Deanship of Graduate Studies
Al-Quds University

Secure Data Sharing Model for Cloud Computing (SSCC)

Israa Mahmoud Abedelfattah Alqatow

M Sc. Thesis

Jerusalem- Palestine

1438 / 2017

Secure Data Sharing Model for Cloud Computing (SSCC)

Prepared By:
Israa Mahmoud Abedelfattah Alqatow

B.Sc. Computer Systems Engineering Palestine Technical University-Khadouri/Palestine

Supervisor: Dr. Rushdi Hamamreh

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Electronics and Computer Engineering, Faculty of Engineering Al-Quds University.

1438 / 2017

Al-Quds University
Deanship of Graduate Studies
Electronic and Computer Engineering Master Program

Thesis Approval

Secure Data Sharing Model for Cloud Computing (SSCC)

Prepared By: Israa Mahmoud Abedelfattah Alqatow
Registration No: 21411738

Supervisor: Dr. Rushdi Hamamreh

Master thesis submitted and accepted, Date 09/01/2017
The names and signatures of the examining committee members are as follows:

1- Head of Committee	: Dr.Rushdi Hamamreh	Signature:
2- Internal Examiner	: Dr.Nidal Kafri	Signature:
3- External Examiner	: Dr.Mousa Farajallah	Signature:

Jerusalem- Palestine

1438 / 2017

Dedication

I dedicate this work to
My mother……..
My father……..
My parents……
My brothers……
My sister……..
My friends……..

Eng. Israa Alqatow

Declaration

I Certify that this thesis submitted for the degree of Master is the result of my own research, except where otherwise acknowledged, and that this study or any part of the same has not been submitted for a higher degree to any other university or institution.

Signed: ……

Israa Mahmoud Abdelfattah Alqatow

Date: 09/01/2017

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Dr. Rushdi Hamamreh. He has offered me great freedom on choosing my favorite research topic and developing my research interests, and continuously provided me help and encouragement with extensive knowledge. His ideas have been a source of inspiration for this work. I also thank the examining committee, all my colleagues and relatives for their support. Finally, I also thank all lecturers.

Many thanks to my mother for her praying and providing support and to my father thank you very much.

Abstract

Cloud Computing is considered to be main concerns nowadays. It plays an important role in providing sources and storing data and its scalability according to the users' needs. Meanwhile, security of data is one of the problems which face Cloud Computing taking into consideration the problem of sharing information by users. This study offers a Secure Data Sharing Model for Cloud Computing (SSCC). It can be applied on Distributed Systems and aims to protect data from unauthorized access.

Our model uses normal text as an input, and then by using many steps, the text becomes encrypted by using encryption strategy with Symmetric key which uses one secret key acquired from trusted third party where every user has his own secret key. These steps include: Encoding, data compression and Padding then turn the text into group of blocks. By using the principle of multiplying blocks, it results the Cipher matrix which contains an output encrypted text.

The data has been collected for the proposed model by doing an experiment on group of data by using two different ways: SSCC1 that based on a fulfilled matrix and SSCC2 that use upper triangular matrix in which affects Upload/Download process. It gave a good impression by comparing it with previous ways where the comparing results between SSCC and Hill Cipher 6.86% improvement in the encryption time. While when we compare SSCC and Advance Encryption Standard (AES) the difference was 3.94% for AES which means that the proposed model has a chance to compete with AES model.

Beside other comparisons which include both cases for each way, the comparison included both the compression of data, and without compression of the same data for both ways in order to find out the effect of compression to do Encryption Time for these data where we find that the time has been reduced to 96.287% by using SSCC1 and 67.146% by using SSCC2.

List of Tables

List of Figures

List of Algorithms

Contents

Chapter One

Introduction

Contents

1.1 Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [21]. This cloud model is composed of five essential characteristics, three service models, and four deployment models[30].Cloud computing deployment models that used in common are: Platform as a Service (PaaS), Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Hardware as a Service (HaaS).

Platform as a Service (PaaS) is a service that use cloud computing platform in developing or using custom applications. Software as a service (SaaS) using this service as a commercial of the shelf (COTS) cloud users can run a software located into virtual servers and use it to gain business. Using Infrastructure as a Service (IaaS) cloud users can pay per use for infrastructure (servers, storage, networking equipment and hardware).Finally, Hardware as a Service depends on time sharing on mainframes and minicomputers [13].

Cloud computing represented by three main types, First is the public cloud where vendor host the infrastructure and it can be accessed and used generally by any public user or organization that can't control where the data are stored, this type of cloud supports pay-as-you-go principle.

Second is the private cloud that is customized for a particular organization/user and the resources are not shared by other organizations, this will make it more secure but expensive and include two subtypes: externally hosted; Cloud that hosted by a third party and can be used only by one organization, and on premise; Cloud that host by the enterprise itself and can be hosted by other externally organizations, this kind is more expensive than others.

Third is the hybrid cloud is a composition of the two cloud types private and public. Private of its own critical data to gain more secure storage and public of its

general type of data that can be accessed from anyone , so that , this type of cloud is more powerful than the previous types.

Some cloud experts generate a new type and called it Community Cloud which restrict infrastructure to be only used by the organization consumers and managed by one of the third parties or greater organization of the community or both [23].

Cloud computing has a set of benefits that make it useful like configurable computing resources , flexibility of the services and economic savings .However , the privacy and security principles represent a very wide issue that must be keep in a safety manner and cloud computing support them using multi-tenancy , outsourcing and resource sharing. This model must save the privacy of each user and keep data safe also from the cloud providers [32].

As shown in Figure 1.1, a survey report that of collected information from 263 IT professionals by asking different questions related to the cloud represent security as a first rank according to IT executive [14].

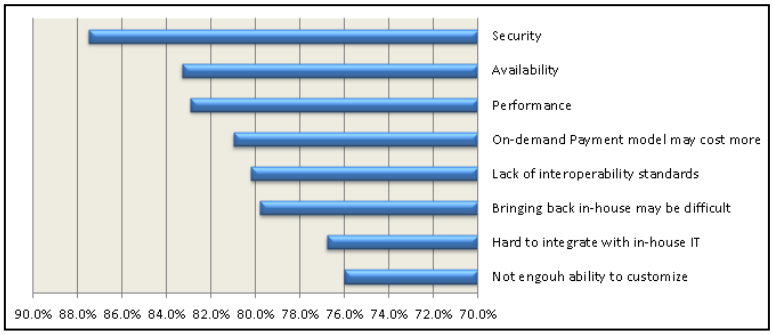

Figure 1.1: Cloud computing issues

Application software databases are moved now towards cloud computing datasets where users may not feel that it is trustworthy [9]:

1. Trust management: trust management depends on data integrity and strength, moving data to a third party may entrust your data and make it issued.

2. Security provider: Provide encrypted data storage after user authentication and authorization work will, but user may worry about his data to be unauthorized

accessed by hackers. Providers can mitigate this problem using substantial resources.

3. Privacy protection: Cloud support virtual computing to utilize the traditional computing model. User's data is saved into several virtual data centers which may be not on the same physical location, so data will face different legal privacy protection systems.

4. Ownership: Users will concern about preserving their rights in the cloud storage. Cloud provider use well-skilled user sided agreement to solve it and then user can decide if he/she wants to accept this agreement or not.

5. Data location and Relocation: Cloud computing support data mobility in a high degree and users don't know the exact location of their data [9].

6. Data integrity: providing more secure data in the data sets will add a high degree of integrity to that storage so that we need a strategy to secure users data.

7. Data recovery : a problem may occur like a server break down may cause loss or damaged data, to solve this problem data cloud users must backed up their data specially critical once on a local computer to recover it later.

8. Performance and availability: in Cloud computing performance and availability of application on the cloud can be measured through a set of levels. Business organization may be worried about this point.

9. Data Backup: Cloud user's data are stored using redundant servers and via routine data backup processes, some users will be worried about their backups. Cloud providers now offering data dumps or by regular downloads.

10. Data portability and conversion : cloud users can changed their service providers, this can be difficult while transferring, porting and converting data cause of the dependent on the data retrieval format provided by cloud providers.

1.2 Motivation

Cloud Computing is standout amongst the most critical themes whose application is being explored in today's chance. One of the noticeable administrations offered in it is the cloud storage [4]. With this storage, information is put away on numerous third party servers, instead of on the committed server utilized as a part of conventional arranged data storage.

All data put away on different outsider servers is not minded by the client and nobody knows where precisely data spared. It is minded by the cloud storage provider that claims that they can secure the data. However nobody trusts them.

Many security models were built to protect the security of data storage in distributed systems; few ones take care of the security in the cloud environment. This will courage us to design a more secure model for data storage in cloud computing within an efficient strategy keeping in our minds the high performance represented by decreasing the encryption/decryption time and the amount of data that can be upload/download to/from the cloud storage at a time.

Moreover, our model is more secure because it added new features to the existing models, for example: padding, compression in the addition of encryption; this will provide a high degree of security which keeps it hard to find the original data in an easy way.

Data will be encrypted before the processing of uploading them to the cloud storage to ensure that it is save from rather than unauthorized or authorized access from the cloud providers, also each cloud user will gain a unique secret key from a trusted Authority to restrict the access for only the authorized users; this will keep the accessing of the data storage from anywhere to everywhere.

1.3 Problem Statement

The nature of data storage in the cloud has many characteristics: Data and applications that are often located off-premise, Users do not control their data and applications, and Shortage of knowledge on how cloud computing impacts the confidentiality of stored data, processed and transmitted in the cloud computing environments. There are many benefits for cloud computing: Reduces costs, increases scalability and flexibility, provides a large-scale infrastructure for high-performance computing, and provides an on-demand access to a shared pool of configurable computing resources [17].

These resources can be rapidly provisioned and released minimal management effort by the consumer (minimal service provider interaction) and able to adapt to user and application needs [37]. However, research results show that security concerns, especially data security and privacy protection issues, remain the major reason of objection for decision makers to adopt cloud computing.

Clients need to store and get out the information safely from the cloud storage keeping that the guarantees of security and information put away on cloud need to be protected. A further point of interest is that if there is security problem at the cloud provider, the client's information will keep on being secure since all information is encoded. Clients additionally require don't care over cloud providers accessing their information unlawfully.

1.4 Research Methodology

During this thesis we search about the Original Hill cipher and its several matrices techniques, take the advantages from that and then add a new modifications to make it more accurate and close to the standard results, two approaches are designed and developed to cover these added features for solving the previous problems according to the original Hill cipher. This features control the matrices structure by creating a suitable new way to hold data with different sizes and the same matrix can be hold a wide range different file sizes and remove the hardness of

creating the key matrix with a set of rules, that mathematically proofed to be always correct.

In this thesis , Simulation phase composed of two parts in order to cover the proposed techniques SSCC1 and SSCC2 and compared the results with other encryption algorithms and the standard , our testing is done within a different variable that was delivered to gain various results depending rather than on the file size or on the matrix size .

Another simulation was done using the same files including two main cases for them, first case using compression which supported using our model the second one is without using compression, then we compare between these cases to proof that the compression played a big role by increasing the file size contained in the matrix which will decrease the traffic while data transfer.

1.5 Objectives

In this section we summarize the objectives of this thesis as the following points:

- To read and analyze existing cryptosystem models.
- To define compression techniques.
- To develop new cryptosystem model that covers both the encryption and key exchange process.
- To simulate and test proposed cryptosystem model including encryption/decryption, compression and key exchange.
- To comparison between our model and the existing models through time, throughput.

1.6 Threat Model

Cloud Computing environment is a network that connect several resources together to maintain one shared pool and can be accessed by different kind of users represented by organizations or individuals can suffer from a set of threats that our model will solve:

1- Denial of the service (DoS): is a type of attack that occurs while data transmission session where the attacker use packet splitting and duplicate insertion and this kind of attack can be described through three main cases [41] where each one can be solved through our model :

- Direct: the increase in the payload of cloud computing operating system will lead to allocate more computational power to hold it [42], using our model we can decrease the needed traffic within a specific session time for data transferred based on compressed data principle.

- Indirect: the attacker work hard to break the availability of the service of one instance so that it affects the availability of other instances [43], our model make it available always using secret key that can't be break easily so that the data will be available and retrieved every time and everywhere this work well against plaintext attack and chosen plaintext attack where attacker hack the plaintext partially to gain information about it.

- Distributed: the attacker controls a large scale of distributed hosts, so that it is the most dangerous type of the denial of the service attack [44], our proposed model support the distributed systems environments by encrypted and splitting the whole data into sub-blocks that saved into several cloud storage.

2- Man in the middle attack (MITM): attacker of the cloud computing environment will steal the connection between two parties (user and provider) and make the provider believe that this connection is secured enough, so that

the provider then send all messages directly to the attacker which can modify the received messages and retransmit them to the user [45], using our model we can make a secure connection between these two parties within the key exchange process.

3- Application security: if the application/software that lies in the cloud and not in the users side has a list of vulnerabilities then the attacker can easily plays its role on stealing the inserted data on that software despite the way [46] [50], cloud computing provider encrypt data using several techniques but the security issue has a high rank, so that we propose our cryptosystem model to encrypt data despite that data encryption using the cloud environment.

Our proposed model will mainly solved the Man in the Middle attack (MITM) by encrypting data before transmitting it to other parties, so that if any unauthorized access to these data it's encrypted and the secret key gained by the key exchange process between user and the third party can't be found on other parties rather than the original two parties (sender, receiver). Also, if someone tries to download these data from the cloud storage upon it is the cloud provider itself, the secret key needed to decrypt it and the algorithm must be known by them.

1.7 Contributions

The aims of this model are done using a set of improvements on the Original Hill Cipher that cover the following points:

- Develop new technique to modify Hill Cipher algorithm using matrices principle and compression technique to improve the traffic speed and keep the throughput in a save mode.

- Another important contribution, which improves the security of Hill Cipher against known plaintext attack. This enhancement in security is possible by using key generation idea based on key-exchange and generation process using Diffie-hellman principle.

- ❖ Remove known plaintext and brute force attacks to protect data from unauthorized access by another users on the same cloud storage or even though the cloud providers themselves.

- ❖ Get benefit from the simplicity of encryption process of Hill cipher to show that SSCC algorithm tacks less time than AES and original Hill cipher.

1.8 L U Factorization proof

Triangular matrix is a matrix with zero elements above or below the main diagonal, on one hand if the non-zero elements below the main diagonal it is called lower triangular matrix denoted by L and on the other hand if all non-zero elements above the main diagonal it is called an upper triangular matrix that denoted by U. The L U Factorization is used in many applications basically in solving the linear systems AX=B.

Let A be a square n×n matrix and can be written using lower and upper triangular matrices then:

A =LU is the L U Factorization of A.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n1} & \cdots & \cdots & \cdots & l_{nn} \end{bmatrix}, U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & u_{nn} \end{bmatrix} \quad (4.6)$$

Example:

Let:

$$A = \begin{bmatrix} 2 & 4 & 2 \\ 1 & 5 & 2 \\ 4 & -1 & 9 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (4.7)$$

then , We apply the element row operation to find the upper triangular matrix U from matrix A in equation 4.7:

$$A = \begin{bmatrix} 2 & 4 & 2 \\ 1 & 5 & 2 \\ 4 & -1 & 9 \end{bmatrix}$$

$R_2 \rightarrow R_2 - 1/2\ R_1$

$R_3 \rightarrow R_3 - 2\ R_1$

$$A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 3 & 1 \\ 0 & -9 & 5 \end{bmatrix}$$

$R_3 \rightarrow R_3 - -3\ R_1$

$$A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 8 \end{bmatrix} \Rightarrow \quad U = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 8 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \tag{4.8}$$

To find the lower triangular matrix L, we use the multiplied coefficients that solve the upper triangular matrix:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ -2 & -3 & 1 \end{bmatrix} \tag{4.9}$$

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}, E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} \tag{4.10}$$

$$E_3\ E_2\ E_1\ A = U \tag{4.11}$$

$$A = (E_3\ E_2\ E_1)^{-1}\ U = L\ U \tag{4.12}$$

The L U Factorization can generate multiple solutions for the same matrix, only if the solver gains the lower triangular matrix and the resultant matrix the upper triangular matrix can be solved easily with a unique solution, so that the attacker can't access to the upper triangular matrix without the lower once [36].

1.9 Literature Review

Modified Hill Cipher [2] gave a solution for the disadvantage points on the original Hill cipher by supporting iterations and interlacing. Despite that iteration will increase the complexity when we work with images as the input data and interlacing leads to confusion on the output of the cipher text, it provides a more secure results for the plaintext rather than images using binary conversion principle which is suitable for integers, so the key matrix can be generated to be always in a non-singular form.

In [1] Biometric traits can be solved in a secure manner using a cryptosystem model based on modified Hill Cipher within a secret key matrix represented by a prime circulant matrix where each element is rotate by one step to the right direction related to the proceeding row vector, it can be written as equation 2.28 :

$$\begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \dots & c_3 \\ c_{n-2} & c_{n-1} & c_0 & \dots & c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & c_3 & \dots & c_0 \end{bmatrix} = \text{circ}(c_0, c_1, c_2, \dots, c_{n-1}) \qquad (2.28)$$

A circulant matrix is called a prime circulant matrix if and only if the gcd of the row vector elements is equal to 1, let row vector elements called a, b, c,d then the condition is denoted by: gcd(a,b,c,d) = 1 .

Another matrix called coefficient matrix denoted by Gc is a coefficient matrix of the matrix G defined by:

$$\text{circ}\big(\text{circ}(\text{row}_1), \text{circ}(\text{row}_2), \dots, \text{circ}(\text{row}_n)\big), \text{such that } \text{row}_1, \text{row}_2, \dots, \text{row}_n$$

The row vectors of matrix G, And $\text{circ}(\text{row}_i)$ is the circulant matrix of the row vector row_i , see the following example [1]:

Let G is 2x2 matrix then Gc is a 4x4 matrix in equation 2.29:

$$G = \begin{bmatrix} g_1 & g_2 \\ g_3 & g_4 \end{bmatrix} \text{; then } Gc = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 \\ g_2 & g_1 & g_4 & g_3 \\ g_3 & g_4 & g_1 & g_2 \\ g_4 & g_3 & g_2 & g_1 \end{bmatrix} \quad (2.29)$$

This matrix is used as a public key which may has a determinant equal to zero to gain more efficient results where this matrix when used to be solved has infinite solutions for the attacker through multi-level polynomial equations like equation 2.30:

$$K = AGA^{-1} \bmod P \quad (2.30)$$

This type of equations is a hard NP-complete problem when P represents a large prime number and it is hard to compute the modulus value of that number, so a simplification form is added to solve this problem using equation 2.31 [1]:

$$Gc\ X = Y \bmod P \quad (2.31)$$

Such that X contains the elements of A and A^{-1}, also Y contains the elements of K, For example: Let A be a circulant prime matrix and G is an invertible matrix using equation 2.32:

$$\begin{bmatrix} g_1 & g_2 & g_3 & g_4 \\ g_2 & g_1 & g_4 & g_3 \\ g_3 & g_4 & g_1 & g_2 \\ g_4 & g_3 & g_2 & g_1 \end{bmatrix} \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} = \begin{bmatrix} k_{11} \\ k_{12} \\ k_{21} \\ k_{22} \end{bmatrix} \quad (2.32)$$

The number of equations is less than the number of unknowns, this leads to infinite number of solutions.

In [2] information hiding can increase the security degree of the cryptosystem model, so that a combination between this model and a steganography method will secure the transmit text message using the LSB insertion technique. Two random techniques are used subsequently, one for XOR the ciphertext and the other for the LSB insertion process into any image.

This will provide a robust cryptosystem model using both encryption and steganography processing which work against cryptanalytic attacks. The XOR process is executed within a special key (password) that is known only by the sender and the receiver, and then the encrypted sequence is permute completing using the XOR operation . The final output of that sequence is converted into an ASCII values and hidden inside any cover image.

The insertion operation of the ASCII values into that cover image is done using a numeric key within a randomized technique .This key is generated and represented by a unique random number that is a resultant value from a particular seed to the key generator. During the transmission phase user need to send his key to the Hill cipher, XOR key and the random number generator.

Encryption at the Sender Side: The mxm key matrix 'K' is taken one a time per session, and if it is a non-invertible matrix with a determinant that equal to zero, an identity matrix is added to it to make it invertible.

The encrypted cipher text is calculated and formed by an mxn matrix using equation 2.33:

$$C = K \times P \tag{2.33}$$

The resultant cipher text is divided into two parts C_1 and C_2 of order mxn computed by the equation 2.34:

$$C_1 = \lfloor C/26 \rfloor \, , \; C_2 = C \bmod 26 \tag{2.34}$$

The usage of the number 26 is to convert the text into an alphabet sequence of letters, so that we gain a doubled number of elements when compared with the original plaintext.

In the XOR technique, the cipher text is entered to an XOR operation with the key (password) of the user and XORed with it to get a new encrypted form of the text the output matrix is also divided into two parts using the division operation on 26, and then storing the quotients in matrix called C_1 and the reminder in other matrix called C_2. The integer value 65 is added to both matrices to form linear elements then

concatenating with each other in one single array with a doubled size .Finally the linear array is converted a gain into a character form.

On the receiver side to retrieve the original text , first we got the cipher text from the stego-object through the key (password) dependent on the block selection algorithm , then XOred using that password to get the Hill Cipher of that text with a doubled length when compared to the original once .After that we subtract the added integer 65 to retrieve alphabetic values that varies between 0-26 , then divide the results into two matrices of the size 3x3 to get C_1 and C_2 with a random key that picked up from a key Generator ,let K = [17 17 5; 21 18 21; 2 2 19] ,Then using the equations 2.35 and 2.36 :

$$C = (26 \times C_1) + C_2. \quad (2.35)$$

$$P = K^{-1} \times C. \quad (2.36)$$

Finally, a value 65 is added to Plaintext take it to ASCII range and the result is Plaintext = [78 65 84 73 79 78 65 76 65] , then convert it back to alphabet range we got “NATIONALA” , where the last character A is added to fix the length of the cipher , so it deleted to get the original plaintext "NATIONAL" .

In the Steganography Hiding**,** text is divided into number of parts that less than number of blocks in the cover image by one, a random numerical key generated using a random sequence generator which specifies the bit hiding degree in the cover image.

Decryption at the Receiver Side: In the Steganography Un-hiding random key sequence is generated using random key generator as in the sender side to unhide the stego-objects from the cover image, using techniques of XOR and decryption to retrieve the original Hill cipher of the text, we apply the XOR operation on the given sequence of bits to get the modified Hill Cipher values of that text.

After that, Divide the given cipher text into two parts and then reshape them into two separated matrices of order mxn the same as C_1 and C_2 on the sender side, then Convert these two matrices back to their numerical values by subtract 65 and called them Y_1 and Y_2 respectively. They also take the same key matrix, and if it is singular convert it to an invertible matrix by adding its identity matrix to it.

Use the equation 2.37 to calculate the cipher text:

$$C= ((Y_1 \times 26) + Y_2). \quad (2.37)$$

Finally, calculate the original plaintext using equation 2.38:

$$P = K^{-1} \times C. \quad (2.38)$$

In [7] One way hash algorithm can be used to clarify the principle of a singular matrix in GF (2) using the Hill cipher multiplication method .This matrix is filled by only one entry for each row and 0 otherwise , so that some of matrices can be represented only using m entries .

In this model, Hill cipher is used symmetric encryption and multiplied by a key matrix and authors proposed a class of matrices in GF (2) based on singular matrices where it is easy to prepare and used. This new model gives an automatic alternative for one-way hash function and a square singular matrix of size m is generated by the summation of two permutation matrices in the form of GF(2) , that means values can be only {0,1} and multiplied by the plaintext to gain the ciphertext based on the Hill Cipher principle .

The algorithm is designed within different two models that both based on the Cipher Block Chaining (CBC) to generate the hash value, the second model varies from the first one by a list of additional operation using a nonlinear function. The plaintext is converted to a binary values before the operation of hash value calculation then divided to column vectors { B_1 , B_2 , … B_N } with a size of mx1 and B_i = (b_{i0} , b_{i1} , ….. b_i m-1) such that $b_{ij} \in \{0,1\}$ using the following protocol in the communication parties: Sender and Receiver.

At the sender side : they prepare a value M and singular matrix P ,then calculate the hash value using the hash function A according to equation 2.39, finally send it to the receiver:

$$H = A (P, M) \quad (2.39)$$

At the receiver side: the got the same value M and a singular matrix P, then calculate the hash value H' using the same hash function A according to equation 2.40:

$$H' = A (P, M) \quad (2.40)$$

The main idea is to compare H and H' to ensure that we have the same value at the two sides.

In [11] A new technique based on the original Hill cipher is proposed to solve the flexibility of the key matrix and enhance the security of this model against the brute force attach by using public ideas that gain more powerful complexity than linear algebra steps which can be done using a specific options .Authors proposed a robust cryptosystem algorithm for singular matrices based on the original Hill cipher ,this technique cover the two main stages of the cryptography model represented by Encryption/Decryption processes.

At the encryption process, convert the character of the original plaintext into numerical values, and then generate the key matrix that needed to the multiplication operation, if this matrix singular we must convert it to a non-singular form by adding the identity matrix. After that, calculate the ciphertext by the multiplication process between the key matrix and the encoded plaintext using equation 2.33.

