

MINIMIZING EXECUTION TIME OF GRID-BASED CRYPTOGRAPHY WITH AES ALGORITHM BY USING THREAD POOL

G. Jai Arul Jose

Research Scholar, Department of MCA, Sathyabama University
Jeeppiaar Nagar, Rajiv Gandhi Road, Chennai – 600 119, Tamil Nadu, INDIA

C. Sajeev

Research Scholar, Department of MCA, Sathyabama University
Jeeppiaar Nagar, Rajiv Gandhi Road, Chennai – 600 119, Tamil Nadu, INDIA

Dr. C. Suyambulingom,

Professor (Rtd.), Tamil Nadu Agricultural University, Coimbatore

Abstract

In this paper, we propose and develop Grid and Thread Pool Based Encryption (GTPBE) application that uses the computational resources of multiple Personal Computers in order to encipher large files with one of the most efficient and secure encryption algorithms, the Advanced Encryption Standard (AES). Thread pool size is determined according to number of Grid parts. Automatic generation of pool is based on Grid nodes which are participating in action. For increasing the efficiency deadlock is constantly analyzed. Simplicity of use and high performance of GTPBE makes it an ideal choice for deploying in any organization that needs this functionality. Experimental results with different sizes of data files demonstrate that the proposed Grid and Thread Pool Based Encryption system is able to perform with high efficiency.

Keywords: *Grid Computing, Cryptography, Thread Pool, AES*

1. Introduction

In the last two decades, we experienced a software and hardware revolution, but still we are always behind the line of end user's expectations. Nowadays, there are multiple symmetric and asymmetric cryptography methods that are efficient, secure and fast. But what would we do when the data file that we want to encrypt is in a magnitude of terabytes? A large organization often prefers its applications to do the jobs in a couple of minutes, but these kinds of computations are not even possible with ordinary desktop PCs. Only a super-computer can do those jobs. Having a super-computer in an organization demands high cost. That is where grid computing comes to help and let us do the computations that essentially belong to a super-computer with 20 or 30 desktop PCs with much lower cost [5]. There are some encryption systems that used the idea to design an application for grid-based environments. One of them is GridCrypt [6] that uses DES, RC4 and Blowfish symmetric key cryptography methods in a grid-based environment. It is a JAVA 1.6 and RMI application that runs above the enterprise grid middleware called Alchemi. Another example is GridNTRU [7] that is in fact a high-performance NTRU (N^{th} degree Truncated Polynomial Ring Unit) algorithm with enterprise grids and uses its own cryptography method. Both of these systems apply their computation distributions (splitting files into many packets) by the way of multithreading. However, the cryptography methods they are using are not much of a good choice now. The proposed Grid and Thread Pool Based Encryption (GTPBE) uses the Advanced Encryption Standard (AES) [8] and is implemented using JAVA 1.6 and RMI. Before we mention the unique features of the GTPBE system in encrypting large files, we make a brief introduction to the AES encryption algorithm.

1.1. The Advanced Encryption Standard

The Rijndael algorithm is a block cipher algorithm that is selected by the U.S. government as the Advanced Encryption Standard (AES) [10]. It is the next generation of the DES cryptography method. It is also the first cryptography method for encrypting NSA's top secret information that is available to the public. The AES algorithm can be implemented so easily and it is one of the most secure algorithms in the world. The only kind of attacks done to AES till now are Side Channel Attacks [11] that are not considered as real security threats. No one can find the key to an AES implementation with this kind of attack. Some researchers are concerned about the use of the AES in security-critical applications [12], but no successful attack has ever broke AES cipher. AES implementations have 128, 192 or 256 bit key lengths. Size of data blocks to be encrypted with AES is always 128 bits. AES has these execution rounds: KeyExpansion using Rijndael's key schedule, the initial round (including AddRoundKey step), the iterative rounds (including SubBytes, ShiftRows, MixColumns, and AddRoundKey steps). At last, the final round includes SubBytes, ShiftRows, and AddRoundKey steps. In AddRoundKey step, each byte is combined with that round's key (which is derived from the cipher key). SubBytes step is a substitution. In this step, each byte will be replaced with another byte (according to the lookup table). In ShiftRows step, each row of the data block will be cyclically shifted with a certain offset. MixColumns combines the four bytes in each column to make new values for each byte. AES with 128 bit key length has 10 rounds. 192-bit AES has 12 rounds, and 256-bit AES has 14 rounds. After these rounds, the result is a new data block that cannot be traced back to the original in any way without having the correct key.

