Palestine Polytechnic University
Deanship of Graduate Studies and Scientific Research
Master of Informatics

Secure Data Sharing Model for Preserving Privacy

by

Sanaa Sarahneh

Supervisor

Dr. Radwan Tahboub

A thesis submitted in partial fulfillment of requirements of the degree of Master of Informatics

August 2017

The undersigned hereby certify that they have read, examined and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of a thesis entitled: **Secure Data Sharing Model forPreserving Privacy**, submitted by **Sanaa Sadi Mohammad Sarahneh** in partial fulfillment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

Dr. Radwan Tahboub (Supervisor), Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Liana Tamimi (Internal committee member), Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Rushdi A. Hamamreh (External committee member), Al-Quds University.

Signature:_____ Date:_____

Thesis Approved

Dr. Murad Abu Subaih

Dean of Graduate Studies and Scientific Research

Palestine Polytechnic University

Signature:_____ Date:_____

DECLARATION

I declare that the Master Thesis entitled **"Secure Data Sharing Model for Preserving Privacy"** is my original work, and hereby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgement is made in the text.

Sanaa Sarahneh

Signature:_____ Date:_____

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for the master degree in Informatics at Palestine Polytechnic University, I agree that the library shall make it available to borrowers under rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from, reproduction, or publication of this thesis may be granted by my main supervisor, or in his absence, by the Dean of Graduate Studies and Scientific Research when, in the opinion of either, the proposed use of the material is for scholarly purposes.

Any coping or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Sanaa Sarahneh

Signature:_____ Date:_____

DEDICATION

To my father and mother who have been a source of encouragement and inspiration to me throughout my life. They always supported me and encouraged me to go for the Master degree. I also dedicate this thesis to my beloved brothers and sisters for their persistent support. I also dedicate this work and give special thanks to my friend Luma Al Sharabati.

ACKNOWLEDGEMENT

As I write the last words of this thesis, I give thanks to God for his protection and giving me the ability to do the work. I greatly appreciate the thesis's supervisor Dr. Radwan Tahboub for his sound advice, careful guidance, continual support, encouragement and time he spent with me to accomplish the thesis successfully.

I also wish to thank the members of my committee, Dr. Liana Tamimi and Dr. Rushdi A. Hamamreh for their suggestion, comments and additional guidance were invaluable to the completion of this work.

Also, I would like to convey my really thanks to Dr. Mousa Farajallah for his guidance, comments, ideas and feedback.

A huge thanks to Eng. Rafat Aljuneidi for his valuable guidance, in doing the experiments for transferring files.

I am deeply and forever indebted to my brothers and sisters for their love, support and encouragement throughout my entire life.

To each of the above, I extend my deepest appreciation.

Abstract

Electronic data interchange can be classified as one of the important areas of information technology, where the need for data sharing increasingly required in almost every field such as health-care, e-government, e-commerce and scientific researches. Data sharing concept can be defined as the process of interchanging, analyzing, retrieving and integrating data among multiple data sources in a controlled access manner. The use of information technology in different areas began to increase. The exchange and sharing different types of information have been also required. Although data sharing facilitates the way that data can be exchanged, security concerns arise as a challenge for conducting data sharing. Many polices including confidentiality, integrity, availability and privacy must be taken into consideration. In this thesis, preserving privacy subject will be considered. This thesis will provide a literature review for different privacy preserving models that facilitate data sharing among different areas. Also, it will propose a new privacy preserving data sharing model that combines compression techniques such as Huffman coding and different encryption algorithms in order to provide privacy in data sharing to facilitate data sharing in different areas. Performance parameters, such as processing and transfer time, and the enhancement for the expected model are also considered. Experiments results over 10 different files sizes show that the proposed model provides privacy to shared files also it reduces large file size since the proposed model use Huffman coding. The proposed model is 87% faster on speed of 1MB/sec and on speed of 16MB/sec it is 38% faster than existing models.

الملخص

في الوقت الحاضر، قامت معظم المؤسسات والشركات بتوسيع عملها في نطاق الشبكة الخارجية لتسهيل تبادل البيانات بين بعضها البعض مثل الرعاية الصحية، التجارة الالكترونية، الحكومة الالكترونية والأبحاث العلمية. ويمكن تعريف مشاركة البيانات على انها عملية تبادل البيانات وتحليلها واسترجاعها ودمجها بطريقة يمكن التحكم بها. وعلى الرغم من ان عملية مشاركة البيانات تُسهل الطريقة التي يمكن من خلالها تبادل البيانات الا انه هناك تحديات لاتمام عملية مشاركة البيانات كالمخاطر الامنية. هناك العديد من السياسات التي يجب اتباعها لتحقيق الامن للمعلومات كالخصوصية.

هذه الرسالة تركز على كيفية المحافظة على الخصوصية عند مشاركة البيانات. هذه الرسالة تقوم بعرض نموذج جديد لمشاركة البيانات والحفاظ على الخصوصية ولتحقيق ذلك عرضت الرسالة مجموعة من النماذج لمشاركة البيانات ومقارنة فيما بينهم. يقوم النموذج الجديد لتبادل البيانات على استخدام مفهوم ضغط البيانات مثل Huffman coding واستخدام خوارزميات التشفير المعروفة وذلك لتوفير الخصوصية عند مشاركة البيانات. وبهدف تقييم النموذج الجديد لمشاركة البيانات تم مقارنته بعدة نماذج موجودة لمشاركة البيانات و تقييمه باستخدام معاملات مثل : تحقيق الخصوصية، سرعة مشاركة البيانات وحجم الملفات التي يتم نقلها. وكانت النتائج ان النموذج المقترح يحافظ على الخصوصية مقارنةً مع نماذج أخرى لمشاركة البيانات، أيضاً النموذج المقترح أسرع في نقل ومشاركة البيانات بـ ٨٧% على سرعة ١ ميجابايت في الثانية، في حين انه على سرعة ١٦ ميجابايت في الثانية أسرع بـ ٣٨ %.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AES	**A**dvanced **E**ncryption **S**tandard
ARP	**A**ddress **R**esolution **P**rotocol
A-RSA	**A**ugmented-RSA
ASCII	**A**merican **S**tandard **C**ode For **I**nformation **I**nterchange
CBC	**C**ipher **B**lock **C**haining
CPA	**C**hosen **P**laintext **A**ttack
DES	**D**ata **E**ncryption **S**tandard
DHT	**D**istributed **H**ash **T**able
EDI	**E**lectronic **D**ata **I**nterchange
EHR	**E**lectronic **H**ealth **R**ecords
FOB	**F**requency **O**f **B**lock
gcd	**g**reater **c**ommon **d**ivisor
IBM	**I**nternational **B**usiness **M**achines corporation
IV	**I**nitialization **V**ector
IP	**I**nitial **P**ermutation
KB	**K**ilo **B**yte
LAN	**L**ocal **A**rea **N**etwork
Mb	**M**ega **b**it
MB	**M**ega **B**yte
MBps	**M**ega **B**yte **p**er **s**econd
MAC	**M**essage **A**uthentication **C**ode
MAC address	**M**edia **A**ccess **C**ontrol address
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
OSI	**O**pen **S**ystem **I**nterconnection
P2P	**P**eer to **P**eer
RLE	**R**un **L**ength **E**ncoding
RSA	Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman
RTT	**R**ound **T**rip **T**ime
SHA	**S**ecure **H**ash **A**lgorithm
SSL	**S**ecure **S**ockets **L**ayer
SWRL	**S**emantic **W**eb **R**ule **L**anguage
SQWRL	**S**emantic **Q**uery-enhanced **W**eb **R**ule **L**anguage
TA	**T**iming **A**ttack
TCP/IP	**T**ransmission **C**ontrol **P**rotocol/ **I**nternet **P**rotocol
VP	**V**irtual **P**latform
WAN	**W**ide **A**rea **N**etwork
XML	e**X**tensible **M**arkup **L**anguage
XOR	e**X**clusive **OR**

Symbols

B	Binary file
$B\prime$	Binary file blinded with randomized component Y
K_B^+	Bob's Public Key
K_B^-	Bob's Private Key
C	Chiphertext
$C\prime$	Ciphertext of the header file blinded with randomized component Y
C_i	Block (i) of a ciphertext
d	Secret exponent
e	Public exponent
F_i	Frequency of a character in a file to be compressed
F_s	File size before compression
CF_s	File size after compression
H	Header file
K_{AES}	AES encryption key
$L(C)$	Huffman code length
M	Message
mod	Modules, Remainder after division
N	RSA factorization
P	Plaintext
p and q	Large prime numbers
P_i	Block (i) of a plaintext
P_{xor_i}	XOred block (i) of a plaintext
Y	Randomized parameter
$\phi(N)$	Euler's totient function

Chapter 1

Introduction

The use of information technology in different areas began to increase since 1980s; the exchange and sharing different types of information have been required at that time. Although data sharing facilitates the way that data can be exchanged, security concerns arise as a challenge for conducting data sharing. Many polices include confidentiality and privacy must be taken into consideration. However data sharing and integration are prevented from being widespread because of privacy concerns. For example in e-commerce areas, companies need to exchange information to boost productivity, but are prevented by fear from competitors. Also sharing data in healthcare areas improve scientific research and enables early detection of disease, but without preserving privacy it is costly and difficult to make healthcare information globally expand [1]. Privacy can be defined as the process to protect information from unauthorized access [2]. Unrestricted data sharing will reduce the privacy of individuals, the challenge is to enforce appropriate security policies that facilitate data sharing as needed. These policies include policies for confidentiality, privacy, and trust. During normal operations, it is important to apply security policies [3–5].

Data sharing is a fundamental enabler of coordination among supply chain partners [6]. Also, data sharing can be defined as the process of interchanging, analyzing, retrieving and integrating data among multiple data sources in a controlled access manner. Data sharing is an important feature for modern organizations due to the increase in the use of communication networks, changes in architectures of enterprise information

systems, as well as the increasing availability of data in computerized form, and perhaps the biggest impact on data sharing can be attributed to the widespread use of the Internet and Internet-related technologies for e-government, e-commerce, scientific research and health-care [6, 7]. In e-commerce, data can be shared for transactions, operations, and analysis. Conducting business transactions is a basic reason for sharing data in e-commerce it is mainly used in Electronic Data Interchange (EDI) [4, 6]. Although data sharing facilitates the way that data can be exchanged, security concerns arises a challenge for conducting data sharing, confidentiality, availability and privacy must be taken into consideration, this means, a controlled access is required to authorize authenticated users or roles to access data. This increase the need for data sharing management and data integration [1]. However data sharing and integration are prevented from being widespread because of privacy concerns. Confidentiality concept and data privacy are almost have the same meaning; which is to limit access to personal information. Different models have been introduced to apply privacy in data sharing and data integration. Each may be the same or different structure of other, such as Semantic Privacy-Preserving Model [8], Capability-based Access Control Model [9], BitTorrent Protocol [10], and OneSwarm Data Sharing Model [2]. Sensitive data must be encrypted when share it, the challenge is to apply policies that preserve privacy. Preserve privacy is to protect data from unauthorized users and to control who can access data [4, 11, 12]. OneSwarm data sharing model is a peer to peer (P2P) data sharing model that provide privacy. Also, it provides better performance than existing data sharing models, it uses Secure Socket Layer (SSL) to provide privacy for the users. Figure 1.1 is an overview for OneSwarm data sharing model structure, it shows a P2P network with secure connection using SSL [2, 13].

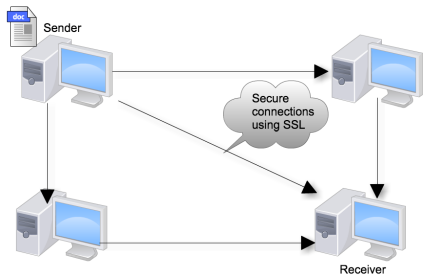


FIGURE 1.1: OneSwarm Structure [2].

Two basic encryption techniques are used to get secure system, symmetric key encryption and asymmetric key encryption. In symmetric key encryption algorithms, the sender

and receiver have the same key for encryption and decryption process while in asymmetric key encryption algorithms, there are two different keys, the private key that must be secret and the public key [14, 15]. Rivest-Shamir-Adleman (RSA) cryptosystem is asymmetric key encryption algorithm [16]. RSA is not semantically secure, encrypts the same message more than once always gives the same ciphertext. Also, RSA does not use to encrypt large data, and it is slower than the symmetric key encryption [17, 18]. A new encryption technique is implemented by [17] called Augmented-RSA (A-RSA). A-RSA has three enhancement factors comparing with RSA. These factors are the security, the execution time, and the size of the ciphertext [17]. Also Secure Socket Layer (SSL) is used to provide privacy, SSL uses symmetric key encryption algorithms such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES). The symmetric key encryption algorithms is faster than the asymmetric key encryption and provide more security [19]. Data Compression is a technique to reduce files size which helps in increasing network bandwidth and reducing transmission time. Also data compression is used in security areas because it decodes the original file such as Huffman coding. Huffman Coding is a lossless data compression algorithm that decode original file to get smaller file size [20–22].

The proposed data sharing model is a model that preserves privacy by using symmetric key encryption algorithm and data compression. The proposed model provide privacy to shared files also it reduces transmission time, and the size of the shared data. Figure 1.2 shows the main components for the proposed model.

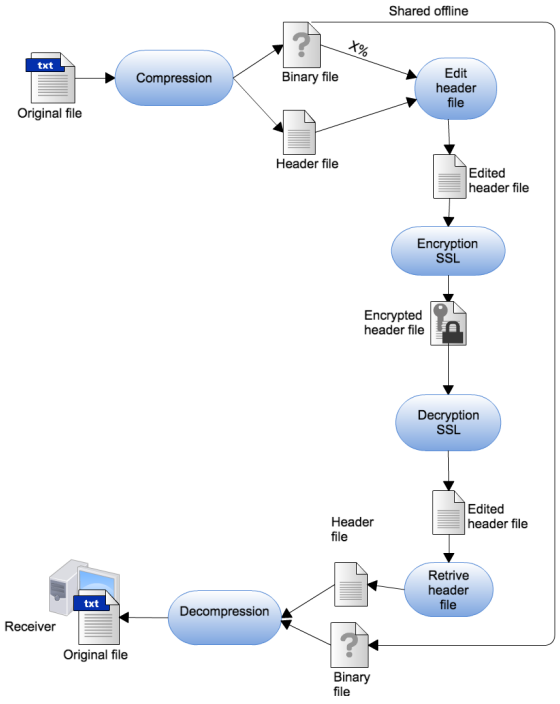


FIGURE 1.2: The Proposed Model Components.

1.1 Problem Statement

Most of existing data sharing models suffer either from privacy problems or from poor performance (computational processes, network bandwidth, storage and volume of transfer) if confidentiality and integrity are considered for the whole shared files and data [2, 13]. Hence increasing privacy may case a performance hit for large files. In this thesis, an attempt to study and analyze existing data sharing models. The idea of using data compression such as Huffman coding to partition shared files into header file and binary file will be considered and tested to increase the performance of a data sharing model like OneSwarm model. Both computation and transfer performance problems will be addressed, studied and compared with the existing data sharing models. To achieve this, the following sub problems are expressed:

- How data sharing models that preserves privacy work?
- How to use compression and cryptographic techniques to share data with better performance?
- How to use cryptographic techniques to preserve privacy for the shared data?

- How to enhance transferring time and computational time for the data sharing models?

1.2 Thesis Objectives

The thesis aims to practice secure data sharing among different organizations focusing on preserving privacy. To achieve this objective, this requires to achieve the following sub objectives:

- To survey preserving privacy data sharing models.
- To study parameters that are used in preserving privacy data sharing models including computational and transfer parameters.
- To study different encryption and compression techniques that are used in preserving privacy models.
- To propose a new data sharing model for preserving privacy.
- To compare the results after applying the new secure data sharing model and see the enhancement for the new suggested model.

