Distributed Text-to-Image Encryption Algorithm

Abusukhon Ahmad
Department of Computer Networks, Al-Zaytoonah Private University of Jordan, Amman, Jordan

Talib Mohammad
Faculty of Information Technology, Royal University for Women, Riffa, Kingdom of Bahrain

Hani Mahmoud Almimi
Department of Information Security, Al-Zaytoonah Private University of Jordan, Amman, Jordan

ABSTRACT
Data encryption techniques are used to protect data against hackers. Text-to-Image encryption algorithm (TTIE) is an encryption algorithm proposed for data encryption. The TTIE algorithm is used to map a given text into an image. The algorithm was analyzed and it was found that the dominant time is the storage time, i.e., saving images on the hard disk). In this paper, it is analyzed that the TTIE algorithm on a single machine when a large data collection is used. A high running time is recorded. To overcome this problem a distributed TTIE (DTTIE) algorithm is proposed in order to investigate reducing the encryption time. In DTTIE a server is responsible for distributing a large data collection (5.77 GBytes) among a cluster of nodes in a round robin fashion. Each node encrypts the document it receives into an image and then stores the resulting image on its local disk. In this paper the speed up of the proposed algorithm DTTIE is calculated.

Keywords
Distributed encryption, Algorithm, Cluster of nodes, Secured communication, Encryption & Decryption, Private key encoding.

1. INTRODUCTION
Securing the data transferred between the client and the server in a global network is one of the most important issues necessary for the continuation of some applications which are performed over the internet such as buying and selling goods on-line.

Cryptography sometime referred to as encipherment is used to convert the plain text to encode or make unreadable form of text [1]. The sensitive data are encrypted on the sender side in order to have them hidden and protected from unauthorized access and then send via the network. When the data are received they are decrypted depending on an algorithm and zero or more encryption keys.

Decryption is the process of converting data from encrypted format back to their original format [2]. Data encryption becomes an important issue when sensitive data are to be sent through a network where unauthorized users may attack the network. These attacks include IP spoofing in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP and packet sniffing in which hackers read transmitted information.

Some of the techniques that are used to verify the user identity, i.e., to verify that a user sending a message is the one who claims to be, are the digital signature and the digital certificate [3]. Digital signature and digital certificate are not the focus of this research.

There are some standard methods which are used with cryptography such as private-key (also known as symmetric, conventional, or secret key), public-key (also known as asymmetric), digital signature, and hash functions [4]. In private-key cryptography a single key is used for both encryption and decryption. This requires that each individual must possess a copy of the key and the key must be passed over a secure channel to the other individuals [5]. Private-key algorithms are very fast and easily implemented in hardware, therefore, these are commonly used for bulk data encryption.

The main components of the symmetric encryption include - plaintext, encryption algorithm, secret key, cipher text and decryption algorithm. The plaintext is the text before applying the encryption algorithm. It is one of the inputs to the encryption algorithm. The encryption algorithm is the algorithm used to transfer the data from plaintext to cipher text. The secret key is a value independent of the encryption algorithm and of the plaintext and it is one of the inputs of the encryption algorithm. The cipher text is the scrambled text produced as output. The decryption algorithm is the encryption algorithm runs in reverse [6, 2, 7]. There are two main categories of private-key algorithms namely block and stream encryption [8].

Public-key encryption uses two distinct but mathematically related keys - public and private. The public key is the non-secret key that is available to anyone you choose. It is often made available through a digital certificate. The private key is kept in a secure location used only by the application user. When data are sent they are protected with a secret-key encryption that was encrypted with the public key. The encrypted secret key is then transmitted to the recipient along with the encrypted data. The recipient will then use the private key to decrypt the secret key. The secret key will then be used to decrypt the message itself. This way the data can be sent over the insecure communication channels [6].

2. RELATED WORK
Bh. P., et. al. [9] proposed encoding and decoding a message in the implementation of Elliptic Curve Cryptography is a public key cryptography using Koblitz's method [10, 11]. In this work, each character in a message is encoded by its ASCII code then the ASCII value is encoded to a point on the curve then each point is encrypted into two cipher text points.