Finally, compute C_1, C_2 where values are specified using equation 2.41 and convert the results of C_1 and C_2 into character again.

$$C_1=\lceil C/P \rceil \text{ and } C_2 = C \bmod P \quad (2.41)$$

At the decryption process, convert the sequences of the cipher text C_1, C_2 back to its numerical values called them Y_1, Y_2, then check the determinant of the key matrix; if is zero this means that we got a singular matrix, and then remove its singularity by adding the identity matrix of it, after that calculate the plaintext using equation 2.42. Finally, convert P back to the character form.

$$P = K^{-1} \times ((Y_1 \times 256) + Y_2) \quad (2.42)$$

In [3], using matrices as a formula for the input text and the key used by the encryption process will improve the results gain from this process, many algorithms use that principle like Row Column Diagonal (RCD) and output a more powerful results than the traditional ways. The RCD engine is used to encode data either in 2D or 3D format which provide a high degree of security , it refers to RCD which means that we do our operations on them within RCD bits given by supervisor and called RCDSUP.

RCD bits contain data despite the needed operations of them that are for row, column, diagonal or transpose of the matrix.

Another type data is generated called data seed (RCDSeed) which is executed on either addition or subtraction operation that is done on row, column and diagonal of the matrix. This model is composed of a set of components as the following:

1. Seed Generator

The seed supervisor is used the seed generator regardless of the running mode UPLOAD/DOWNLOAD to provide the RCDSUP and RCDSeed for both the CLOUD and the HOST.

2. RCD Encryption Decryption Module (RCDEDM)

This model performs the Encryption and Decryption operations for data that is specified through RCD bits.

3. Seed validator

The validation process is done by the HOST to validate the RCD replay that comes from the Cloud side containing address which is unique and use the the validation data that always stored in the Seed buffer.

4. Seed buffer

This component plays a big role by storing the current statues of the process cycle and it used by the HOST to store validation data, RCD bits that represent all the seeds at the moment.

5. Controller

The decision of sending data through which line is taken by the controller, these lines can be either a Secure Control Line (SCL) where RCD bits, seed values, RCD reply and control signals are send through it or a Secure Data Line (SDL) where it is the channel for all transmitted data.

6. Unique Address Recorder

The HOST requests its unique address from this recorder in order to give it for the RCD supervisor or EDM module to validate RCD Reply and data.

7. RCD TABLE

Located at the Cloud side and store the status of the sub-clouds contains the stored files, available space, load and other parameters. This status is needed when an UPLOAD request to the sub-cloud occur and the central Cloud handle this request and directed it to the suitable sub-cloud then update the RCD table.

The Encryption Decryption Module (EDM) is a module used for the Encryption/Decryption process for the incoming RCD bits that decide the operation needed to be performed either on Row, Column ,diagonal or transpose of the matrix within the process of uploading/downloading to the Data matrix . These conditions are satisfied by the RCD bits which is called RCDSUP and generated by a RCD supervisor to tell the RCD engine what operation must be done and either it needs to be encrypted or decrypted.

The RCD encryption engine take an mxm matrix that contain data input and output $(m+2) \times (m+2)$ matrix contains data output, these two rows are added as a carry values resultant from either addition or subtraction operations on values.

First, when the first operation is done on the data matrix, we padded it with 0's for all size, so that two rows and two columns are added to this matrix. Now, the first element in the first row is prepared to contain the carry of the operation on the diagonal and the reset element will keep the carries of the columns operations. Also, the most left column will handle the carry of the diagonal and the rest contains the carriers of the row operations.

1.10 Thesis Organization

The organization of this thesis can be summarized as follows:

1. Chapter one includes brief overview of cloud computing security, motivation of thesis, problem statement, proposed solution, research methodology, and objectives of the thesis.
2. Chapter two gives an introduction to cryptosystem models including symmetric or single key encryption and asymmetric or public key encryption techniques.
3. Chapter three introduces cloud architecture and the key exchange techniques, comparing between them and the chosen one in our model.
4. Chapter four presents our proposed model using a set of steps that describes the phases of the proposed algorithm in details and a pseudo code for two our techniques that clarify it briefly.
5. Chapter five presents the testing and simulation results for both two techniques of our model (SSCC1 and SSCC2) comparing them with each other to get the optimal solution , then complete the testing by compare the results with others algorithms .
6. Chapter six presents conclusions of our results and future work for this model.
7. Finally, The list of references used during this thesis.

Chapter Two

Cryptosystem models

Contents

2.1 Introduction

In this Chapter we present an overview about Cryptosystem models, the word "Cryptography" may use alternatively with the words "Cryptology" or "Cryptanalysis" but each of them has its own meaning that differ slightly using the term "Crypto" which comes from Greek "Krypto" to mean hidden and ending with "graphy" which mean writing, so the whole word cryptography is mean to hidden writing that done as an output of the encryption process in the secret systems [5].

The encrypted message or text is called ciphertext or cipher. There is someone who can legitimately decipher this text and get the original one using a "key". Only the key holder can simply get the original plaintext which kept secret and can be provided within a trusted third party through a secure manner.

The word "Cryptanalysis" is referred to the process of analysis of the cipher or hidden text to get the original one by decrypting or expose what is hidden when the analyst of the system doesn’t have the legitimate key that used directly to retrieve the original text. Also, the word "Cryptology" is more correct to be used because it merge the meaning of either the encryption techniques or the analysis of these techniques, in common people using the term Cryptography instead [29].

Cryptography is a term that reflects the meaning of information hiding and verification using formals that depends on a set of protocols that used to results a specific form of data to keep it in away from illegal access special when we communicate with each other using sensitive information.

The process of converting the original plaintext from useful understating form to opaque form of understanding called cipher text represent the Encryption phase of the message , also using the opposite direction of the process to retrieve the original useful plaintext is called the Decryption phase, two phases are used to create the cryptography concept [6].

Cryptography is an area in a computer security that support the protection of data; keeping it in a safe manner with a confident formula to obtain and preserving of the main objectives of the computer security includes cryptography services: integrity, confidentially and availability of the resources; which contains either software or hardware. Cryptography is used to satisfy security objective using only one applicable security model, there are three main objectives confidentiality, integrity and availability (CIA) represent the base of the computer security principle:

1. Confidentiality: this concept can clarify two different terms; first is Data confidentiality that ensure the avoidance of unauthorized access to the users private data or data storage in cloud computing environment, Second is privacy to restrict the disclose of data from whom to whom and these data are collected and stored together.

2. Integrity: this concept also can cover two related terms include: Data Integrity where data is kept with only changes from an authorized users and System integrity where the system execute its function in a correct manner and prevent illegitimate manipulation from unauthorized users.

3. Availability: means that the service work correctly and available every time it will be requested from the authorized users.

Cryptography as a special case of the computer security has in general the same goals like: message confidentiality (privacy) and integrity and adds new goals because it is more specific contains: Sender authentication ensures the identity of the sender from the message content and in some cases the path the message was traveled, and sender non-repudiation: the sender can't deny the process of sending the message.

These goals are not applicable for all cryptographic systems, because some of them require one or more goals to be achieved as needed using a set of variants including: Access control; to satisfy for whom the data will be shared and the availability of the message [29].

Cryptosystem model is referred to a model that applied an Encryption methodology and a key exchange technique to generate a more secure model. In the field of Cryptography there are two several techniques for encryption/decryption processes, i.e. Symmetric and Asymmetric key Cryptography [5].

The transformation of cloud computing has given open doors for examination in all parts of cloud computing. Research in the safe cloud storage is intensified by the reality that user's data might be kept at a few areas for either repetition/adaptation to internal failure or in light of the fact that the administration is given through a chain of administration providers.

Secure data storage framework is composed, executed and assessed basically. The configuration is given every single required point of interest, which uses off-the-shelf crypto-graphical strategies and gives essential upload/download facilities.

2.2 Standard Cipher Algorithms

Standard Cipher Algorithms is a type of Symmetric or single key cryptography same key called a secret key is used for encryption and decryption. Figure 2.1, demonstrate the simplified model for symmetric key encryption technique, the plain text is encrypted using that key and the output differ when we use a different key also the decryption process done using the same key or the inverse of that key [4].

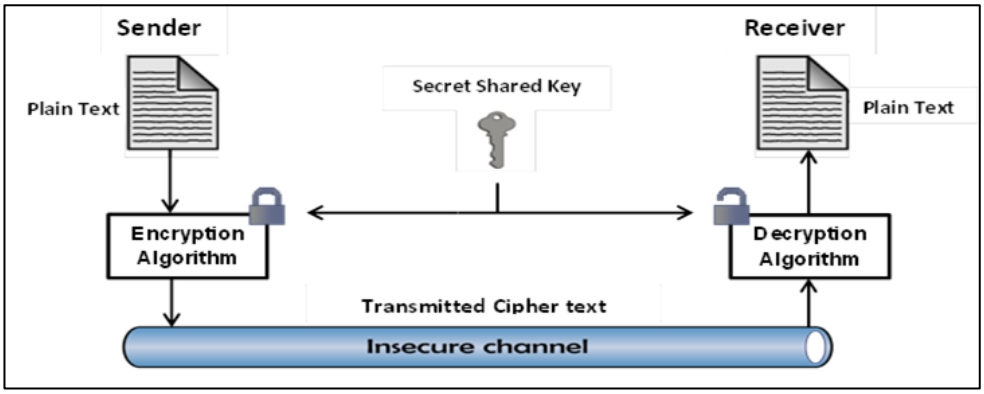

Figure 2.1: A simplified model for the symmetric encryption

In a symmetric cryptography the encryption and decryption process is done using one generated secret key that commonly shared between only two parties, this type of cryptography is composed of two main types one of them is mainly for cloud computing environment and the other is for any other distributed system that based on matrices principle.

Cloud computing environment uses many encryption algorithms to convert the cloud storage data to a cipher values that can't be guessed from users that shared the same resources. It uses generally the symmetric key encryption for several algorithms. Each one has its own role and support suitable case [51].

2.2.1 Advance Encryption Algorithm (AES)

The AES supports the scalability of the hardware resources and represent symmetric cryptography techniques that are fast and can be implemented easily [51]. This algorithm represent the standard of encryption algorithms used in cloud computing environment and depends on number of cycles used that leads to different key sizes and based on 4x4 matrix size, 10 cycles needs a key length of 128 bit also 12 cycles supports a key length of 192 bit, finally 14 cycles needs a key length of 256 bit [52]. The key that is provided as input is expanded into an array of forty-four 32-bit words. Four distinct words (128 bits) serve as a round key for each round [29], see Figure2.2.

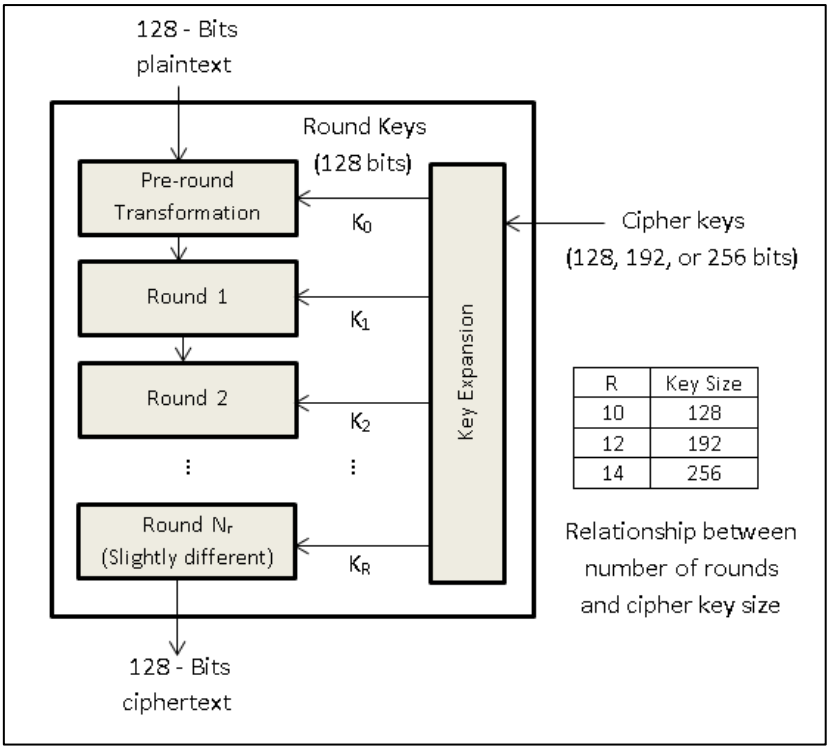

Figure2.2: AES encryption process [29]

2.2.2 Data Encryption Standard Algorithm (DES)

The DES use symmetric key encryption [53] strategy where a secret key is used for encryption/decryption process it use 64 bit of block cipher and a key of 56 bits, the encryption process is done by both confusion and diffusion processes to result the cipher text blocks that divided into 32 bits which input to F-function then combine the output with other half of data got from XOR operation to generate the whole result and then passed through 16 rounds the decryption process is done using the same operation in the reverse order but the key used is too short and can be broken also its suitable for hardware rather than software applications [52] ,see equations 2.1, 2.2 and Figure2.3.

$$L_i = R_i \quad (2.1)$$

$$R_i = L_i \text{ XOR } f(R_i) \quad (2.2)$$

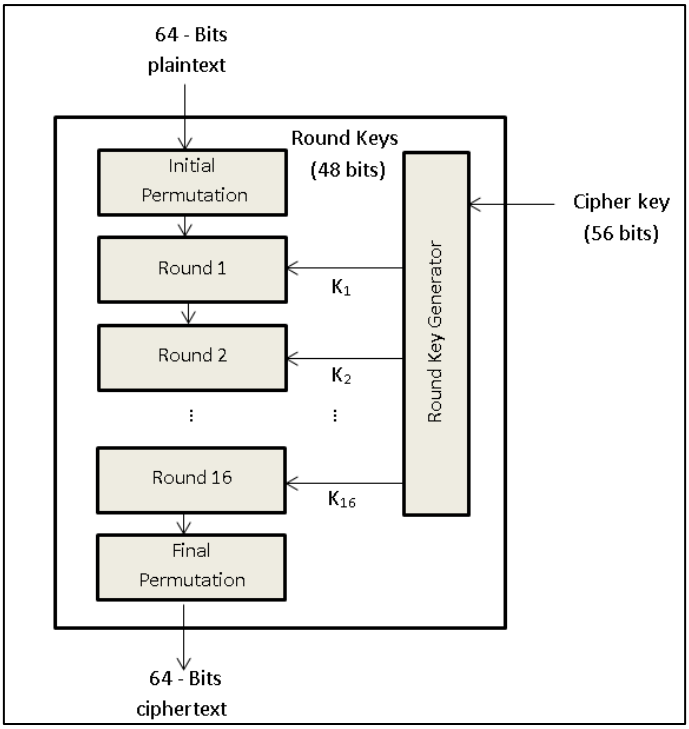

Figure2.3: DES encryption process [29]

2.2.3 Triple- DES (3DES)

The triple –DES or 3DES as a type of symmetric key cryptography [53] is use three key to encrypt the plaintext message and apply the DES three times to generate the ciphertext, these will increase the security degree and keep the cipher text in a safe manner [52] , see equation 2.3.

$$C_i = E_{K3} (D_{K2} (E_{K1} (P_i))) \qquad (2.3)$$

If $K_2 = K_3$ this is DES, see Figure2.4.

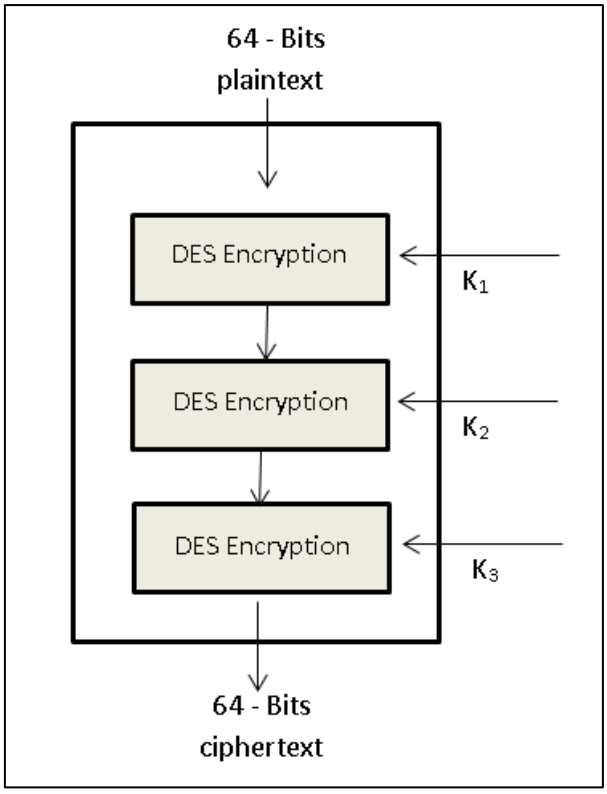

Figure2.4: Triple DES encryption process [29]

2.2.4 Blowfish Algorithm

It is also a symmetric key algorithm that works as DES using one secret key [51], but it use a larger one that varies from 32 to 448 bit that is hard to brake when compared with the previous techniques like DES, it support file sizes that are multiple of eight and if not a padded bits will be added to access the suitable file size. As DES the 64 bits is divided into two equal sized parts of 32 bits and has 16 rounds. The bits are passed to P-array function to generate the left side of the message [52]. First, Divide P into two 32-bit halves: L, R then starts with the first Round

We XORed the part of from the left side of the plaintext with a subkey see equation 2.4.

$$L = L \text{ XOR } Ki \tag{2.4}$$

Then the Right side of the plaintext is equal to the Left side output from a specific function f and XORed with the Right side see equation 2.5.

$$R = f(L) \text{ XOR } R \tag{2.5}$$

After that swap the left and right sides together .The last two steps XORed the Right and Left sides with K_{16} and K_{17} respectively, see equations 2.6, 2.7.

$$R = R \text{ XOR } K_{16} \tag{2.6}$$

$$L = L \text{ XOR } K_{17} \tag{2.7}$$

Finally, Recombine L and R sides to shape the ciphertext, see Figure for more illustration.

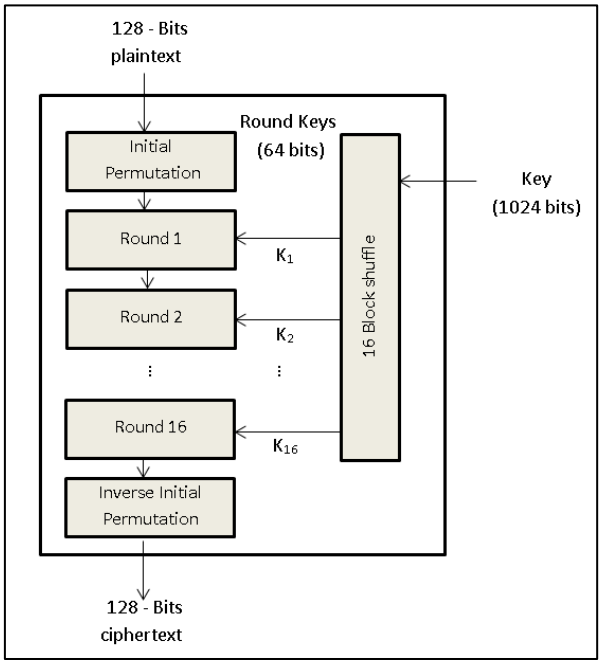

Figure2.5: Blowfish encryption process [29]

2.2.5 Hill Cipher Algorithm (HC)

This cipher is developed by the mathematician Lester Hill is One of the symmetric key cryptography that use the matrix manipulation [15] principle and depending on key matrix that must be invertible to use it in both sides; encryption/decryption. Moreover, not all matrices are invertible so a singular matrix can't be eligible to be chosen as a key matrix. Hill cipher defines its simplicity by using letters frequencies to represent the text message and the results of multiplication are given by a high speed and also high throughput. Encryption: To Encrypt Plain text with a fixed-length of Block size m, the values of P_{nxn} matrix entries are vary between $(0, q - 1)$ included where q represent number of entries, and P must be an invertible

(nonsingular) matrix, Each block of the Plain text matrix also contains entries between (0, q-1) included and represent a vector of n dimension .This process is done using the linear equation 2.10:

$$C = P \times K \, mod \, N \tag{2.10}$$

Decryption: To Decrypt cipher text vector C, we need first to find the inverse matrix K^{-1} to K, where that matrix must be nonsingular. Then we can decrypt the incoming cipher text message using the following equation 2.11 [8]:

$$P = C \times K^{-1} \, mod \, N \tag{2.11}$$

The singular matrix can be converted to non-singular matrix [31], so Hill cipher becomes more efficient because it can encrypt/decrypt any text using the key matrix. The Hil Cipher is stronger than Playfair because it not used the principle of the letters frequencies this will hide these details completely, and as the matrix size increase, more letters and its frequencies are hidden. Despite Hill-cipher is robust against ciphertext it is easy to be broken using plaintext attack.

2.3 Key management Algorithms

The Key management algorithms mainly based on the asymmetric or public key encryption; user must have two type of keys to complete the encryption/decryption process; public key for encryption and private key for decryption between two users or parties like UserA and UserB. UserA encrypt her plaintext using the public key of UserB before sending it to him then decrypt it using its own private key, so that no one can decrypt the ciphertext without the private key of UserA to keep his privacy [10], see Figure 2.6.

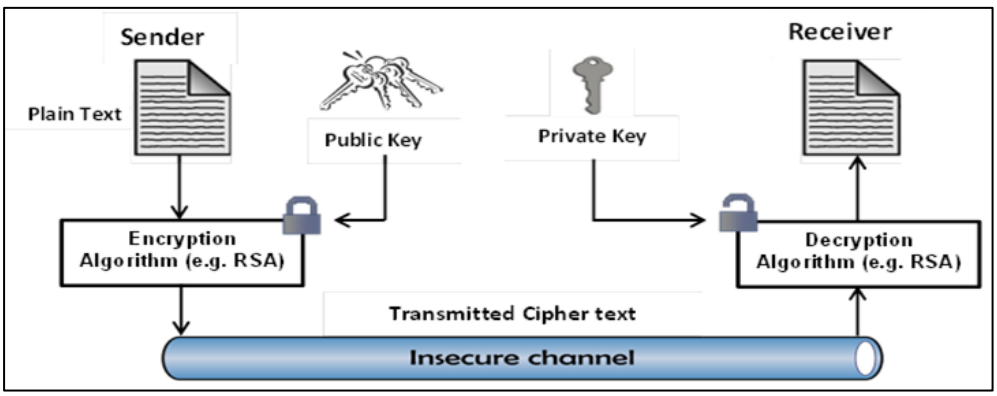

Figure 2.6: A simplified model for asymmetric encryption

There are different methods to implement this type of cryptography system .These are RSA, ECC, Diffie-Hellman and Digital Signature.

2.3.1 Rivest Shamir Adleman Algorithm (RSA)

The difficulty of getting the plaintext message back from the ciphertext and the public key is related to the difficulty of factoring a very large product of two prime numbers. The RSA [10] key Generation can be done using the following steps [26]:

First we choose two large primes p and q, then using the equation 2.12to compute n:

$$n = p \times q \tag{2.12}$$

After that the phi function is solved preparing the e value using the equation 2.13:

$$\phi(n) = (p-1) \times (q-1) \tag{2.13}$$

Select the public exponent $e \in \{1,2,..,\phi(n)-1\}$ based on the relationship specified in the equation 2.14:

$$\gcd(e, \phi(n)) = 1 \tag{2.14}$$

Compute the private key d using equation 2.15:

$$d \times e \equiv \bmod \phi(n) \tag{2.15}$$

Finally, the public key denoted by k_{pub} can be computed using n and e, also the private Key denoted by k_{pr} is equal to d; RSA is used for the encryption/decryption process as the following:

Encryption: The Encryption process can be done using the given: Public Key (n, e) = k_{pub} and Plaintext = P. Then the Encryption function is represented using the equation 2.16:

$$C = E_{pub}(P) \equiv P^e \bmod n. \tag{2.16}$$

Decryption: RSA Decryption process is done using the given: Private Key d= k_p and Ciphertext = C.Then the Decryption function computed using equation 2.17:

$$P = D_{pr}(C) \equiv C^d \bmod n. \quad (2.17)$$

2.3.2 Elliptic Curve Cryptography Algorithm (ECC)

The ECC [27] used the curve principle to generate two distinct keys called the public and private keys, this will be generated using the Elliptic curve properties of points on that curve, a set of these points can be described by the equation 2.18 [12]:

$$y^2 = x^3 + ax + b \quad (2.18)$$

Where: $4a^3 + 27b^2 \neq 0$ (this is required to exclude singular curves) .

There are two group operations of the Elliptic Curve: Point addition: to compute a new point, using a tangent line that is going through the two points P and Q and Point Doubling where the two points are equal P and Q.

2.3.3 Diffie-Hellman Algorithm (DH)

First public-key type scheme proposed by Diffie and Hellman along with the exposition of public key concepts, now know that James Ellis (UK CESG) secretly proposed the concept. It is a practical method for public exchange of a secret key between two parties where each one has the same key using a set of steps to specify both the key generation and distribution principles [29]. Also the attacker can't guess this value because it depends on private keys of the two parties let us called them A and B, the following scenario clarify how the Diffie-Hellman key exchanges work:

First step: prepare global public elements: ***q*** as a prime number and ***a*** is primitive root of q less than q. Second step: the Key Generation for **User A** can be done by selecting private key Kpr_a < q then calculate public key Kpu_a using equation 2.19:

$$K_{pu\_a} = a \times K_{pr\_a} \bmod q \quad (2.19)$$

The same steps are repeated for **User B** and the public key is calculated using equation 2.20:

$$K_{pu\_b} = a \times K_{pr}\_b \bmod q \quad (2.20)$$

Then, the secret key of **User A** can be calculated using equation 2.21:

$$K_s = (K_{pu\_b}) \times K_{pr\_a} \bmod q \quad (2.21)$$

Also, the Calculation of secret key for **User B** can be calculated using equation 2.22:

$$K_s = (K_{pu\_a}) \times K_{pr\_b} \bmod q \quad (2.22)$$

Notice that the same result of secret key is gained by the two cases.

