve if there are. After opening the pool, there is a function called SPMD (or Single Program Multiple Data). "MATLAB executes the SPMD body denoted by statements on several MATLAB workers simultaneously. Inside the body of the SPMD statement, each MATLAB worker has a unique value of labindex, while numlabs denotes the total number of workers executing the block in parallel. Within the body of the SPMD statement, communication functions for parallel jobs (such as labSend and labReceive) can transfer data between the workers." These two functions works similar to the MPI send/receive functions. MATLAB uses as many workers as it finds available in the pool or as specified. labSend(data, destination) sends the data to the specified destination. Data can be any MATLAB data type. Destination identifies the labindex of the receiving lab, and must be either a scalar or a vector of integers between 1 and numlabs; it cannot be labindex (i.e., the current lab). On the other hand, labReceive(source) receives the data sent from the source or a labIndex which has sent the data. Finally, the function gather is also used to collect all the output data that has been already processed by the workers[15].

Due to that Hornet Cluster does not support parallelization using MATLAB, this program has been applied on only two cores in a personal computer that has only two cores, though the results were satisfactory because of the improvement that has been noticed on the running time of the program from the sequential to the parallel program.

The execution of the program has been done sequentially and in parallel. The test examples included different input size varying from 900 to 12,000. The execution time in each test gives agreeable values that satisfy the theoretical assumptions we were based on.

TABLE I
MODE: L = LOW, M = MEDIUM, H = HIGH
TE = ENCRYPTION TIME AND TD = DECRYPTION TIME.

Number of Inputs	L		M		H	
	te	td	te	td	te	td
n	Sequential					
925	1.66	1.09	2.41	1.83	2.67	2.35
1895	1.72	1.26	2.87	1.91	5.54	5.67
3764	4.76	2.05	5.95	3.2	32.29	30.74
6467	7.15	4.28	9.95	7.69	104.59	106.96
10495	11.55	7.09	15.97	10.36	326.25	334.97
12133	12.31	8.15	19.07	13.23	450.5	448.64
	Parallel					
925	0.65	0.59	0.91	0.89	1.71	1.64
1895	0.83	0.75	0.99	0.92	3.28	3.37
3764	1.03	1.19	1.7	1.78	16.33	16.09
6467	2.01	1.85	4.77	4.15	45.11	51.4
10495	6.18	5.61	8.49	6.83	96.11	102.21
12133	8.75	7.48	10.35	9.89	161.71	157.95

The following graphs show the difference between sequential time and parallel time for each level of security. The input size promotes a growth in the execution time as it gets larger. In each level, there are different operations to be performed as mentioned previously. In the low level, five operations are there whose execution time gets up whenever the input size gets increased.

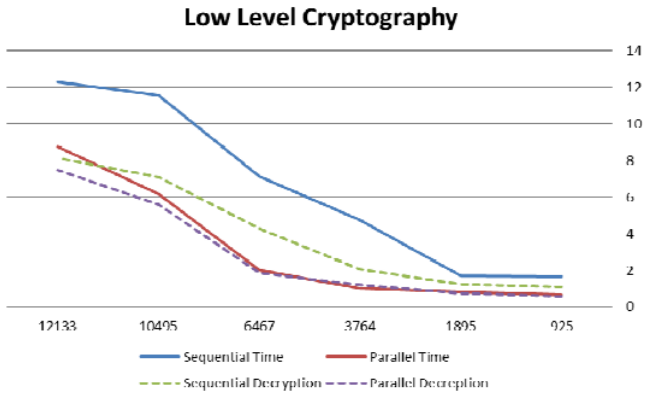

Fig. 1 Low level cryptography

In each level of cryptography, it is noticeable that time is higher in High level than the Medium level; the Medium is higher than the Low level. That is because of the number of operations used in each level.

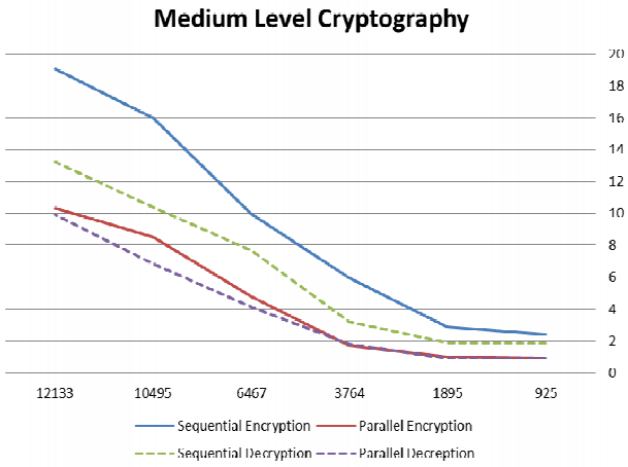

Fig. 2 Medium level cryptography

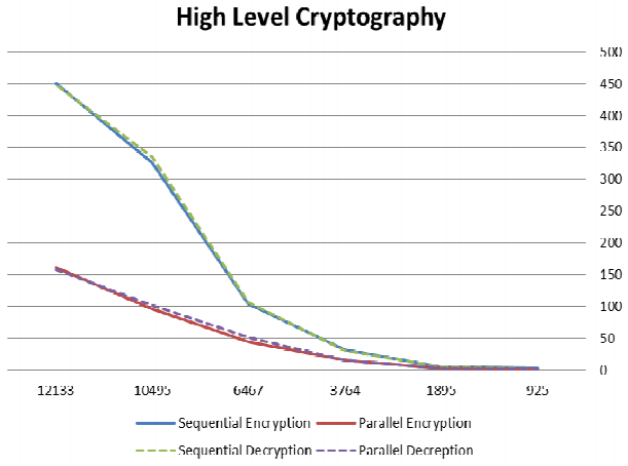

Fig. 3 High level cryptography

The proposed scheme has the following features:

• Variable Key Length: it depends on the input size whether it's small or large.