Singh and Gilhorta [5] proposed encrypting a word of text to a floating point number that lie in range 0 to 1. The floating point number is then converted into binary number and after that one time key is used to encrypt this binary number.

Kiran et al. [12] proposed a new technique of data encryption. Their technique is based on matrix disordering which was relied on generating random numbers used for rows or columns transformations. In this work, the original plaintext was ordered into a Two-directional circular queue in a matrix A of order m x n. A number of column and row transformations were carried-out on the matrix and to do so a random function was used to generate positive integer say X and then X is converted to a binary number. Rows or columns transformation is made based on the values of the individual bits in the binary number resulted from the X value.

Another random number was generated in order to determine the transformation operation. The random number was divided by three (as we have three types of transformation operations) and the modulus (0, 1, or 2) was used to determine the operation type. The operation type could be 0; means circular left shift, 1; means circular right shift, and 2; means reverse operation on the selected rows. In case, rows were selected to perform a transformation operation (the selection is made depending on the bit value of X) two random numbers r1 and r2 were generated where r1 and r2 represent two distinct rows. Another two random numbers were generated c1 and c2 that represent two distinct columns. Two columns c1 and c2 were generated in order to determine the range of rows in which transformation had to be performed. After the completion of each transformation a sub-key was generated and stored in a file key which was later sent to the receiver to be used as decryption key. The sub-key format is (T, Op, R1, R2, Min, Max) where,

T: transformation applied to either row or column

Op: operation type coded as 0, 1, or 2, e.g., shift left array contents, shift right array contents, and reverse array contents

R1 and R2: two random rows or columns

Min, Max: minimum and maximum values of range for two selected R1, R2

Abusukhon et. al. [13, 14] proposed a novel data encryption algorithm called Text-to-Image Encryption algorithm (TTIE) in which a given text is encrypted into an image. Each letter from the plain text is encrypted into one pixel. Each pixel consists of three integers and each integer represents one color (for example, Red, Green, and Blue). Each color has a value in the range from 0 to 255. The private key is generated randomly by creating three random integers (one pixel) for each letter. For example, letter "A" may be represented by the RGB value (0, 7, 0). All letters of a given text are transformed into a two dimensional array of pixels say M, then M is shuffled a number of times by performing row and column swapping. In this work, they analyzed the TTIE algorithm by calculating the number of possible permutations to be guessed by hackers.

Abusukhon, A. [15] investigated block cipher encryption where a given text is divided into a number of blocks and then each block is encrypted into a sub-image using a specific key. All sub-images are joined together in order to introduce one image.

Abusukhon et al. [16] analyzed the encryption time for the TTIE algorithm. They concluded that the most dominant time is the time required to write the encrypted data (images) on the hard disk. In their work, data were encrypted using a single machine.

The work in this paper differs from their work. In their work they used a single machine for data encryption. In this paper, the distributed TTIE algorithm (DTTIE) in which the data are encrypted in parallel using a cluster of nodes is proposed. This is done in order to reduce the total time required for encrypting a large data collection on the hard disk. In addition, in the experiment, it was analyzed that the TTIE algorithm using a data collection of size 1.96 Gigabytes. In this paper, a data collection of size 5.77 Gigabytes was used.

3. WORK IN THIS PAPER

In this paper, Java NetBeans has been used as a vehicle to carry out experiments.

3.1 Machine Specifications

The experiments are carried out using identical machines with the following specifications; processor Intel (R) core (TM)2, Duo CPU T5870 @ 2.00GHz, installed memory (RAM) 2.00GB operating system Windows 7 professional and hard disk 25.2 GB (free space).

3.2 Data Collection

A simple Java code is written in order to randomly generate a data collection of size 5.77 GBytes for the experiments. The data collection is stored in 500 folders on the server's hard disk. The total number of documents generated for these experiments is 25,000,000 text files.