2.3.4 EEC and Diffie-Hellman Key Exchange Cryptography

As mentioned in section 2.3.2.2 the Elliptic Curve is a type of asymmetric public key cryptography [29], it is also a technique for key exchange which include the key generation and distribution phases, as specified in the following steps:

User select a suitable curve Eq(a,b) then Select a base point $G=(x_1,y_1)$ with large order ***n*** such that $n \times G = 0$. Second **User A** and **User B** select private keys $K_{pr\_a} < \boldsymbol{n}$, $K_{pr\_b} < \boldsymbol{n}$.

After that they compute public keys: $K_{pu\_a} = K_{pr\_a} \times G$ and $K_{pu\_b} = K_{pr\_b} \times G$ to prepare the shared secret key between them using equations 2.23 and 2.24:

$$K_s = K_{pr\_a} \times K_{pu\_b} \text{ and } K_s = K_{pr\_b} \times K \quad (2.23)$$

Also the same results in the equation 2.15:

$$K_s = K_{pu\_a} \times K_{pu\_ab} \times G \quad (2.24)$$

Attacker would need to find K_s which hard.

2.4 Cryptosystem models for Cloud Computing

Symmetric and Asymmetric key Cryptography is both used in several areas. Each one has its own role, but it differs from each other through a set of factors that affect its role. These factors are: key length, Rounds, Block size, Security rate, Execution time, see Table2.1 for more details.

Table2.1: Comparison between symmetric and asymmetric Algorithms

Factors	RSA	DES	3DES	AES
Key length	Based on number of bits in N = p× q	56 bits	168 , 112 bits	128,192 and 256 bits
Rounds	1	16	48	10 or 12 or 14
Block Size	variance	64 bits	64 bits	128 bits
Security rate	Good	Not enough	Adequate	Excellent
Execution time	Slowest	Slow	Very slow	More fast

Also, when we compared the RSA and Diffie-Hellman key exchange , each one has its own key length where Diffie-hellman needs more bits so that it's more secure than RSA as a key distribution algorithm , see Figure 2.7 we illustrate that Diffie-hellman needs less time to exchange keys between two parties.

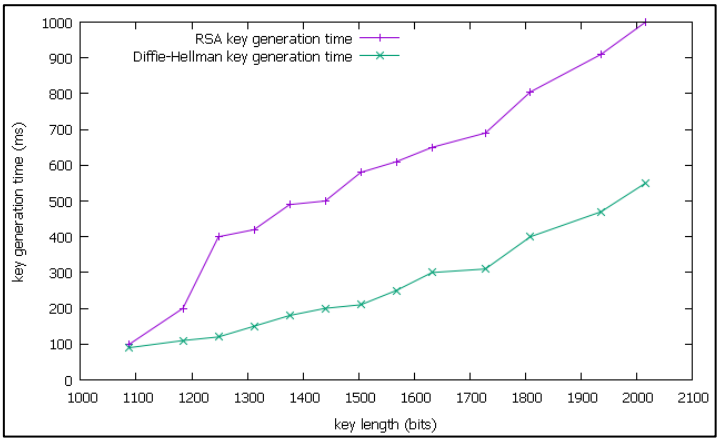

Figure2.7: Comparison between RSA and Diffie-Hellman Key Generation

2.5 Anti- vulnerability methods

This section includes matrix compression method that used in our proposed algorithm with modification and a set of several padding algorithms that support the

security degree of our plaintext, use padded bits will add raw data to the original one that make it robust against any other attacks.

2.5.1 Padding techniques

Padding is used to complete a specified stream of bits in order to make it useful in a specific manner .That is to retrieve data easily after sending them to another party. It is known that the usage of padding was to satisfy the authentication process of the message .This process called Message Authentication Codes (MACS).The recipient checks the padded bits to confirm that it comes from the original sender and no one has stolen these data in order to modify it. There are two types of padding as mentioned below:

1. Bit Padding: is a technique that adds single '1' bit and complete stream of bits with '0's to access the specified block size [54] for example if the plaintext stream of bits b's are :

   … | bbbb bbbb bbbb bbbb bbbb bbb**1 0000 0000** |
   The padded bits are specified with the bold bits that start with '1'.

2. Byte Padding: Used for encoding process when the plaintext message is based on integral number of bytes B's and contains :

   A. Zero Padding: all padded bytes are filled with '0's to complete a specified length of block size [58] , see example below:

   … | BB BB BB BB BB BB BB BB | BB BB BB BB **00 00 00 00** |.

   B. ISO/IEC 7816-4: There are two main types of padding [16]: the old method that adds zero values to the string of bits to access a fixed length. And the other first add one value let it be 80 then fill the remained values with zeros until the needed length accessed as an example ISO/IEC with different versions.

   … | BB BB BB BB BB BB BB BB | BB BB BB BB **80 00 00 00** |.

C. ISO 10126: Random bytes are added at the last byte and the number of padded bytes should be specified with the last byte [56].
   Example:
   … | BB BB BB BB BB BB BB BB | BB BB BB BB **81 A6 23 04** |.

D. ANSI X.923: The plaintexts of bytes are padded with '0's and the final byte represents the number of padded bytes [55].
   Example:
   … | BB BB BB BB BB BB BB BB | BB BB BB BB **00 00 00 04** |

E. PKCS7: padding is done on all bytes the number of padded bytes represent the value that padded through it [57], so that the padded bytes can be one of the following forms:
   … | BB BB BB BB BB BB BB BB| BB BB BB BB **04 04 04 04** |.

2.5.2 Compression

Is a type of lossless compression of numerical matrices by a representing data in a new mathematical model by converting number into stream of binary bits of a variable lengths to improve the operations on the original data with minimum computational costs by minimize the needed space in memory and the operation time.

Nowadays, in the digital computers all integer values are stored as a stream of binary bits (base 2) and the block size (chunk) is take a power of two, numbers of bits. Most of the modern computers are 64 bits word length. Thus, the total memory size allocated for the matrix P_{mxn} is $64 \times m \times n$.

We can retrieve matrix elements using the row and column indices and this will waste more overheads between the memory and the processor, so that each row is composed only using one decimal value. This means that the m×n matrix will be converted to column vector.

There are many methods to solve this compression process , either for static block size along the matrix depends on the minimum number of bits needed to represent the maximum number between a set of values through the first row , this method is called The Supreme Minimum (SM) or using a variable length block (VLB) [24].

The Supreme Minimum (SM): This is a matrix compression method depends on the value of the greatest element in the matrix represented by the minimum number of bits and called it the supreme, see 2.25, 2.26.

$$P = \begin{bmatrix} 20 & 660 & 11 & 256 & 2 & 721 & 700 & 500 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.25}$$

$$b(supM) = \begin{cases} 1, & \text{if } supM \in \{0,1\} \\ \lfloor \log_2(supM) \rfloor + 1, & \text{if } supM > 1 \end{cases} \tag{2.26}$$

Table2.2: Elements Binary representation according to 2.25

Element	Value	Binary	Bit length
P1,1	20	10100	5
P1,2	660	1010010100	10
P1,3	11	1011	4
P1,4	256	100000000	9
P1,5	2	10	2
P1,6	721	1011010001	10
P1,7	700	1010111100	10
P1,8	500	0111110100	10

Bitstream1=

$$\underbrace{1100}\ \underbrace{000000101}_{11}\underbrace{100000000}_{2}\underbrace{100000000}_{256}\underbrace{101101000}_{721}\underbrace{101001010}_{660}\underbrace{000111110100}_{500}$$

Bitstream2=$\underbrace{101011}_{b}\ \underbrace{1100}_{a}\ \underbrace{0000001011}_{11}$

Each bit stream can contain only 64 bit so that the element P1, 7 is divided into two parts: a located on the first stream of bits and b located on the second stream of bits. Let you see the following complete example of the process:

Assume:
$$A = \begin{bmatrix} 7 & 14 & 5 & 8 \\ 12 & 3 & 6 & 9 \\ 3 & 1 & 10 & 11 \end{bmatrix} \quad (2.27)$$

$$(AA)_2 = \begin{bmatrix} 0111 & 1110 & 0101 & 1000 \\ 1100 & 0011 & 0110 & 1001 \\ 0011 & 0001 & 1010 & 1011 \end{bmatrix} = \begin{bmatrix} 0111111001011000 \\ 1100111001101001 \\ 0011000110101011 \end{bmatrix} = \begin{bmatrix} 32344 \\ 52841 \\ 12715 \end{bmatrix} \quad (2.28)$$

Ternary system is used instead to make the compression process harder to guess than binary, notice that in some cases, switching the conversion system to Ternary system will retrieve smaller values, but not always, using a larger matrices sizes that depends on the increasing number of columns, so that number of digits increase and using the Ternary system the decimal number will be represented using three possible values 0, 1 and 2, see equation 2.29.

$$(AA)_3 = \begin{bmatrix} 21 & 112 & 12 & 22 \\ 110 & 10 & 20 & 100 \\ 10 & 1 & 101 & 102 \end{bmatrix} = \begin{bmatrix} 211121222 \\ 1101020100 \\ 101101102 \end{bmatrix} = \begin{bmatrix} 16496 \\ 27144 \\ 7571 \end{bmatrix} \quad (2.29)$$

2.6 Summary

Our proposed model SSCC is based on the original Hill Cipher using matrices principle, where the encoded plain text is added into a matrix and multiplied by the key matrix which contains the secret key of that plain text.

The original version of Hill cipher use two nxn matrices that filled completely by entries. The model adds a new feature that the matrices has their own properties to make it difficult to be broken by the attacker, that's why we create an upper triangular matrix for the input secret key and a lower triangular matrix for the input plain text and multiply them to attain the ciphertext.

Chapter Three

Cloud Computing Architecture: Background and definition

Contents

3.1 Introduction

Cloud Computing is an environment that composed of a set of central remote servers that communicate with each other through the internet network to serve either data or applications . It provides the accessibility of their resources while the internet is available to their users that can be ordinary or business users. Cloud Computing apply the availability concept of its resources where these resources (Storage, memory, Processing and bandwidth) are centralized to support a more efficient environment [28].

The outsourcing of data to cloud computing environment and deal with a third party that may be known or not will leads to a security issue where data is processed in a global scope, so that this issue must be solved using a cryptosystem model that keep data in the encryption mode before they uploaded to the Cloud storage .User can access their data through a specific name and make its own modifications (upload, edit) within a virtual server that view all data as a one unit and on the same location this is what is called the transparently concept .

Cloud Storage is the main component in the cloud computing environment where that data and applications located on multiple servers that is may not be dedicated on the same network.

The location where that data placed doesn't exists as the user see, it represent on the reality view by a set of computers that connects together through a network to form the cloud. A dynamic change of that location will occur periodically and affect the actual used storage, but user don't see these changes and work with it like a static location on his own computer and can easily make modification on his data.

A management system controls the Cloud storage architecture and composed of a master server that controls a set of storage servers. According to this level of abstraction only one of data servers is connected to internet.

3.2 Cloud Computing Models

In this section, a set of cloud computing architectures are mentioned briefly to clarify the differences between them and show who it works by view their steps and who it support the security concerns to be robust enough against the unauthorized access from the unknown third parties . The selected Cloud computing architecture is depends mainly on the purpose of that cloud storage and the needed security level, so that critical data or application needs a very robust system to ensure that it is safe enough and to gain the satisfaction of the end users or organizations.

Cloud computing environment mainly composed of five separated components [40], the two major components are: First is the **Cloud Service consumer**: represented by a company that interested in the cloud based solutions, it requests a service either from the Cloud Service Provider or from the Cloud broker; and second is the **Cloud Service Provider (CSP)**: provides a set of layers consists of :Service Layer, Resource abstraction and control layer, physical resource layer, Cloud service management, privacy and security layers.

Also, Cloud computing has **Cloud Broker**, **Cloud Auditor** and **Cloud carrier**; each one plays its role to satisfy the major aim of the Cloud computing principle like resource sharing and other services, Cloud Broker integrate these cloud services it works as a cloud provider when act with the consumer this role is supported by the Cloud Auditor which check the privacy and security concerns, finally the cloud carrier makes the connectivity between the consumer and the CSP.

The first three zones are consists of the Cloud service provider (SCP) within its domain and the last one is an external zone that can be accessed from the outside see Figure 3.1.

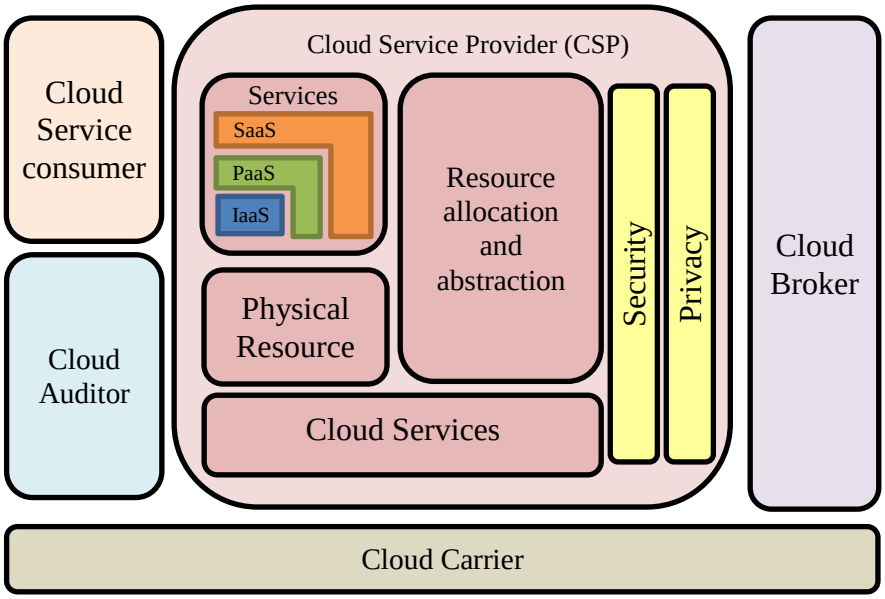

Figure 3.1: NIST Cloud Architecture design [21]

3.2.1 Information Dispersal model (ID)

In [20] the period of 'Big Data', computerized substance is created at an uncommon pace by organizations and users alike. Usage of IT has driven up the interest for portable storage devices (PSD), they are additionally seeing an exponential development in the utilization of individual cloud storage administrations. PSD, in spite of the fact that advantageous to use because of it's fitting in plug-and-play, is regularly subject to loss and theft.

Secret sharing or information dispersal model (ID) slices information into n number of slices and scatters the slices to large numbers of storage nodes. Without adequate number, m, of slices, the information can't be recreated. These methods depend on (m, n) threshold plan. An enemy needs to take in any event m of the n cuts to recreate the file. With less than m cuts, the attacker gets no data. By giving secrecy without encryption, we can utilize a more adaptable and secure authentication based security framework that can be effectively changed in light of new dangers and vulnerabilities. Interestingly, evolving encryption plan involves the regularly inconceivable errand of re-encryption furthermore, keeping up an excessively complex key generation framework.

This architecture is based on using a single secret key that can be found using public key of User B called K_{pub_b} and private key of User A called $K_{pr\_a'}$ to complete the process a key exchange will occur between two parties as shown in Figure 3.2, where:

ID_A: Identifier of A.
N_1, N_2: nonce to identify this transaction uniquely.

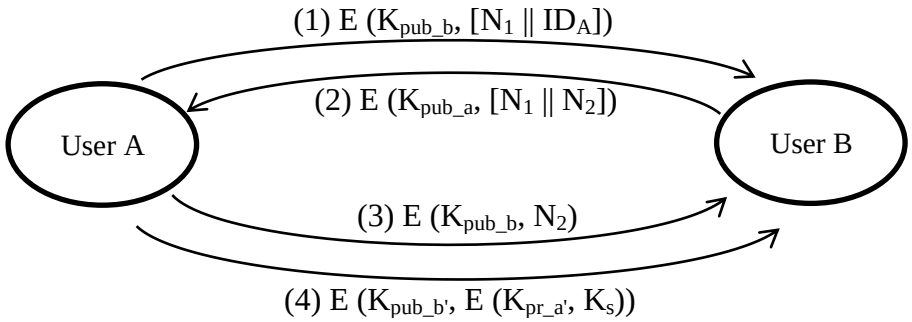

Figure 3.2: ID key exchange

3.2.2 Lookup table based Secure Cloud Computing model (LUT)

In [34] a great deal of cryptographic procedures have been proposed to reduce the data security issues in cloud , however the greater part of these works concentrate on comprehending a particular security issue, for example, data sharing, correlation, searching, and so forth. Using the LUT of the FPGA (field-programmable gate array), we can produce a more secure cloud storage for both data and program depending on the k-meaning clustering.

The evaluation of every function in the computer infrastructure is done within a Look up table scheme. This scheme work as a traditional table which consists of a set of elements (entries) and each one has two main properties (index ,value); where indices is used to represent inputs and values used to simplified output .

Several types of functions can be converted into this LUT pattern easily like constant functions, successor functions or any other primitive functions. Also, if the function contains more than one type at the same formula, we can decompose it into partial functions and solve each of them separately then composed them again to get the final result.

The conversion process of the function into the LUT scheme need to collect all the possibilities of the input set, then apply them into the function to calculate their results that represent the output set .The number of inputs may be greater than two inputs, this means that we need a vector to represent the input element which can directed link to multiple output.

This architecture based on key exchange of public keys PK_A , PK_B between two users and parameters of each user called $K_{Aparameter}$ and $K_{Bparameter}$ as shown in Figure 3.3, User A and User B retrieve their parameters from the Trusted 3rd party, then send their encrypted public keys to each other.

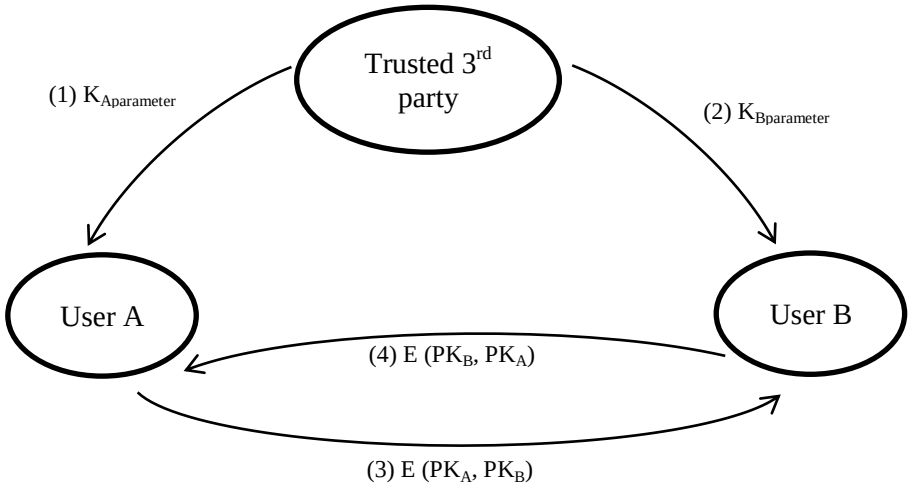

Figure 3.3: LUT Key exchange

3.2.3 DNA Matching model

In [19] Cloud computing aims to satisfy a set of features like performance, availability, scalability and low cost at the same time within low processing efforts at the client side and handle it on the cloud side. User needs only a simple input/output device and the most of the computation process is done on the cloud infrastructure. This algorithm supports a hybrid Cloud computing architecture using the DNA matching.

A more secure cloud is built using DNA matching system within garbled circuit evaluation technique and RSA to improve the overhead to the cloud storage in the public and private cloud and decrease the needed execution time.

User data is represented by a DNA sequence that sent to the private Cloud and the encryption process is done using the RSA algorithm that generate the cipher string is processed to Garbled Circuit to gain the garbled sequence of the input DNA sequence, then it is moved to public Cloud and a comparison matching is done between this sequence and the original DNA sequence using Edit distance algorithm to complete the evaluation process. The Decryption process is done back on the private Cloud after the completeness of the verification process using matching principle. Figure 3.4 illustrates the key exchange process between two parties and via a Trusted Third Party User A and B request their public keys PU_A, PU_B from it and each request has an ordered number $Request_1$, $Request_2$, … and so on.

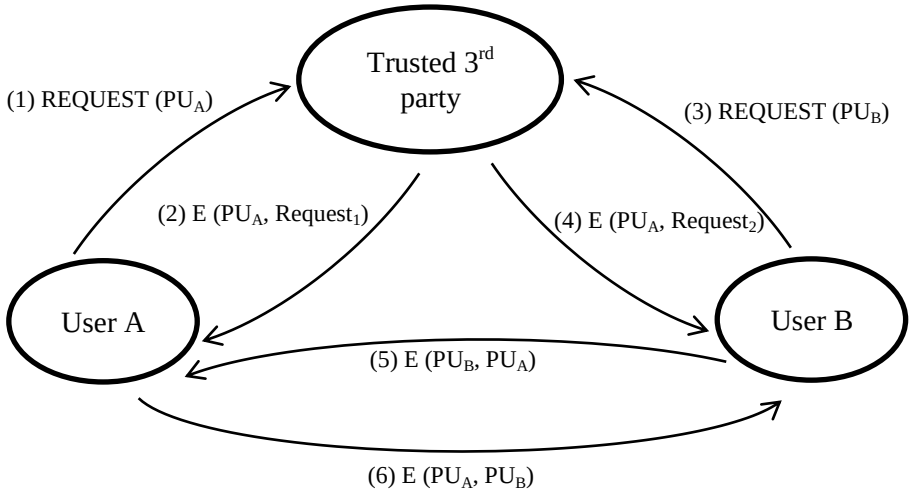

Figure 3.4: DNA Key exchange

3.2.4 Client-Server User Authentication and Encryption Model

In [59], authors proposed a new model for cloud computing environment that do the encryption process before uploading to the cloud storage using RSA algorithm output small e to share an encrypted key that keep the transparent between using two different keys between parties represented by client server architecture, then it check the authentication degree using modified Diffie-Hellman key exchange to work with only one secret key in order to authenticate and validate each other.

The authentication leads to identity the legal users and control there privileges to access the shared resources. This model will add features to data protection, and security of the transmitted data which support the cloud computing services [60], Figure 3.5 illustrates the key exchange between two parties User A and B and secret shared encrypted secret key K_s using small integer value e is exchanged.

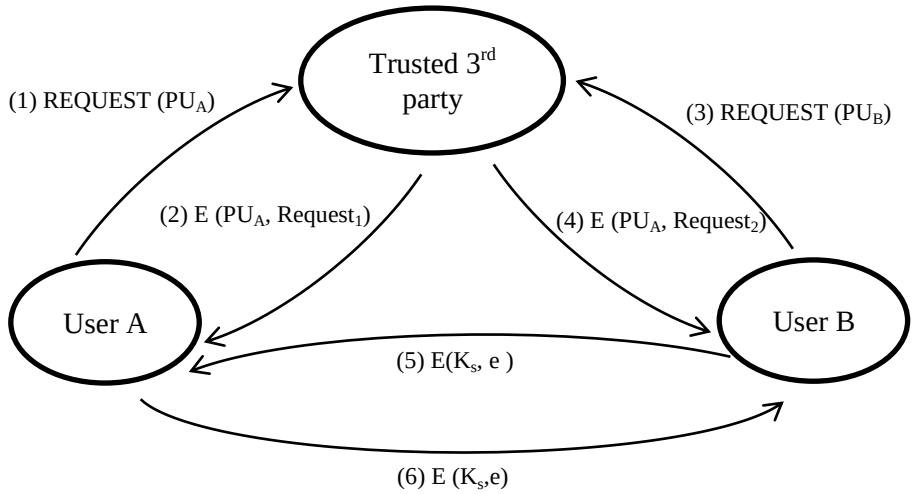

Figure 3.5: Client-Server User Authentication and Encryption model key exchange

3.3 Summary

In this chapter, we introduce the general form of Cloud Computing Architecture, and then several Cloud Computing architectures are mentioned to clarify the variety in the Cloud computing environment and demonstrate the key exchange process for each one separately.

In our model we generate a composite secret key that will be distributed across the key matrix, but searching through a key of 100 bit still harder than 4 x 25 bit where the key is divided into two parts for example, so we padded the key using random bits to decrease this problem.

Chapter Four

Proposed Model:

Cryptosystem model for Secure Data Sharing in Cloud Computing (SSCC)

Contents

4.1 Introduction to SSCC

In this chapter we introduce our proposed algorithm (SSCC) that's a Cryptosystem model for Secure Data Sharing in Cloud Computing which has two inputs (Plain text, Secret Key) and one output (Cipher text) as a result of the encryption process. The plain text can be represented by File = f $[m_1,m_2,\ldots m_n]$ where f is the file from the user data where each users can store a lot of files into the cloud storage and each file composed of a set of blocks $m_1,m_2,\ldots m_n$ and n is the number of the last block in this file .

The other input is the secret key (K_s). The user can obtains its key from a trusted third party let us called it CaaS, which plays its role by request a new secret key from a key Generator (keyGen) for each user that can be either a new user or old users. So, a new key is retrieved for this purpose.