1.3 Thesis Methodology

Thesis methodology is as follows:

- Make surveys for previous work in this area and for different data sharing privacy preserving models, and using understanding and interpretation of this work.
- Study different encryption and compression techniques, then implement these techniques to have new privacy preserving data sharing model.
- Combine compression and encryption techniques to build privacy preserving data sharing model with better performance.
- Study the parameter (computational processes time, network bandwidth, storage and volume of transfer) for the new preserving privacy model.

1.4 Thesis Contributions

Most of existing data sharing models suffer either from privacy problems or from poor performance. Because of these problems, the thesis introduces a solution to overcome these challenges. The main contributions of this thesis are summarized as follows:

- New data sharing model that preserves privacy to shared data using compression and encryption techniques is presented.
- The proposed model provides better performance compared to the existing data sharing models by combining compression and encryption techniques.
- The proposed model reduces sharing data time by using compression techniques. Also compression technique is used to reduce data size, which enhances network bandwidth utilization.

1.5 Publications

The following are the publications for the data sharing model:

- Sarahneh, S., Tahboub, R., (2017), "Secure Data Sharing Model for Preserving Privacy", *Journal of Theoretical and Applied Information Technology*, 95.
- Sarahneh, S., Tahboub, R., (2015), "Secure Data Sharing Polices and Architecture Preserving Privacy", *The 7th International Conference on Information Technology*, Jordan, 2015. Amman: Al-Zaytoonah University.
- Sarahneh, S., Tahboub, R., (2015), "Secure Data Sharing Model Using New Technique for Preserving Privacy", *Journal of Internet Technology and Secured Transactions*, 4, (4), 472-434.

1.6 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 presents a background for data sharing concept and security in data sharing also in this chapter we made a survey of some important data sharing models that preserve privacy and different encryption and compression techniques.

Chapter 3 introduces a new privacy preserving data sharing model. Then it presents the generic standard data sharing Model and the proposed data sharing model. Then the main components for the proposed model are described. Finally, it discusses how the model preserves privacy, saves storage and decrease the transfer time.

Chapter 4 presents the experimental results of applying the proposed preserving privacy data sharing model. Several experiment scenarios were performed to evaluate our model in terms of parameters such as transferring time, computation time, files size and preserving privacy.

Chapter 5 summarizes the general idea of the proposed model. Then future work is presented in the next section.

Chapter 2

Background and Literature Review

This chapter presents a background for data sharing concept and security in data sharing. Also in this chapter a survey of some important data sharing models and different encryption techniques are clarified. The chapter is organized as follows: Section 2.1 illustrates data sharing concepts and different data sharing models. Section 2.2 clarifies network technologies. Section 2.3 clarify important definitions of cryptography and network security then it presents a survey for different encryption techniques. Finally, different compression techniques are illustrated in section 2.4.

2.1 Data Sharing

This section clarifies data sharing concept then it lists different data sharing models and a comparison between them.

2.1.1 Data Sharing Concepts

Data sharing concept emerges to introduce a new era of cloud computing processes, e-commerce, e-government, e-operations, e-everything. This term was coined since 1970s [4, 12]. From the early 1980s, the use of IT in the construction industry and broader engineering sector began to increase and find application in many different areas [23].

The exchange of many different types of information was also required at that time. As a result, a series of generic product data models were developed. Data Sharing was also defined as a fundamental to computer-supported cooperative work. People share information through explicit communication channels and through their coordinated use of shared database [24]. It was also been described as a fundamental enabler of coordination among supply chain partners. Therefore, data sharing can be defined as the process of interchanging, analyzing, retrieving and integrating data among multiple data sources in a controlled access manner as shown in figure 2.1. The data source is considered a destination in case of obtaining data from other data sources [6].

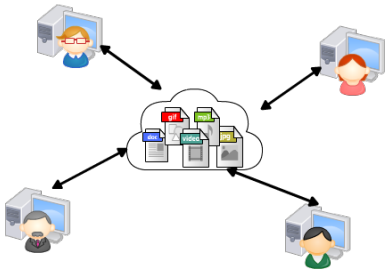


FIGURE 2.1: Sharing Data Between Users [6].

2.1.2 The Need for Data Sharing

Data sharing is an important feature of modern organizations due to the increase in the use of communication networks, the changes in architectures of enterprise information systems, as well as the increase of availability of data in computerized form. And perhaps the biggest impact on data sharing can be attributed to the widespread use of the Internet and Internet-related technologies for e-government and e-commerce [6, 7]. They clarify; e-government involves sharing data for transactions with citizens, other agencies and outside vendors and businesses.

Then in addition, in e-commerce, data can be shared for transactions, operations, and analysis [6]. Conducting business transactions is the basic reason for sharing data in e-commerce. It is mainly used in Electronic Data Interchange (EDI), business to-business marketplaces, as well as consumer purchases over the web. The focus of data sharing for operational purposes leads to the optimization of business processes over the entire chain and to benefit all participants in the chain [6, 11]. Information that shared among supply chain partners may include inventory sales, demand forecasts, order status, and production schedules. Analysis, business intelligence, and decision-support represents

the third purpose for data sharing in e-commerce, information available for analysis is increased through the sharing of data, for example banks share data with affiliates and telemarketers, another example about retailers who allow suppliers to access their inventory data for analysis purposes [6]. It was indicated that it is highly desirable to share data among the members of the medical community; because data is a very valuable, hard to produce, and in some cases irreproducible resource. Data sharing reduces the cost of reproducing redundant data collections as much as minimizing the efforts paid in performing this [25].

Although data sharing facilitates the way that data can be exchanged, security concerns arises a challenge for the conducting of data sharing, confidentiality and privacy must be taken into consideration, this means, a controlled access is required to authorize authenticated users or roles to access data. Each data source represents a database, each database may use an application -for example- to access another database, this application is assigned specific permissions to access specific view of a specific database, permissions that identifies what kind of access must be granted to this application, (e.g. to read, or write, or even to have full access), for this purpose, a database of databases is needed to allow the sharing of data among the different databases as [25] indicates. This increases the need for the data sharing management and data integration. To conduct the data sharing management, a specific model of the data sharing must be followed [2, 7, 12].

2.1.3 Preserving Privacy Models in Data Sharing

Different models have been introduced to apply data sharing each may be the same or different structure of other, some models introduced to extend previous models, others developed to overcome limitations and challenges of previous models. This section provides a literature review for different data sharing models.

2.1.3.1 Semantic Privacy Preserving Model

A semantic privacy preserving model provides authorized view based query answering [8]. For that reason model consider a large number of servers. Therefore a unified global

data sharing and protection service can be achieved at the Virtual Platform (VP). Privacy protection policies represent a long term promise made by an enterprise to its users and is determined by business practice and legal concerns, which is expressed as combination ontology and rule [8].

This model is proposed with three layers, where the bottom layer provides data sources from the relational databases. The middle layer provides a semantics enabled local schema for each independent service domain. The top layer is served at the VP, which provides a unified global view of privacy preserving data sharing and integration services [8]. The ontology mapping and merging algorithm with a source description that creates a global ontology schema (mediated), which is a reconciled view of the information that provides query services to end users, at the VP by merging different local ontology schemas for data sharing. A query is defined as an Semantic Query-enhanced Web Rule Language (SQWRL) data log rule in the Semantic Web Rule Language (SWRL) based policy to access to a global ontology, and each SQWRL data service query for a global ontology at the VP is mapped to multiple queries as SQWRL data log rules for each local schema [8].

The challenge of designing a semantic privacy protection model is to ensure soundness and a completeness of data sharing and protection in multiple servers [8]:

- For the soundness criterion, this model does not allow unintended data being released to the data users through the global policy schema at the VP.
- As for the completeness criterion, the model does not miss any eligible shared data when a user asks for a data request service at the VP. Therefore, shareable data obtained the VP should equal data obtained directly from each server.

Figure 2.2 shows the three layers of the model, where the bottom layer provides data sources from the relational databases, the middle layer provides a semantics enabled local schema for each independent service domain. The top layer is served at the VP, which provides a unified global view of privacy preserving data sharing and integration services. The top layer at the VP has a global policy schema, including a global schema aligned and merged from several local schemas and a set of rule integration at the middle layer. The VP provides conceptual data access and protection services that give users a unified

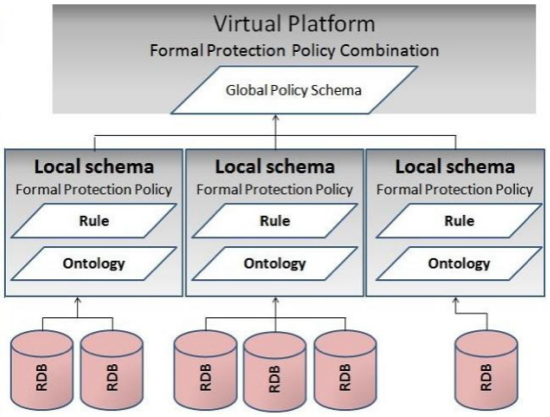


FIGURE 2.2: A Semantic Privacy Preserving Model [8].

conceptual global view" with access control power for each data request. Ontology based data sources are external, independent, and heterogeneous, and each local ontology was combined with logic program based rules for each server in the middle layer. Mapping language, which semantically links a global schema and integrated rule, set in the top layer to each server's ontology local schema and privacy protection rules in the middle layer [8].

2.1.3.2 Capability based Access Control Model

Capability based Access Control, in this model each capability consists of a Name, which identifies a single object in the Internet, and group of access rights for that object [9]. In this model, the system sits between applications and the underlying file system [9]. It presents applications a view based interface to the file system. It executes queries over the local file system and communicates with other peers to evaluate distributed queries. The model is depicted through the following steps [9]:

1. The system registers each new view and capability in a local catalog, this capability has three parts shown in figure 2.3:
    - A 128-bit global view Identification ID: this ID created by concatenating a hash of the local nodes Media Access Control (MAC) address with a locally unique for all time view ID, this view ID uniquely identifies an individual view in the Internet.

- A 128-bit random password: associated with each capability a 128-bit random password that ensures the capabilitys authenticity.
- A 32-bit IP hint field: that contains the IP address of the node that can locate the object addressed by the capability in the Peer to Peer (P2P) Network, in general, they expect that objects will not move in their network, and the IP hint will be the address of the node that created the capability and still holds its definition. If the hint fails, then it must fall back on a conventional distributed hash table scheme for location.

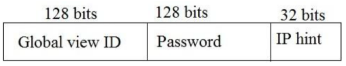


FIGURE 2.3: Capability for a View [9].

2. The per node catalog table generated by the system holds view and capability information. It contains two tables ViewTable and CapTable as shown in figure 2.4. The ViewTable entry contains the global view ID, the view definition, and other attributes (such as the human readable view name). The CapTable entry stores the global view ID of the named view, the password, and the access rights.

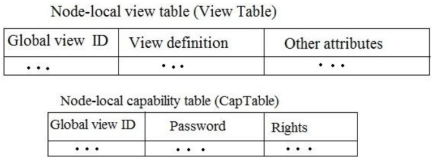


FIGURE 2.4: Capability and View Catalog Tables [9].

3. Users grant each other access to their data simply by exchanging capabilities to their views.

4. When the system receives a capability, it uses the IP hint to determine whether the capability is for a local view. If the capability is local, the system checks whether the (global view ID, password) pair in the capability matches a (global view ID, password) pair in CapTable. If so, the capability is valid, and the system then examines the access rights in CapTable to see if the requested operation is permitted. If the capability is not found in CapTable or the operation is not permitted, the request fails. If the capability is for a remote view, the system forwards the request to the appropriate node in the peer to peer network.

2.1.3.3 BitTorrent

BitTorrent is data sharing model created in 2003, it uses P2P network architecture which allows users to join a "swarm" of hosts to download and upload from each other simultaneously and shares files efficiently using file swarming [10, 26]. All peers in a swarm are interested in downloading the same files. The user who want to download notifies the tracker about the file needed to download, the tracker replay with a list of peers which downloads the same file, then the users in swarms connect to each other using this information as shown in figure 2.5.

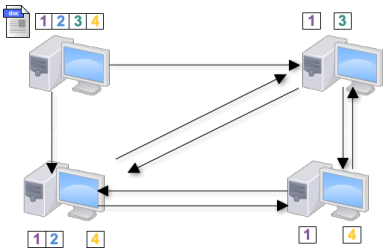


FIGURE 2.5: BitTorrent Overview [10].

In order to initially connect to a swarm, peers download a metadata file, called a torrent which contains file info: name and size of the file to be download, hash info of the data blocks (typically 256 KB), the hash info used to check data integrity. Also the torrent file contains the address for a tracker. The tracker is a server that keeps track of the peers in a swarm, it does not have a copy of the file but it is managing the file share process [10]. Each user sends to all other users in the swarm what fragments of the file it has, the peers exchange data fragments with a set of directly connected peers. In BitTorrent, peers who finish downloading the file being shared and make it available to others are called seeders and peer who are still downloading the shared file are called leechers [26].

The following steps clarify how the message exchange [26]:

- A leecher sends a handshake message which contains a torrent file to another peer.
- Peers exchange bit-fields. A bit-field is a bit-string data structure that encodes which fragments of the file that the peer has.
- If the requesting peer wants to have any of the fragments, the peer sends an interested message.

- If the responding peer is able to serves the request the responding peer sends an unchoke message.
- The requesting peer sends a request message containing the piece number, offset, and length of the data.
- The responding peer sends the data requested as shown in figure 2.6.

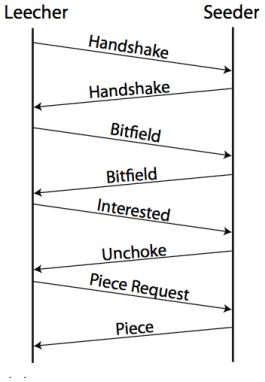


FIGURE 2.6: Initiate a Block Transfer at BitTorrent [26].

2.1.3.4 Privacy Preserving P2P Data Sharing with OneSwarm

OneSwarm data sharing model is a P2P design for data sharing that overcomes the lack of privacy in P2P data sharing models such as BitTorrent and to overcome poor performance [2, 13]. OneSwarm made a trade off between privacy and performance, also it provides secure connection than BitTorrent [2, 13]. OneSwarm builds trusted links through social network peers instead of direct links. OneSwarm users are free to control the performance and privacy by using one of the three cases that OneSwarm provides [2].

There are three cases for OneSwarm described by [2] and shown in figure 2.7, the first one is public distributed data, in this case the data is not private. The second is sharing data with permissions. The last one, sharing data without attribution. In public distribution anyone in the network can reach the shared data, this case is similar to BitTorrent model. With permission case, only the authorized users can download files, in this case all users know each others. While without attribution case is depend on hiding identity of source and destination, data is relayed through unknown number of nodes, and it is for sensitive data.

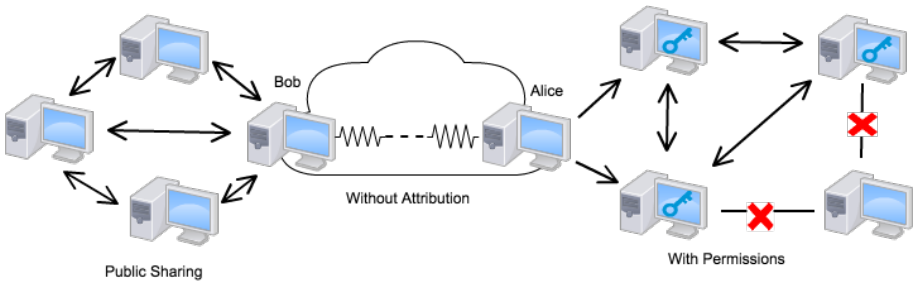


FIGURE 2.7: Cases for Data Sharing by OneSwarm [2].

The topology for OneSwarm the users define the links by exchanging public keys, this identifies each user and creates direct encrypted P2P connections, also OneSwarm uses social graph and community server for key distribution, Distributed Hash Table (DHT) serves as name resolution service, each client maintains encrypted entities advertising their IP address and port to authorized users, the topology is used for each transfer. In each transfer each OneSwarm client restricts direct communication to a small number of persistent contacts and locates different data sources using object lookup through overlay, this topology is used to enhance privacy, while to enhance performance in OneSwarm protocol, multiple paths to each data source are used.