3.3 Network Architecture

In the experiments, a local area network containing a server and seven nodes connected through a switch was built. This architecture is shown in Fig. 1.

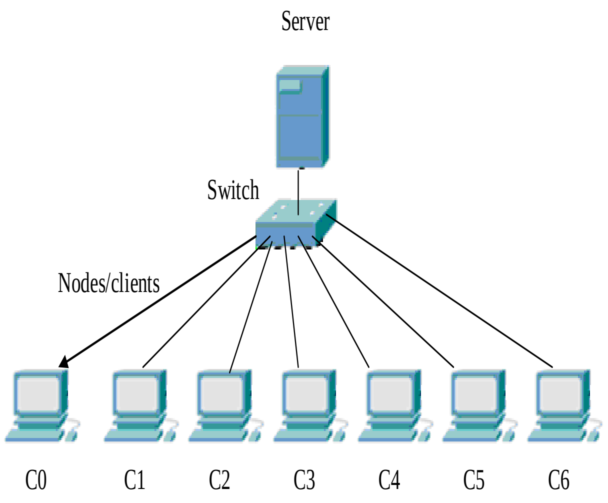

Fig 1: Network Architecture

3.4 Research Methodology

First, the baseline system was created for the experiments. The TTIE encryption algorithm was run on a single node (the server in this paper). The total time of the experiment was calculated when 5.77 GBytes was encrypted into images of type .png

In the experiments, the total time into six times as described in [16] was divided. This is done in order to facilitate comparing the system performance with the system performance proposed in [16], i.e., the baseline system:

3.4.1 Reading Text Files from the Hard Disk (RF)

Time required to read all text files (the plaintext) from the hard disk

3.4.2 Switch Time (ST)

Time required to map each letter in the text file into a pixel of three integers where each integer represents a specific color (i.e. Red, Green, or Blue)

3.4.3 Change to Two Dimensions (C2d)
Time required to change one dimensional array (this array represents all pixels of an image) into two dimensional array. This step is necessary for shuffling the matrix and thus makes it difficult for hackers to guess the encrypted text

3.4.4 Shuffle Time (SH)
Time required to shuffle a given matrix (columns and rows swapping)

3.4.5 Change to Pixels (CP)
Time required to change each three contiguous values into one pixel

3.4.6 Create Image (CI)
Time required to create an image from an array of integers and then store it into the hard disk

3.4.7 Scrambling (SC)
Time required to scramble the matrix (swap a number of rows and columns with other rows and columns)

Second, a set of experiments was carried out on a cluster of nodes (2, 3, 4, 5, 6, 7, and 8 nodes) in order to encrypt the data collection of size 5.77 GBytes into images. The data collection was stored on the server. The server distributes the data collection among a cluster of nodes in a round robin fashion. Each node encrypts the data it receives and then stores the result images on its local hard disk. The results from baseline system and the results from a cluster of nodes for measuring the speed up of our proposed algorithm (DTTIE) were used.

3.5 Data Encryption using a Single Node
In this experiment the baseline system for the experiments was built. The data collection (5.77 GBytes) was encrypted into images and stored on a single node. The total time for running this experiment is 41.004 hours. This time is divided as shown in Table 1.

Table 1. Time analysis for the TTIE algorithm (single node is used)

Time component	Time (Hours)
RF	19.399
ST	0.157
C2d	0.268
SH	0.316
CP	0.205
CI	21.081
SC	0.099

The above Table shows that the dominant times are RF and CI.

The main objective of this paper is to reduce the total time (mainly the CI time) required for encrypting a large data collection. It is proposed that storing the large data collection on a server then the server distributes the data collection among a cluster of nodes where each node encrypts part of the data collection. The following sections show the total time of data encryption when more than one node is used.