Our model SSCC communicate with the Cloud provider and send the encrypted part of file to it as $x_i = E(k_s,f)$ so each file f is encrypted using user own key k_s and stored in x_i to send directly to the cloud provider which directed it to the cloud storage as the whole file X , you can follow the scenario illustrated in Figure 4.1.

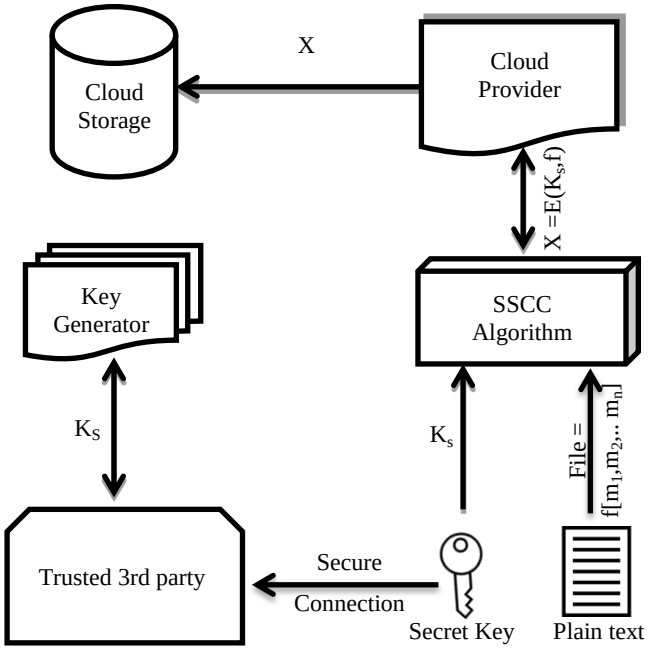

Figure 4.1: SSCC Block Diagram

SSCC uses matrices manipulation principle to produce ciphertext .This algorithm uses two matrices: plain text (P) and key (K) as a type of triangular matrices to gain i.e., benefit from their properties. The key matrix (k) contains the secret key of the owner where each user has his own unique key to support high security guarantees for the data file sharing systems through low management and maintenance costs of the outsourced data files. The users' memberships will change frequently, so we need a robust system to handle these users and save them from the collisions that may occur by untrusted cloud users.

Also, we need a secure communication channels and a group manager Certificate Authorities that provide private keys for dynamic groups in the cloud, when a new member wants to join or revoked from this group, the private keys of the other members will not be changed .And the revoked member can't access to any data files if it come from untrusted third party [38].

SSCC Algorithm

Our SSCC Algorithm has eight sequential steps to transform the plain text message and then upload it to the Cloud storage as a cipher text file. The phases of the SSCC algorithm are as the following:

1. **Input**: User input a plaintext file (M) and a secret key that considered a password for his data (Ks). This key can be gained also from a trusted third party to make the process portable.
2. **Encode**: the plaintext encoded such that each character is represented using a decimal integer value starts from 1 up to 26. To keep the matrix in the upper triangular form we discard the 0 value, let us called these resultant values E. Then E is filled into matrix of m × n [24], see Table4.1 which illustrate the values of alphapatic characters that we used in our testing.

Table4.1: Character Encoding

Character	Value
a	1
b	2
c	3
d	4
e	5
f	6
g	7
h	8
i	9
j	10
k	11
l	12
m	13
n	14
o	15
p	16
q	17
r	18
s	19
t	20
u	21
v	22
w	23
x	24
y	25
z	26

3. **Split**: plain text is split into n numbers of sub blocks (m_1, m_2 , ….. m_n) , depending on a blockSize value (B).
4. **Compress**: Each sub block is converted into its ternary value then a composition of this stream of values is converted to one decimal value that represents a block.

These set of blocks is inserted into an upper triangular matrix, see the Figure below that explain the compression process completely:

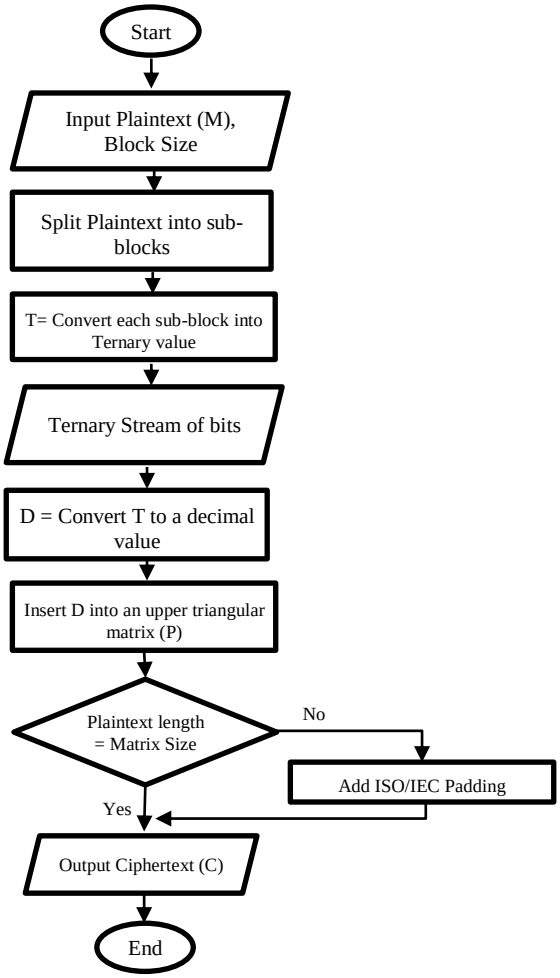

Figure 4.2: Compress phase

5. **Pad**: The zeros elements in this matrix is replaced with padding values using ISO/IEC padding technique to complete the matrix , i.e., all elements above the diagonal are nonzero elements.
6. **Generate and Result**: The used key for the encryption is by utilizing several algorithms. A keyGen uses the Diffie-hellman key exchange principle and then stores it into a trusted third party. The value is unique for each user, and then the key matrix is prepared to be used with the plaintext matrix.
7. **Encryption** : The process of encryption is represented by the multiplication between two matrices; Plaintext(P) and Key (K) to generate a third matrix called Cipher text (C) ,then decode the result matrix in order to prepare the ciphertext file.
8. **Output**: ciphertext (c) is uploaded to the cloud storage.

These 8 phases are done sequentially where the input of each phase depends on the output of the previous phase except the first one. The forward direction through these phases result the Encryption process sees Figure 4.3.

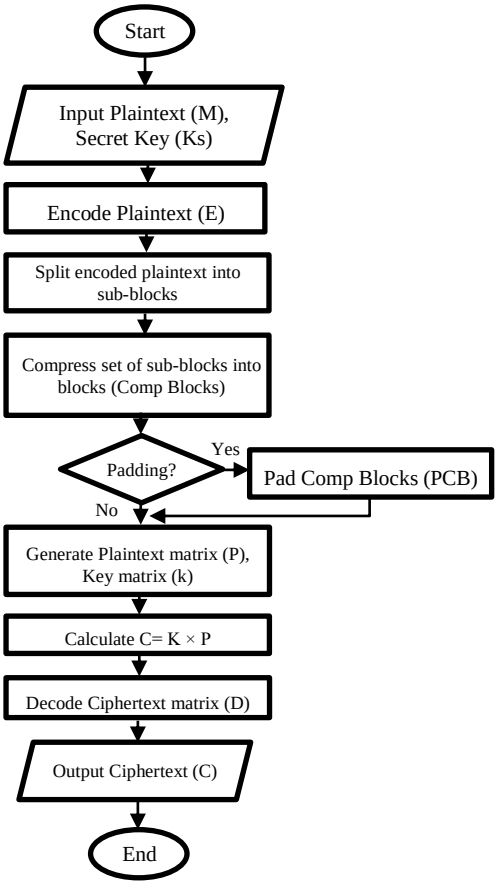

Figure 4.3: Encryption process flow chart

Pseudo code at the UPLOAD to the Cloud storage (Encryption)

M = readFile("Plaintext.txt")

E=encode(M) % a=0 , b=1 , ……. , z=26 and fill them in E_{mxn} *matrix*

T= dec2ter(E) % convert decimal values to values of base 3values

B= split(T, BlockSize) % split into equal Sized sub-blocks

compBlocks = ter2dec(B , n) % convert form ternary to decimal

K= upperTriang(key) %insert key vector into upper triangular matrix

pcb = padCompBlocks(compBlocks) % add padding if needed

P = lowerTriang(pcb) % insert pcb vector into lower triangular matrix

R=K × P % generate the ciphertext matrix

C = decode (R) % convert decimal values to characters

uploadToCloud(C,"ciphertext.txt")

Algorithm 4.1: Algorithm for Encryption process for SSCC1

Also the reverse direction within these phases represents the decryption process see Figure 4.4.

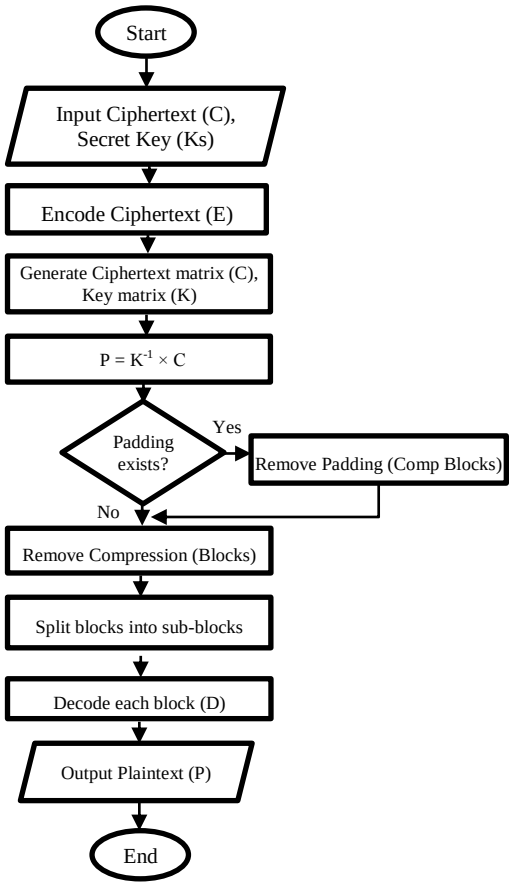

Figure 4.4: Decryption process flow chart

Pseudo code for the DOWNLOAD from the Cloud storage (Decryption)

```
C= downloadFromCloud("ciphertext.txt")
E=endcode( C )
padCompBlocks=lineSolve(E,K) %matrices L U decomposition
pcb = extractVector(padCompBlocks)
comBlocks=removePadding(pcb)
T=dec2ter(comBlocks)
B=split(T,blockSize)
E=ter2Dec(B)
P=decode(E)
SaveTo(P,"paintext.txt")
```

Algorithm 4.2: Algorithm for decryption process for SSCC1

Our model has two techniques to be accomplished that vary by the plaintext matrix form. The first technique is called SSCC1 which has an upper triangular matrix

with zero elements above the diagonal. The second technique is called SSCC2 which has non-zero elements for all entries in this matrix based on the LU Factorization principle as mentioned in the previous section. The next two sub-sections provide more details about them.

4.2 First technique SSCC1

The first technique of SSCC is called SSCC1. This Technique implies the last 8 phases with more details related to the form of the needed matrices. Recall that we have two matrices. First is for the secret key k_s called K matrix with nxn size and lower triangular matrix properties where all elements over the diagonal are zeros. The second matrix hold the numeric values of the plain text and called P matrix with size nxn and in the upper triangular matrix properties.

The output of this technique can be computed by the multiplication of the last to matrices to get the ciphertext and called it C matrix of n x n size.

Triangular matrices are needed to gain the benefits from the LU decomposition theory in the linear algebra and that's done through the LU factorization process to retrieve either the plain text or the key matrix. And as the key matrix in the lower triangular form, this will force it to be always invertible as we fill it in that form to emphasis the non-singularity of the Key matrix. The Encryption / Decryption process will work well and in the forward and reverse directions due to the proprieties of LU decomposition form.

4.2.1 Mathematical model of SSCC1

In this section we focus on the model that used for two different stages; UPLOAD and DOWNLOAD.

UPLOAD encrypted file to the Cloud storage (Encryption)

User needs to follow the previous steps in the forward direction to complete the process of uploading his data to the cloud storage.

Let the key matrix (K), $K=\begin{bmatrix} k_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix}$

And the plaintext matrix (P), $P=\begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \dots & P_{nn} \end{bmatrix}$

Then;

$$\begin{bmatrix} k_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix} \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \dots & P_{nn} \end{bmatrix} = \begin{bmatrix} C_{11} & \dots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \dots & C_{nn} \end{bmatrix}$$

$$K \times P = C \qquad (4.9)$$

Where,

K: nxn lower triangular matrix that contains the secret key for one user. $K_s = [k_1 \ k_2 \ \dots\dots \ k_m]$; where k_{si} is an integer value represent a subpart of that key.

P: nxn upper triangular matrix that contains encoded and compressed data, so that each element represents a block of data, using the following formula:

$$p_{ij} = \text{compression}(\text{encode}(\text{pcb})) \qquad (4.10)$$

Where,

pcb is a padded compression block.

encode (pcb) to convert it into integer values rather than characters

i = 1.. n , j = 1.. n.

C: nxn Cipher matrix.

DOWNLOAD encrypted file from the Cloud storage (Decryption)

If Ciphertext matrix (C), $C = \begin{bmatrix} C_{11} & \dots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \dots & C_{nn} \end{bmatrix}$

User has the key and the incoming cipher matrix; so P can be easily found by solving a system of linear equations to find K^{-1}.

$$C = K \times P \tag{4.11}$$

Where:

p_{ij} as mentioned in equation 4.10.

Example for SSCC1

UPLOAD to the Cloud storage (Encryption)

Let user needs to upload a file contains the following plaintext:

Plaintext = "helloisraa"

Then each character is encoded to its value according to Table4.1:

$$\text{Blocks} = \begin{bmatrix} 8 & 5 & 12 & 12 & 15 \\ 9 & 19 & 18 & 1 & 1 \end{bmatrix}$$

Each entry in the Blocks matrix is converted from Decimal to Ternary value to generate a matrix let us called it T:

$$T = \begin{bmatrix} 022 & 012 & 110 & 110 & 120 \\ 100 & 201 & 200 & 001 & 001 \end{bmatrix}$$

Each row entries in matrix T are column wise merged to represent a decimal value to generate a new matrix let us called it T':

$$T' = \begin{bmatrix} 022100012201110 \\ 200110001120001 \end{bmatrix}$$

Each element in matrix T' is converted back from Ternary to decimal value added to a matrix called CompBlocks:

$$\text{CompBlocks} = \begin{bmatrix} 4432845 \\ 9803269 \end{bmatrix}$$

The matrix CompBlocks entries are added to another matrix P in an upper triangular form and if any entry missed it padded to complete the matrix using ISO/IEC padding as mentioned in section 2.5.1:

$$P = \begin{bmatrix} 4432845 & 9803269 \\ 0 & 80 \end{bmatrix}$$

The secret key is generated using MKG as mentioned in section 4.4 and returned in a decimal value K_s as example:

$K_s = (9)_{10}$

Convert the secret key decimal value to binary stream of bits K_s':

$K_s' = (1001)_2$

Pad the K_s' with random 0's and '1 to complete a suitable key length for the matrix K which must be with the same size of the Plaintext matrix P to generate K_s'':

$K_s'' = (100101100)_2$

The resultant stream of bits are read from left to right then divided into sub keys of only three bits:

SubKeys ={1, 5, 1}

The SubKeys elements are inserted into a lower triangular matrix K:

$$K = \begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}$$

The ciphertext matrix is gained as a result of the multiplication operation of K and P matrices:

$$C = \begin{bmatrix} 4432845 & 9803269 \\ 22164225 & 49016425 \end{bmatrix}$$

DOWNLOAD from the Cloud storage (Decryption)

The ciphertext matrix returned from encoding the downloaded text file from the Cloud Storage, notice that we must avoid using 1x1 and 2x2 matrix, because the first row will be remain the same as you see in the following example:

$$C=\begin{bmatrix} 4432845 & 9803269 \\ 22164225 & 49016425 \end{bmatrix}$$

Obtain the key matrix from a trusted third party and insert it into a K matrix as mentioned in the UPLOAD stage:

$$K=\begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}$$

After Linear Solve using LU Decomposition, using the ciphertext and Key matrices to get the plaintext matrix P:

$$P=\begin{bmatrix} 4432845 & 9803269 \\ 0 & 80 \end{bmatrix}$$

This matrix P contains compressed blocks of data with padding, we need to remove the padded entries and insert entries without padding into column vector called CompBlocks:

$$CompBlocks=\begin{bmatrix} 4432845 \\ 9803269 \end{bmatrix}$$

Convert the entries in the CompBlocks from Decimal to Trinary values called T:

$$T=\begin{bmatrix} 022100012201110 \\ 200110001120001 \end{bmatrix}$$

Split the stream of bits into 3 by 3 elements called it T':

$$T'=\begin{bmatrix} 022 & 012 & 110 & 110 & 120 \\ 100 & 201 & 200 & 001 & 001 \end{bmatrix}$$

Convert each value back to its original Decimal value to generate Blocks matrix:

$$Blocks=\begin{bmatrix} 8 & 5 & 12 & 12 & 15 \\ 9 & 19 & 18 & 1 & 1 \end{bmatrix}$$

Decode each decimal value to get the characters of our text, and then return the whole plaintext as an output:
Plaintext = "helloisraa"

4.3 Second technique SSCC2

The Second technique of SSCC is called SSCC2 implies the last 8 phases with more details related to the form of the needed matrices, as we have two matrices; First is for the secret key k_s called K matrix with nxn size and square matrix properties where all elements are nonzero, the second matrix exists to hold the numeric values of the plain text and called P matrix with size nxn and also a square matrix properties represented by the nonzero for all elements.

The output of this technique can be computed by the multiplication between the last to matrices to get the cipher text and called it C matrix with also nxn size where C = K P. The key matrix at this technique is needed to be an invertible matrix ; this can be done be check the determinant of the matrix at the end of the filling process of elements if it is equal to zero a new key matrix will be generated that satisfied that condition .

4.3.1 Mathematical model of SSCC2

In this section we illustrate our model using the second technique SSCC2 for both stages UPLOAD and DOWNLOAD.

UPLOAD to the Cloud storage (Encryption)

User needs to follow the previous steps in the forward direction to complete the process of uploading his data to the cloud storage as mentioned for SSCC1 but the matrix format change to be as a fulfilled form.

Let Key matrix (K), $K=\begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix}$, and the plaintext matrix (P) is

$$P=\begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \dots & P_{nn} \end{bmatrix}$$

Then,

$$\begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix} \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \dots & P_{nn} \end{bmatrix} = \begin{bmatrix} C_{11} & \dots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \dots & C_{nn} \end{bmatrix}$$

$$K \times P = C \tag{4.11}$$

Where,

K: nxn lower triangular matrix that contains the secret key for one user. K_s = [k_1 k_2 ……. k_m]; where k_i is an integer value represent a subpart of that key.

P: nxn upper triangular matrix that contains encoded and compressed data, so that each element represents a block of data, as mentioned in the equation 4.10.

DOWNLOAD from the Cloud storage (Decryption)

User download the text file from, it converted to ciphertext matrix using the encoding process then,

$$C = \begin{bmatrix} C_{11} & \dots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \dots & C_{nn} \end{bmatrix}$$

User has the key and the incoming cipher matrix; so plaintext can be easily found by solving a system of linear equations, See equation 4.11.

Example for SSCC2

UPLOAD to the Cloud storage (Encryption)

Plaintext = "helloisraa"

Blocks = $\begin{bmatrix} 8 & 5 & 12 & 12 & 15 \\ 9 & 19 & 18 & 1 & 1 \end{bmatrix}$

T = $\begin{bmatrix} 022 & 012 & 110 & 110 & 120 \\ 100 & 201 & 200 & 001 & 001 \end{bmatrix}$

T = $\begin{bmatrix} 022100012201110 \\ 200110001120001 \end{bmatrix}$

Comp Blocks = $\begin{bmatrix} 4432845 \\ 9803269 \end{bmatrix}$

P = $\begin{bmatrix} 4432845 & 9803269 \\ 80 & 0 \end{bmatrix}$ // Padding is started from the first entry of the 2nd row

$K_s = (9)_{10}$

$K_s = (1001)_2$

K_s with padding =$(100101100)_2$

Sub Keys ={1 , 5 , 1 }

K= Lower matrix = $\begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}$

C= $\begin{bmatrix} 4432845 & 9803269 \\ 22164305 & 49016345 \end{bmatrix}$

DOWNLOAD from the Cloud storage (Decryption)

C= $\begin{bmatrix} 4432845 & 9803269 \\ 22164305 & 49016345 \end{bmatrix}$

K= Lower matrix = $\begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}$

After Linear Solve using LU Decomposition:

P =Upper matrix = $\begin{bmatrix} 4432845 & 9803269 \\ 80 & 0 \end{bmatrix}$

Comp Blocks = $\begin{bmatrix} 4432845 \\ 9803269 \end{bmatrix}$

T = $\begin{bmatrix} 022100012201110 \\ 200110001120001 \end{bmatrix}$,T = $\begin{bmatrix} 022 & 012 & 110 & 110 & 120 \\ 100 & 201 & 200 & 001 & 001 \end{bmatrix}$

Blocks = $\begin{bmatrix} 8 & 5 & 12 & 12 & 15 \\ 9 & 19 & 18 & 1 & 1 \end{bmatrix}$ = Plaintext = "helloisraa"

At the DOWNLOAD from the Cloud storage (Decryption)

```
C= downloadFromCloud("ciphertext.txt")
E=endcode( C )
padCompBlocks=lineSolve(E,K) %matrices L U decomposition
pcb = extractVector(padCompBlocks)
comBlocks=removePadding(pcb)
T=dec2ter(comBlocks)
B=split(T,blockSize)
E=ter2Dec(B)
P=decode(E)
SaveTo(P,"paintext.txt")
```

Algorithm 4.3: Algorithm for decryption process for SSCC2

4.4 Matrix Key Exchange model MKG

Matrix key generator (MKG) is a proposed new function that we developed to generate several secret keys for the SSCC model; it is work as an aided subsystem to our main cryptosystem model, where each user has his own unique key to encrypt his data. This MKG takes its input from the Diffie-Hellman algorithm that output a secret unique key and distributes it between two different parties (Trusted 3rd party and the key Generator).

This secret key K_s also padded with extra random bits to make it with the suitable length as a power of two stream of bits then it is divided into sub-key then inserted as an entries inside the key matrix, this mean that a composite key is used to encrypt the given plain text matrix which need a hard effort from the attacker to be hacked, see Figure 4.6 that demonstrate the abstraction view of the key exchange process.

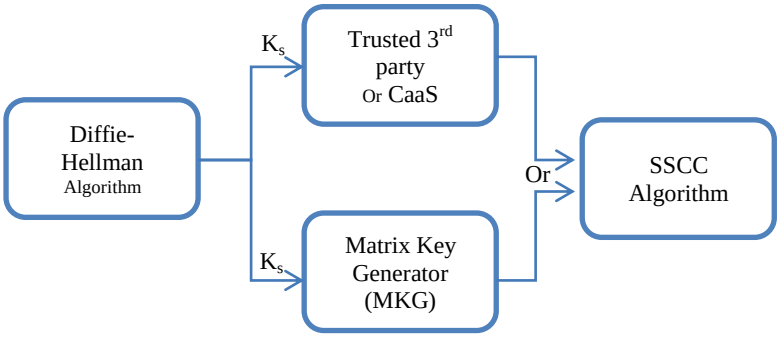

Figure 4.6: Block Diagram for Key Exchange process

Pseudo code for MKG

In MKG the secret key has several lengths which is related mainly to the matrix size, as the matrix size increase the key length must be increased to protect a

convenient security degree, a random padded stream of bits are added to access the needed length and support the security of the secret key.

Table4.2: Secret Key length

n x n A matrix size	*A matrix number of elements	Key length (bits) Without padding	Key length (bits) With padding
1x1	1	8	16
2X2	3	24	32
3x3	6	48	64
4x4	10	80	128
5x5	13	104	128
6x6	21	168	256
7x7	28	224	256
8x8	36	228	256
9x9	45	360	512
10x10	55	440	512

*Note that the number of non-zero elements in the triangular matrix with a specified size nxn is calculated using:

NumberOfElements = $\sum_{i=1}^{n} i$, where n is number of rows or columns because we work with square matrices forms.