1.2 Grid Computing

Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data, and storage or network resource across dynamic and geographically dispersed organization. Why we go for Grid computing?

- To exploit the inherent distributed nature of an application.
- To decrease the turn around or response time of a huge application.
- To allow the execution of an application, that is outside the capability of a single architecture.
- To exploit the affinity between an application component and Grid resource with a specific functionality.

2. Grid and Thread Pool Based Encryption (GTPBE) System

GTPBE uses remote method invocation for connecting each node in the grid. It maintains thread resource pool for allocating each grid part into thread. Thread pool size is effectively maintained by GridManagerProcess. GridPart files and GridPartNodes are the parameters considered for maintaining thread pool size. Background daemon thread is created for watching deadlock condition occurs during thread process. This distributed system which runs on all of the grid network's machines and each node can insert a request for encrypting or decrypting a large file into the grid. The request will be sent to all of the grid nodes. Some of them may be busy doing the computations for previous encryption requests, some of them may be even down, but this does not concern us because there are lots of computational nodes out there and many of them are idle or at least they can do the encryption computations in their backgrounds. All the jobs in the GTBHE system are thread processes and they can run concurrently with other processes. The free nodes that can accept the large file encryption request will send a message to the data owner to inform that they can participate in this particular computation. On the other side, The GTPBE component on the data owner machine splits the file into small data packets (data packet size is totally flexible and selectable by the user) and distributes them into the grid network. Now each participating node can grab a piece and start to do the computations needed to encrypt file parts. When the job is done, that node informs the owner with a message that the encrypted file part is ready and then sends the encrypted part back to the owner. So, the grid manager is basically sending unprocessed file parts and receiving encrypted file parts back and forth. It also can participate in that computation (or even other computations for encrypting other files) itself. There is a deadline for giving back the encrypted file part. If the grid node does not send the part back till that time, GBHE considers it a failure and returns that file part to the list of unprocessed file parts .So, if one of the nodes experiences a failure, there would be no harm to completing the distributed computation. It other words, there is no single point of failure in the proposed GTPBE system. The only case that the job fails is when the data owner itself fails. Finally, after encrypting all the file parts and giving them back to the owner, GTPBE merges all the file parts and constructs the desired large encrypted data and that particular distributed computation is finished. It is assumed that the GTPBE runs in a trusted environment. Each node that identifies itself for the GTPBE is assumed a valid grid node. The

GTPBE binds its valid grid nodes in an RMI registry process by giving secure access using stub and skeleton objects. Execution flowchart for the GTPBE is shown in Figure 1.

GTPBE has five major components: UtilityComponent, GridManager, GridManagerGUI, GridNode, GridNodeGUI.

UtilityComponent contains WorkerNonDaemonThreadComponent, ThreadPoolResourceComponent and DaemonDeadlockAnalyzeThreadComponent. Designing an independent Project for the distributed algorithm allows us to change the application's functionality to anything else we want.

GridManager is a component that is the main service provider for this solution. It splits the large files into smaller parts. It calls UtilityComponents for making Thread Pool and allocates each Worker Thread from the Thread Pool to each Grid File Parts and sends file parts to participating GridNodes and takes them back from them after being encrypted.

GridManagerGUI & GridNodeGUI Java Swing is used for designing graphical user interface for both the GridManager and GridNodes. Swing is a special feature available in Java. It provides some powerful classes to design graphical user interface. The classes used in swing will not depend on the operating system.

GridNode creates secure Skeleton and Stub objects for making secure connection with GridManager. Then Stub is bound with RMI registry. Here the Splitted file parts are encrypted and sends the encrypted file parts to the waiting thread.