3.6 Data Encryption using a Cluster of Node
In this section, a set of experiments was run on a cluster of nodes consisting of 2, 3, 4, 5, 6, 7 and 8 nodes. The data collection is stored on the server. The server is connected to a cluster of nodes using a switch. The server distributes the data collection evenly among a cluster of nodes in a round robin fashion. Each node when receiving a text file (plaintext) encrypts it into an image and then stores it on its local disk. The experiments begin by running the DTTIE algorithm on two nodes, then 3, 4, 5, 6, 7, and 8 nodes as described in sections 3.6.1 through 3.6.7

3.6.1 Encryption using Two Nodes
In this experiment, the server reads the data collection from its local disk and then sends it to the client. The client encrypts the text files into images and then stores the images on its local hard disk. The total time of the experiment is 22.897 hours. Table 2 shows the results of this experiment described by the slowest node in the system.

Table 2. Time analysis for the TTIE algorithm (two nodes are used)

Time component	Time (Hours)
ST	0.129
C2d	0.030
SH	0.377
CP	0.024
CI	22.083
SC	0.117

The above Table shows that the dominant time is the CI time as described in [16]

3.6.2 Encryption using Three Nodes
In this experiment, a server and two clients are used to encrypt the data collection. The total time of this experiment is 10.333 hours. Table 3 shows the time of this experiment described by the slowest node in the system.

Table 3. Time analysis for the TTIE algorithm (three nodes are used)

Time component	Time (Hours)
ST	0.058
C2d	0.013
SH	0.170
CP	0.010
CI	9.909
SC	0.052

3.6.3 Encryption using Four Nodes
In this experiment, a server and three clients are used in order to encrypt the data collection. The total time for this experiment is 4.022 hours. Table 4 shows the results of this experiment described by the slowest node in the system.

Table 4. Time analysis for the TTIE algorithm (four nodes are used)

Time component	Time (Hours)
ST	0.034
C2d	0.008
SH	0.105
CP	0.007
CI	3.062
SC	0.033

3.6.4 Encryption using Five Nodes
In this experiment, a server distributes the data collection among four clients. Each client encrypts the data it receives into images and stores them on its local hard disk. The total time for this experiment is 1.931 hours. Table 5 shows the time described by the slowest node in the system.

Table 5. Time analysis for the TTIE algorithm (five nodes are used)

Time component	Time (Hours)
ST	0.024
C2d	0.006
SH	0.074
CP	0.005
CI	1.554
SC	0.023

3.6.5 Encryption using Six Nodes

Table 6 shows the time of this experiment described by the slowest node in the system.

Table 6. Time analysis for the TTIE algorithm (six nodes are used)

Time component	Time (Hours)
ST	0.019
C2d	0.005
SH	0.059
CP	0.004
CI	1.064
SC	0.019

In this experiment, a server distributes the data collection among five clients. Each client encrypts the data it receives into images and stores them on its local hard disk. The total time for this experiment is 1.488 hours.

3.6.6 Encryption using Seven Nodes

In this experiment, a server distributes the data collection among six clients where each client encrypts its sub-collection into images. The total time for this experiment is 1.456 hours. Table 7 shows the results of this experiment described by the slowest node in the system.

Table 7. Time analysis for the TTIE algorithm (seven nodes are used)

Time component	Time (Hours)
ST	0.018
C2d	0.004
SH	0.050
CP	0.003
CI	0.884
SC	0.015

3.6.7 Encryption using Eight Nodes

In this experiment, a server distributes the data collection among eight nodes where each node encrypts the data it receives into images. The total time for this experiment is 1.498. Table 8 shows the results of this experiment described by the slowest node in the system.

Table 8. Time analysis for the TTIE algorithm (eight nodes are used)

Time component	Time (Hours)
ST	0.014
C2d	0.003
SH	0.043
CP	0.003
CI	0.755
SC	0.014

4. SPEED UP

In this section, the speed up (SP) for these experiments is calculated. Speed up is one of the parallel measurements. Speed up is calculated by the following formula:

$$SP = T_1 / T_P \qquad \ldots \qquad (1)$$

Where, T_1 is the time running the algorithm on a single node and T_P is the time running the algorithm using P nodes. Linear speed up or sometimes called ideal speed up is achieved when SP = P. Table 9 and Fig. 2 show the speed up when P= 2, 3, 4, 5, 6, 7, and 8 nodes. Fig.3 shows the different components of the data encryption time when 1, 2, 3, 4, 5, 6, 7, and 8 nodes are used.