Based on the results on Table 4.2, the results show that increase the value of n which represent nxn or the size of the matrix lead to increasing the key length, see Figure 4.7:

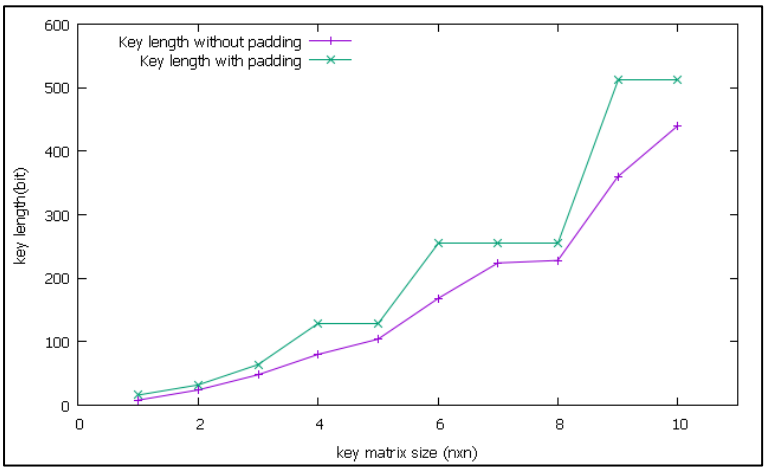

Figure 4.7: Relationship between n and key length

Pseudo code is used to show the main steps of our new algorithm MKG as the following:

Algorithm: MatrixKeyGenrator(Ks , matrixCapacity)

Input: A secret Key (Ks) from Diffie-hellman algorithm.
Output: A lower triangular matrix with size nxn called a key matrix

Ks' = dec2bin(Ks) // convert the secret key to its binary value

While length(Ks') < matrixCapactiy do
Ks' = Ks' + random(1) // add padding to the secret key to access the specified length
K = splitKey(Ks' , 3) // divide the secret key into parts containing only 3 bits
K' = bin2dec(K) // Convert the resultant parts into their decimal values
Output = fillMatrix(K') // insert the decimal values into the key matrix as a lower triangular form

return Output

Algorithm 4.4: Algorithm for matrix key generator (MKG)

4.5 Summary

In this chapter we present our model. In the first section, we provide an introduction that well clarifies the main idea about it. The second section, an algorithm of our model explains the main steps (phases) as each one was clarified carefully. We also mentioned the two main techniques of our model named SSCC1 and SSCC2 respectively; each technique has its own mathematical model that is introduced to be different from the other.

Finally, a general pseudo code is written to clarify the sequence of steps as functions and parameters, to make it easy to follow during reading it. Our model was illustrated by employing a set of different ways to understand and to ensure the simplicity of the given algorithm. Besides its correctness to prove that it is good enough to be used later.

Chapter Five

Simulation and Testing

Contents

5.1 Simulation results for Encryption process

In this chapter we present the results and discussion our simulation in an Intel ASUS P6100 laptop, 2.00GHz 2 core(s), 4GB RAM/ Microsoft windows7/MATLAB 7.10.0 simulator.

A different file sizes are chosen and fed to the algorithm to generate a cipher text for the input file and then calculate the processing time for each case. First, we start with small file sizes to calculate the capacity for different matrices, so we start with 1x1 to 10x10 matrices to calculate the processing time needs for the encryption process; these matrices can be either an upper triangular matrix represented by the 1st technique (SSCC1) or a fulfilled matrix represented by the 2nd technique (SSCC2).

Our simulation was covered two cases, first when the compression phase is included in our model and applied for the two techniques separately and the second was when we skip the compression phase mainly from our model.

The degree of compression was measured using a finite value called it Compression Value (CV) which affect the results in a significant effect, Table5.1 shows the results of different compression values 18, 27, and 135 bytes that clarify that when the compression value increase the amount of data transfer will increase, which will save a lot of encryption time, this results was gained from the testing of both techniques and a proportional relationship between matrix (n× n) and file size (Bytes).

Table5.1: File Sizes in bytes for different compression values (CV) using a set of matrices

n×n (matrix Size)	File Size(Bytes)					
	Comp. value = 18 Byte (9x2)		Comp. value = 27 Byte (9x3)		Comp. value = 135 Byte (9x15)	
	First technique SSCC1	Second technique SSCC2	First technique SSCC1	Second technique SSCC2	First technique SSCC1	Second technique SSCC2
1x1	18	18	27	27	135	135
2x2	54	72	81	108	405	540
3x3	108	162	162	243	810	1215
4x4	180	288	270	432	1350	2160
5x5	270	450	405	675	2025	3375
6x6	378	648	567	972	2835	4860
7x7	504	882	756	1323	3780	6615
8x8	648	1152	972	1728	4860	8640
9x9	810	1458	1215	2187	6075	10935
10x10	990	1800	1485	2700	7425	13500

The illustrations of the previous table are emphasized using Figure 5.1 and clarify the differences between curves that represent several sets of data.

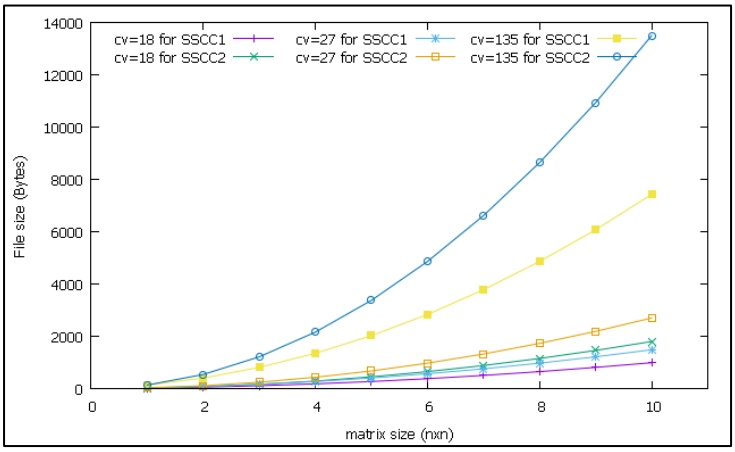

Figure 5.1: File sizes in a related of matrix size using different Compression values

5.1.1 Simulation results with compression

Simulation results can be measured including the compression phase in our model SSCC when we either use the first or the second technique. We applied this phase and got results from our system; first we take matrices with different sizes to compute the amount of data carried through each one and at the same time measure the encryption time for them separately, see Table5.2.

Table5.2: Encryption time for SSCC2 using different file sizes, compression value = 9x15 = 135 byte

Matrix size (nColumns × nRows)	File Size(bytes)		Encryption time (sec)
	1st technique (SSCC1)	2nd technique (SSCC2)	
1x1	135	135	0.001
2x2	405	540	0.004
3x3	810	1215	0.007
4x4	1350	2160	0.012
5x5	2025	3375	0.018
6x6	2835	4860	0.024
7x7	3780	6615	0.033
8x8	4860	8640	0.041
9x9	6075	10935	0.049
10x10	7425	13500	0.063
11x11	8910	16335	0.075
12x12	10530	19440	0.088

Note that, each value of File Size is calculated of a matrix that is either fulfilled or triangular filled in bytes within the following formula, Figure 5.2 illustrates that there is a non-linear growth between matrix size and encryption time:

File Size = Compression Value x numberOfElementsInTheMatrix (Bytes)

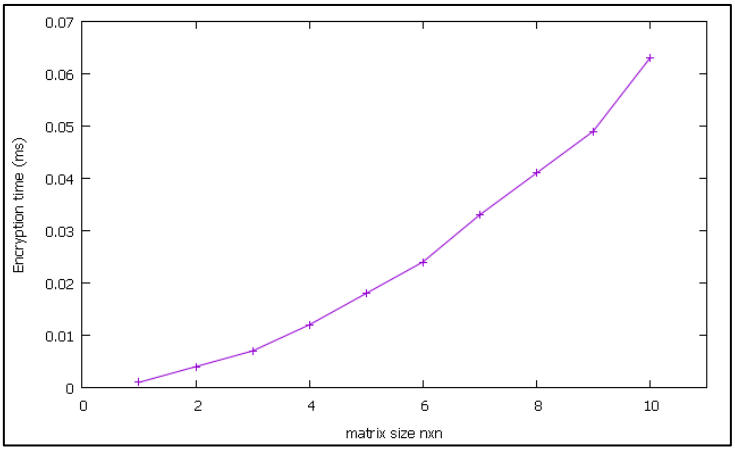

Figure 5.2: Relationship between the matrix size and the needed Encryption time in (ms) including the compression phase in SSCC model

The encryption time was increased as the size of the matrix increase , we test our model on several file sizes that varies from 32 to 320 kilo bytes using a base matrix with different size from 1x1 to 10x10 at each case , base matrix means that the whole file is divided into a set of matrices abound this size of matrix and add padding at the end if its needed to complete elements in the matrix, and then compare between the results to get the optimal one for this encryption process .

For the 1x1 matrix you can see the results in Table5.3:

Table5.3: The encryption time for different file sizes depends on the 1x1 as a base matrix.

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2370	0.2370
64	0.4740	0.4740
96	0.7111	0.7111
128	0.9481	0.9481
160	1.1851	1.1851
192	1.4222	1.4222
224	1.6592	1.6592
256	1.8962	1.8962
288	2.1333	2.1333
320	2.3703	2.3703

It is easy to work with 1x1 as a base matrix where each block of data is represented in a separate matrix that is the same using our two techniques SSCC1 ,

SSCC2 because of the existence of only one element, see Figure 5.3, also throughput can be measured for the File Size = 320 Kb using the formula 5.1.

$$Throughput = \frac{File\ Size\ (\ Bytes\ )}{Encryption\ Time\ (\ Sec\ )} \quad (5.1)$$

$$Throughput_{1\times1} = \frac{320\ \times\ 1024}{2.3703\ \times\ 10^{-3}} = 138.244\ MB/Sec$$

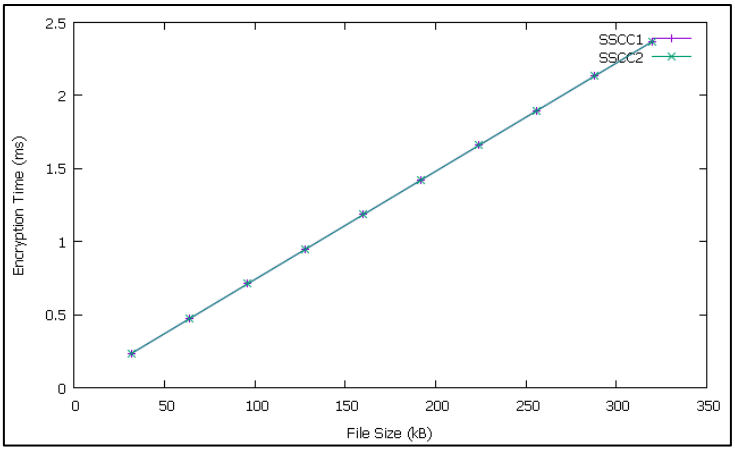

Figure 5.3: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 1x1 matrix

For the 2x2 matrix you can see the results below in Table5.4:

Table5.4: The encryption time for different file sizes depends on the 2x2 as a base matrix.

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.3160	0.2370
64	0.6320	0.4740
96	0.9481	0.7111
128	1.2641	0.9481
160	1.5802	1.1851
192	1.8962	1.4222
224	2.2123	1.6592
256	2.5283	1.8962
288	2.8444	2.1333
320	3.1604	2.3703

The usage of 2x2 matrix will generate a small difference in results of the encryption time between the first technique SSCC1 and the second SSCC2, see Figure 5.4.

$$Throughput_{2\times2} = \frac{320\ \times\ 1024}{2.3703\ \times\ 10^{-3}} = 138.244\ MB/Sec$$

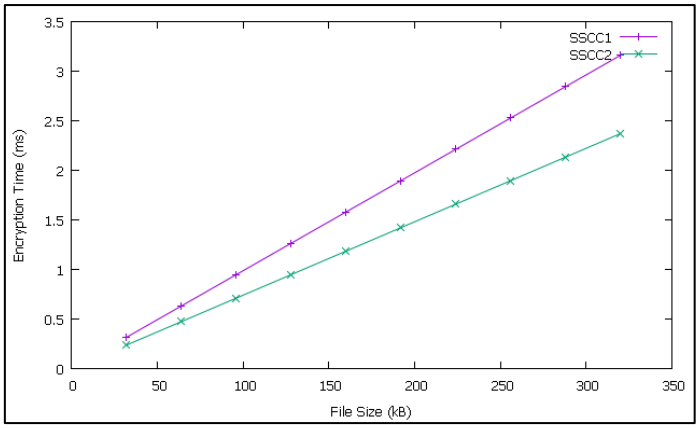

Figure 5.4: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 2x2 matrix

For the 3x3 matrix you can see the results in Table5.5:

Table5.5: The encryption time for different file sizes depends on the 3x3 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2765	0.1843
64	0.5530	0.3687
96	0.8296	0.5530
128	1.1061	0.7374
160	1.3827	0.9218
192	1.6592	1.1061
224	1.9358	1.2905
256	2.2123	1.4748
288	2.4888	1.6592
320	2.7654	1.8436

Another big difference is done using this size of matrix 3x3 and the better values are using the first technique SSCC1 that support compression according to Figure 5.5.

$$Throughput_{3\times3} = \frac{320 \times 1024}{1.8436 \times 10^{-3}} = 177.739\ MB/Sec$$

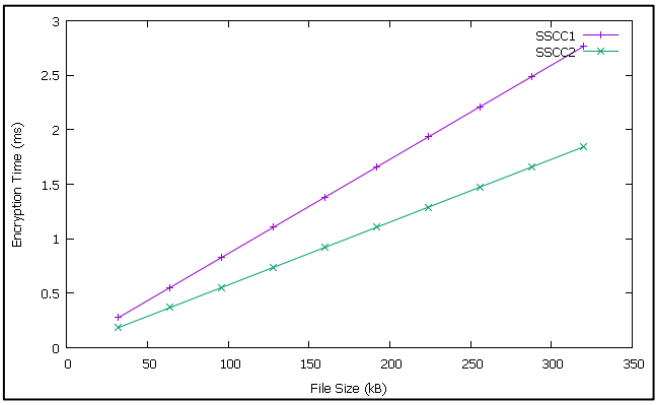

Figure 5.5: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 3x3 matrix

For the 4x4 matrix you can see the results in Table5.6:

Table5.6: The encryption time for different file sizes depends on the 4x4 as a base matrix.

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2844	0.1777
64	0.5688	0.3555
96	0.8533	0.5333
128	1.1377	0.7111
160	1.4222	0.8888
192	1.7066	1.0666
224	1.9910	1.2444
256	2.2755	1.4222
288	2.5600	1.6000
320	2.8444	1.777

As we see in Figure 5.6 , a more gap is appear between the two curves but we got that the difference between it and the previous one isn't big especially the SSCC1 curve.

$$Throughput_{4\times4} = \frac{320 \times 1024}{1.7770 \times 10^{-3}} = 184.400\ MB/Sec$$

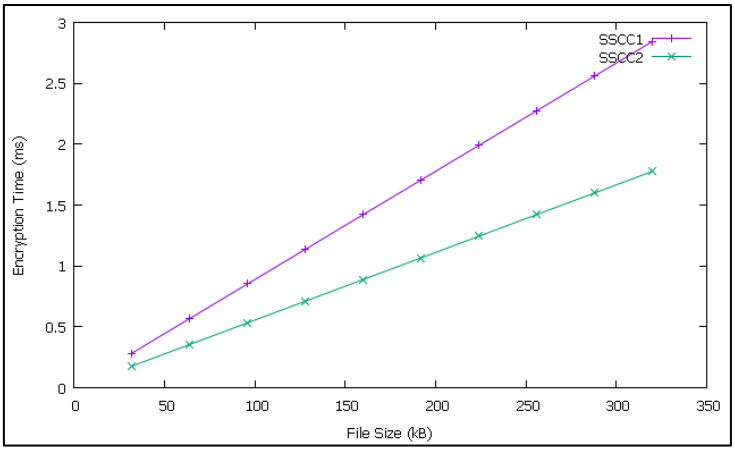

Figure 5.6: Relationship between the file size in (kB) and the needed encryption time in (ms) for 4x4 matrix

For the 5x5 matrix you can see the results below in Table5.7:

Table5.7: The encryption time for different file sizes depends on the 5x5 as a base matrix.

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2844	0.1706
64	0.5688	0.3413
96	0.8533	0.5120
128	1.1377	0.6826
160	1.4222	0.8533
192	1.7066	1.024
224	1.9911	1.1946
256	2.2755	1.3653
288	2.5600	1.5360
320	2.8444	1.7066

In Figure 5.7 the encryption time values are approximately the same as in Figure 5.6, this means that we can hold more data with the same encryption time that compared to 5x5 matrix.

$$Throughput_{5\times5} = \frac{320 \times 1024}{1.7066 \times 10^{-3}} = 192.007\ MB/Sec$$

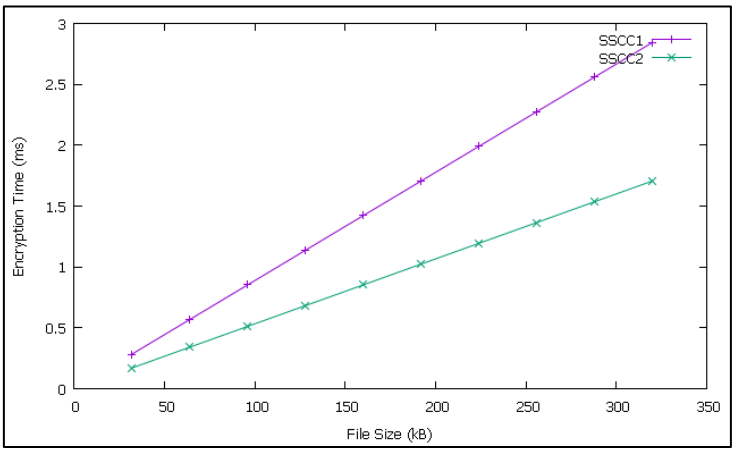

Figure 5.7: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 5x5 matrix

For the 6x6 matrix you can see the results in Table5.8:

Table5.8: The encryption time for different file sizes depends on the 6x6 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2708	0.1580
64	0.5417	0.3160
96	0.8126	0.4740
128	1.0835	0.6320
160	1.3544	0.7901
192	1.6253	0.9481
224	1.8962	1.1061
256	2.1671	1.2641
288	2.4380	1.4222
320	2.7089	1.5802

The 6x6 matrix plays its role by decreasing a bit of the encryption time, which will affect the whole process as in Figure 5.8.

$$Throughput_{6x6} = \frac{320 \times 1024}{1.59802 \times 10^{-3}} = 207.366\ MB/Sec$$

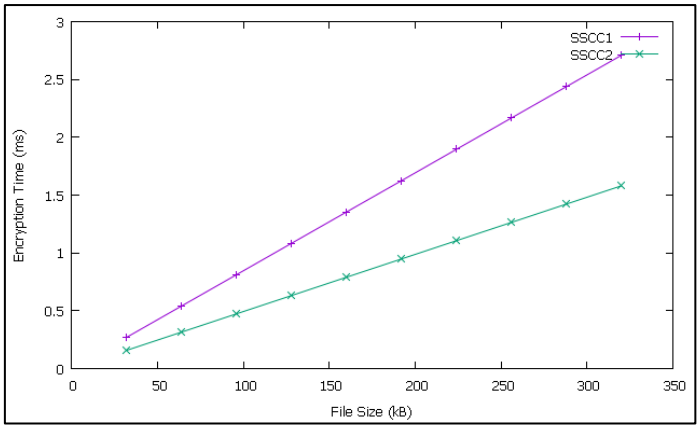

Figure 5.8: Relationship between the file size in (kB) and the needed encryption time in (ms) for 6x6 matrix

For the 7x7 matrix you can see the results in Table5.9:

Table5.9: The encryption time for different file sizes depends on the 7x7 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2793	0.1596
64	0.5587	0.3192
96	0.8380	0.4789
128	1.1174	0.6385
160	1.3968	0.7981
192	1.6761	0.9578
224	1.9555	1.1174
256	2.234	1.2770
288	2.5142	1.4367
320	2.7936	1.5963

The results again directed to be increased again using this size of matrix 7x7 see Figure 5.9, which leads to skip it for better results in the coming graphs.

$$Throughput_{7x7} = \frac{320 \times 1024}{1.5963 \times 10^{-3}} = 205.274\ MB/Sec$$

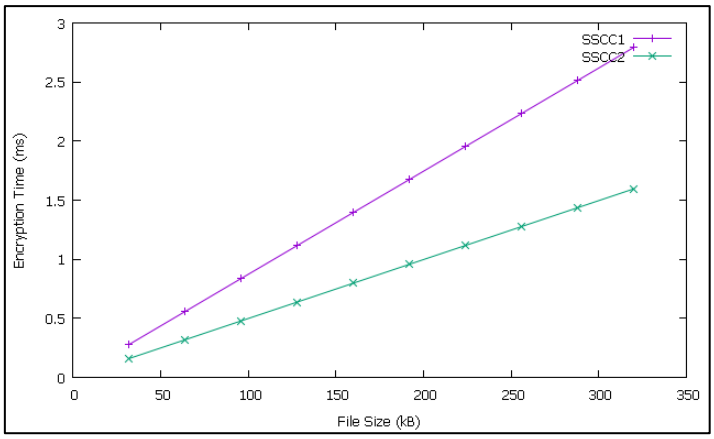

Figure 5.9: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 7x7 matrix

For the 8x8 matrix you can see the results below in Table5.10:

Table5.10: The encryption time time for different file sizes depends on the 8x8 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2699	0.1542
64	0.5399	0.3037
96	0.8098	0.4555
128	1.0798	0.6074
160	1.3497	0.7592
192	1.6197	0.9111
224	1.8897	1.0629
256	2.1596	1.2148
288	2.4290	1.3666
320	2.6995	1.5185

Using 8x8 matrix will return approximately the same results.

$$Throughput_{8x8} = \frac{320 \times 1024}{1.5185 \times 10^{-3}} = 215.791\ MB/Sec$$

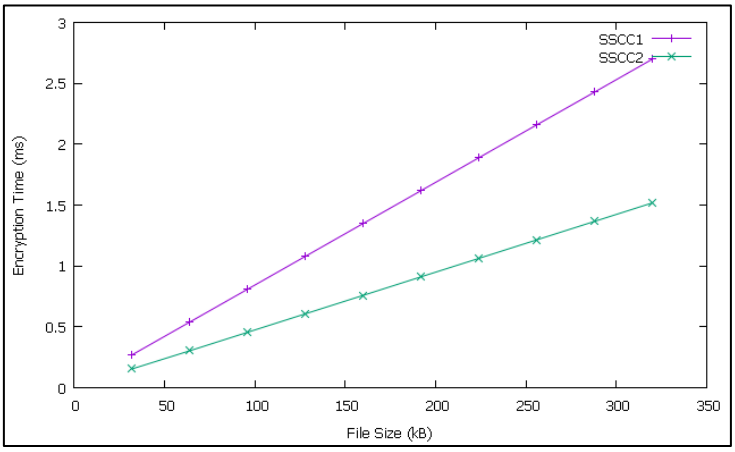

Figure 5.10: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 8x8 matrix

For the 9x9 matrix you can see the results below in Table5.11:

Table5.11: The encryption time for different file sizes depends on the 9x9 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	0.2581	0.1434
64	0.5162	0.2868
96	0.7743	0.4302
128	1.0324	0.5736
160	1.2905	0.7170
192	1.5486	0.8604
224	1.8067	1.004
256	2.0649	1.1471
288	2.3230	1.2905
320	2.5811	1.4340

The good results are found using 9x9 matrix where the encryption time is decreased and at the same time a larger file sizes are hold, so that we can call it the best case of our approach when using the SSCC2 second technique of our model.

$$Throughput_{9x9} = \frac{320 \times 1024}{1.4340 \times 10^{-3}} = 288.507\ MB/Sec$$

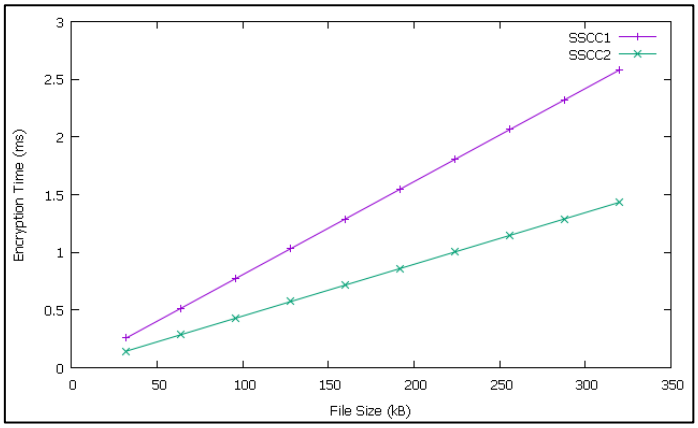

Figure 5.11: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 9x9 matrix

For the 10x10 matrix you can see the results in Table5.12:

Table5.12: The encryption time for different file sizes depends on the 10x10, 11x11 and 12x12 as base matrix

File Size (Kb)	Encryption time (ms)for 10x10		Encryption time (ms)for 11x11		Encryption time (ms)for 12x12	
	1st technique SSCC1	2nd technique SSCC2	1st technique SSCC1	2nd technique SSCC2	1st technique SSCC1	2nd technique SSCC2
32	0.2715	0.1493	0.2693	0.1469	0.2674	0.1448
64	0.5430	0.2986	0.5387	0.2938	0.5348	0.2897
96	0.8145	0.4480	0.8080	0.4407	0.8022	0.4345
128	1.0860	0.5973	1.0774	0.5876	1.0697	0.5794
160	1.3575	0.7466	1.3468	0.7346	1.3370	0.7242
192	1.6290	0.8960	1.6161	0.8815	1.6045	0.8691
224	1.9006	1.0453	1.8855	1.0284	1.8719	1.0139
256	2.1721	1.1946	2.1548	1.1753	2.1394	1.1588
288	2.4436	1.3440	2.4242	1.3223	2.4068	1.3037
320	2.7151	1.4933	2.6936	1.4692	2.6742	1.4480

In matrices 10x10, 11x11 and 12x12 as illustrated in Figure 5.13 the problem of the big encryption time will solved but it is still larger than the 9x9 matrix, so we chose the 9x9 as the best case of all cases.

$$Throughput_{10x10} = \frac{320 \times 1024}{1.4933 \times 10^{-3}} = 219.433\ MB/Sec$$

$$Throughput_{11x11} = \frac{320 \times 1024}{1.4692 \times 10^{-3}} = 223.032\ MB/Sec$$

$$Throughput_{12x12} = \frac{320 \times 1024}{1.4480 \times 10^{-3}} = 226.298\ MB/Sec$$

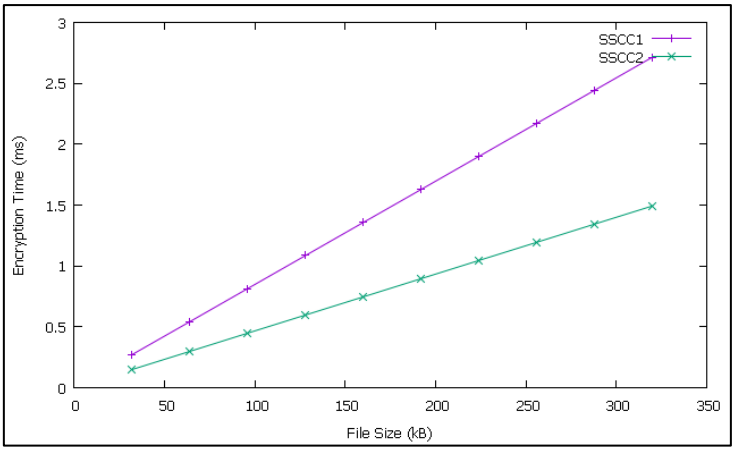

Figure 5.12: Relationship between the file size in (kB) and the needed Encryption time in (ms) for 10x10 matrix

We make our comparison between these matrices and conclude that the usage of the 1x1 matrix will obtain the optimal solution of the encryption time but a smaller amount of data that can be carried through this type of matrix, so we search about the second optimal solution and we gain the 9x9 matrix that provide both a better encryption time than others and with a very good throughput while the process of data transfer , also the usage of 2x2 matrix will provide a very bad results comparing it with others , see Figure 5.13.