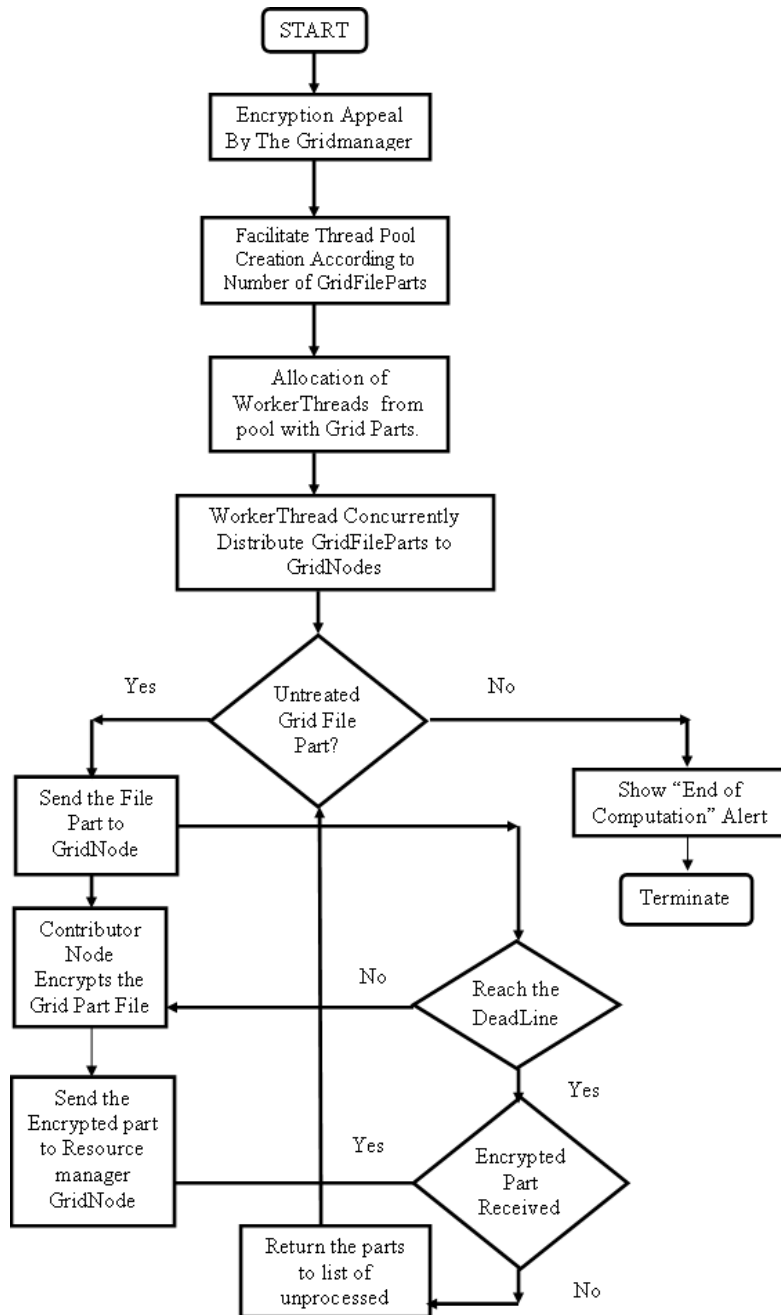


Figure 1: The Structure of GTPBE System

3. Results and Discussion

The proposed GTPBE system encrypts a huge data file in a couple of minutes. The system is tested the in multiple execution cases. Test files are selected with the following sizes: 265MB (Figure 2), 503MB (Figure 3). There are five executor nodes in the test environment, with these specifications:

1. Intel Pentium IV 2.6GHz Celeron processor, 512 MB of RAM, running Windows XP Media Center operating system.
2. AMD Athlon 1.8 GHz 64bits 2800 processor, 512 MB of RAM, running Windows XP Professional operating system.

3. AMD Athlon 2.11 GHz 64 bit X2 Dual processor, 512MB of RAM, running Windows XP Professional operating system.
4. Intel Pentium IV 1.6 GHz Dual Core processor, 512MB of RAM, running Windows XP Home Edition operating system.
5. Intel Pentium IV 2.2 GHz Dual Core processor, 512MB of RAM, running Windows Vista Home Premium operating system.