• Effective Security: since the keys are truly random and used only once; no matter what operation is performed frequently.

• Flexible Key Generation Technique: the algorithm ensures that the random generator sets its certain point in different locations each time it runs.

• Less Coding Complexity.

• Reduce time complexity from O(n2) to be approximately O(n/p), where p is the number of processors.

• Less Complexity in analysis

• Balanced the job among all Workers so each worker takes the same amount of data and performs same numbers of operations.

• Super linear case occurs in some situations but it's acceptable. In worst case, the speed up rises around 30%.

TABLE I
SPEED UP BASED ON INPUT SIZE IN TABLE1

Speed up %		
L	M	H
55%	57.50%	33.26%
47%	96%	40.68%
67.40%	62%	48.56%
66.23%	49.45%	54.38%
37%	41.80%	70%
21%	37.33%	64.45%

IV. METHOD

The algorithm works on a local or remote cluster (where multiprocessors are available). For using a cluster, a scheduler distributes the data among all nodes; where some operations will be performed according to user's selections. The algorithm has multilevel of security namely as Low, Medium and High; where in each level, a certain number of operations occur. Since the hornet cluster doesn't have a license for Matlab Parallel Computing ToolBox, the scheme tested only on a local machine with different size of input. Fig. 4 illustrates how the toolbox works.

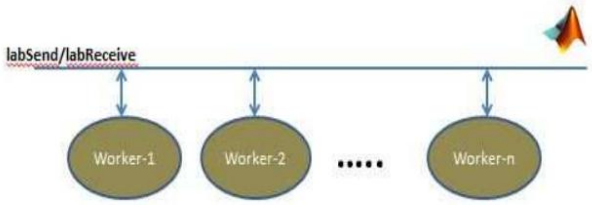

Fig. 4 Message passing between workers in MATLAB Parallel Toolbox

The algorithm goes as following:

a. One node works as a master while others work as slaves.

b. The master node reads the input file which contains the data, computes its length and divides it by the number of nodes which is determined previously.

c. The master chunks the data to be likely equal if possible, sends each chunk to its destination node.

d. All nodes including the master itself will perform certain operations as well as time taken and save the results.

e. The master node receives data at the end whether it's encrypted or decrypted from all slaves and puts it in a matrix. Then compares the time and shows the results.

The developed software has a user friendly interface where users can select among different levels of encryption methods. Each level represents a bundle of cryptosystems that are strong enough and unbreakable. The algorithm supports three levels or stages of cryptography which they are low, medium and high. Input format and Output formats must matches. Text must be selected; an error box pops up if they don't match and program terminates. Data is stored in a file named (file1.txt). A user determines the input size by saving any text in that file. The idea of this scheme is based on dealing with any text by taking every letter as a string. The numbers of keys vary from stage to another; and the input size increases to be almost more than the double size which raises the security's efficiency.

V. Conclusion

Since there is no previous work on parallel mode of one time pad (or O.P.T), the output of this scheme is reasonably satisfactory. The execution time has been reduced perfectly when the comparison was put under the scope between the sequential and parallel modes in this application. Obviously, the execution time will be reduced when more nodes are used. The speedup was achieved as expected in the theoretical manner. The necessity of applying the high performance computing concepts on many models and applications led to think of this idea to help having a text (e.g. sensitive data, message, etc.) encrypted and/or decrypted in a practical time.

References

[1] A. S. TANENBAUM, Computer Networks, 4th EDITION. Pearson Education Inc, New Jersey, USA. 2003.

[2] I. CURRY, An Introduction to Cryptography and Digital Signatures, Version 2. Entrust Inc, USA. 2001.

[3] D. E. Culler and J. P. Singh, Parallel Computer Architecture. Morgan Kaufmann Publishers, California, USA. 1999.

[4] N. T. COURTIOS and J. PIEPRZYK, "Cryptanalysis Of Block Cipher with Over defined System of Equations," Department of computing, Macquarie University, Sydney, NSW 2109, AUSTRALIA. 2002.

[5] R. G. KAMMER, Computer Security, Cryptography, Federal Information processing and standards Publication, USA. 1999.

[6] M. BELLARE and P. ROGAWAY, "Introduction to Modern Cryptography," Davis, CA 95616, USA. May 2005

[7] M. S. ANOOP, "Public Key Cryptography- Application Algorithms and Mathematical Explanations," Tata Elxsi LTD, INDIA. 2003.

[8] T. DAVIS, "RSA Encryption," 24603 Olive Tree Lane Los Altos Hills, CA 94024, USA. OCTOBER 2003.

[9] A. Pellegrini, V. Bertacco and T. Austin, "FaultBased Attack of RSA Authentication," Todd Austin Publication, MICHIGAN, USA. 2007.

[10] A. V. Meier, "the ElGamal Cryptosystem," A CRC Press, NEW YORK, USA. 2005.

[11] T. ELGAMAL, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-31, NO. 4, 1985

[12] A. Menezes , P. Oorschot and S. Vanstone , Hand book of Applied Cryptography. A CRC Press Company, New York, USA. 1997.

[13] N. Megiddo, "Applying Parallel Computing Algorithms in the Design of Serial Algorithms", Tel Aviv University, Journal of the Association for Computing Machinery, Vol 30, No 4, October 198.

[14] E. Horwitz, S. Sahni and S. Rajasekaran, Computer Algorithms, 2nd Edition. Silicon Press, 2008.

[15] Parallel Computing toolbox of MATLAB at: http://www.mathworks.com/products/parallel-computing/