Linking peers with trust relationships is explained by [2] it uses 1024 bit RSA cryptosystem public/private key pair which is generated in installation phase, public key serves as its identity among friends, manual key sharing between two users; the automatic key sharing discovers and exchange keys over local area network or by email invitation to friends. Managing untrusted peers by private community server and public community server, the private is to maintain a list of registered users and to provide authorized subscribes with a current set of public keys, the public is to allow new users to easily obtain a set of untrusted peers. Identity in OneSwarm protocol are managed by the DHT which consist of hashed IP and port and entries for a client encrypted with the public key.

Naming and locating data in OneSwarm used SSL for connection as [2] say, file list messages is exchanged on first connection then compressed XML attributes which contain name, size and other meta data for particular peer. Shared files are named using 160 bit Secure Hash Algorithm (SHA), for public data user obtains hashes from email, websites and keywords search, while for private data user must obtain both hash and key used for decryption of data. The risk in OneSwarm model is that the attacker can join with

limited number of nodes, also can check the traffic flow to/from, also may modify or injected data. Limiting hacker to snoop in from by not assigning peer dynamically, also defining trusted and untrusted links to keep the information private, end to end path between users change rapidly helps to prevent hacking using historical data [2].

2.1.4 Security and Privacy in Data Sharing

Security in computer is defined as the protection of automated information system to preserve the integrity, availability, and confidentiality of information system resources. Security considerations apply not only to data held in database, as [27] indicate, database security encompasses hardware, software, people, and telecommunications [27]. They add, loss or damage of data results as a reason of intentional or unintentional acts, intentional acts such theft and fraud result in either loss of confidentiality or loss of privacy, they also add, unintentional acts such viewing of unauthorized data cause loss of privacy and loss of confidentiality. Unrestricted data sharing will reduce the privacy or confidentiality of users [3, 28].

One of the methods to struggle threats is make sure that sensitive data must always be encrypted when stored or transmitted, another is to provide backup facilities to assist with the recovery of the database, views mechanism provides a powerful and flexible security mechanism by hiding parts of the database from users of data share [27]. Sharing data must be under the data sharing agreement which provides secure data sharing [3, 28] .

The challenge is to apply the appropriate security policies that can facilitate data sharing [3]. During any processes, it is important to apply confidentiality and privacy.

The protection of information from unauthorized users becomes an important issue on the Internet due to the increasing of data sharing, but it still a challenging issue [2, 29]. Although there are clear benefits of sharing data in health sector, there are also risks. There are benefits for sharing data from medical areas, but the data must shared in private way. Designers of Electronic Health Records (EHR) should consider a way to preserve privacy for sensitive data. For example, Cancer patients whose donate their personal data for research purposes will help medical researchers so we need for preserving privacy to share these data with other researchers privately [30]. To preserve

data privacy a basic solution is to encrypt data files and then upload the encrypted data, then distribute the decryption keys only to authorized users [29, 31].

2.1.5 Comparison Between Data Sharing Models

Based on the comparison between the four models as shown in table 2.1, the Capability based Access Control model [9] has disadvantages, mainly: there is no fixed method for translation, difficulties in integrity control, cannot prevent the user from keeping and distributing the shared data, and decentralized control [9]. These disadvantages make the implementation of the model hard. Concerning semantic privacy preserving model [8] overcomes the previous disadvantages, and provides data integration, secure sharing through authorized view, in addition, each organization enable data sharing without affecting its clients [8]. While BitTorrent [10] shares files easily, also breaking the file into pieces allows it to be distributed as efficiently as possible [26]. But BitTorrent has a privacy challenge. On the other hand OneSwarm data sharing model [13] provides flexibility for the user to manage the level of privacy for file sharing, and reduces cost of privacy, also OneSwarm provides high availability and it's performance is better than BitTorrent, but it has delay in response [28, 29, 31].

2.2 Networking Technologies

Data or messages are transmitted or shared between users by networks. The message to be transmitted is the basic unit of network communications, the message can be one or more cells, frames or packets [32–34]. Computer networks can be classified to Local Area Networks (LANs) or Wide Area Network (WANs) [32, 34]. Networking technologies includes everything between LANs and WANs as shown in figure 2.8 [32].

The Transmission Control Protocol/Internet Protocol (TCP/IP) is the basic communication protocol of the Internet. The TCP/IP is four layer model created with reference to the seven layer Open System Interconnection model (OSI) [32, 34].

The delivery of an IP packet to its final destination is accomplished by means of either direct or indirect delivery [32]. Direct delivery is when source and destination on the same network. Direct delivery happens when the IP address of the destination packet

TABLE 2.1: Comparison Between Data Sharing Models.

Model Name	Strength Points	Weakness Points
Semantic Privacy Preserving Model	• Each organization enables data sharing without affecting its clients. • Data integration. • Provide secure sharing through authorized views.	• Inconsistency problems.
Capability based Access Control Model	• Provide flexible protection mechanism for controlling access to shared views. • Reuse of queries. • Independent of the user.	• Cannot prevent users from keeping and distributing the shared data.
BitTorrent	• Share files easily. • Efficient.	• Not secure.
OneSwarm	• Efficient, robust. • Users flexible. • Preserve privacy.	• There are delayed response to queries from untrusted peers.

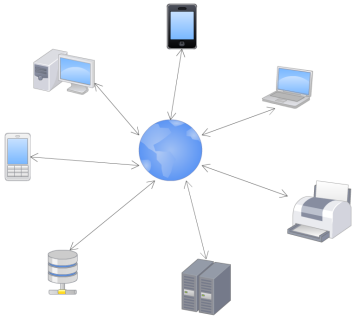


FIGURE 2.8: Networking Technologies.

same as the addresses of the connected networks. Address Resolution Protocol (ARP) is used to find the destination physical address [32]. While in an indirect delivery, the packet goes from router to router through until it reaches router that is connected to the same physical network. The sender uses the destination IP address and a routing

table to find the IP address of the next router, the sender then uses the ARP to find the physical address of the next router [32].

2.2.1 Network Bandwidth and Transfer Parameters

Efficiency for the network can be measured by network bandwidth (throughput) [34]. Network bandwidth is the number of bits that can be transmitted over network in a certain period of time. Bandwidth described in the following equation [34]:

$$Bandwidth = \frac{file\ size(MB)}{Time(sec)} \tag{2.1}$$

Also network performance can be measured by latency (delay). Latency is how long the message takes time to travel from one user of a network to the other [34]. Latency is expressed in the following equation [34]:

$$Latency\ Time = Propagation\ Time + Transmit\ Time + QueueTime \tag{2.2}$$

where queue: the queuing delays time inside the network,
and the propagation time is:

$$Propagation\ Time = \frac{Distance}{Speed\ Of\ Light} \tag{2.3}$$

where Distance: the length of the wire in meter.

Transmit time is :

$$Transmit\ Time = \frac{Size}{Bandwidth} \tag{2.4}$$

where Size: the size of the packet in bits.

2.3 Cryptography and Network Security

Cryptography is a science that contains different ways for converting the original data to coded data in order to protect data [33, 34]. It is important to use cryptography to secure communications between users or to transmit and store sensitive data [33]. The National Institute of Standards and Technology (NIST) defines computer security as: the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources [33, 35]. Many fields and organizations such as e-government need information security to prevent data theft, boost productivity, avoids data loss, prevent unauthorized access and prevents identity theft [12, 33]. Figure 2.9 summarize different cryptography techniques at sender side and receiver side to get secure communication systems, including: key distribution process using RSA, encryption process using DES, hashing process using MD5 and digital signature process by RSA [32, 33].

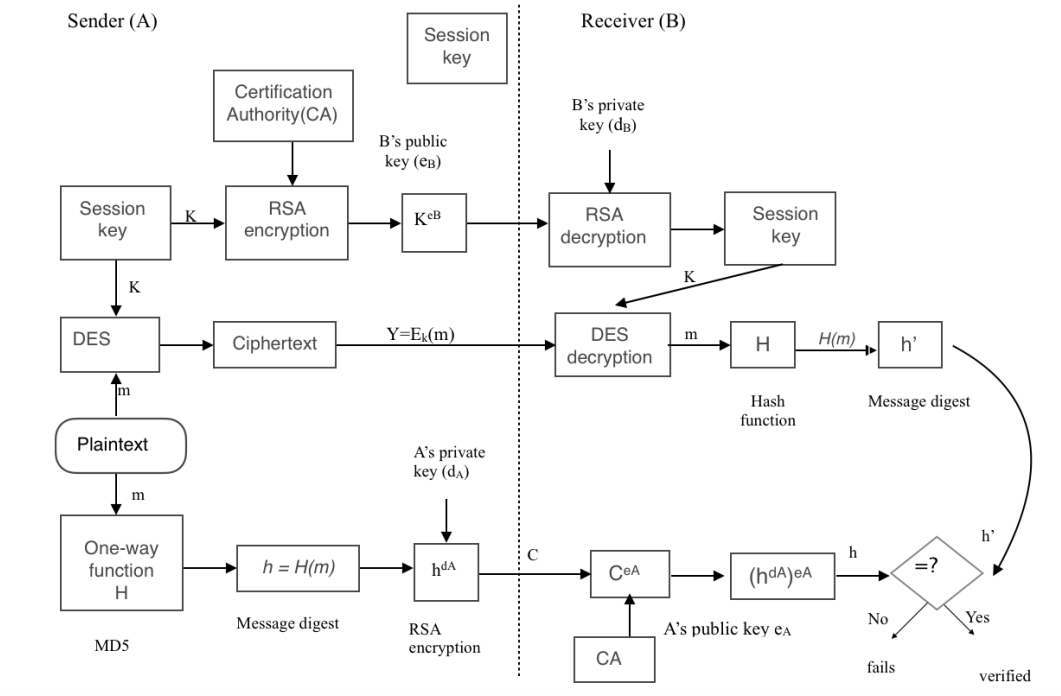


FIGURE 2.9: Secure Communication Systems [32].

2.3.1 Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) was developed by Netscape in 1994 to provide a secure transmission between applications [33]. SSL provides confidentiality by using symmetric key encryption such as AES and DES also it provides message integrity, figure 2.10 shows the general processes for SSL.

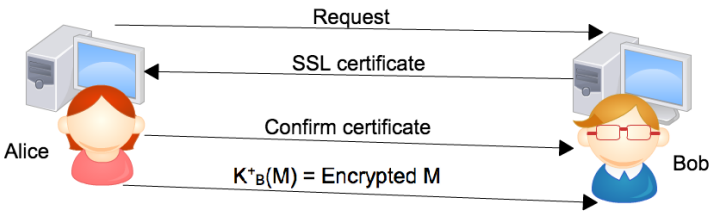


FIGURE 2.10: SSL Overview [33].

SSL has two layer of protocols as shown in figure 2.11:

1. The lower layer protocol has four main operations:
    - Fragmentation: divide the message to be transmitted into blocks.
    - Compression: apply a lossless compression to the blocks, this operation is optional.
    - Compute the Message Authentication Code (MAC) over the compressed blocks.
    - Encryption: use symmetric key encryption.
    - Append SSL record header: consists of the following fields Content Type, Major Version, Minor Version, and Compressed Length. then the message is transmitted.
2. The higher layer protocol, which consists of three layer of protocols:
    - The change cipher space protocol: consist of a single message with single byte, it used to update the cipher that used on this connection.
    - The alert protocol: consist of two bytes, the first indicates to close the connection or to continue based on the message, the second describes indicates a specific alert.

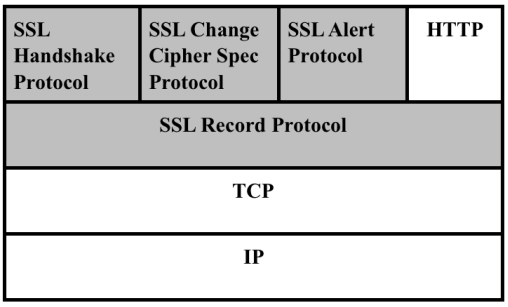


FIGURE 2.11: SSL Protocols [33].

- The Handshake protocol: it allows the server and client to authenticate each other and decide the encryption, MAC algorithm and cryptographic to be used. it consists of multiple messages between server and client, as shown in figure 2.12.

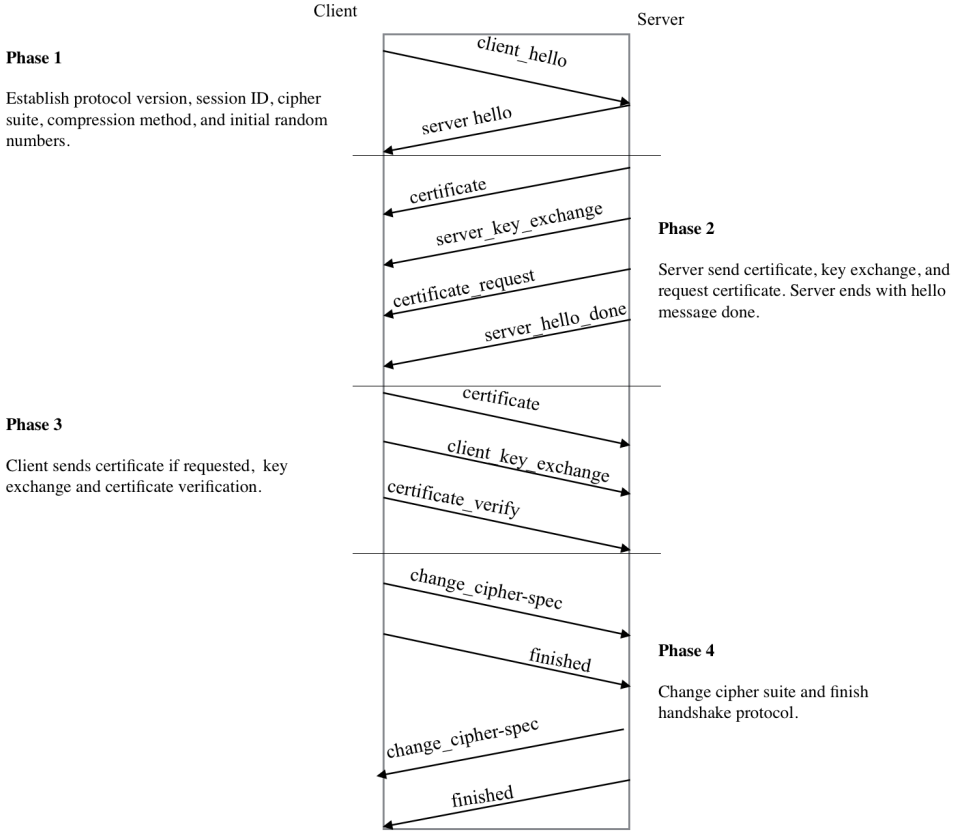


FIGURE 2.12: Handshake Protocol [33].

2.3.2 Encryption Techniques

Encryption is the process of converting the original data (Plaintext) to a coded data (Ciphertext) while the decryption is the process of converting the encrypted data (Ciphertext) to its origin (Plaintext) [33].

There are two types of encryption techniques used to get secure communications: symmetric key encryption and asymmetric key encryption. Symmetric key encryption the same key is used for encryption and decryption process such as Advanced Encryption Standard (AES) [36] and Data Encryption Standard (DES) while asymmetric key encryption use two different keys such as Rivest-Shamir-Adleman (RSA) cryptosystem [16], and Augmented-RSA (A-RSA) Cryptosystem [17]. Understanding encryption technique will help to develop more technique to protect information. This section provides a literature review for these techniques.

2.3.2.1 RSA Algorithm

In 1977, Ron Rivest, Adi Shamir and Leonard Adleman developed RSA as an encryption technique. RSA is the most widely used asymmetric key encryption among transactions over insecure channels, which used to protect data from being transferred between devices as shown in figure 2.13. It used in digital signature, key exchange and encrypting data [17].