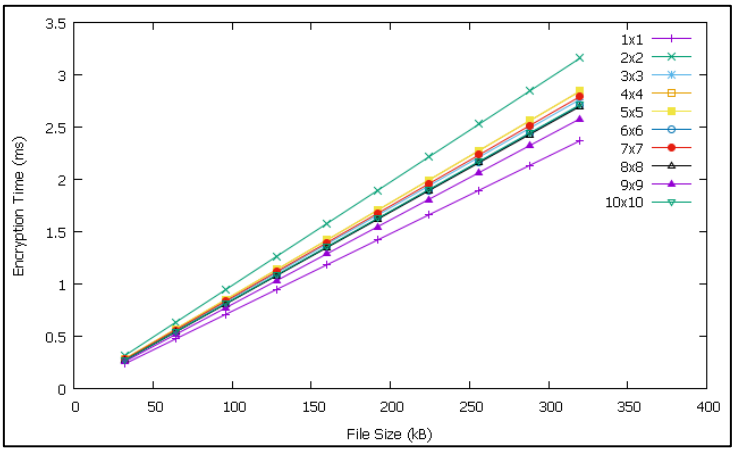

Figure 5.13: Relationship between the file size in (kB) and the needed Encryption time in (ms) for all matrices and using the SSCC1 technique with compression

Another last comparison in this section is done between several matrix sizes but using the second technique SSCC2 and the results indicates that the optimal solution was found using the 9x9 matrix and the second once was the 10x10 matrix and again the usage of 2x2 matrix represent the worst case of this model, see Figure 14.

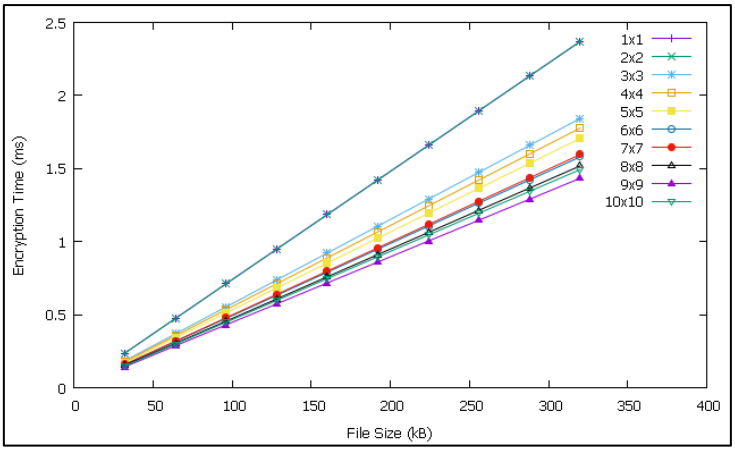

Figure 5.14: Relationship between the file size in (kB) and the needed encryption time in (ms) for all matrices and using the SSCC2 technique with compression

Finally, a comparison between using 1x1 and 9x9 matrices based on the transmission time to approximate the network traffic at each case using the 5.2 formula and let bandwidth=100Mbps :

$$\text{Transmission time (delay)} = \frac{\text{File size (bit)}}{\text{Bandwidth (bit/sec)}} \tag{5.2}$$

For 1x1 matrix (see Table 5.2 for File Size):

$$\text{Transmission time (delay)} = \frac{135 \text{ byte}}{100\text{Mb/sec}} = \frac{135 \times 8 \text{ bit}}{100 \times 10^6 \text{bit/sec}} = 0.0108 \text{ msec}$$

Using both SSCC1 and SSCC2, we gain the same results because there is only one element at this matrix

For 9x9 matrix (see Table 5.2 for File Size):

a. Using SSCC1

$$\text{Transmission time (delay)} = \frac{6075 \text{ byte}}{100\text{Mb/sec}} = \frac{6075 \times 8 \text{ bit}}{100 \times 10^6 \text{ bit/sec}} = 0.486 \text{ msec}$$

b. Using SSCC2

$$\text{Transmission time (delay)} = \frac{10935 \text{ byte}}{100\text{Mb/sec}} = \frac{10935 \times 8 \text{ bit}}{100 \times 10^6 \text{bit/sec}} = 0.8748 \text{ msec}$$

The results show that the matrix 9x9 needs more time than 1x1 matrix to be transmitted but it can hold more data than it.

5.1.2 Simulation results without compression

Simulation results can be measured excluding the compression phase in our model SSCC for both techniques SSCC1 and SSCC2. We skipped this phase and got results from our system; first we take matrices with different sizes to compute the amount of data carried through each one which mainly decreased when compared with the previous approach where the compression process included and at the same time measure the encryption time for them separately, see Table5.13.

Table5.13: Encryption time for SSCC using different File sizes

Matrix size (nColumns x nRows)	File Size(bytes)		Encryption time (sec)
	1st technique (SSCC1)	2nd technique (SSCC2)	
1x1	9	9	0.001
2x2	27	36	0.005
3x3	54	81	0.012
4x4	90	144	0.024
5x5	135	225	0.030
6x6	189	324	0.041
7x7	252	441	0.055
8x8	324	576	0.071
9x9	405	729	0.088
10x10	495	900	0.100

Note that, each value of File Size is calculated of a matrix that is either fulfilled or triangular filled in bytes as illustrated in Figure 15 and within the following formula: File Size = numberOfElementsInTheMatrix (bytes).

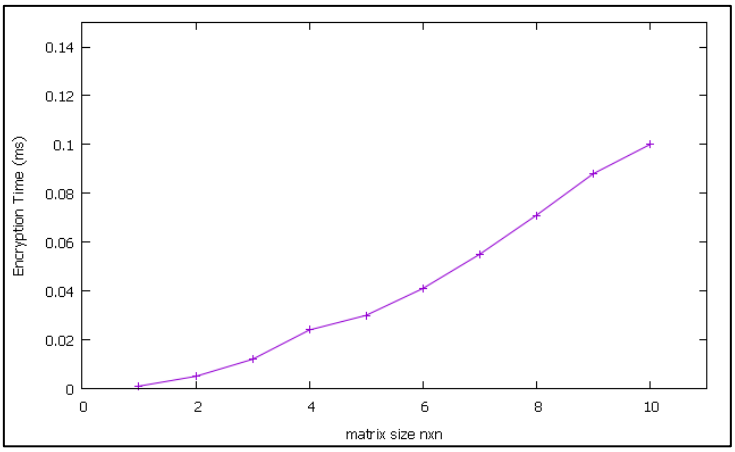

Figure 5.15: Relationship between the matrix size and the needed encryption time in (ms) excluding the compression phase in SSCC model

For the 1x1 matrix you can see the results in Table5.14:

Table5.14: The encryption time for different file sizes depends on the 1x1 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	3.5555	3.5555
64	7.1111	7.1111
96	10.6666	10.6666
128	14.2222	14.2222
160	17.7777	17.7777
192	21.3333	21.3333
224	24.8888	24.8888
256	28.4444	28.4444
288	32	32
320	35.5555	35.5555

We notice that a huge difference between values as the file size increase, because each element of the matrix contains only one byte of data and the matrix is composed of only one element , so the data transfer is done one byte at a time which will waste time as in Figure 5.16.

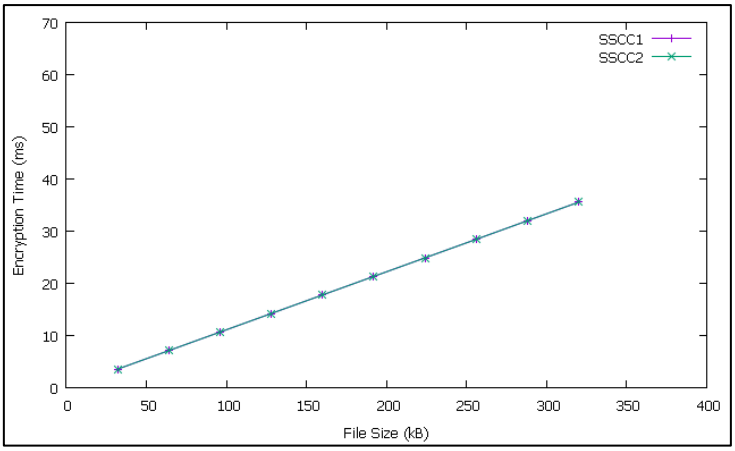

Figure 5.16: Relationship between the file size in (kB) and the needed encryption time in (ms) for 1x1 matrix

For the 2x2 matrix you can see the results below in Table5.15:

Table5.15: The encryption time for different file sizes depends on the 2x2 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	5.9259	4.4444
64	11.8518	8.8888
96	17.7777	13.3333
128	23.7037	17.7777
160	29.6296	22.2222
192	35.5555	26.6666
224	41.4814	31.1111
256	47.4074	35.5555
288	53.3333	40
320	59.2592	44.4444

In 2x2 matrix , the encryption time continue its increasing process but there is a difference in output between the two techniques SSCC1 , SSCC2 where the SSCC2 return a less time than SSCC1 , this indicate positively as in Figure 5.17.

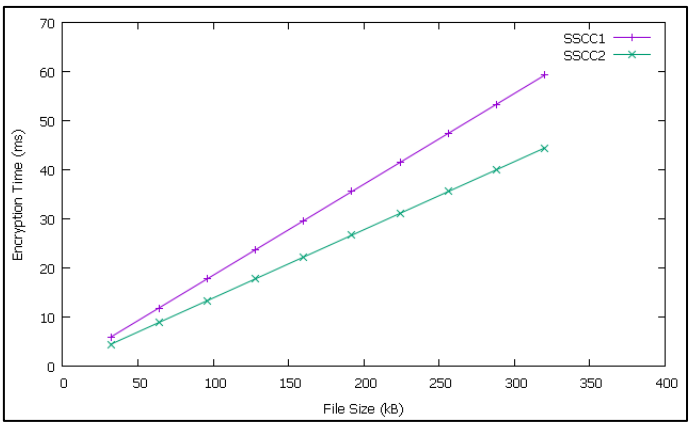

Figure 5.17: Relationship between the file size in (kB) and the needed encryption time in (ms) for 2x2 matrix

For the 3x3 matrix you can see the results below in Table5.16:

Table5.16: The encryption time for different file sizes depends on the 3x3 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	7.1111	4.7407
64	14.2222	9.4814
96	21.3333	14.2222
128	28.4444	18.9629
160	35.5555	23.7037
192	42.6666	28.4444
224	49.7777	33.1851
256	56.8888	37.9259
288	64	42.6666
320	71.1111	47.4074

The absence of the compression phase affect the results mainly , so that as the matrix size increase which leads to an increase in the file size will reflects negatively on the encryption time see Figure 5.18.

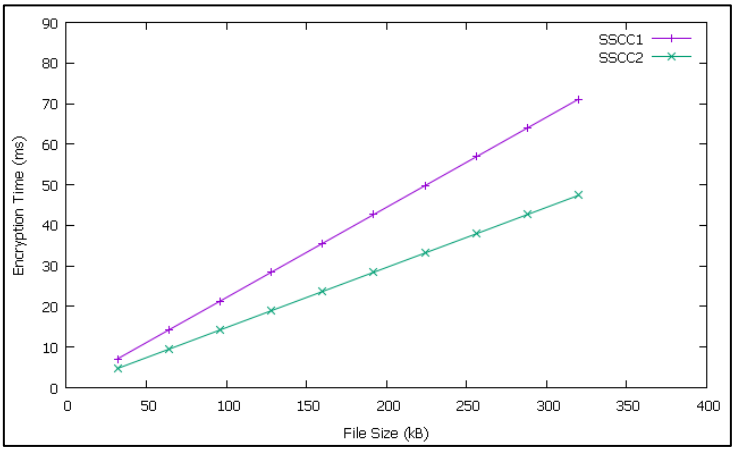

Figure 5.18: Relationship between the file size in (kB) and the needed encryption time in (ms) for 3x3 matrix

For the 4x4 matrix you can see the results in Table5.17:

Table5.17: The encryption time for different file sizes depends on the 4x4 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	8.5333	5.3333
64	17.0666	10.6666
96	25.6	16
128	34.1333	21.3333
160	42.6666	26.6666
192	51.2	32
224	59.7333	37.3333
256	68.2666	42.6666
288	76.8	48
320	85.3333	53.3333

As we see in Figure 5.19 , the values grows up deeply with respect to matrix size that represent the file size , this will prove that the compression phase must be included to gain the perfect results .

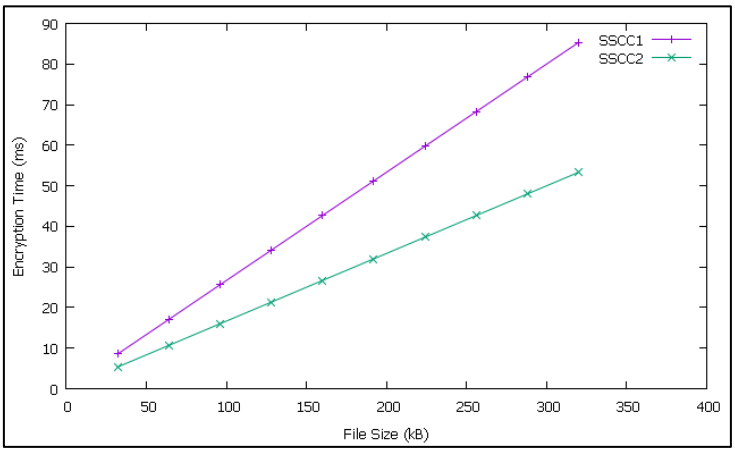

Figure 5.19: Relationship between the file size in (kB) and the needed encryption time in (ms) for 4x4 matrix

For the 5x5 matrix you can see the results in Table5.18:

Table5.18: The encryption time for different file sizes depends on the 5x5 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	7.1111	4.2666
64	14.2222	8.5333
96	21.3333	12.8
128	28.4444	17.0666
160	35.5555	21.3333
192	42.6666	25.6
224	49.7777	29.8666
256	56.8888	34.1333
288	64	38.4
320	71.1111	42.6666

In 5x5 matrix, the values move down and decreased when compared with the last cases that conclude 1x1 to 4x4 matrices, this will provide a good feedback see Figure 5.20.

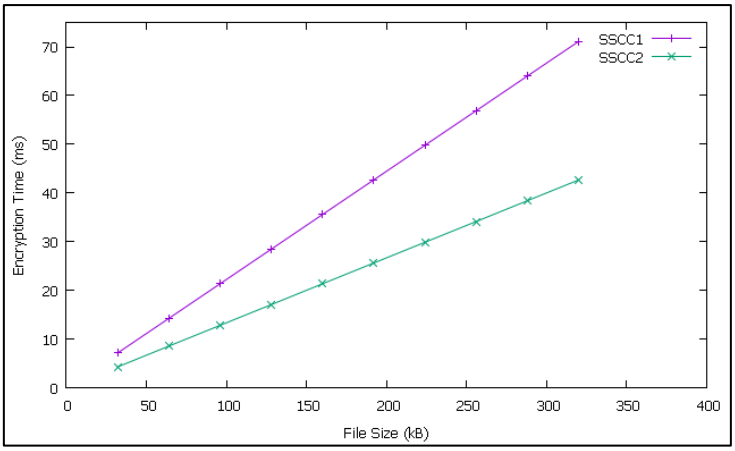

Figure 5.20: Relationship between the file size in (kB) and the needed encryption time in (ms) for 5x5 matrix

For the 6x6 matrix you can see the results in Table5.19:

Table5.19: The encryption time for different file sizes depends on the 6x6 as a base matrix.

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	6.94176	4.0493
64	13.8835	8.0987
96	20.8253	12.1481
128	27.7671	16.1975
160	34.7089	20.2469
192	41.6507	24.2962
224	48.5925	28.3456
256	55.5343	32.3950
288	62.4761	36.4444
320	69.4179	40.4938

As shown in Figure 5.21, the results continue their decreasing process that the difference gap increase as the matrix size increase.

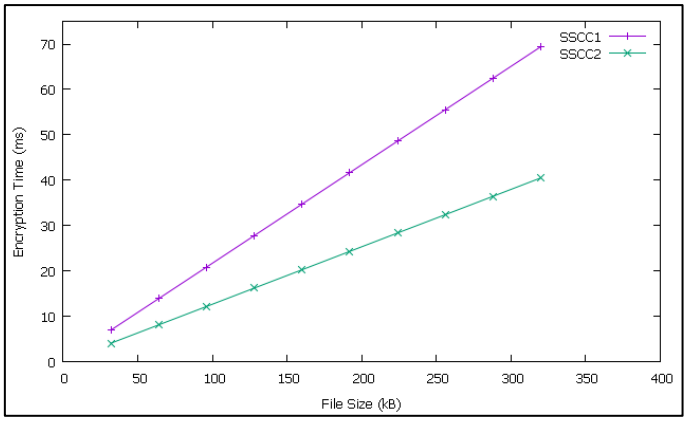

Figure 5.21: Relationship between the file size in (kB) and the needed encryption time in (ms) for 6x6 matrix

For the 7x7 matrix you can see the results in Table5.20:

Table5.20: The encryption time for different file sizes depends on the 7x7 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	6.9841	3.9909
64	13.9682	7.9818
96	20.9523	11.9727
128	27.9365	15.9637
160	34.9206	19.9546
192	41.9047	23.9455
224	48.8888	27.9365
256	55.8730	31.9274
288	62.8571	35.9183
320	69.8412	39.9092

According to the Table5.20, we notice that the results remain approximately as the previous case when the matrix size is 6x6, but using the 7x7 matrix is better because it can contain more data.

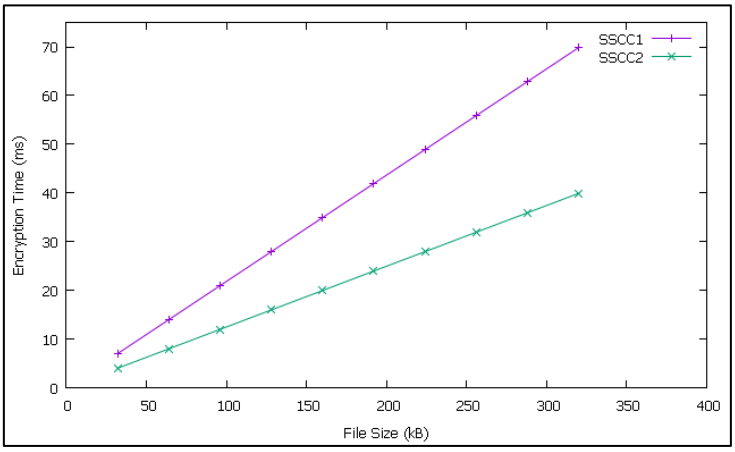

Figure 5.22: Relationship between the file size in (kB) and the needed encryption time in (ms) for 7x7 matrix

For the 8x8 matrix you can see the results in Table 5.21:

Table5.21: The encryption time for different file sizes depends on the 8x8 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	7.0123	3.9444
64	14.0246	7.8888
96	21.0370	11.8333
128	28.0493	15.7777
160	35.0617	19.7222
192	42.0740	23.6666
224	49.0864	27.6111
256	56.0987	31.5555
288	63.1111	35.5000
320	70.1234	39.4444

In 8x8 matrix, the values return to its increasing manner specially for the SSCC1 technique and remain the same in the SSCC2 technique.

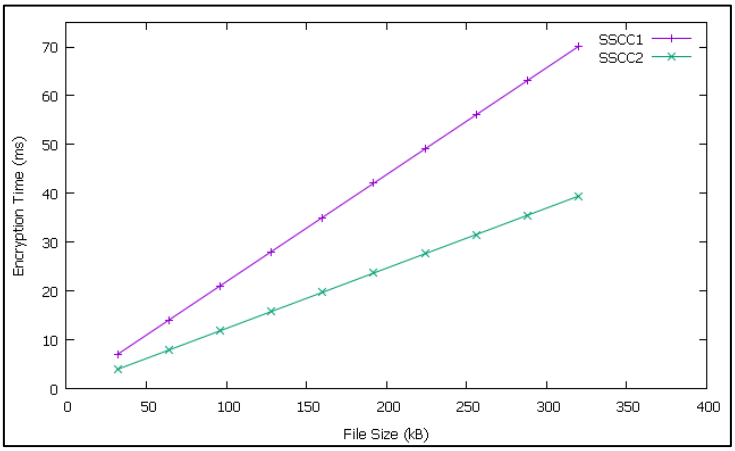

Figure 5.23: Relationship between thefile size in (kB) and the needed encryption time in (ms) for 8x8 matrix

For the 9x9 matrix you can see the results in Table5.22:

Table5.22: The encryption time for different file sizes depends on the 9x9 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	6.9530	3.8628
64	13.9061	7.7256
96	20.8592	11.5884
128	27.8123	15.4513
160	34.7654	19.3141
192	41.7185	23.1769
224	48.6716	27.0397
256	55.6246	30.9026
288	62.5777	34.7654
320	69.5308	38.6282

The results of Table5.22 are illustrated in Figure 5.24, and no big changes on data, so that we make sure to use the larger size of the matrix to complete the process of data transfer with larger file sizes.

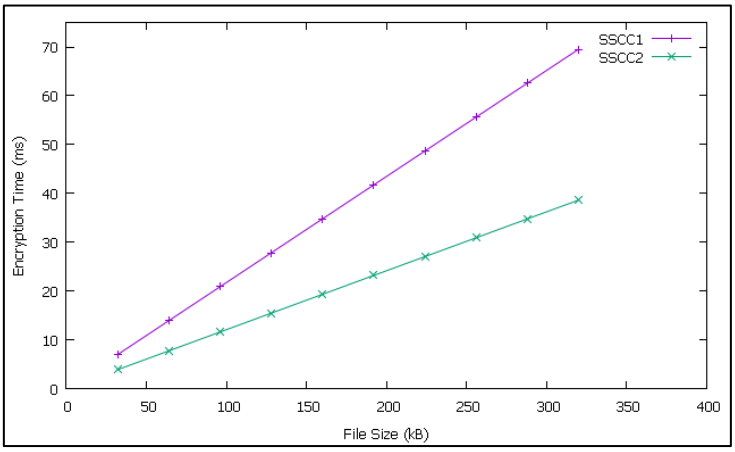

Figure 5.24: Relationship between the file size in (kB) and the needed encryption time in (ms) for 9x9 matrix

For the 10x10 matrix you can see the results in Table5.23:

Table5.23: The encryption time for different file sizes depends on the 10x10 as a base matrix

File Size (Kb)	Encryption time (ms)	
	1st technique SSCC1	2nd technique SSCC2
32	6.4646	3.5555
64	12.9292	7.1111
96	19.3939	10.6667
128	25.8585	14.2222
160	32.3232	17.7777
192	38.7878	21.3333
224	45.2525	24.8888
256	51.7171	28.4444
288	58.1818	32.0000
320	64.6464	35.5555

Finally, in 10x10 matrix is the optimal matrix for data transfer because it provides the less encryption time with a more a larger data files, so that as the results of the previous cases we choose it as the best case according to Figure 5.25 indication.

$$Throughput_{10\times10} = \frac{320 \times 1024}{35.5 \times 10^{-3}} = 9.23\ MB/Sec$$

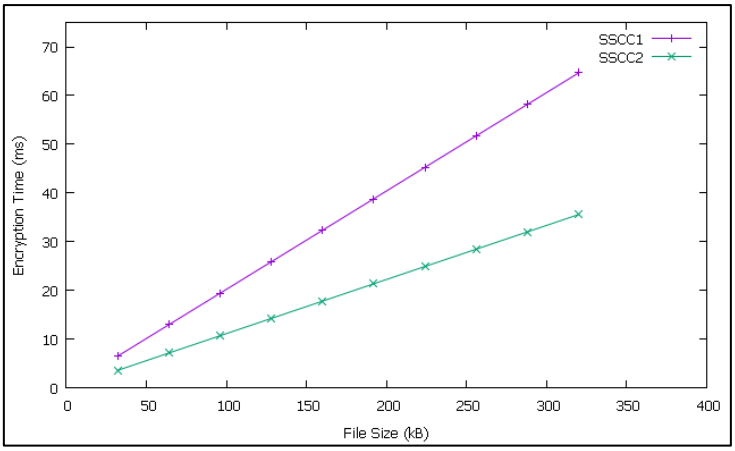

Figure 5.25: Relationship between the file size in (kB) and the needed encryption time in (ms) for 10x10 matrix

Moreover, We make a comparison between all cases that include different matrix size at the same time for our first technique SSCC1and conclude that the usage of the 1x1 matrix will obtain the better encryption time but no one will use it because of it small capacity and high traffic which will waste time in overheads , then 2x2 matrix has the second order but also with a small capacity ,the third one is 10x10 matrix which represent the best one because it combine the two advantages represented by larger file sizes with low encryption time , see Figure 5.26.