In all of the execution cases, Intel Pentium IV 2.2 GHz Dual Core processor is the data owner. In the case with five GridNodes, Intel Pentium IV is also a participant node. The test environment connects the grid nodes with a 100Mbps local area network.

Five tests are performed for the two cases from one node to five nodes and Figures 3 and 4 shows the average time of those three tests in each case.

As it is seen in the two cases, by adding fourth and fifth executor nodes, we do not have a considerable execution time difference.

In the first test case, the execution time becomes 1.398 times less by adding each node. This number is 1.328 times for the second case.

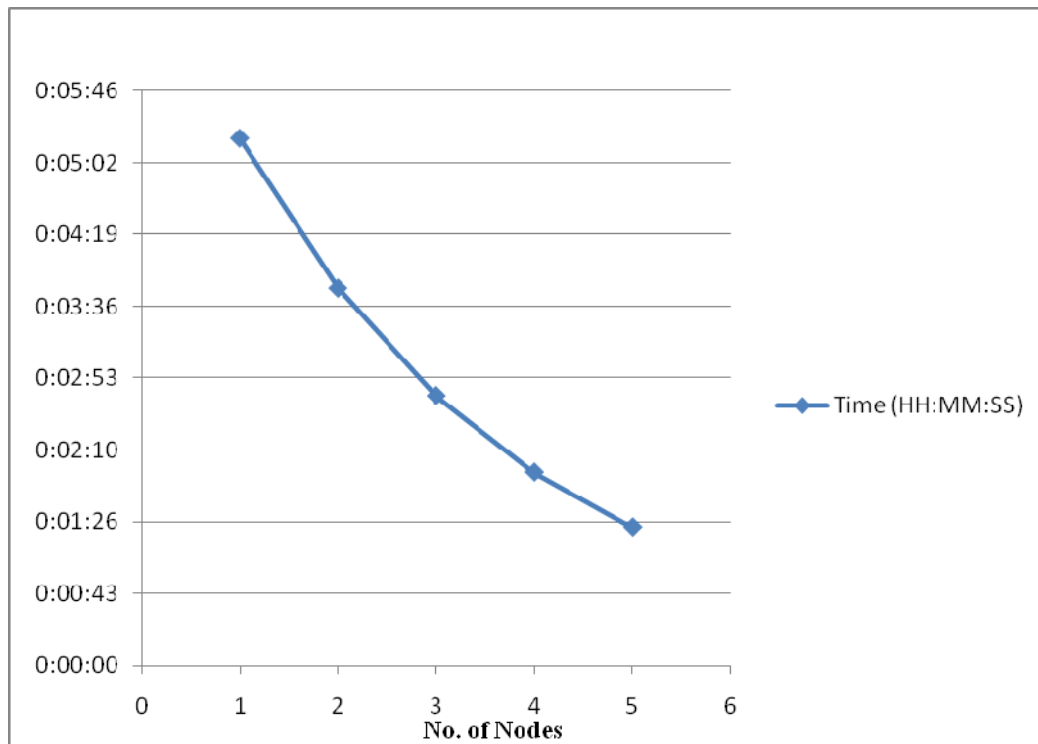


Figure 2: Average Execution Time for the First Case

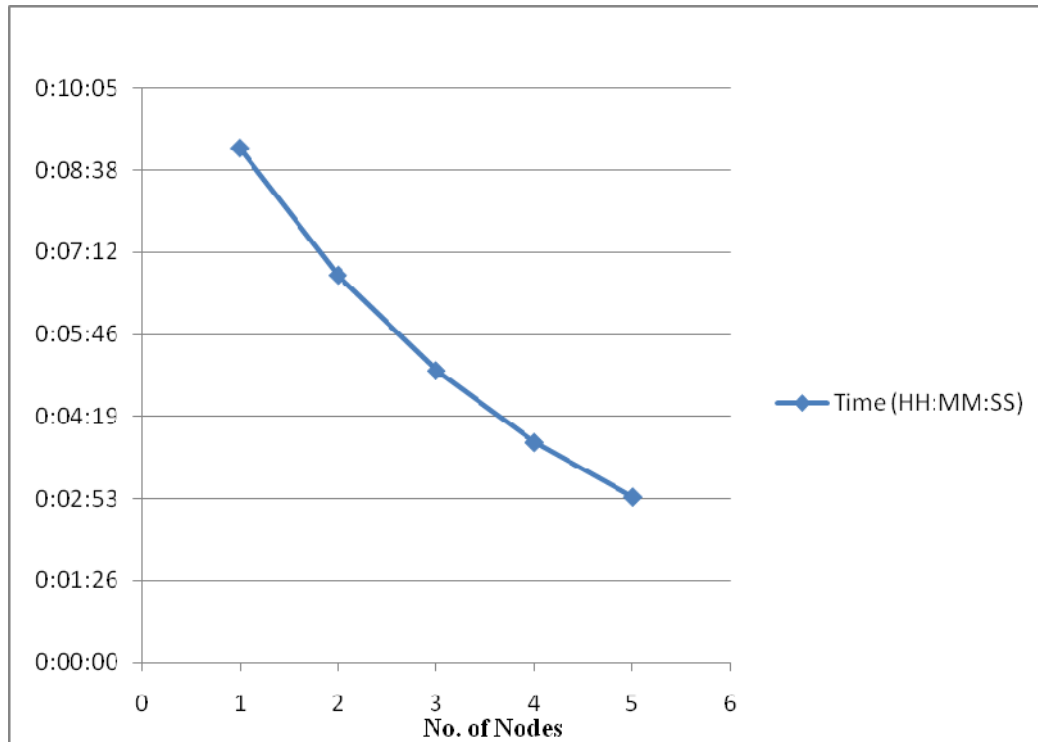


Figure 3: Average Execution Time for the First Case

4. Conclusion

A Grid and Thread Pool Based Encryption (GTPBE) System is proposed to perform file encryption. The proposed system is called The Grid and Thread Pool Based Encryption (GTPBE) which uses the AES algorithm for file encryption. The proposed GTPBE system is tested with different file sizes and different number of grid nodes. The experimental results show that the GTPBE reduces the execution time considerably.

References

- [1] William Stallings, Cryptography and Network Security Principles and Practices, Fourth Edition, Prentice Hall, 2005
- [2] Popek, G., and Kline, C. "Encryption and Secure Computer Networks." ACM Computing Surveys, December 1979.
- [3] Feistel, H. "Cryptography and Computer Privacy." Scientific American, May 1973.