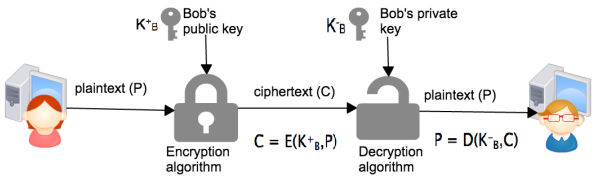


FIGURE 2.13: Asymmetric Key Encryption (RSA) [33].

RSA Algorithm relies on the difficulty of factoring a large integers [16]. RSA is built from two large prime numbers these prime numbers are manipulated to give a public key and a private key. Anyone can encrypt a message using the public key, the private key is kept secret and used to decrypt the message [16]. To encrypt a message with RSA, suppose that Alice want to send message (P) to Bob, Alice must use the Bob public key (K_B^+) to encrypt the message, and Bob uses his private key (K_B^-) to decrypt the

ciphertext (C), The private key and public key are generated by the receiver. Algorithm 1 summarize RSA Algorithm.

Algorithm 1 RSA Algorithm

1: Choose two distinct large prime numbers p and q.
2: Compute N = pq.
3: Compute Euler's totient function $\phi(N)$ = (p-1)(q-1).
4: Choose a random integer e, such that $1 < e < \phi(N)$ and GCD(e, $\phi(N)$) = 1.
5: Compute d, such that ed = 1 mod $\phi(N)$.
6: Publish the public key (N,e), while keep the private key (N,d), p, q, and $\phi(N)$ secret.
7: Compute the ciphertext (C), such that C = M^e mod N, To encrypt the message M.
8: To Decrypt the ciphertext (C), compute M = C^d mod N.

Although RSA is the most popular encryption method, it takes long time for encrypting and decrypting large data, also it is not secure because encrypting the same message more than once gives the same ciphertext [17, 35].

2.3.2.2 A-RSA Cryptosystem

New encryption method called Augmented-RSA (A-RSA) [17]. A-RSA depends on RSA algorithm and Rabin algorithm, in this cryptosystem new component is added to RSA algorithm, this component is encrypted by Rabin algorithm. Also Huffman coding algorithm is used to reduce data redundancy before adding the randomized component which enhances the execution time. The compression process using Huffman coding facilitates storing and transmission of large data. The main idea of A-RSA is that it uses Huffman coding to compress the message, this coding process will produce two files the Binary file (B) and Header file (H), the binary file depends on the header file to retrieving the original data. Instead of encrypts the entire message, A-RSA encrypt the header file using RSA and blinding the binary file by the randomization component Y [17] as shown in figure 2.14 .

If Alice wants to send a message M to Bob, she chooses a randomized parameter (Y), then she compresses the message using the Huffman coding to generate the header file and binary file. She computes the ciphertext C for the header file using RSA (Bob public key). To make the encryption semantically secure she computes the Mixture ($C\prime$), such as $C\prime = C \oplus Y$, also she blinds the binary file with Y to generate $B\prime$ as $B\prime = B \oplus Y$. Alice encrypts Y by Rabin Cryptosystem. Finally, Alice sends packet contains ciphertext of Y , $C\prime$, and $B\prime$ to Bob.

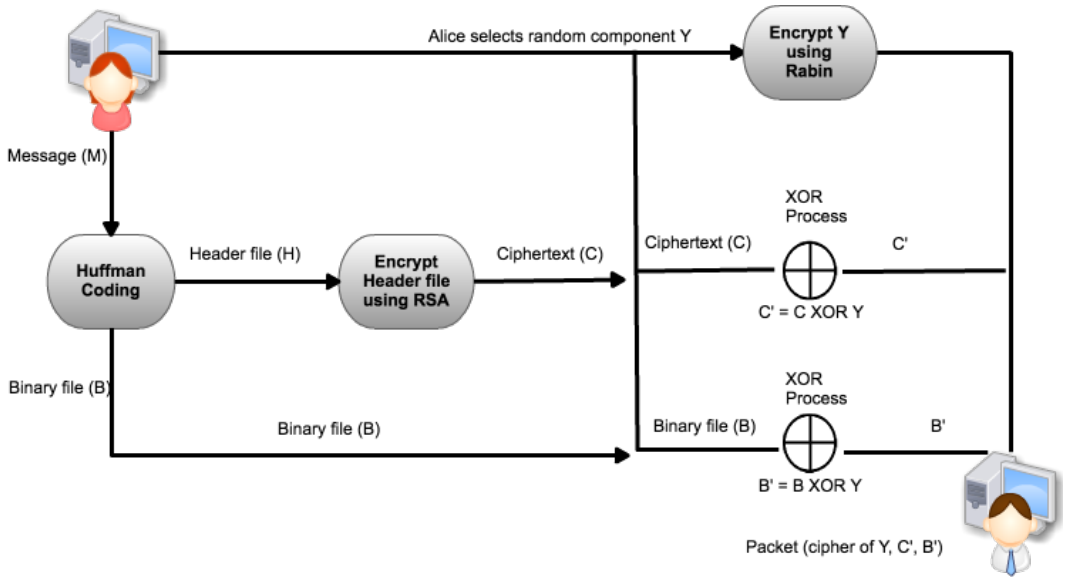


FIGURE 2.14: A-RSA Cryptosystem [17].

While the decryption process as follows, Bob uses his Rabin private key to decrypt the randomized component Y, then removes Y from the Mixture ($C\prime$) to get the encrypted header C as C $=C\prime \oplus Y$. Bob gets the header file H by using his RSA private key to decrypt C, then Bob recovers B from $B\prime$ as B$=B\prime \oplus Y$, then he uses H to recover the original message M from the binary file [17]. Algorithm 2 summarize A-RSA Algorithm.

Algorithm 2 A-RSA Algorithm.

Encryption Process:

1: Generate a random component Y for each message
2: Compress message using Huffman code. The outputs are: Header file (H) and Binary file (B).
3: Encrypt H by RSA and the result is C $= H^e$ mod N, where $0 < H < N1$.
4: Blind C by Y to generate the mixture $C\prime$=C $\oplus$Y.
5: Blind binary file B by Y to generate $B\prime$ =B$\oplus$Y.
6: Encrypt Y by Rabin and the $Y\prime = Y^2$ mod N.

Decryption Process:

1: Decrypt Y using Rabin cryptosystem
2: Compute C from the mixture C $= C\prime \oplus Y$.
3: Compute B$=B\prime \oplus Y$.
4: Decrypt C using RSA decryption to generate H $= C^d$ mod N.
5: By Huffman coding reconstruct the message using B and H.

2.3.2.3 DES Algorithm

The Data Encryption Standard (DES) was developed by International Business Machines corporation (IBM) in 1974 then it was adopted by the National Institute of Standards and Technology (NIST) in 1977. DES is a symmetric key encryption as shown in figure 2.15, data are encrypted in 64-bit per block using short key 56-bit [32].

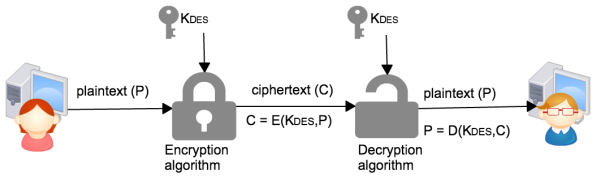


FIGURE 2.15: Symmetric Key Encryption (DES) [33].

Figure 2.16 clarifies the DES algorithm, the left-hand side of the figure shows main operations that done on the message [33].

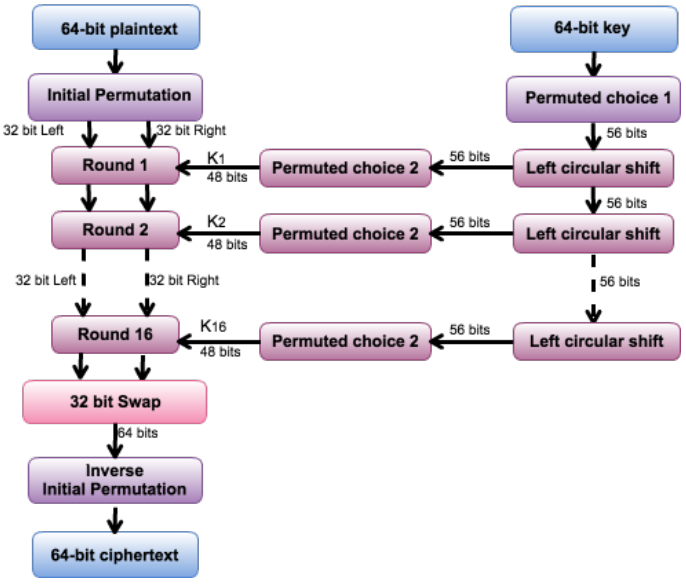


FIGURE 2.16: DES Algorithm [33].

The 56-bit key was an easy target for the attacker to try every possible key on a block of ciphertext until he obtained the plaintext (Brute-Force Attack) [33], so DES has been replaced with a new standard called The Advanced Encryption Standard [36].

2.3.2.4 AES Algorithm

The Advanced Encryption Standard (AES) is a symmetric key encryption published by (NIST) in 2001 as shown in figure 2.17, it was a replacement for Data Encryption Standard (DES), because DES is weak and vulnerable to brute force attack [36].

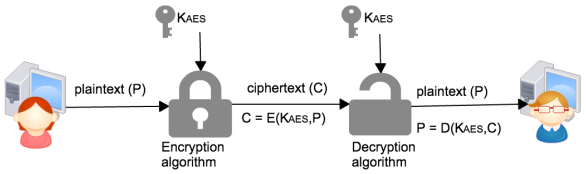
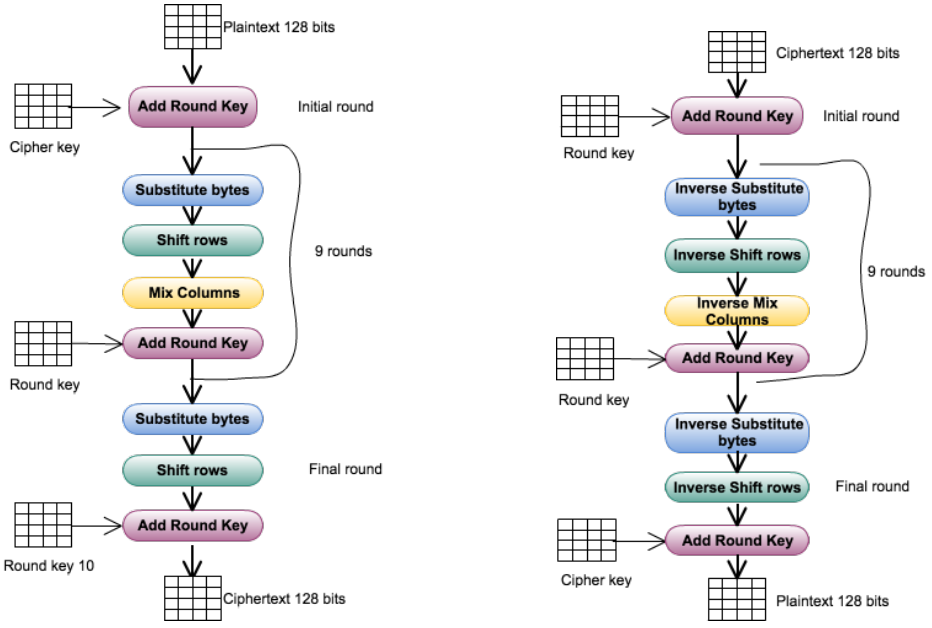


FIGURE 2.17: Symmetric Key Encryption (AES) [33].

AES operates on a block with a fixed size 128 bits, these bits are arranged at 4*4 matrix called State, and it allows for three different key size 128, 192, and 256 bits, number of rounds in AES depends on the key size, AES-128 has 10 rounds, AES-192 has 12 rounds and AES-256 has 14 rounds, [19]. Figure 2.18 (a) shows the encryption process for AES while figure 2.18 (b) shows the decryption process.



(a) Encryption Process. (b) Decryption Process.

FIGURE 2.18: AES Algorithm [35].

2.3.3 Comparison Between Different Encryption Techniques

Table 2.2 presents a comparison between RSA, A-RSA, DES and AES based on encryption type, key size, block size, speed, security and Vulnerabilities. Based on the comparison the AES and DES algorithms are faster than the RSA and A-RSA.[14, 15, 18].

TABLE 2.2: Comparison Between RSA, A-RSA, DES and AES.

Encryptic Techniques	**Encryption Type**	**Key Size**	**Block Size**	**Speed**	**Security**
RSA	Asymmetric	Depends on number of bits in p and q	Variable	Slowest	Secure
A-RSA	Asymmetric	Depends on number of bits in p and q	Variable	Slow	Secure
DES	Symmetric	56 bits	64 bits	Slow	Not secure enough
AES	Symmetric	128, 192, or 256 bits	128 bits	Fast	Secure

2.4 Data Compression Techniques

Data compression is a way to save or transmit data in a smaller size than its original also data compression can be defined that it is the art of representing data in a compact form rather than its original [20]. Data compression is used to reduce storage needed to save data, increase channel bandwidth utilization and reduce transmission time. Also data compression is used in security areas because compressed data is different than the original data [37]. Data compression has two categories, lossy data compression and

lossless data compression. Data in lossy compression are not recovered as its original, some of data is lost. it is used for images and sound where a loss in bits or resolution is acceptable. While in lossless data compression, data is recovered exactly as the original data without any loss. It used for text files. Compression efficiency can be calculated as follows [20]:

$$compression\ efficiency = \frac{size\ of\ compressed\ data}{size\ of\ original\ data} \tag{2.5}$$

2.4.1 Lossless Data Compression Algorithms

This section clarified three lossless data compression algorithms, Huffman coding algorithm, Run-Length Encoding (RLE) algorithm and Abraham Lempel-Jacob Ziv (Lempel-ziv) algorithm. According to studies for the different data compression algorithms, Huffman coding is more efficient than RLE and Lempel-Ziv [21], [38]. Using RLE to compress a file with no repeated characters will increase file size also using Lempel-Ziv algorithm to compress small files may increase the compressed file size due to the size of the dictionary [21].

2.4.1.1 Huffman Coding

Huffman coding is the most used of lossless data compression algorithm , that uses small number of bits to encode common characters. Huffman coding developed by David Huffman in 1952 to reduce size of data, reduce storage needed and reduce transmission time, latency and bandwidth [22].

Huffman coding algorithm reads file to be compressed, stores all characters on a list according to their frequency counts, then it uses the frequency of the characters to build bottom-up tree its leaves are the weights. Huffman tree used to represent frequent symbols with fewer bits and infrequent symbols with more bits. The result of applying Huffman coding on data is binary file and header file as shown in figure 2.19.

The header file contains a list of all characters and their frequency counts. Binary file and header file depend on each others, if the header file does not exist we can not recover the origin data.

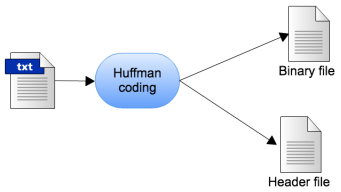


FIGURE 2.19: Summarize Huffman Coding.

2.4.1.2 Run-Length Enoding

The RLE is the simplest lossless data compression algorithm, it replace the repeated characters in the message with single character and its count [20]. RLE is good for data that contains many repeated characters. For example, this message "SSSSAAMMRRRRR" is represented in Run-Length coding as follows: "4S2A2M5R". Run-Length algorithm represent the original message in 8 character while the original message is 13 character [22]. In RLE, the compressed file size may increase if the original file doesn't have repeated characters. It is efficient in White/Black images because it have many repeated pixel with same color also it is efficient in binary files (0s and 1s).