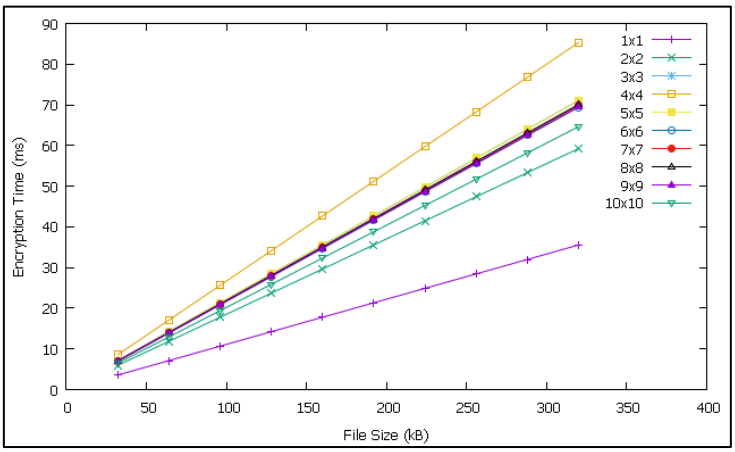

Figure 5.26: Relationship between the file size in (kB) and the needed encryption time in (ms) for all matrices and using the SSCC1 technique without compression

In the second technique SSCC2, the gap between curves decrease mainly and indicates the opposites relationship between the file size and the encryption time, as the matrix increase the encryption time decrease, the 10x10 matrix refer to the best case and 4x4 matrix refer to the worst case in this technique because it contains more padded bytes than data, see Figure 5.27.

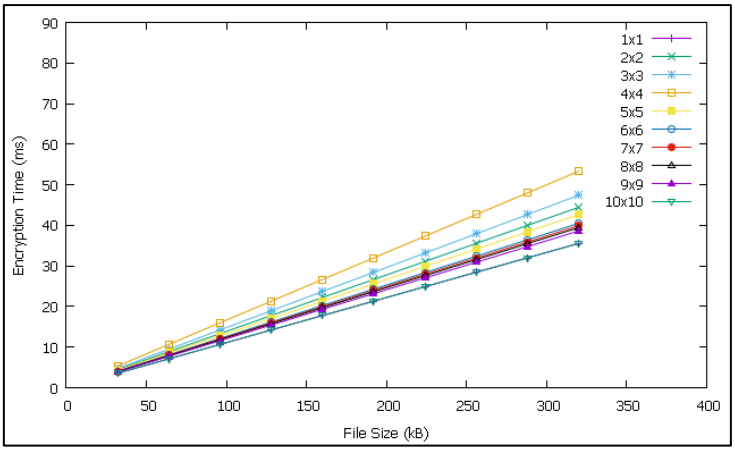

Figure 5.27: Relationship between the file size in (kB) and the needed encryption time in (ms) for all matrices and using the SSCC2 technique without compression

5.1.3 Comparison between SSCC1, SSCC2 with and without compression

Compression play a big role in our process and embedded it within the encryption process will increase the efficiency of the output, the testing is done either with or without compression and the results show the difference between them separately, we compose the two cases with compression and without together to clarify the big gap between them for each matrix size individually.

For 1x1 matrix, and as shown in Figure 5.28 a big gap between the curves that used when compression is applied or not.

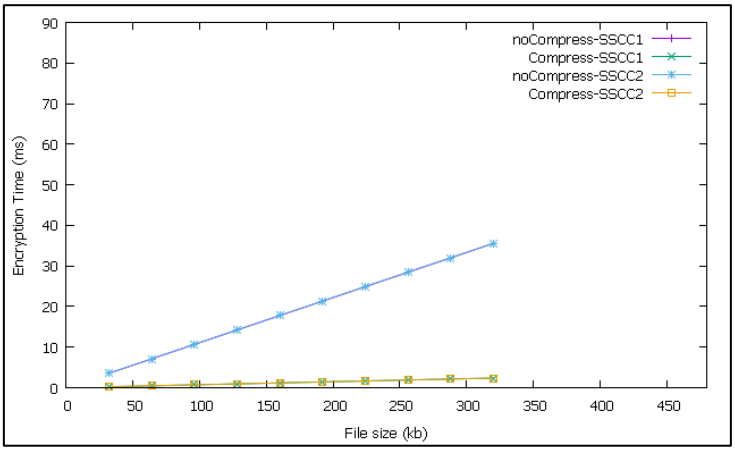

Figure 5.28: Relationship between the file size in (kB) and the needed encryption time in (ms) for 1x1 matrix and using both SSCC1 and SSCC2 techniques with and without compression

According to Figure 5.29 the 2x2 matrix is used and the values with compression are stay the same when compared with 1x1 matrix, but the gap between the two curves with and without compression was increased.

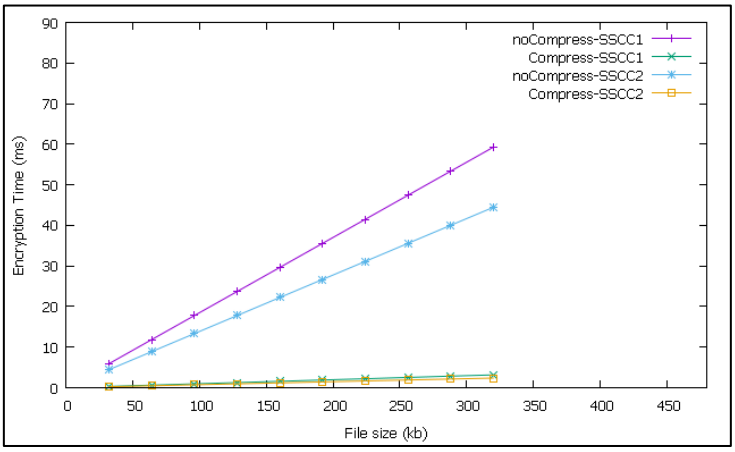

Figure 5.29: Relationship between the file size in (kB) and the needed encryption time in (ms) for 2x2 matrix and using both SSCC1 and SSCC2 techniques with and without compression

Figure 5.30 shows that a bit changes in values for SSCC1 results which used the triangular matrices to represent data but the results of SSCC2 remain approximately the same.

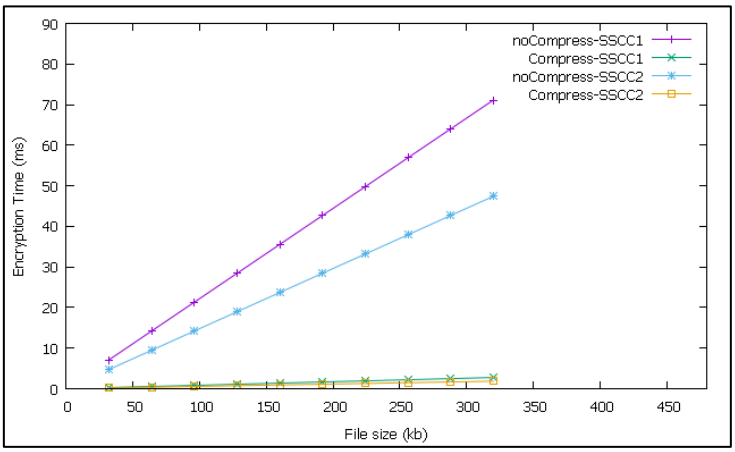

Figure 5.30: Relationship between the file size in (kB) and the needed encryption time in (ms) for 3x3 matrix and using both SSCC1 and SSCC2 techniques with and without compression

The encryption time increase within the matrix size increase when the compression phase discard, take the 4x4 matrix as an example we can notice the result clearly, see Figure 5.31.

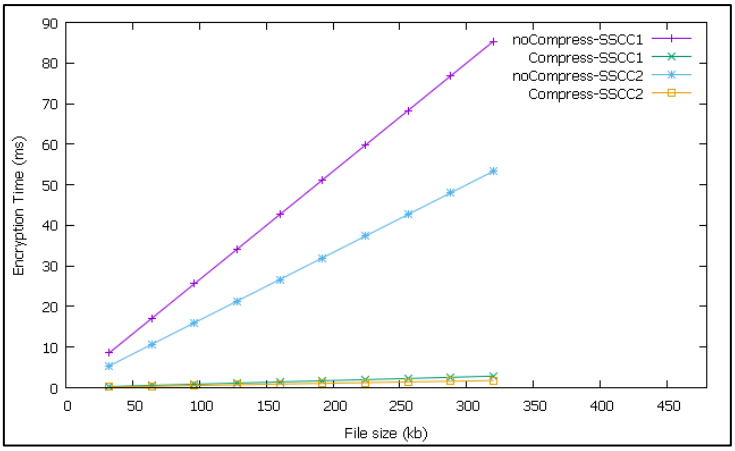

Figure 5.31: Relationship between the file size in (kB) and the needed encryption time in (ms) for 4x4 matrix and using both SSCC1 and SSCC2 techniques with and without compression

In 5x5 matrix and within Figure 5.32, the curves goes to be decrease a bit more than 4x4 , 3x3 , 2x2 , 1x1 but it still has a high encryption time when compared with the results from compression .

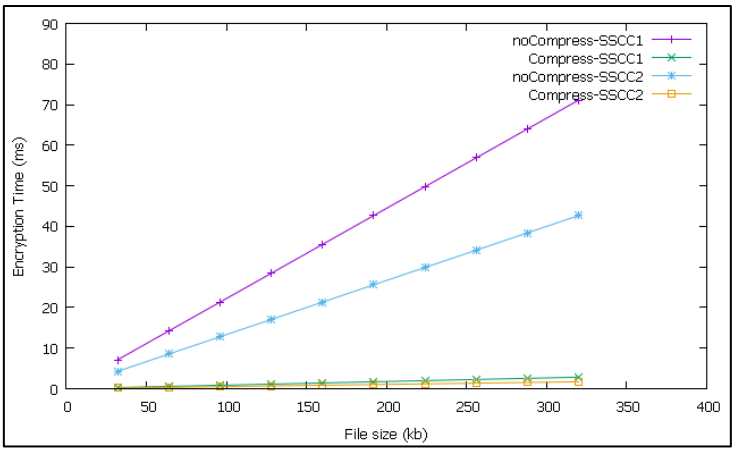

Figure 5.32: Relationship between the file size in (kB) and the needed encryption time in (ms) for 5x5 matrix and using both SSCC1 and SSCC2 techniques with and without compression

A bit more decrease is done through the 6x6 matrix, Figure 5.33 shows the added effects on the previous 5x5 matrix. This matrix leads to better values.

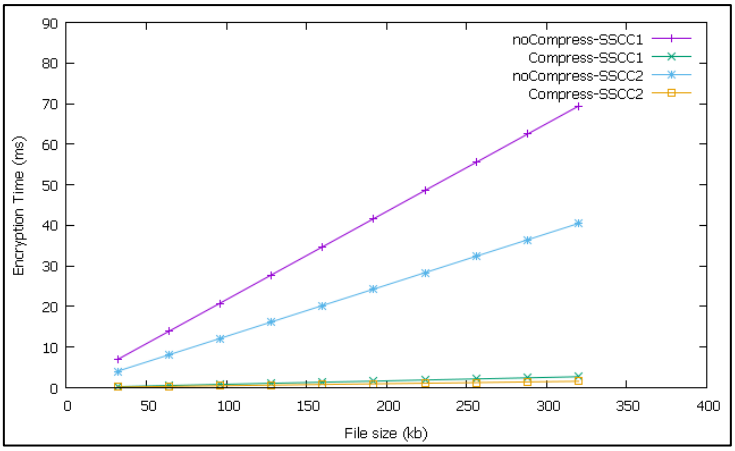

Figure 5.33: Relationship between the file size in (kB) and the needed encryption time in (ms) for 6x6 matrix and using both SSCC1 and SSCC2 techniques with and without compression

As shown in Figure 5.34 , the result of using 7x7 matrix doesn't make any changes on the previous 6x6 matrix , it appear as a little difference was made , so that we compare between them using the file sizes in kilo bytes .

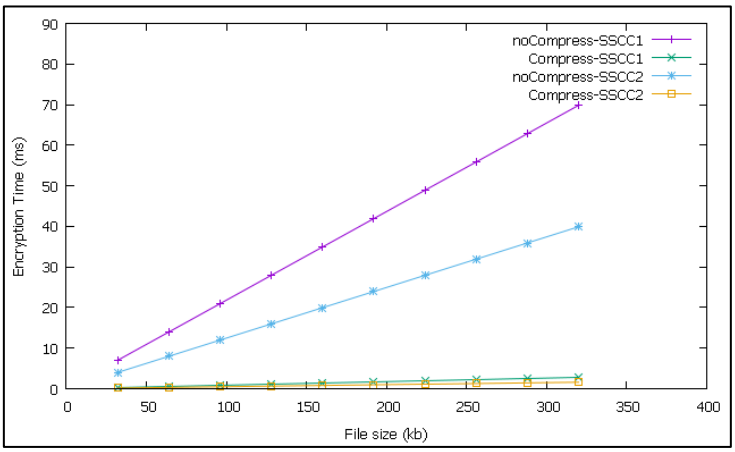

Figure 5.34: Relationship between the file size in (kB) and the needed encryption time in (ms) for 7x7 matrix and using both SSCC1 and SSCC2 techniques with and without compression

Also , when we take the larger matrix 8x8 as a base matrix we gain the a more data represented by file size that can be encrypted within the same encryption time of 7x7 as illustrated in Figure 5.35.

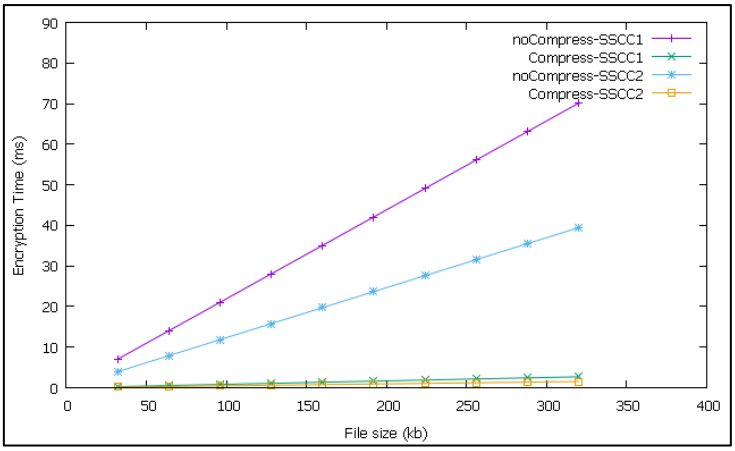

Figure 5.35: Relationship between thefile size in (kB) and the needed encryption time in (ms) for 8x8 matrix and using both SSCC1 and SSCC2 techniques with and without compression

The 9x9 matrix, is the optimal matrix for performing our system because it include more data at a time and less encryption time. This result is gained through our experiments on matrices that varies from 1x1 to 10x10 and this optimal result is using SSCC2 that include the compression phase on its design, see Figure 5.36.

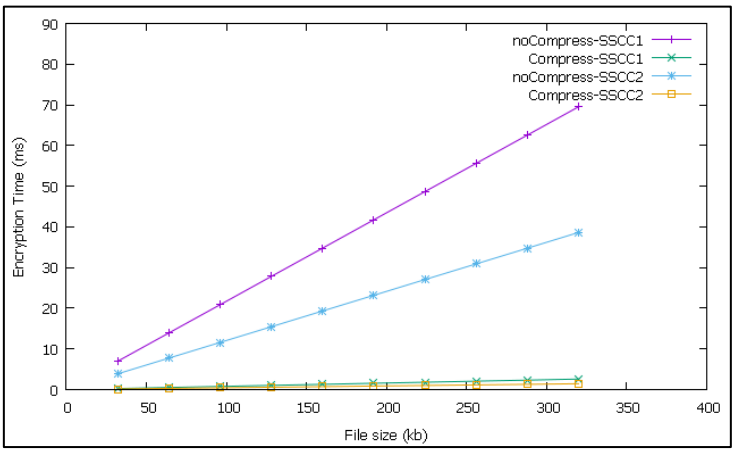

Figure 5.36: Relationship between the file size in (kB) and the needed encryption time in (ms) for 9x9 matrix and using both SSCC1 and SSCC2 techniques with and without compression

Our testing is done using 10x10 matrix as a base matrix which reflect that it is better than 9x9 matrix using SSCC1 technique using triangular matrices , but this technique cover less data than SSCC2 that use a complete full matrix an support compression phase despite the support of it in the SSCC1, see Figure 5.37.

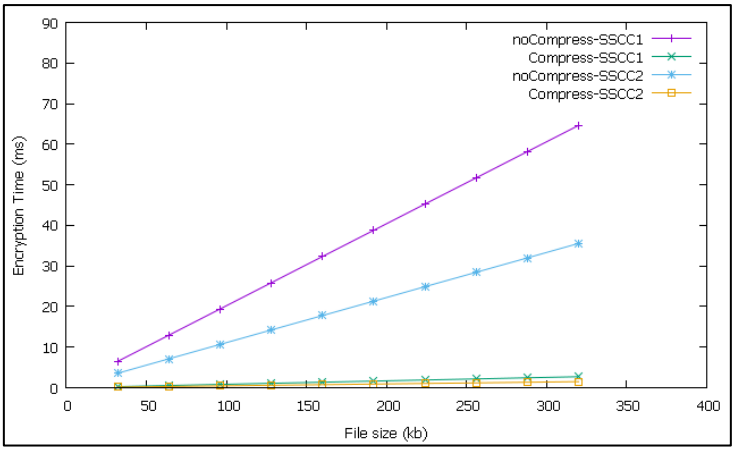

Figure 5.37: Relationship between the file size in (kB) and the needed encryption time in (ms) for 10x10 matrix and using both SSCC1 and SSCC2 techniques with and without compression

Finally, a comparison between using 1x1 and 9x9 matrices based on the transmission time to approximate the network traffic at each case using the following formula and let bandwidth=100Mbps based on the formula 5.2:

For 1x1 matrix:

$$\text{Transmission time (delay)} = \frac{9\ \text{byte}}{100\text{Mb/sec}} = \frac{9 \times 8\ \text{bit}}{100 \times 10^6\text{bit/sec}} = 0.00072\ \text{msec}$$

Using both SSCC1 and SSCC2, we gain the same results because there is only one element at this matrix

For 9x9 matrix:

a. Using SSCC1

$$\text{Transmission time (delay)} = \frac{495\ \text{byte}}{100\text{Mb/sec}} = \frac{495 \times 8\ \text{bit}}{100 \times 10^6\text{bit/sec}} = 0.0396\ \text{msec}$$

b. Using SSCC2

$$\text{Transmission time (delay)} = \frac{900\ \text{byte}}{100\text{Mb/sec}} = \frac{900 \times 8\ \text{bit}}{100 \times 10^6\text{bit/sec}} = 0.072\ \text{msec}$$

The results show that the matrix 9x9 needs more time than 1x1 matrix to be transmitted but it can hold more data than it. Also, if we compared results between matrices with and without compression we notice that we need less time using without compression technique but with high traffic cause to the smaller sizes of each matrix either 1x1 or 9x9.

In Figure 5.38 a relationship between the Encryption time and the matrix size is used to illustrate that the optimal matrix is 9x9, where the minimum encryption time.

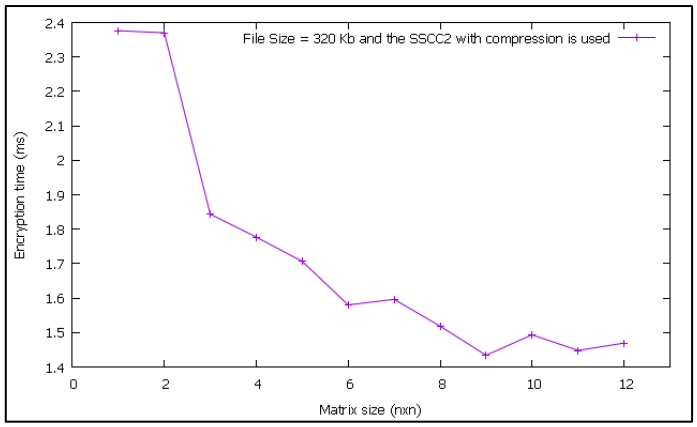

Figure 5.38: Encryption time for different matrix sizes using SSCC2 and File Size 320 Kb

5.1.4 Comparison between SSCC and other models

In this simulation a different block sizes are used in the process for splitting the input text into sub blocks then composite them into several compressed blocks. We test our algorithm using two techniques and compare between them and others see Figure 5.39.

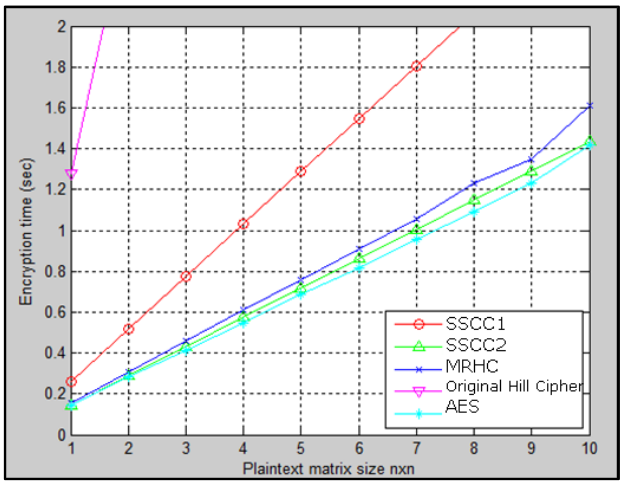

Figure 5.39: Encryption time for different algorithms , Compression Value = 135 bytes

These techniques are: full or triangular plain text matrix .We started from 32 Kb and end with 160 kb with fixed matrix size = 9x9, we choose this size of matrix to be a reference because we gain the optimal results using it when compared with other sizes that varies from 1x1 to 10x10 with compression, see Table 5.24.

Table5.24: Average time for different algorithms, matrix size 9x9

Algorithm	Mean of the algorithm time (ms)	Time for 1 KB in (μs)
SSCC1	1519	8.065
SSCC2	789	4.48
MRHC [11]	846	4.81
AES [11]	758	4.31

$d_{MRHC,SSCC2} = 4.81 - 4.48 = 0.33$ ms

$d_{AES,SSCC2} = 4.31 - 4.48 = -0.17$ ms

Our results is gained according to test the SSCC model using different inputs which will affect the traffic of uploading/downloading data and provide a good modification in time compared with the last traditional cryptosystem models, a comparison between SSCC and MRHC result an improvement percentage of 6.86%, while the comparison between SSCC and AES result a percentage of 3.94% to the AES, this means that our model has a good chance to approach the standard, using the following formulas:

$$\text{ET(decrease)\%} = \frac{ET_{MRHC} - ET_{SSCC2}}{ET_{MRHC}} \text{ x } 100\% = \frac{4.81 - 4.48}{4.81} \text{ x } 100\% = 6.86\%$$

$$\text{ET(increase)\%} = \frac{ET_{AES} - ET_{SSCC2}}{ET_{AES}} \text{ x } 100\% = \frac{4.31 - 4.48}{4.31} \text{ x } 100\% = 3.94\%$$

Using our techniques the needed time for completely encrypt the plaintext is decreased such that when we apply the first technique (SSCC1) either before or after compression process using the 9x9 matrix and file size 320 kb using the following formula :

$$ET\% = \frac{CET - NCET}{CET} \text{ x } 100\% \quad (5.2)$$

Where,

ET: The percentage of decrease in the Encryption Time.

CET: Compression Encryption Time.

NCET: Non-Compression Encryption Time.

Based on the formula (5.2):

$$\text{ET\%} = \frac{38.6282 - 1.4340}{38.6282} \text{ x } 100\% = 96.287\%$$

Using the compression process the encryption time will be decreased by96.287%.

5.2 SSCC Assumptions

Let us use a file size between 32kb to 320 kb and fill them into matrices 1x1 to 10x10 sizes to contain the original plain text that is divided into a set of sub-blocks, key matrix must be with the same size of the plain text where each plaintext matrix need a key matrix to encrypt it.

The Secret key is gained from the key Generator and has a length that filled the matrix either when it is completely filled or half matrix the padding will be used to add more security degree to the key that varies (8 to 48) bits without padding and (16 to 512) bits with padding.

5.3 Brute-force attack

Brute force attack is an attack from the outside world to the cipher text message in order to get the original plain text by trying multiple keys to guess the sent message. Encryption algorithm must be robust enough against this type of attacks by calculating the time required for a key to be broken [29].