Table 9. Speed up

Number of nodes	Speed up
2	1.790802
3	3.968257
4	10.19493
5	21.23459
6	27.55645
7	28.16209
8	27.3725

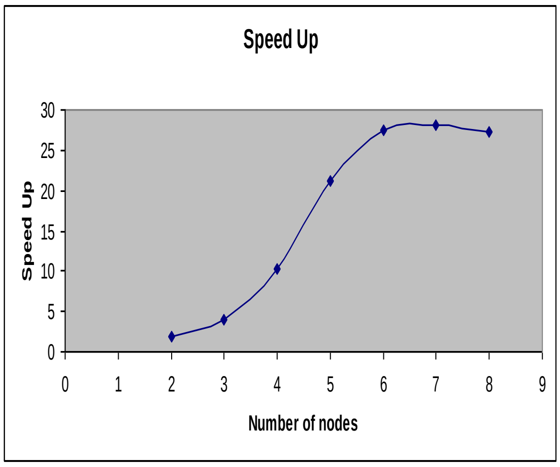

Fig 2: Speed up (data encryption for 5.77 GBytes using eight nodes)

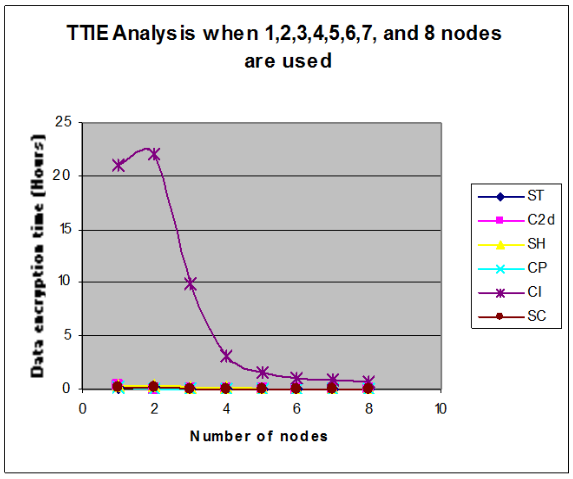

Fig 3: TTIE analysis when 2, 3, .. , and 8 nodes are used

5. ANALYZING THE ENCRYPTION TIME

As shown in Table 1, when a single node is used, the CI value = 21.081. In Table 2, when two nodes are used (a server reads the data collection and then send it to the client), the CI value = 22.083. The value of CI in Table 2 is greater than the value of CI in Table 1 (see Fig. 3). This is because of the communication time between the server and the client. In Tables 3 through 8, the value of CI is extremely decreased as the number of nodes is increased. This is because data encryption is executed in parallel. As shown in Fig. 2, the speed up is a positive slope when the number of nodes is greater than 2. This is because the data collection is distributed among a cluster of nodes and thus data encryption is carried out in parallel.

6. CONCLUSIONS AND FUTURE WORK

The results from our previous work [16] showed that the dominant time (when using the TTIE algorithm) is the time required to store images on the hard disk. In this paper, it is analyzed that the TTIE algorithm proposed by [13] when a large-scale data collection is used. The results from our work showed that the dominant time is the time required to create the encrypted text (create an image) and store it on the hard disk. In addition, in this paper, it is investigated that improving the efficiency of the TTIE algorithm by distributing a large-scale data collection (multi Giga Bytes) among a number of nodes where each node encrypts a partition of the data collection into images and then stores the images on its local disk. Our results showed that the encryption time is extremely decreased as the number of nodes is increased (Fig. 3). In addition, our results showed that the speed up is a positive slope when the number of nodes is greater than 2 (Fig. 2).