2.4.1.3 Lempel-Ziv Algorithm

Lempel-Ziv algorithm [20] was developed by Jacob Ziv and Abraham Lempel, it creates a dictionary of strings for the file to be compressed. Lempel-Ziv looks for the frequently occurring combination of characters and bulid a dictionary for these combinations then replace them by a single character. The algorithm is efficient for the large size files, while in small size files the length of the dictionary may exceed the size of the original file [22]. The following steps describe how Lempel-Ziv works:

1. Initialize the dictionary with the available characters.

   Assume the sequence "AABABBBABAABABBB", the following table 2.3 is the initialization dictionary for the sequence:

TABLE 2.3: Initialization Dictionary of Lempel-Ziv Algorithm

Strings	**Location**
A	1
B	2

2. Then separate the string into strings that have not appeared before, as follows. "A/AB/ABB/B/ABA/ABAB/BB"

3. Finally, encode the strings by their location in the dictionary, as shown in table 2.4. The result after applying Lempel-Ziv algorithm, the original message is 16 bits while the compressed message is 7 bits:
   Sequence : AABABBBABAABABBB
   Code: 1342567

TABLE 2.4: The Dictionary of Lempel-Ziv Algorithm

The Strings	**Location**
A	1
B	2
AB	3
ABB	4
ABA	5
ABAB	6
BB	7

Chapter 3

Proposed Data Sharing Model for Preserving Privacy

This chapter introduces a new privacy preserving data sharing model, the proposed data sharing model focuses on providing privacy for files that will be shared by users. Section 3.1 presents the generic standard data sharing model. Section 3.2 presents the proposed data sharing model then the main components for the proposed model are described. Finally, Section 3.3 discusses how the model preserves privacy, saves storage and decreases the transfer time.

3.1 Generic Standard Data Sharing Model

Sharing sensitive data between members in a team in different areas such as healthcare, government, scientific research and e-commerce needs to share them in a secure way before transferring any sensitive data. OneSwarm is an excellent example as a generic data sharing model that make a trade off between performance and privacy. It controls privacy through friend-to-friend sharing. OneSwarm transfers data fragments through multiple nodes instead of direct path, so the sender and receiver are unknown to each other as shown in figure 3.1, the figure shows a OneSwarm network which contains of four friends (nodes), the file transfer to multiple paths until it reach the destination [2].

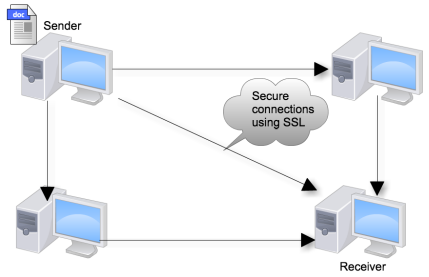


FIGURE 3.1: OneSwarm Structure [2].

3.1.1 Privacy for Generic Model

There are three cases for OneSwarm described by [2], the first one is public distributed data, in this case the data is shared publicly. The second is sharing data with permissions limits access. The third case is to share data without attribution. In public distribution anyone in the network can download file free, same as BitTorrent. With permission case only users with permission can download files. While without attribution case uses privacy preserving keyword search, this case is used for sensitive material.

The third case of OneSwarm provides privacy to shared data by using SSL to get secure connections between users. Also it transfers data through multiple nodes instead of direct path, so the sender and receiver are unknown to each other as shown in figure 3.1. The trusted links between users are built using 1024 bit RSA keys, and the identities in OneSwarm are managed by the DHT which contains the hashed IP and port [2].

Although OneSwarm was developed to provide privacy to shared files and to overcome the privacy and performance problems in BitTorrent, OneSwarm still has problems such as delay in response, this happens because of using multiple nodes, and privacy problems [2, 26].

3.1.2 Computation and Transfer Time Parameters for Generic Model

This section describes the performance for the generic model in terms of computation time and transfer time. Total sharing time for OneSwarm model is expressed in equation 3.1.

$$T_{shar} = T_{comp} + T_{trans} \tag{3.1}$$

Where T_{shar}: the sharing time,
T_{comp}: the computation time,
T_{trans}: the transfer time.

The computation time is:

$$T_{comp} = T_{Enc}(File) + T_{Dec}(File) + T_{Pro} \tag{3.2}$$

Where $T_{Enc}(File)$: the encryption time for the file,
$T_{Dec}(File)$: the decryption time for the file assuming SSL infrastructure,
T_{Pro}: a processing time.

The transfer time is the sum of all times that can be used for transferring file fragments from one node to another.

3.2 Generic Proposed Model Using Compression and Encryption

The generic proposed data sharing model was developed to provide privacy to the shared files without affecting the performance using compression and encryption. As discussed in section 3.1.2 the computation time, transfer time and privacy of the standard generic model can be improved. The proposed model is peer to peer model that shares data in secure way so no one can read data except the authorized users. The proposed model combined compression technique and encryption algorithms to provide privacy for the shared data. It focuses on preserving privacy for shared data by combining Huffman coding algorithm and SSL encryption.

When Alice wants to upload a file to network, the file is compressed using Huffman coding. Huffman coding generates two files, header file and binary file. The binary file

is shared offline while the header is edited to include X% of data of the binary file then it is secured using SSL which can be either DES, AES or A-RSA. When Bob wants to download a file that shared by Alice, Alice sends the encrypted header file to Bob, so he can decrypt the downloaded file (binary file) and read it. Bob decrypts the header file, then the decrypted header file is used to recover the original data using Huffman decoding. Only the authorized users who have the encrypted header file can read the shared binary file as shown in figure 3.2.

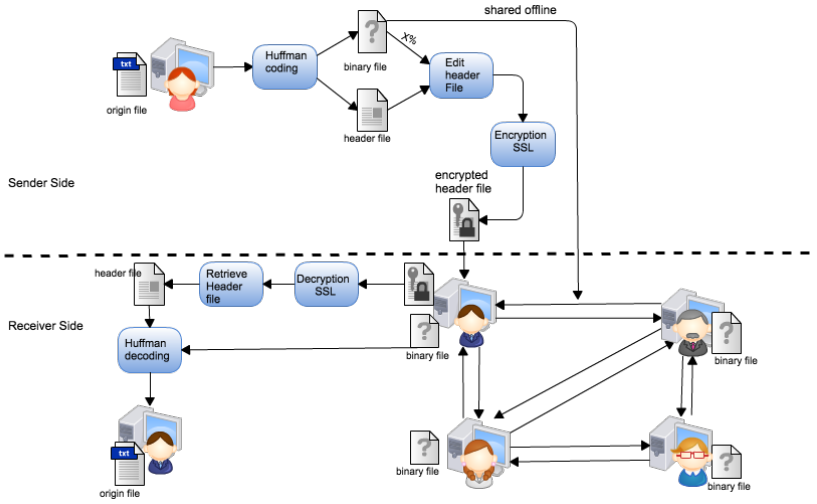


FIGURE 3.2: The Proposed Model Overview.

3.2.1 Proposed Model Components

The proposed model has two main components: The first is the compression, decompression which is used to reduce data size and to reduce transmission time. The second component is to provide privacy using encryption, decryption algorithms. Also X% of data have been added to header file before encrypting it to make the algorithm stronger against attacks such as direct attacks, in these attacks the attacker tries to guess the secret keys that have been used in encryption process by trying all possible combinations to find these keys, so using X% will make it harder for the attacker to guess the header file and it will take longer time.

3.2.1.1 Compression using Huffman Coding

Huffman coding is a lossless data compression algorithm. It used to reduce size of data, transmission time and bandwidth [20]. Huffman coding generates two file binary file

and header file as shown in figure 3.3. Binary file and header file depend on each others, if the header file does not exist we can not recover the original data.

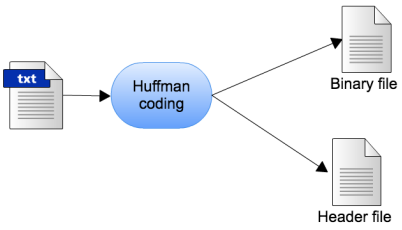


FIGURE 3.3: Summarize Huffman Coding.

The following steps summarize Huffman coding process:

1. Scan the file to be compressed, then store the frequency for each character in a list. Let us take an example of a file that contains 9 different character and their sum counts in the file is 1200, table 3.1 contains the frequency for each character.

TABLE 3.1: The Frequency of each Character in a File.

Character	**Frequency (F)**
H	20
F	40
B	60
G	60
C	100
D	120
A	200
I	200
E	400

2. From the table pick two characters with the lowest frequency.

3. Based on the two characters build a Huffman sub-tree that has the two characters as child nodes and create a parent node with the sum of the childrens frequency, as shown in figure 3.4.

4. Insert the parent node to the table as a new value and delete the children from the table.

   The second step, third and fourth is repeated until the table has no values.

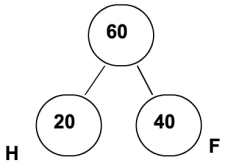


FIGURE 3.4: Huffman Sub-Tree.

5. Assign (0 or 1) to every path, for left path assign 0 and for right path assign 1.

6. Assign a code for each character, code is completed when a leaf node is reached, figure 3.5 presents the complete Huffman tree.

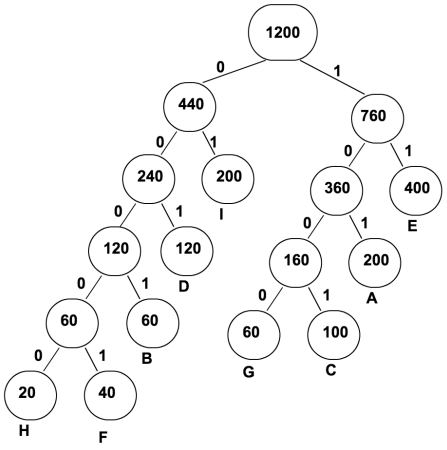


FIGURE 3.5: Complete Huffman Tree.

7. After that, rescan text again and encode file using the Huffman codes, table 3.2 represents the Huffman code for each character.

TABLE 3.2: Huffman Code for each Character.

Character	**Huffman code (C)**
H	00000
F	00001
B	0001
G	1000
C	1001
D	001
A	101
I	01
E	11

After the processes is completed Huffman coding generates two files the header file and the binary file. The header file is an array of 256 elements each element represent the frequency of each character in the original file and the index of each element represents the decimal ASCII for the characters. While the binary file contains the Huffman code for each character.

The header file for the previous example is shown in figure 3.6.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 200 60 100 120 400 40 60 20 200 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 3.6: Header File.

Where "200" represent the frequency of the element that it's decimal in ASCII is "65" which is "A" and "60" represent the frequency of the element that it's decimal in ASCII is "66" which is "B", and so on.

While the binary file for a part from the file "HFAEAIAEEEDDIFBC" is shown in figure 3.7.

00000000011011110101101111111001001010000100011001

Figure 3.7: Binary File.

Table 3.3 represents the total length in bits for the compressed file and the total length in bytes for the original file.
Number of bits in the original file is:
$1200 character * 8 bits/character = 9600 bits.$
while the number of bits in the compressed file in binary file is 3340 bits which computed by:

$$TotalLength = Frequency(F_i) * CodeLength(L(C_i)) \tag{3.3}$$

where F_i: the frequency of a character in a file to be compressed,
and $L(C_i)$: the code length for a block of a ciphertext.

This means, by compressing this file using Huffman coding, 65% of space is saved.

TABLE 3.3: Number of Bits in Original File and Number of Bits in Binary file.

Character	**Frequency (F)**	**Code Length (L(C))**	$F_i * L(C_i)$
H	20	5	100
F	40	5	200
B	60	4	240
G	60	4	240
C	100	4	400
D	120	3	360
A	200	3	600
I	200	2	400
E	400	2	800
Sum	1200 byte		3340 bits

The following equation is used to compute saving storage percentage, in the previous example the saving storage percentage is 65%, this means that the compressed file is 65% smaller than the original file:

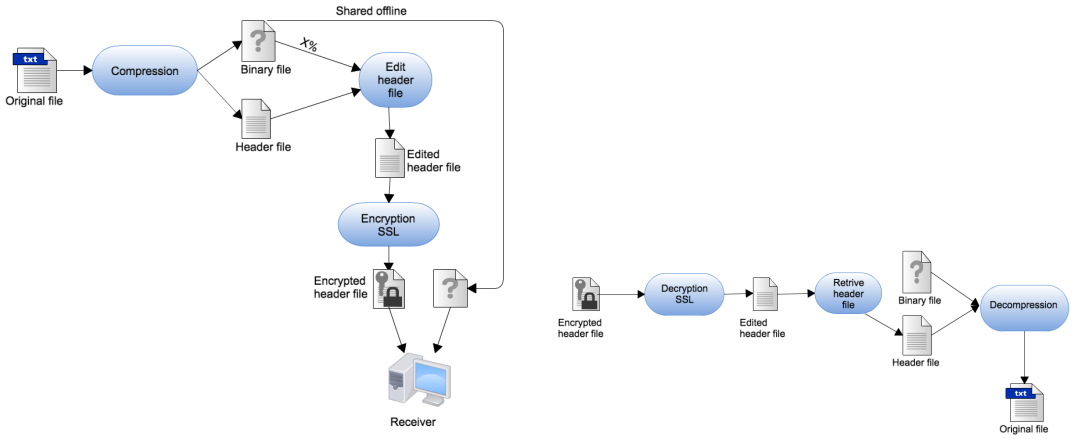
$$Saving\ Percentage = \frac{F_s - CF_s}{F_s} * 100\% \tag{3.4}$$

where F_s: the file size before compression,
and CF_s: the file size after compression.

For decompression process, read the header file that contains each character and its count in the original file, use this information to build the Huffman tree from top to bottom. Then trace the path from root to its leaf node, when finding 0 move to the left on the tree, when finding 1 move to the right on the tree until reaching a leaf node then stop. Repeat the process until all characters in the binary file is decoded.

3.2.1.2 Security for The Proposed Model

The proposed model uses SSL encryption to encrypt and decrypt the header file, it can be AES, DES or A-RSA. To enhance the algorithm against attacks, X% of the data of binary file is added to the header file before encrypt it, also to enhance security Cipher Block Chaining (CBC) was used [39]. In (CBC) every block of the plaintext will be XORed with the encrypted previous block this make the plaintext analysis difficult for the attacker comparing with other operation modes because every block of the ciphertext depends on the previous block. Figure 3.8 (a) shows the compression and encryption process for the proposed model at the sender side and figure 3.8 (b) shows the decryption and decompression process for the proposed model at the receiver side.



(a) Summarize Compression and Encryption Process. (b) Summarize Decryption and Decompression Process.

FIGURE 3.8: Processes for the Proposed Model.

Algorithm 3 summarizes the processes for the proposed model:

Algorithm 3 Processes for the Proposed Model.

Compression and Encryption Processes.

1: Compress original file using Huffman Coding.
2: Generate the binary file (B) and the header file (H).
3: Add X% of the data of (B) to (H).
4: Then, use the encryption algorithm to encrypt the output of the previous process, to get the ciphertext (C).

Decryption and Decompression Processes.

1: Decrypt the ciphertext (C) to get header file.
2: Retrieve the Header file (H).
3: Use header file (H) to decompress binary file (B), the output is the original file.

3.2.1.3 Privacy for The Proposed Model

The proposed data sharing model provides privacy to the shared data, the following points describe how the proposed data sharing model provides privacy:

1. Huffman coding which encode the original data, this increase data security because the file that will be shared is coded and different from the original file [17].
2. Encryption algorithm to encrypt the header file.
3. To improve security Cipher Block Chaining (CBC) is used. CBC makes the plaintext analysis difficult for the attacker comparing with other operation modes due to every block of the plaintext is dependent on the previous block [33].
4. In addition, the proposed model add X% of data of binary file to header file to enhance the algorithm against the attacks such as direct attacks, in these attacks the attacker tries to guess the secret keys that have been used in encryption process by trying all possible combinations to find these keys, so using X% will make it harder for the attacker to guess the header file and it will take longer time.

3.2.2 Computation and Transfer Time Parameters for Proposed Model

This section compares the total sharing time for the generic standard models and models that use compression and encryption for the entire file with the proposed model which uses Huffman coding as compression algorithm and encrypts the header file. The section shows the enhancement for the sharing time for the generic proposed model which shares the binary file offline and uses Huffman coding as compression technique and encryption algorithm to encrypt the header file then it.