Table5.25 shows different size of key matrices and the time need to break that key and the time is measured using two cased with and without padding, also we can notice that the security concerns is approved when the matrix size is greater than 5× 5 because the key length must be greater than 1024 bit without padding and greater than 4x4 with padding.

Number of alternative keys =2^n.
n: key length in bits .
Operation time = 64 kb (Using our platform).
Time need to break key = Operation time × Number of alternative keys (sec).

Table5.25: Time needed to break the key with different length for SSCC using different matrix sizes

nxn	Key length without padding (bit)	No. of alternative keys Without padding	Time need to break key (sec)	Key length with padding (bit)	No. of alternative keys with padding	Time need to break key (sec)
1x1	8	2^{8}	$16.3840x10^{6}$	32	2^{32}	$2.7488x\ 10^{14}$
2x2	24	2^{24}	$1.0737x10^{12}$	64	2^{64}	$1.1806x10^{24}$
3x3	48	2^{48}	$1.8014x10^{19}$	128	2^{128}	$2.1778x10^{43}$
4x4	80	2^{80}	$7.7371x10^{28}$	128	2^{128}	$2.1778x10^{43}$
5x5	104	2^{104}	$1.2980x10^{36}$	256	2^{256}	$7.4107x10^{81}$
6x6	168	2^{168}	$2.3945x10^{55}$	256	2^{256}	$7.4107x10^{81}$
7x7	224	2^{224}	$1.7254x10^{72}$	256	2^{256}	$7.4107x10^{81}$
8x8	360	2^{360}	$150.3067x10^{111}$	512	2^{512}	$8.5810x10^{158}$
9x9	440	2^{440}	$181.7096x10^{135}$	512	2^{512}	$8.5810x10^{158}$
10x10	528	2^{528}	$562.3642x10^{161}$	1024	2^{1024}	$64k\ x\ 2^{1024}$

The key length varies and depends on the matrix size, as the matrix become larger and contains more data; we need a high key length to protect these data. We calculate these values for the previous two cases (full and half plain text matrix).

Chapter Six

Conclusion and Future work

Contents

6.1 Conclusion

Our proposed model SSCC has two techniques: The first one is using a triangular matrix for the plain text and it is called (SSCC1) and the other one is using a full matrix for the plain text and it is called (SSCC2). Although, the key matrix remains the same as a lower triangular matrix for both techniques, this comparison has done to satisfy a large file size through the same matrix by increasing the number of blocks existed on that file or increasing the compression value, so that one block will contain more bytes.

It's known that the SSCC tests the process with and without compression phase to ensure the big role of the compression phase in the proposed model. Also, SSCC aims to contribute the existing Hill Cipher algorithm by reducing time needed for the process of encryption as well accesses a robust and secure mode.

Moreover, SSCC provide a set of modification on the existing Hill Cipher such as: Encoding, compression for matrix blocks, plain text padding and encryption using a composite secret key. This key is generated by the matrix key Generator (MKG) and based on the Diffie-hellman key exchange principle, so that attacker has no direct choices to check because the algorithm is complex and composite of a set of complicated steps.

Compression phase leads to a huge data transform per unit of time where each entry in the plain text matrix will represent a set of compressed blocks. The compression value (CV) is variable and can be increased as needed to create high speed traffic within the network. Our proposed model keeps the privacy of individual users using unique secret key. It is not only for Cloud Computing but also can be applied on distributed systems.

6.2 Future work

In Future, we will add features for different types of data that is either texts, numbers or images and videos where the compression will play a major role in decreasing the multimedia files that have a large file sizes. The key exchange scheme will be improved using the ECC and Diffie-hellman key exchange as it makes it hard to guess the key against the Brute force attack.

Also, SSCC will be developed as a web based application and a mobile application to make it flexible and portable for every user and everywhere. Users need only their own login information to access the data in order to modify them through an easy set of steps. Dealing with a robust secure system users will be able to block unauthorized access even though the Cloud provider itself. These mechanisms will courage users to upload their data to cloud environment with no worries about their sensitive data.

Bibliography

[1] B. Acharya, H. Agrawal, A. Modi, ,U. K Agrawal."Combined Implementation of Robust Cryptosystem for Non-invertible Matrices based on Hill Cipher and Steganography", Proc. of Int. Conf. on Advances in Computer Science, 2010.

[2] B. Acharya, M. Sharma, , S. Tiwari,V. Minz. "Privacy protection of biometric traits using modified hill cipher with involutory key and robust cryptosystem", Procedia Computer Science,2010.

[3] S. Agrawal, S. Joshi, B. Purohit. "Secure Data Communication In A Cloud Environment using Row Column Diagonal (RCD)", International Conference on Soft Computing Techniques and Implementations, 2015.

[4] D. Arockiam, S. Monikandan. "Data Security and Privacy in Cloud Storage using Hybrid Symmetric Encryption Algorithm". International Journal of Advanced Research in Computer and Communication Engineering,2013.

[5] D. Arockiam, S. Monikandan. "Efficient Cloud Storage Confidentiality to Ensure". International Conference on Computer Communication and Informatics,2014.

[6] L. Batten."Public Key Cryptography: Applications and Attacks", IEEE Press Series, 2013.

[7] A. Berisha, B. Baxhaku, A. Alidema."A Class of Non Invertible Matrices in GF (2) for Practical One Way Hash Algorithm", International Journal of Computer Applications, 2012.

[8] A. Kalai Selv, Dr.M. Sathik." Polynomial Based Secret Sharing Scheme for Image Encryption Based on Mathematical Theorem", International Journal of Advanced Research in Computer Science, 2011.

[9] D. Dinadayalan, S. Jegadeeswari, D. Gnanambigai. "Data Security Issues in Cloud Environment and Solutions", World Congress on Computing and Communication Technologies, 2014.

[10] H. Gururaja, M. Seetha, A. Koundinya, A. Shashank, C. Prashanth. "Comparative Study and Performance Analysis of Encryption in RSA, ECC and Goldwasser-Micali Cryptosystems". International Journal of Application or Innovation in Engineering & Management,2014.

[11] R. Hamamrah, M. Farajallah. "Design of a Robust Cryptosystem Algorithm for Non-Invertible Matrices Based on Hill Cipher", IJCSNS International Journal of Computer Science and Network Security, 2009.

[12] D. Hankerson, A. Menezes, S. Vanstone. "Guide to Elliptic Curve Cryptography", Springer proffisonal computing, 2004.

[13] K. Hashizume, D. Rosado, E. Fernández-Medina, E. Fernandez. "An analysis of security issues for cloud computing", Journal of Internet Services and Applications, 2013.

[14] S. Kuyoro, F. Ibikunle, O. Awodele, "Cloud Computing Security Issues and Challenges", International Journal of Computer Networks, 2011.

[15] I. Ismail. "How to b the Hill cipher",2006.

[16] L. Janczewski. "Internet and Intranet Security Management: Risks and Solutions", Idea Group Publishing, 2000.

[17] I. Khalil, A. Khreishah, M. Azeem. "Cloud Computing Security: A Survey", available at: http://www.mdpi.com/,2014.

[18] J. Li, D. Xie, Z. Cai. "Secure Auditing and Deduplicating Data in Cloud", IEEE Conference on Computer Communications,2015.

[19] S. Lunawat, A. Patankar . "Efficient Architecture for Secure Outsourcing of Data and Computation in Hybrid Cloud", International Conference on Reliability, Optimization and Information Technology,2014

[20] K. Mar, C. Law, V. Chin. "Secure Personal Cloud Storage", the 10th International Conference for Internet Technology and Secured Transactions, 2015.

[21] P. Mell, T. Grance. "The NIST Definition of Cloud", National Institute of Standards and Technology, 2011.

[22] U. MogheI, B. Chaturvedi, P.Lakkadwala. "Cloud Secure Resource Sharing Algorithm from Object Based Sharable Environment", International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2015.

[23] M. Nazir, P. Tiwari, S. Tiwari,"Cloud Computing: An Overview", available at: http://ebooks.hctl.org/cloud-computing/chapter-1.pdf, 2015.

[24] C. Paix˜ao, F. Coelho. "Matrix compression methods", available at: https://peerj.com/articles/,2010.

[25] Q. Quan , W. Tian-hong, Z. Rui, X. Ming-jun"A Model of Cloud Data Secure Storage Based on HDFS",Computer and Information Science (ICIS),2013.

[26] S. Singh, A. Garg, AnshulSachdeva. "Comparison of Cryptographic Algorithms: ECC & RSA",International Journal of Computer Science and Communication Engineering,2013.

[27] R. Sinha, H. Srivastava, S.Gupta. "Performance Based Comparison Study of RSA". International Journal of Scientific & Engineering Research,2013.

[28] D. Slamanig ,C. Hanser. "On Cloud Storage and the Cloud of Clouds Approach", Internet Technology And Secured Transactions,2012.

[29] W. Stallings,"Cryptography and Network Security", Prentice Hall, 2005.

[30] Y. Sun, J. Zhang,Y. Xiong,G. Zhu. "Data Security and Privacy in Cloud Computing", International Journal of Distributed Sensor Networks, 2014.

[31] S. Tang, F. Liu,"A one-time pad encryption algorithm based on one way hash and conventional block cipher", International Conference on Consumer Electronics, Communications and Networks (CECNet), 2012.

[32] C. Tsai, U. Lin, A. Chang, C. Chen,"Information security issue of enterprises adopting the application of cloud computing", Networked Computing and Advanced Information Management (NCM), 2010.

[33] P. Urien,"Cloud of Secure Elements: An Infrastructure For The Trust of Mobile NFC Services",Wireless and Mobile Computing, Networking and Communications (WiMob), 2014.

[34] L. Xu, P. Khoa, S. Kim, W. Ro,W. Shi. "LUT based Secure Cloud Computing-an Implementation using FPGAs", ReConFigurable Computing and FPGAs (ReConFig),2014.

[35] Z. Zhu, R. Jiang."A Secure Anti-Collusion Data Sharing Scheme", Transactions On Parallel And Distributed Systems, 2016.

[36] D. Zill,W. Wright, "Advanced Engineering Mathematics", Jones and Bartlett, 2016.

[37] D. Zissis, D. Lekkas."Addressing cloud computing security issues", Future Generation Computer Systems, 2012.

[38] B. Zwattendorfer, A. Tauber. "Secure Cloud Authentication UsingEids", Cloud Computing and Intelligent Systems (CCIS),2012.

[39] A. Khaldi, K. Karoui, N. Tanabène, H. Ghezala. "A secure cloud computing architecture design". International Conference on Mobile Cloud Computing, Services, and Engineering, 2014.

[40] R. B. Bohn, J. Messina, F. Liu, J. Tong, J. Mao."NIST Cloud Computing ReferenceArchitecture", National Institute of Standards and Technology,2011.

[41] Y. Huiming, P. Nakia, S. Dexter, and Y. Xiaohong. "Cloud computing and security challenges",In 50th Annual Association for Computing Machinery Southeast Conference,2012.

[42] A. Hammami, N. Simoni, and R. Salman. "Ubiquity and QoS for cloud security", International Conference on Parallel Processing Work-shops (ICPPW),2012.

[43] M. Jensen, J. Schwenk, N. Gruschka, and L.L. Iacono."On technical security issues in cloud computing". International Conference on Cloud Computing,2009.

[44] J. Bansidhar, A. Santhana, and B. Joshi. "Securing cloud computing environment against DDoS attacks". In Computer Communication and Informatics (ICCCI), 2012.

[45] S. William. "Network Security Essentials",Pearson Education India, 2008.

[46] P. Srivastava, S. Singh, A. Pinto, S. Verma,V. Chaurasiya, and R. Gupta. "An architecture based on proactive model for security in cloud computing", Recent Trends in Information Technology (ICRTIT), 2011.

[47] L. Savu. "Cloud computing: Deployment models, delivery models, risks and research challenges", Computer and Management (CAMAN), 2011.

[48] L.Qing ,Z. Wang and et al," Applications integration in a hybrid cloud computing environment: modelling and platform", Enterprise Information Systems,2013.

[49] L. Ertaul, S. Singhal, and G. Saldamli. "Security challenges in cloud computing", International Conference on Security & Management, 2010.

[50] A. Bhardwaj and V. Kumar,"Cloud security assessment and identity management", Computer and Information Technology (ICCIT), 2011.

[51] K. Jasim, S. Abbas, M. El-Horbaty, and M. Salem, "Efficiency of Modern Encryption Algorithms in Cloud Computing", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),2013.

[52] A. Pansotra, and S. Singh, "Cloud Security Algorithms", International Journal of Security and Its Applications,2015.

[53] A. Bhardwaja, G. Subrahmanyamb, V. Avasthic, and H. Sastryd ." Security Algorithms for Cloud Computing", International Conference on Computational Modeling and Security (CMS 2016),2016.

[54] R. Rivest,"The MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992.

[55] M. Rosulek, "Block Cipher Modes of Operation", Mike Rosulek,2016.

[56] C. Jansen, P. Vlist, "Message encipherment with minimal expansion and redundancy—doing better than ISO-10126", Elsevier Ltd, 1996.

[57] R. Housley,"Cryptographic Message Syntax (CMS)", Internet Engineering Task Force (IETF),2010.

[58] B. Johnson, E. Cygnacom,"Improving Hash Function Padding", National Institute of Standards and Technology,2005 .

[59] F. Moghaddam, I. Ghavam, D. Varnosfaderani, S. Mobedi. "A Client-Based User Authentication and Encryption Algorithm for Secure Accessing to Cloud Servers", IEEE Student Conference on Research and Development (SCOReD), 2013.

[60] S. Gupta and J. Sharma, "A Hybrid Encryption Algorithm based on RSA and Diffie-Hellman", Computational Intelligence & Computing Research (ICCIC),2012.

نموذج مشاركة البيانات بشكل امن في الحوسبة السحابية.

إعداد : إسراء محمود عبدالفتاح القطو.

إشراف: د. رشدي حمامرة.

ملخص:

تعتبر الحوسبة السحابية (Cloud Computing) من أهم التوجهات في الوقت الحالي ، حيث تلعب دورا هاما في توفير المصادر والمساحات التخزينية القابلة للتوسعة (scalability) حسب حاجة المستخدم ، لكن مشكلة أمن المعلومات (Security) تعد من اهم المشاكل التي تواجه الحوسبة السحابية لاسيما انه تتم مشاركة المصادر من قبل المستخدمين .الحوسبة السحابية هذه الأيام تأخذ موضع الصدارة باعتبارها ذات أهمية كبيرة كمكان تخزيني للبيانات، ويسمى بالتخزين السحابي (Cloud Storage) حيث يتم وضع البيانات في مكان خارجي بعيدا عن المكان الأصلي لتخزين البيانات كما هو معتاد . كل البيانات يتم حفظها بأماكن غير معروفة للمستخدمين حيث لا يتم تحديد المكان بدقة ، ومن مهمة مزود مساحات التخزين في الحوسبة السحابية توفير مكان امن لهذه البيانات ، لكن المستخدمين ما زالوا يواجهون قلقاً اتجاه مستوى الحماية لهذه البيانات وإمكانية الاطلاع عليها من قبل المزود .

في هذا البحث تم اقتراح نموذج مشاركة البيانات بشكل امن في الأنظمة الموزعة بشكل خاص للحوسبة السحابية (Secure Data Sharing Model for Cloud Computing) . ويهدف هذا النموذج والمسمى SSCC لحفظ البيانات بطريقة آمنة وحمايتها من الوصول غير المصرح به (unauthorized access). يأخذ نموذجنا نص عادي كمدخل (input) ثم بعد ذلك وباستخدام مجموعة من الخطوات يتم تشفيرها (encryption) باستخدام إستراتيجية التشفير بالمفتاح المتماثل (symmetric key) الذي يستخدم مفتاح سري واحد (secret key) مكتسب من طرف ثالث (Third party)موثوق به حيث يكون لكل مستخدم مفتاح خاص به .هذه الخطوات تشمل : الترميز(Encoding)، وضغط البيانات (Data Compression)، وحشوه من النص (Padding) ثم يحولها إلى مجموعة من كتل مصفوفة (blocks) وباستخدام مبدأ ضرب المصفوفات يولد مصفوفة الشفرات (Ciphertext matrix) والذي يحتوي على نص مشفر كمخرج (output) من نص عادي.

تقدم هذه الأطروحة طريقة جديدة للتعامل مع البيانات قبل تخزينها في المكان التخزيني التابع للحوسبة السحابية بحيث تسمح للمستخدم الوصول والحصول على المعلومات بأمان من سحابة التخزين ، بحيث تضمن أمنها وحمايتها حتى لو تم تخزينها في أماكن غير معروفة خصوصا إذا كان هناك مشكلة أمنية في مزود الخدمة

حيث انه يتم ترميز وضغط وتشفير البيانات قبل إرسالها لمكان التخزين وبالتالي لا يهم في ما اذا كان هناك وصول غير موثوق للبيانات سواء كان من قبل المزود أو من قبل أطراف خارجية أخرى .

وتم الحصول على النتائج للنموذج المقترح من خلال عمل تجربة على مجموعة من البيانات باستخدام طريقتين مختلفتين الأولى تعتمد على ادخال البينات على مصفوفة مربعة وتعبئتها بشكل كامل وتسمى (SSCC1) و الثانية تعتمد على تعبئتها بشكل جزئي على شكل مصفوفة مثلثة علوية upper triangular matrix و تسمى (SSCC2) بحيث أثرت على عملية التحميل \ التنزيل (Upload\Download) في الطريقة الثانية لان كمية البيانات المنقولة اكبر وأعطت انطباعا جيدا مقارنة بالطرق السابقة لتشفير البيانات حيث كان ناتج المقارنة من ناحية الوقت اللازم للتشفير بين (SSCC) و (HC) تحسين بمقدار 6.86% بينما عند المقارنة بين(SSCC) و (AES) كان الفرق بينهما بمقدار3.94% لصالح (AES) وهذا يعني ان النموذج المقترح لديه فرصة لا بأس بها لمنافسة النموذج المعياري .

بالإضافة إلى مقارنات أخرى تشمل حالتين لكل طريقة من الطرق النموذج المقترح (SSCC1) و (SSCC2)حيث كانت تشمل المقارنة للطريقة الأولى (SSCC1) مع الضغط للبيانات وبدونه لنفس الملفات ومقارنة الطريقة الثانية (SSCC2)مع الضغط وبدونه لنفس الملفات لإظهار اثر ضغط البيانات على الزمن اللازم لعملية التشفير(Encryption Time) لهذه الملفات حيث تم تقليل الوقت بنسبة 96.287% باستخدام الطريقة الأولى ونسبة 67.146% باستخدام الطريقة الثانية .

Acronyms and Abbreviations

SSCC	Secure data Sharing in Cloud Computing
MRHC	Mousa Rushdi Hill Cipher
AES	Advanced Encryption Standard
COTS	Commercial of the Shelf
PaaS	Platform as a service
SaaS	Software as a service
IaaS	Infrastructure as a service
HaaS	Hardware as a service
IT	Information Technology
KeyGen	Key Generator
P	Plaintext matrix
K	Key matrix
C	Ciphertext matrix
RSA	Rivest Shamir Adleman
EEC	Elliptic Curve Cryptography
gcd	greatest common divisor
mod	modulus arithmetic
RCD	Row Column Diagonal
EDM	Encryption Decryption Model
SeCloud	Secure Cloud
ISO	International Organization for Standardization
IDA	Information Dispersal Algorithm
LUT	Lookup Table
DNA	Deoxyribonucleic acid
HDFS	Hadoop distributed file system
CoSE	Cloud of secure elements
CS – RSA	Cloud Secure - Resource Sharing Algorithm
STROKE	Secure idenTity acRoss bORders linKEd
MKG	Matrix Key Generator
E_{pub}	Encryption within public key
D_{pr}	Decryption within private key

Some of MATLAB Codes

Blocks_matrix.m

```
function [blocks]=blocks_matrix(blockSize,B)

% matrix for encoded values
blocks = zeros(ceil(length(B)/blockSize),blockSize);
%outer indexing
j=1;
for i = 1 : blockSize: length(B)-blockSize

  blocks(j,:)=double(B(i:i+blockSize-1))-96;

  j=j+1;
end
  blocks(j,1:length(B(i+blockSize:end)))=double(B(i+blockSize:end))-96;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Checkprimpitive.m

```
% To find the primitive root of the given value using powermod function

function foundAlpha=checkPrimitive(q)
p=0;
for alpha= 2 : q-1
  a=zeros(1,length(q-1));
  for i=1:q-1
    a(i)=powermod(alpha,i,q);
  end
  b=zeros(1,q-1);
  b=unique(a);
    if length(a)==length(b)
    foundAlpha=alpha;
    break;
    end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

compressedBlocks.m

```
function [compBlocks] = compresed_Blocks(blockSize,C,blocks,B)

% matrix for compressed values
compBlocks = zeros(ceil(length(B)/blockSize),1);
%convert to binary and compression
```

```
j=1;
C = cellstr(dec2base(blocks,3));
CC='';

for i = 1 : length(C)

    CC=strcat(CC,C(i)) ;
    if  mod(i,blockSize)== 0
     compBlocks(j)=base2dec(CC,3);
     %disp(CC);
     CC='';
     j=j+1;
    end
end
%disp(CC);
if j<=ceil(length(B)/blockSize)
compBlocks(j)=base2dec(CC,3);
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Decrypt.m


% function call [output]=decryp(lower2_matrix,C,blockSize)
function output2=decryp(K,C,blockSize)
tic;
P=abs(linsolve(K,C'))
P=int64(P);
CompBlocks=P(P~=0 & P~=80)
Blocks=dec2base((CompBlocks'),3);
display(length(Blocks))
if length(Blocks) == (3*blockSize)-1
Blocks=strcat('0',dec2base((CompBlocks'),3));
end

Blocks2=Blocks';
size_Blocks=size(Blocks);
j=1;k=size_Blocks(2);
output=zeros(blockSize,size_Blocks(1));
output2=zeros(size_Blocks(1),blockSize);
for i = 1 : size_Blocks(1)

 output(:,i)=base2dec(findCharacters(Blocks2(j:k)),3);
 j=j+size_Blocks(2);
 k=k+size_Blocks(2);
end
stringSize=blockSize*size_Blocks(1);
for i = 1 : stringSize
```

```
output2(i)=output(i);
end
output3=reshape(output2',stringSize,1);
display(char(output3(output3~=0)+96)')
toc;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

keyGen.m

```
function [lower2_matrix,counter]=key_Gen(n , x , y ,counter , lenComp)
%calling [lower2_matrix,counter]=key_Gen(11 , 2 , 3 ,counter
,length(compressed_Blocks))
%------ Diffie_Hellman_main -----------
%n=11; %must be a prime number
g=checkPrimitive(n); % primitive root of n
%x=2; % private key for the first person
%y=3; % private key for the second person

%a=powerDiff(g,x,n);
b=powerDiff(g,y,n);
%display(powerDiff(a,y,n));

%-------- Perpare Key matrix -----------
%CC{1}=20;
% slove it to find big values or duplicate well
CC{1}=powerDiff(b,x,n);
%write a full algorithm %duplicate key value


%KK=strcat(int2str(randVal),dec2bin(CC{1}));
K=dec2bin(CC{1});

%display(counter)
while length(K) ~= ((counter+1)*(counter+1))
randVal=int2str(round(rand(1)));
K=strcat(K,randVal);
%i=i+1;
end

%fliplr to read the stream of bits from left to right (more values)
[keys]=key_matrix((counter+1),fliplr(K),lenComp)
[lower2_matrix]=lu_matrix(keys)
%--------------------------------------
End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

keyMatrix.m

```
function [keys]=key_matrix(counter,B,LengthcompBlocks)
```

```
% matrix for encoded values
keys = ones(1,LengthcompBlocks);
%outer indexing
j=1;step=3;
for i = 1 : LengthcompBlocks
  if j <= length(B)-3
  keys(i)=bin2dec(B(j:step));
  step=step+3;
  j=j+3;
  end
end
keys(keys ==0 )=1;
%  keys(j)=bin2dec(B(i+counter:end));

End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Lumatrix.m

```
%half plain text matrix
% With padding 80  .. to complete matrix
function [lower_matrix]=lu_matrix2(compBlocks)
count=135;
lower_matrix = zeros(count,count);
if length(compBlocks)<count
compBlocks=[compBlocks 80*ones( 1,count-length(compBlocks))]
lower_matrix = triu(ones(count),1);
lower_matrix(~~lower_matrix)=compBlocks;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Test.m

```
clc;
%%message length must be greater than blocksize by 1
%%_______________Divide message into Blocks and
encoding__________________%
%Import text file
fileID = fopen('Hello_n2.txt');

C = textscan(fileID,'%s','bufsize', 200000);
%fclose(fileID);
celldisp(C);
B=char(C{1});%----- Prepare plain text matrix -----%Store plaintext into cell array
blockSize=135; %  amount of compression must equal n
tic;
blocks=blocks_matrix(blockSize,B)
[compBlocks] = compresed_Blocks(blockSize,C,blocks,B)
%if length(compBlocks)<blockSize
```

```
%compBlocks=[compBlocks 80*ones( 1,blockSize-length(compBlocks))];
%end
lower_matrix=vec2mat(compBlocks,9)
C=lower2_matrix*lower_matrix';
%[C]=lu_Multi(lower2_matrix,lower_matrix');
toc;
%[lower_matrix]=lu_matrix(compBlocks)
%upper_matrix=lower_matrix'
%lengthComp=length(compBlocks)
%display(counter)

%format bank to display full results
%linsolve(lower2_matrix,C') to find the plain text matrix
```