- [4] Charlie Kuafmann, Network Security: Private Communication in a Public World, Second Edition, PHI.
- [5] I. Foster, & C. Kesselman, The grid: blueprint for a new computing infrastructure (San Francisco, CA: Morgan Kaufmann Publishers, 1998).
- [6] A. Setiawan, D. Adiutama, J. Liman, A. Luther, & R. Rajkumar Buyya, GridCrypt: high performance symmetric key cryptography using enterprise grids, Grid Computing and Distributed Systems Laboratory, Dept. of Computer Science and Software Engineering, The University of Melbourne, Australia, Available: <http://www.gridbus.org/papers/gridcrypt.pdf>
- [7] N. Challa, & J. Pradhan, GridNtru: high performance PKCS, Proc. 26th World Academy of Science, Engineering And Technology, Dec. 2007, 540-543.
- [8] J. Daemen, & V. Rijmen, The design of rijndael: AES - the advanced encryption standard (Springer-Verlag, 2002).
- [9] D. Chappell, Introducing windows communication foundation, Chappell & Associates Microsoft Whitepapers, Sept. 2007.
- [10] J. Daemen, & V. Rijmen, AES proposal: rijndael, 1999, Available: <http://www.daimi.au.dk/~ivan/rijndael.pdf>
- [11] D.J. Bernstein, Cache-timing attacks on AES, Department of Mathematics, Statistics, and Computer Science, The University of Illinois, Chicago, 2005, Available: <http://cr.ypt.to/antiforgery/cachetiming-20050414.pdf>
- [12] N. Ferguson, R. Schroepel, & D. Whiting, A simple algebraic representation of rijndael, Proc. Selected Areas in Cryptography, Lecture Notes in Computer Science, 2001, 103-111.
- [13] C. McMurtry, M. Mercuri, N. Watling, & M. Winkler, Microsoft windows communication foundation unleashed (SAMS Publishing, 2007).