Total sharing time for a model that using compression and encryption is expressed in equation 3.5.

$$T_{sharCom} = T_{compCom} + T_{transCom} \tag{3.5}$$

Where $T_{sharCom}$: the sharing time when using compression and encryption,
$T_{compCom}$: the computation time,
$T_{transCom}$: the transfer time.

The computation time is:

$$T_{compCom} = T_{compr}(File) + T_{Enc}(File) + T_{Dec}(File) + T_{decompr}(File) + T_{Pro} \tag{3.6}$$

Where T_{compr}: the compression time,
$T_{Enc}(File)$: the encryption time for the file,
$T_{decompr}$: the decompression time for the file at the receiver side,
$T_{Dec}(File)$: the decryption time for the file at the receiver side,
T_{Pro}: a processing time.

Total sharing time for the proposed model using Huffman coding and encryption algorithm is expressed in equation 3.7.

$$T_{sharHuff} = T_{compHuff} + T_{transHuff} \tag{3.7}$$

Where $T_{sharHuff}$: the sharing time when using Huffman coding as compression algorithm,
$T_{compHuff}$: the computation time,
$T_{transHuff}$: the transfer time.

The computation time for compression, decompression using Huffman coding and encryption, decryption is:

$$T_{compHuff} = T_{Huffcomr} + T_{comp}(Binary) + T_{Enc}(Header) + T_{Dec}(Header) + T_{Huffdecomr} + T_{Pro} \tag{3.8}$$

Where $T_{Huffcomr}$: the compression time using Huffman coding,
$T_{comp}(Binary)$: the computation time for the binary file,

$T_{Enc}(Header)$: the encryption time for the header file,

$T_{Dec}(Header)$: the decryption time for the header file at the receiver side,

$T_{Huffdecomr}$: the decompression time at the receiver side using Huffman coding,

T_{Pro}: a processing time.

Based on the studies, the $(T_{compCom}, T_{compHuff} \geq T_{comp})$ because there is an additional computation time (compression time) but the $(T_{sharCom}, T_{sharHuff} << T_{shar})$ because $T_{sharCom}$ encrypts and shares the compressed file while $T_{sharing}$ encrypts and shares the entire file. On the other hand, using Huffman coding as compression technique reduces the $T_{sharHuff}$ because in this case only the header file is encrypted and shared.

3.3 Discussion

Transferring time, reducing data size and preserving privacy for the proposed model are discussed in this section.

3.3.1 Reducing Data Size

Huffman coding was used as compression algorithm in the proposed model. Huffman coding helps in reducing data size, because Huffman coding remove the redundancy in data, it stores just each character with its count, instead of storing the same character many times, also it use fewer bits to represent frequent characters and use more bits to represent infrequent characters [17, 22]. This make the compressed file has smaller size than the original file as discussed in section 3.2.1.1. Which means that the proposed model saves storage by reducing the shared file size as explained in the following equation.

$$Saving\ Percentage = \frac{F_s - CF_s}{F_s} * 100\% \tag{3.9}$$

where F_s: the file size before compression,

and CF_s: the file size after compression.

3.3.2 Transferring Time

Using Huffman coding to compress data generates header file and binary file, the proposed model encrypts just the header file which its size is much smaller than the original file and the binary file is approximately 50% smaller than the original file so the time for encrypting header file is less than encrypting the entire file. Also sharing time for the header file is much less than sharing time for the file because the shared file is much smaller than the original file as discussed in section 3.2.2.

3.3.3 Preserving Privacy in the Proposed Model

To provide privacy to the shared data, the proposed model uses:

1. Huffman coding which encode the original data, this increase data security because the file that will be shared is coded and different from the original file [17].
2. Encryption algorithm to encrypt the header file using SSL infrastructure.
3. To improve security Cipher Block Chaining (CBC) is used. CBC makes the plaintext analysis difficult for the attacker comparing with other operation modes due to every block of the plaintext is dependent on the previous block [33].
4. In addition the proposed model add X% of data of binary file to header file to enhance the algorithm against the attacks such as direct attacks, in these attacks the attacker tries to guess the secret keys that have been used in encryption process by trying all possible combinations to find these keys, so using X% will make it harder for the attacker to guess the header file and it will take longer time.

Chapter 4

Experiments and Results

This chapter presents the experiments results for applying the proposed preserving privacy data sharing model. Several experiment scenarios were performed to evaluate the proposed model in terms of parameters such as transferring time, computation time, files size and preserving privacy. Section 4.1 represents the experimental settings for the generic proposed model. Section 4.2 shows the experiments results after applying different scenarios. Finally section 4.3 lists main differences between the proposed model and other data sharing models.

4.1 Experimental Setup

This section summarizes the overview of the experiments setup.

4.1.1 The Proposed Model using SSL

The proposed model is peer to peer model that shares data in secure way so no one can read data except the authorized users. The proposed model combined compression technique and encryption algorithms to provide privacy for the shared data.

The proposed model has two main components: The first is the compression which is used to reduce data size and to reduce transmission time. The second component is to provide privacy using encryption algorithms. Also X% of data have been added to header file before encrypt it to make the algorithm stronger against attacks. The

proposed model uses SSL encryption to encrypt the header file, it can be AES, DES and A-RSA. To enhance the algorithm against attacks, X% of the data of binary file is added to the header file before encrypt it as shown in figure 4.1.

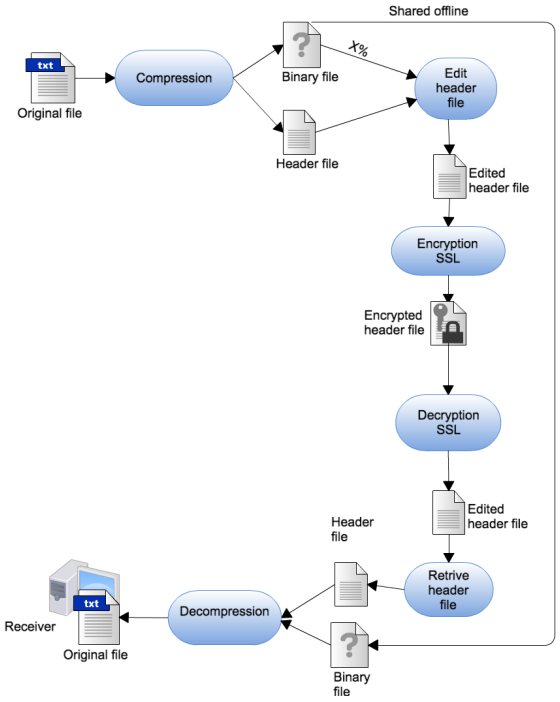


FIGURE 4.1: Generic Proposed Model.

The next section discussed the scenarios that used to test the proposed model.

4.1.2 Test Scenarios

The proposed model has been implemented using C++ programming language, using XCode version 8.2.1. Ten test files with different sizes is generated to test different scenarios (1MB - 10MB).

The scenarios were tested on macOS Sierra version 10.12.3 with: Intel Core i5, CPU 2.90 GHz, 8 GB memory and system type is 64-bit operating system.

Sharing files was tested between two users on two different network with two different speeds (1MB/sec and 16MB/sec). The experiments was performed three times, average values are used to express the results.

The test scenarios for the proposed model is summarized as follows:

- AES (file): Sharing the file after encrypt it using AES, this scenario simulates OneSwarm data sharing model as shown in figure 4.2.

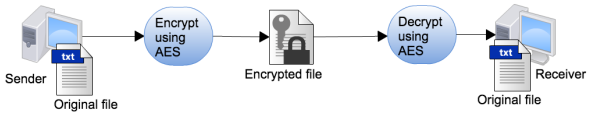


FIGURE 4.2: Encrypt File Using AES.

- DES (file): Encrypt the file using DES instead of AES then share the encrypted file as shown in 4.3.

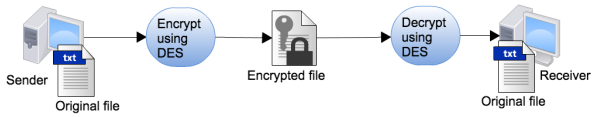


FIGURE 4.3: Encrypt File Using DES.

- RSA (file): Encrypt the file using RSA then share the encrypted file, this scenario is not common because encrypting a message by RSA takes long time as shown in 4.4.

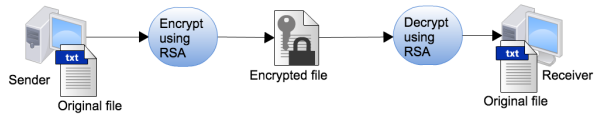


FIGURE 4.4: Encrypt File Using RSA.

- A-RSA (file): Encrypt the file using A-RSA then share it, to test sharing model using A-RSA as shown in 4.5 [17].

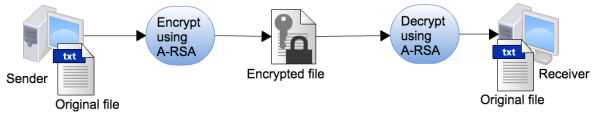


FIGURE 4.5: Encrypt File Using A-RSA.

- AES (Header file): Compress the file using Huffman coding to generate header file and binary file, then encrypt the header file using AES, after that the binary file

is shared offline while the encrypted header file is used to decompress the file of destination as shown in 4.6.

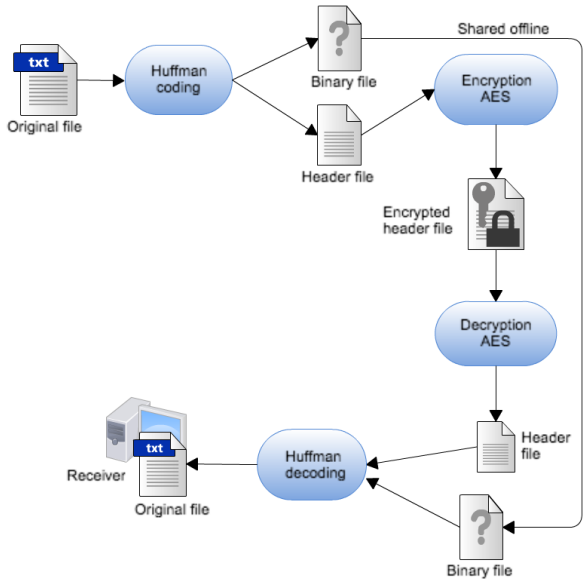


FIGURE 4.6: Encrypt Header File Using AES.

- DES (Header file): Compress the file using Huffman coding to generate header file and binary file, then encrypt the header file using DES, after that the binary file is shared offline while the encrypted header file is used to decompress the file of destination as shown in 4.7.

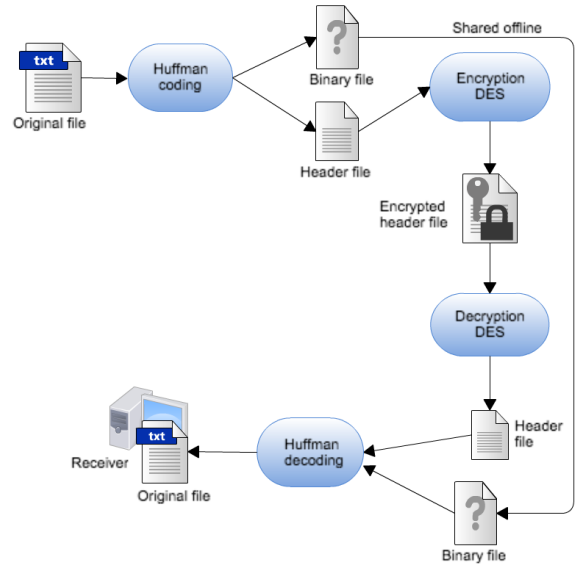


FIGURE 4.7: Encrypt Header File Using DES.

- AES (Header file + X% Binary file): Compress the file using Huffman coding to generate header file and binary file, then add X% of data of the binary file to the header file, then encrypt the modified header file using AES, then share the binary file while the encrypted header file is used to decompress the file of the destination, in this scenario we assumed that the X% of data is equal to 10% as shown in 4.8.

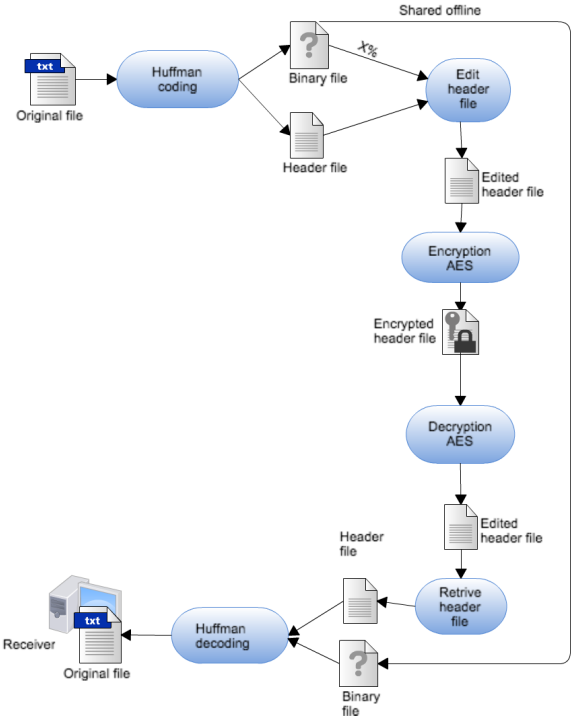


FIGURE 4.8: Encrypt Edited Header File Using AES.

- DES (Header file + X% Binary file): Compress the file using Huffman coding to generate header file and binary file, then add X% of data of the binary file to the header file, after that encrypt the modified header file using DES, then share the binary file offline while the encrypted header file is used to decompress the file of the destination, in this scenario we assumed that the X% of data is equal to 10% as shown in 4.9.

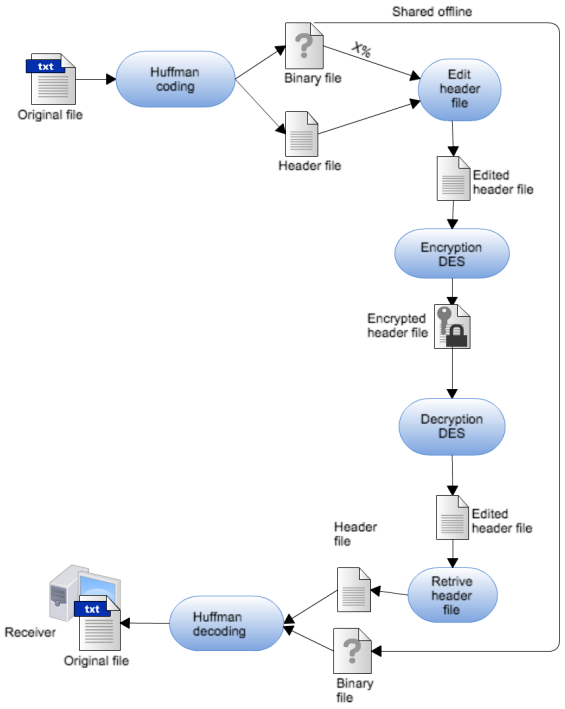


FIGURE 4.9: Encrypt Edited Header File Using DES.

4.2 Results Analysis

This section presents the finding results with respect to the size of the file and header file, sharing time, and preserving privacy.

4.2.1 File and Header Size Analysis

The proposed model uses Huffman coding as compression algorithm to reduce file sizes, so the shared files are smaller than the original files, which is helping in saving storage and reducing sharing time. The following equation clarifies how compression algorithms save storage, table 4.1 shows the size of ten files and the files size after encrypting them using the scenarios, the second and third column show the Header file size and Binary file size.

$$Saving\ Percentage = \frac{F_s - CF_s}{F_s} * 100\% \tag{4.1}$$

where F_s: the file size before compression,

and CF_s: the file size after compression.

Figure 4.10 shows that using compression algorithm is good choice before encrypting large data. Using compression with encryption algorithms reduced file size to 45% compared to the original file size so this helps in reducing storage space and reducing sharing time, according to the following.

$(T_{shar}(HeaderFile) << T_{shar(File)})$

because $(Header\ File\ Size << Original\ File\ Size)$.