The DTTIE algorithm is useful when a large-scale data collection is required to be encrypted online e.g. cloud computing or offline at a high speed.

In future, we propose to investigate the efficiency of stream cipher and block cipher encryption when a cluster of nodes is used to encrypt a large-scale data collection using the DTTIE algorithm.

7. ACKNOWLEDGMENTS

The authors like to acknowledge and extend their heartfelt gratitude to Al-Zaytoonah Private University of Jordan for their help and financial support to carry out this work successfully.

8. REFERENCES

[1] Lakhtaria, K., (2011) "Protecting computer network with encryption technique: A study", *International Journal of u- and e-service, Science and Technology*, Vol. 4, No. 2, pp 43-52.

[2] Chan, A., (2011) "A security framework for privacy-preserving data aggregation in wireless sensor networks", *ACM transactions on sensor networks*, Vol. 7, No. 4.

[3] Goldwasser, S., Micali, S., L.Rivest, R., (1998) "A digital signature scheme secure against adaptive chosen-message attacks", *SIAM Journal of Computing* Vol. 17, No.2, pp 281-308.

[4] Zaidan, B., Zaidan, A., Al-Frajat, A., and Jalab, H., (2010) "On the differences between hiding information and cryptography techniques: An overview", *Journal of Applied Sciences* Vol. 10, No. 15, pp 1650-1655.

[5] Singh, A., Gilhorta, R., (2011) "Data security using private key encryption system based on arithmetic coding", *International Journal of Network Security and its Applications (IJNSA)* Vol. 3, No. 3, pp. 58-67.

[6] Stalling, W., (2005) *Cryptography and network security principles and practices*, 4th ed. Prentice Hall. Available at: http://www.filecrop.com/cryptography-and-network-security-4th-edition.html, [Accessed on 25-Mar-2014].

[7] Shannon, C.E., (1948) "Communication theory of secrecy systems", *Bell System Technical Journal*, pp 656-715.

[8] Bellare, M., Kilian, J., Rogaway, P., (1994) *The security of cipher block chaining.* In: Proc. of the conference on Advances in cryptology (CRYPTO'94), Lecture Notes in Computer Science, Vol. 839, pp 341-358.

[9] Bh, P., Chandravathi, D., Roja, P.P., (2010) "Encoding and decoding of a message in the implementation of Elliptic Curve cryptography using Koblitz's method", *International Journal of Computer Science and Engineering*, Vol. 2, No. 5, pp 1904-1907.

[10] Koblitz, N., (1987) "Elliptic Curve cryptosystems", *Mathematics of computation* Vol. 48, No. 177, pp 203-209.

[11] Koblitz, N., (1994) *A Course in number theory and cryptography*. 2nd ed. Springer-Verlag.

[12] Kumar, K.M. , Azam, M.S., Rasool, S., (2010) "Efficient digital encryption algorithm based on matrix scrambling technique", *International Journal of Network Security and its Applications* (IJNSA) Vol. 2, No. 4, pp 30-41.

[13] Abusukhon, A., Talib, M., (2012) *A Novel Network Security Algorithm Based on Private Key Encryption*, International Conference on Cyber Security, Cyber Warfare and Digital Forensic. Kuala Lumpur, Malaysia.

[14] Abusukhon, A., Talib, M., and Issa, O., (2012) "Secure network communication based on text to image encryption", *International Journal of Cyber-Security and Digital Forensics (IJCSDF), The Society of Digital Information and Wireless Communications (SDIWC)* Vol. 1, No. 4, pp 263-271.

[15] Abusukhon, A., (2013) "Block Cipher encryption for Text-to-Image encryption algorithm", *International Journal of Computer Engineering and Technology (IJCET)* Vol. 4, pp 50-58.

[16] Abusukhon,A., Talib, M, Nabulsi,M., (2012) "Analyzing the efficiency of Text-to-Image encryption algorithm", *International Journal of Advanced Computer Science and Applications (IJACSA)* Vol. 3, No. 11, pp 35 – 38.