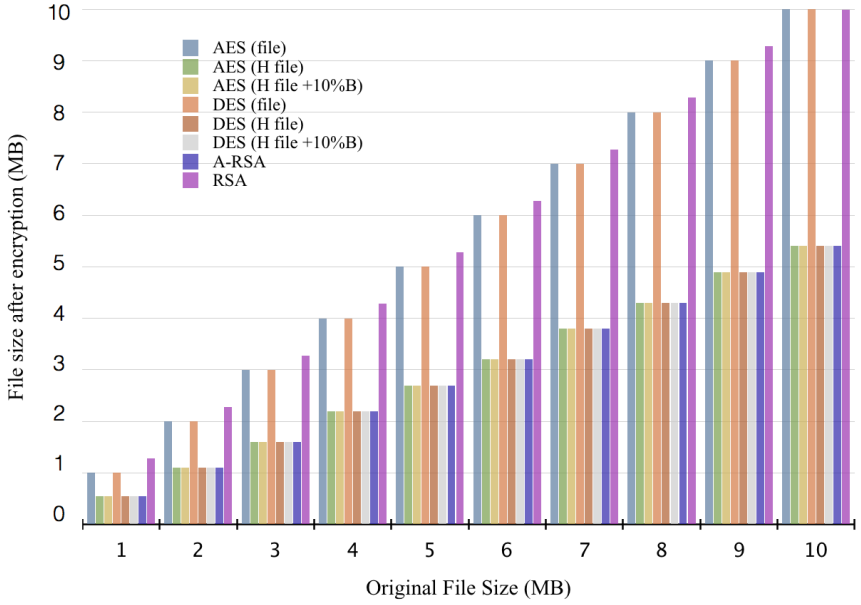


FIGURE 4.10: File Size in Different Scenarios.

Table 4.1: File Size

file Size (MB)										
file Size (MB)	**Header file Size (KB)**	**Binary file Size (MB)**	**AES(file)**	**AES(H file)**	**AES(H file + 10%B)**	**DES(file)**	**DES(H file)**	**DES(H file + 10%B)**	**A-RSA**	**RSA**
1	1	0.549	1	0.55	0.55	1	0.55	0.55	0.55	1.28
2	1	1.099	2	1.1	1.1	2	1.1	1.1	1.1	2.28
3	1	1.599	3	1.6	1.6	3	1.6	1.6	1.6	3.28
4	1	2.199	4	2.2	2.2	4	2.2	2.2	2.2	4.28
5	1	2.699	5	2.7	2.7	5	2.7	2.7	2.7	5.28
6	1	3.199	6	3.2	3.2	6	3.2	3.2	3.2	6.28
7	1	3.799	7	3.8	3.8	7	3.8	3.8	3.8	7.28
8	1	4.299	8	4.3	4.3	8	4.3	4.3	4.3	8.28
9	1	4.899	9	4.9	4.9	9	4.9	4.9	4.9	9.28
10	1	5.399	10	5.4	5.4	10	5.4	5.4	5.4	10.28

4.2.2 Computation Time for Compression and Encryption Process

This section shows the computation time for the mentioned scenarios where the computation time for AES (file), DES (file), RSA (file) and A-RSA(file) is:

$$T_{comp} = T_{Enc}(File) + T_{Dec}(File) + T_{Pro} \tag{4.2}$$

Where $T_{Enc}(File)$: the encryption time for the file,
$T_{Dec}(File)$: the decryption time for the file,
and T_{Pro}: a processing time.

While the computation time for AES (Header file) and DES (Header file).

$$\begin{aligned} T_{compHuff} = T_{Huffcomr} + T_{Enc}(Header) + \\ T_{Dec}(Header) + T_{Huffdecomr} + T_{Pro} \end{aligned} \tag{4.3}$$

Where $T_{Huffcomr}$: the compression time using Huffman coding,
$T_{Enc}(Header)$: the encryption time for the header file,
$T_{Dec}(Header)$: the decryption time for the header file at the receiver side,
$T_{Huffdecomr}$: the decompression time at the receiver side using Huffman coding
and T_{Pro}: a processing time.

And the computation time for AES (Header file + X% Binary file) and DES (Header file + X% Binary file):

$$\begin{aligned} T_{compEditHuff+} = T_{Huffcomr} + T_{comp}(Binary) + T_{Enc}(Header) + \\ T_{Dec}(Header) + T_{Huffdecomr} + T_{Pro} \end{aligned} \tag{4.4}$$

Where $T_{Huffcomr}$: the compression time using Huffman coding,
$T_{comp}(Binary)$: the computation time for the binary file,
$T_{Enc}(Header)$: the encryption time for the header file,
$T_{Dec}(Header)$: the decryption time for the header file at the receiver side,

$T_{Huffdecomr}$: the decompression time at the receiver side using Huffman coding and T_{Pro}: a processing time.

Based on the previous equations, $T_{compHuff}$ and $T_{compEditHuff}$ will consumes more time than the T_{comp} because of the compression time. Table 4.2 shows the average execution time for encrypting 10 files. Each file is encrypted three times, the average of these readings was computed.

Figure 4.11 shows that encryption time for A-RSA consumes longer time compared to the others. AES (file) and DES (file) consumes less computation time than AES (H file) and DES (H file) because compression process consumes more time. While there is no significant difference between encrypting header file and encrypting (header file + 10% Binary file) using AES or DES.

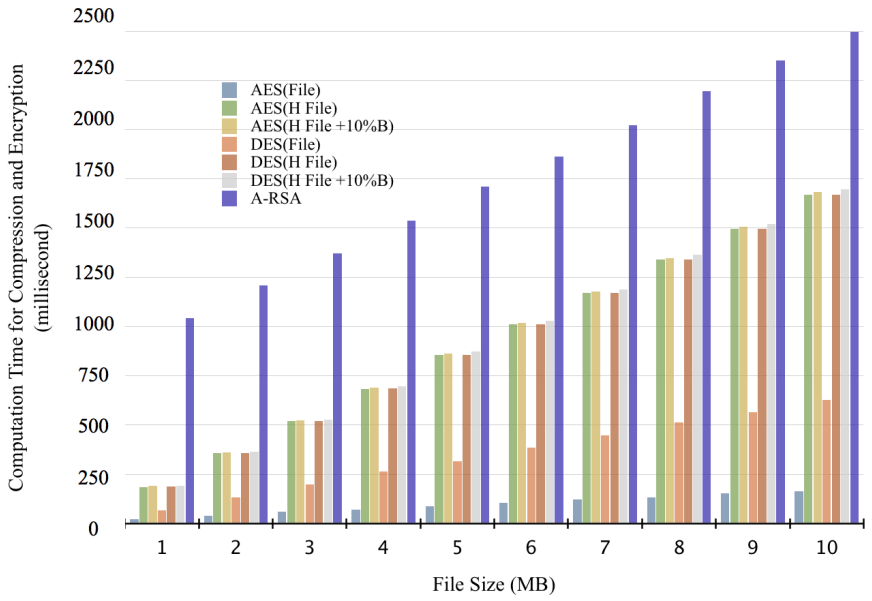


FIGURE 4.11: Computation Time for Compression and Encryption Process.

Table 4.3 shows the average execution time for decrypting 10 files. Each file is decrypted three times, the average of these readings was computed. According to the results the computation time for AES (file), DES (file) consumes less time than AES (H), DES (H), AES (Header file + 10% Binary file) and DES (Header file + 10% Binary file) due to the decompression process.

TABLE 4.2: Computation Time for Compression and Encryption Process.

Encryption Time (millisecond)												
file Size	AES(file)	AES(H file)			AES(H file + 10%B file)	DES(file)	DES(H file)			DES(H file + 10%B file)	RSA	A-RSA
		Huffman	AES	Sum			Huffman	DES	Sum			
1Mb	22	185	4	189	191	68	185	5	190	191	38110	1043
2Mb	38	353	4	357	360	134	353	5	358	364	72103	1211
3Mb	61	515	4	519	523	199	515	5	520	528	113104	1373
4Mb	70	680	4	684	690	263	680	5	685	696	144106	1538
5Mb	88	852	4	856	862	318	852	5	857	873	187106	1710
6Mb	104	1007	4	1011	1018	384	1007	5	1012	1028	232107	1865
7Mb	122	1166	4	1170	1178	448	1166	5	1171	1190	282111	2024
8Mb	133	1336	4	1340	1349	512	1336	5	1341	1365	311113	2194
9Mb	152	1492	4	1496	1506	567	1492	5	1497	1522	334116	2350
10Mb	165	1666	4	1670	1682	628	1666	5	1671	1699	377117	2524

Table 4.3: Computation Time for Decompression and Decryption Process.

Decryption Time (millisecond)												
file Size	AES(file)	AES(H file)			AES(H file + 10%B file)	DES(file)	DES(H file)			DES(H file + 10%B file)	RSA	A-RSA
		Huffman	AES	Sum			Huffman	DES	Sum			
1Mb	27	290	4	294	296	71	290	5	295	298	78070	1620
2Mb	42	863	4	867	869	137	863	5	868	1073	161063	2393
3Mb	62	1320	4	1324	1328	173	1320	5	1325	1333	244064	3150
4Mb	81	1880	4	1884	1916	230	1880	5	1885	1896	316066	3910
5Mb	99	2333	4	2337	2343	277	2333	5	2338	2355	399066	4663
6Mb	120	2980	4	2984	2791	344	2980	5	2985	3001	475067	5410
7Mb	140	3520	4	3524	3532	391	3520	5	3525	3545	550071	6150
8Mb	158	4143	4	4147	4156	442	4143	5	4148	4171	645073	6973
9Mb	183	4600	4	4604	4614	466	4600	5	4605	4629	729076	7730
10Mb	191	5243	4	5247	5258	505	5243	5	5248	5274	767077	8473

Figure 4.12 shows that decryption time for A-RSA consumes more time from the others. AES(file) and DES(file) consumes less time in decryption than AES(H file) and DES(H file) that's because decompression process consumes more time. While there is no significant difference between decrypting header file and decrypting (header file +10%) using AES or DES.

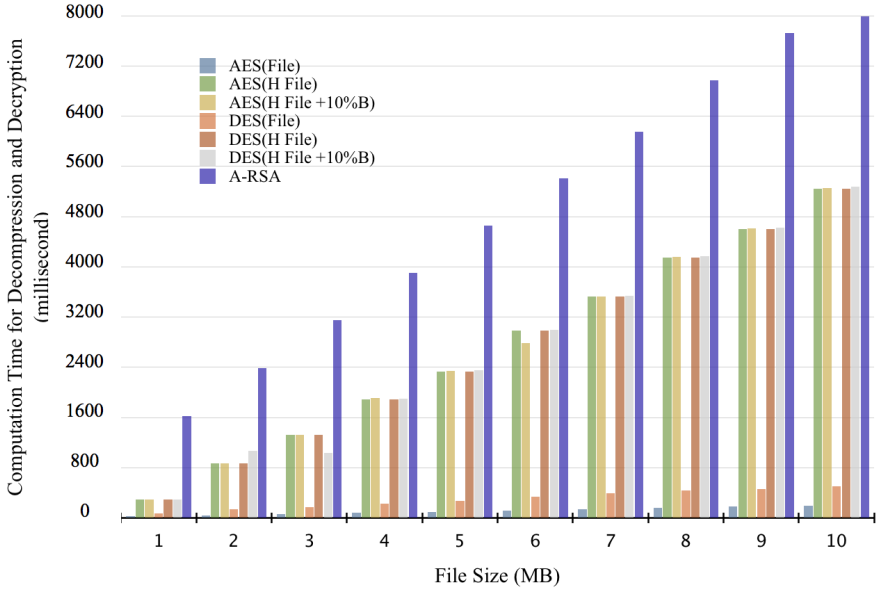


FIGURE 4.12: Computation Time for Decompression and Decryption Process.

4.2.3 Transferring Time

Transferring time is the time for transferring a file from a user to another. Table 4.4 shows transferring time for the ten files using two different speeds (1MB/sec and 16MB/sec) for the scenarios mentioned before. The results shows that the transferring time is approximately equal for AES (file), DES (file) and RSA (file) because the files in these scenarios have the same size. Also the transferring time for AES (H file), DES (H file) and A-RSA has the same transferring time. And the transferring time is equal for AES (Header file + 10% Binary file) and DES (Header file + 10% Binary file).

Efficiency for the network can be measured by network bandwidth. The bandwidth is the number of bits that can be transmitted over network in a certain period of time. Bandwidth described in the following equation [34]:

$$Bandwidth = \frac{file\ size(MB)}{Time(sec)} \tag{4.5}$$

TABLE 4.4: Transferring Time.

Transferring Time (millisecond)						
file Size	AES/DES(file) and RSA		AES/DES(H file) and A-RSA		AES/DES(H file + 10%B file)	
	Speed1 (1MB/sec)	Speed2 (16MB/sec)	Speed1 (1MB/sec)	Speed2 (16MB/sec)	Speed1 (1MB/sec)	Speed2 (16MB/sec)
1Mb	8570	1090	32	25	466	35
2Mb	17160	2120	32	25	940	71
3Mb	25020	3210	32	25	1367	103
4Mb	33310	4200	32	25	1872	141
5Mb	41520	5210	32	25	2295	173
6Mb	49700	7150	32	25	2719	205
7Mb	57840	7240	32	25	3227	243
8Mb	65940	8240	32	25	3650	275
9Mb	74310	9300	32	25	4159	313
10Mb	82590	10360	32	25	4582	345

Based on the transmission time results for AES (file) and DES (file) on speed 1MB/sec the bandwidth is: $0.120MBps$, this means that the network able to deliver $0.120MB$ every second while on speed 16MB/sec the bandwidth is $0.946MBps$.

While the network bandwidth for AES (H file), DES (H file) and A-RSA on speed 1MB/sec is $171.8MBps$ and on speed 16MB/sec is $220MBps$. And the bandwidth for AES (Header file + 10% Binary file) and DES (Header file + 10% Binary file) on speed 1MB/sec is $2MBps$ and on speed 16MB/sec is $28MBps$. This means the bandwidth when using compression algorithm is better.

Figure 4.13 shows transferring time on 1MB/sec speed. Based on the results AES (file), DES (file) and RSA consumes more time. While using compression is faster by 98%. And using compression then edit header file is faster than AES (file) and DES(file) by 94%.

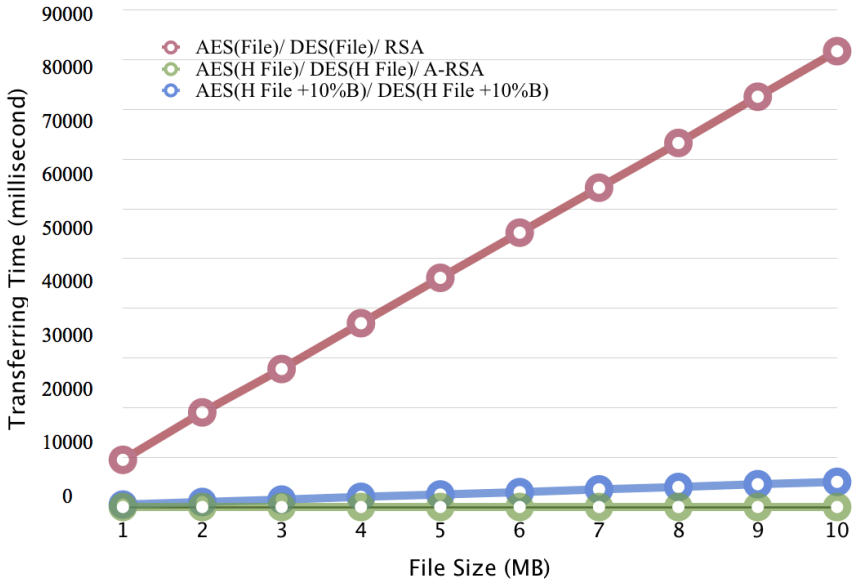


FIGURE 4.13: Transferring Time on 1MB/sec Speed.

Figure 4.14 shows transferring time on 16MB/sec speed. Based on the results AES(file), DES(file) and RSA consumes more time. While using compression is faster by 98%. Also using compression then edit header file is faster than AES (file) and DES(file) by 98%.

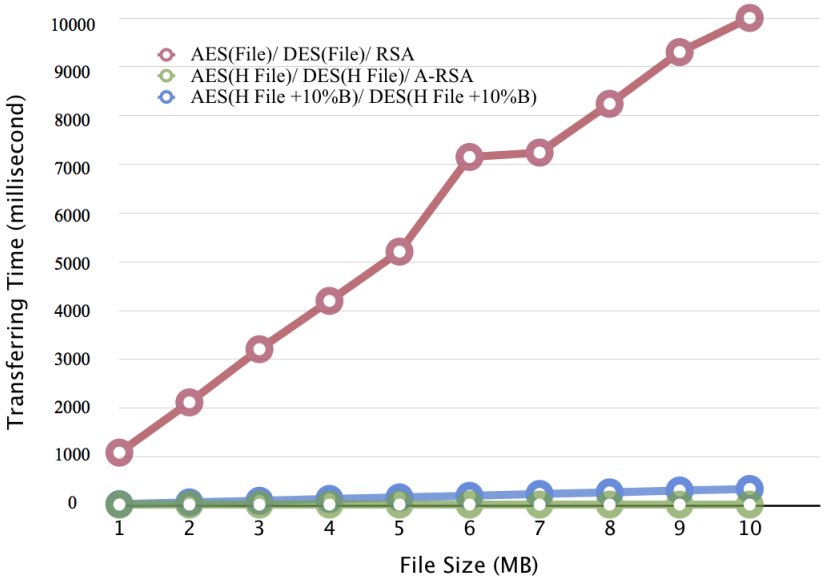


FIGURE 4.14: Transferring Time on 16MB/sec Speed.

4.2.4 Sharing Time

This section shows the sharing time for the mentioned scenarios where the sharing time is:

$$T_{shar} = T_{comp} + T_{trans} \tag{4.6}$$

Where T_{shar}: the sharing time,
T_{comp}: the computation time
and T_{trans}: the transfer time.

Table 4.5 shows the sharing time for the ten files on 1MB/sec speed. The results shows that the sharing time for AES(file), DES(file) and A-RSA consumes more time. While transferring time when using compression algorithm consumes less time assuming that the binary file is shared offline.

TABLE 4.5: Sharing Time on 1MB/sec Speed.

Sharing Time (millisecond)							
file Size	AES(file)	AES(H file)	AES(H file + 10%B file)	DES(file)	DES(H file)	DES(H file + 10%B file)	A-RSA
1Mb	8619	515	953	8709	517	955	2695
2Mb	17240	1256	2169	17431	1258	2377	3636
3Mb	25143	1875	3218	25392	1877	2928	4555
4Mb	33461	2600	4478	33803	2602	4464	5480
5Mb	41707	3225	5500	42115	3227	5523	6405
6Mb	49924	4027	6528	50428	4029	6748	7307
7Mb	58102	4726	7937	58679	4728	7962	8206
8Mb	66231	5519	9155	66894	5521	9186	9199
9Mb	74644	6132	10279	75343	6134	10310	10112
10Mb	82946	6949	11522	83723	6951	11555	11029

Figure 4.15 shows that sharing time on 1MB/sec speed when using compression and add X% is 87% faster than sharing file without compression assuming that the binary file is shared offline. While using compression only is 92% faster than sharing file without compression.

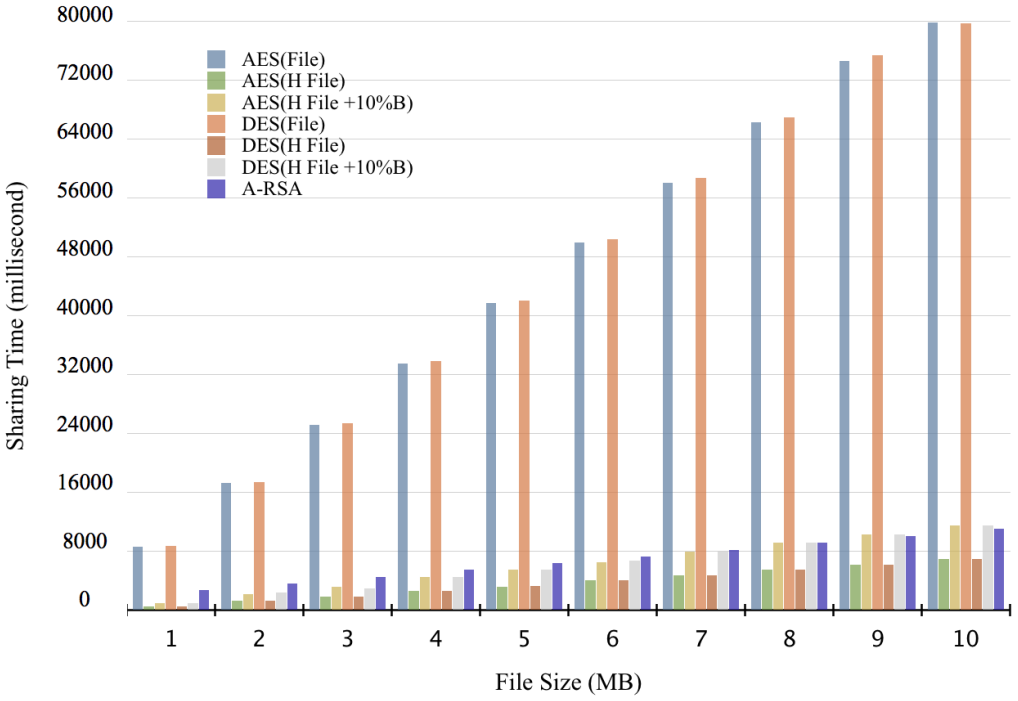


FIGURE 4.15: Sharing Time on 1MB/sec Speed.

Table 4.6 shows the sharing time for the ten files on 1MB/sec speed. The results shows that the sharing time for AES(file), DES(file) and A-RSA consumes more time. While transferring time when using compression algorithm consumes less time assuming that the binary file is shared offline.

TABLE 4.6: Sharing Time on 16MB/sec Speed.

Sharing Time (millisecond)							
file Size	AES(file)	AES(H file)	AES(H file + 10%B file)	DES(file)	DES(H file)	DES(H file + 10%B file)	A-RSA
1Mb	1139	508	522	1229	510	524	2663
2Mb	2200	1249	1300	2391	1251	1508	3604
3Mb	3333	1868	1954	3582	1870	1664	4523
4Mb	4351	2593	2747	4693	2595	2733	5448
5Mb	5397	3218	3378	5805	3220	3401	6373
6Mb	7374	4020	4014	7878	4022	4234	7275
7Mb	7502	4719	4953	8079	4721	4978	8174
8Mb	8531	5512	5780	9194	5514	5811	9167
9Mb	9634	6125	6433	10333	6127	6464	10080
10Mb	10716	6942	7285	11493	6944	7318	10997

Figure 4.16 shows that sharing time on 1MB/sec speed when using compression and add X% is 38% faster than sharing file without compression assuming that the binary file

is shared offline. While using compression only is 41% faster than sharing file without compression.

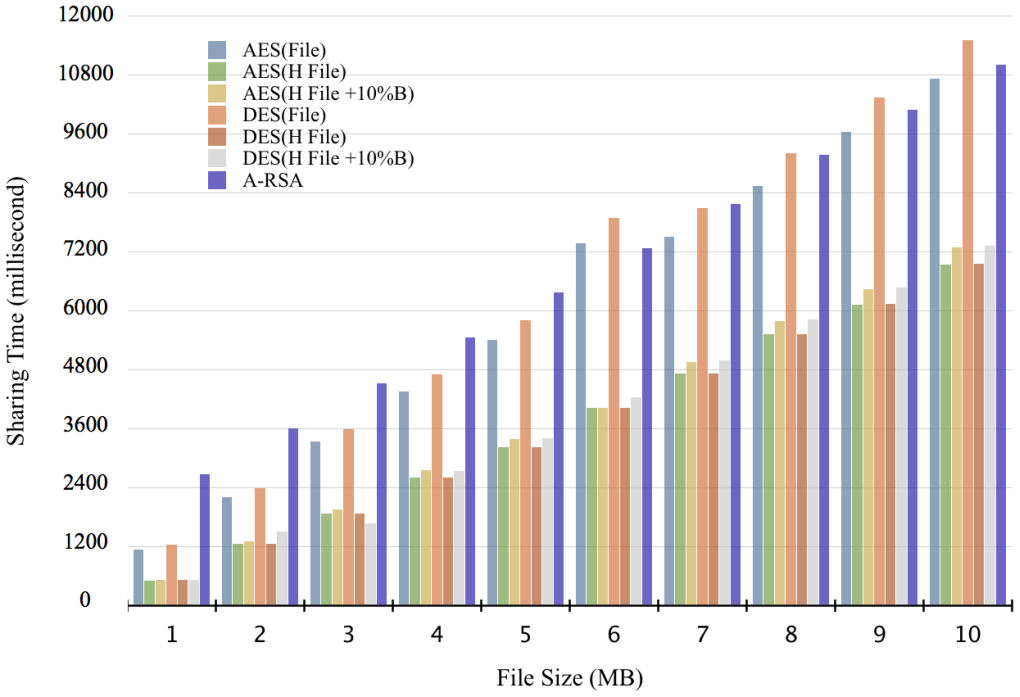


FIGURE 4.16: Sharing Time on 16MB/sec Speed.

4.2.5 Security Analysis

In this section we describe how the proposed data sharing model resists to attacks through proofs.

1. The Frequency Of Blocks (FOB) attack: If one of the blocks repeated within the same plaintext, then the blocks will has the same ciphertext. The proposed data sharing model is protected from the frequency of blocks (FOB) attack, because the model uses compression technique before encrypt the message. Because of that, the characters in the compressed message has a uniform distribution so no repeated blocks are produced for the same plaintext.

2. Chosen Plaintext Attack (CPA): This attack can be happened, when the adversary chooses arbitrary message, and the attacker is able to get the source system and insert the chooses message into the system to encrypt it, so the attacker has the plaintext with corresponding ciphertext, he can build sets of plaintexts-ciphertexts

[33]. The proposed model is protected from chosen plaintext attack because it depends on the X% data which added to the header file before encrypting it, where each message has different X%. Thus, encrypting the same message more than once produces different ciphertext.

3. Timing Attack (TA): In timing attack the attacker can use the time taken by a hardware to decrypt the message. The proposed model is secure against timing attack, since the attacker still needs the X% of the binary file to decrypt the ciphertext.

4.2.6 Preserving Privacy

To provide privacy to the shared data, the proposed model uses:

1. Huffman coding which encode the original data, this increase data security because the file that will be shared is coded and different from the original file [17].
2. Encryption algorithm to encrypt the header file.
3. To improve security Cipher Block Chaining (CBC) is used. CBC makes the plaintext analysis difficult for the attacker comparing with other operation modes due to every block of the plaintext is dependent on the previous block [33].
4. In addition the proposed model add X% of data of binary file to header file to enhance the algorithm against the attacks.

4.3 Main Differences between The Proposed Model and other Data Sharing Models

The proposed preserving privacy data sharing model differs from other data sharing models in several points, that are summarized as follows:

1. Privacy: The proposed model provides privacy since it uses encryption algorithms and Huffman coding as compression technique.

2. Sharing time: the proposed model is 87% faster on 1MB/sec speed than other models and it is 38% faster on network speed of 16MB/sec, since it uses file compression, assuming the binary file is shared offline.

3. Save storage: the proposed model is 45% less in size than files size in other models, since it uses Huffman coding as compression technique to reduce file size before encrypting it.

Chapter 5

Conclusion and Future Work

In this chapter, the work performed in this thesis is concluded in the first section. Future research directions are suggested in the second section.

5.1 Conclusion

Electronic data interchange can be classified as one of the important areas of information technology, where the need for data sharing increasingly required in almost every field such as health-care, e-government, e-commerce and scientific researches. Data sharing concept can be defined as the process of interchanging, analyzing, retrieving and integrating data among multiple data sources in a controlled access manner. The use of information technology in different areas began to increase, the exchange and sharing different types of information was also required. Although data sharing facilitates the way that data can be exchanged, security concerns arise as a challenge for conducting data sharing, many polices include confidentiality, integrity, availability and privacy must be taken into consideration. This thesis focused on preserving privacy. This thesis provided a literature review for different preserving privacy models that facilitate data sharing among different areas:

- Semantic Privacy-Preserving Model.
- Capability-based Access Control Model.
- BitTorrent Model.

- OneSwarm Model

Also, it provided a literature review for different encryption technique:

- RSA.
- Augmented-RSA.
- Data Encryption Standard.
- Advanced Encryption Standard.

Then it clarified lossless data compression techniques such as:

- Huffman Coding.
- Run-Length Enoding.
- Lempel-Ziv Algorithm.

Then, it proposed a new privacy preserving data sharing model that combined Huffman coding and SSL encryption that it can be DES, AES or A-RSA in order to provide privacy in data sharing also to facilitate data sharing in different areas.

The proposed model is tested on ten files using different scenarios to compare the results, the scenarios are:

- AES (file): encrypt the file using AES before sharing it, to simulate OneSwarm.
- DES (file): encrypt the file using DES instead of AES before sharing it.
- RSA (file): encrypt the file using RSA before sharing it, which is not common.
- A-RSA (file): encrypt the file using A-RSA before sharing it, to test A-RSA algorithm.
- AES (Header file): compress the file to generate header file and binary file then encrypt the header file using AES before sharing it.
- DES (Header file):compress the file to generate header file and binary file then encrypt the header file using DES before sharing it.

- DES (Header file + X% Binary file): compress the file to generate header file and binary file after that add X% of data to the header file then encrypt the edited header file using DES before sharing it.
- AES (Header file + X% Binary file): compress the file to generate header file and binary file after that add X% of data to the header file then encrypt the edited header file using AES before sharing it.

Experiments results show that the proposed model saves storage by 45% compared to other scenarios that sharing the full file without compression.
Also, experiments result on speed 1MB/sec shows that sharing time for the proposed model is 87% faster than sharing the complete file. While on 16MB/sec it is 38% faster than the others, also the proposed model saved 45% storage.
In addition, the proposed model provide privacy to the shared files due to:

1. Huffman coding which encode the original data, this increase data security because the file that will be shared is coded and different from the original file [17].
2. Encryption algorithm to encrypt the header file.
3. To improve security Cipher Block Chaining (CBC) is used. CBC makes the plaintext analysis difficult for the attacker comparing with other operation modes due to every block of the plaintext is dependent on the previous block [33].
4. In addition the proposed model add X% of data of binary file to header file to enhance the algorithm against the attacks such as direct attacks, in these attacks the attacker tries to guess the secret keys that have been used in encryption process by trying all possible combinations to find these keys, so using X% will make it harder for the attacker to guess the header file and it will take longer time.

5.2 Future Work

The proposed model could be further developed in:

1. Develop the proposed model to include images, sounds and videos compression.

2. Implement new data sharing model using compression and hashing, test the performance and compare the results with the proposed model.

3. Test the model on real applications like BitTorrent and OneSwarm.

Appendix A

Detailed experiments results

In this appendix, The experiments result in detailed was described.

TABLE A.1: The Size of Header File and Binary File Generated from Huffman Coding.

File Size (MB)	Header File Size (KB)	Binary File Size (MB)	File size after compression (MB)
1	1	0.549	0.55
2	1	1.099	1.1
3	1	1.599	1.6
4	1	2.199	2.2
5	1	2.699	2.7
6	1	3.199	3.2
7	1	3.799	3.8
8	1	4.299	4.3
9	1	4.899	4.9
10	1	5.399	5.4

TABLE A.2: Execution Time for the encryption process of AES(File).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	23	22	22	22
2	37	39	38	38
3	65	62	56	61
4	73	70	69	70
5	87	93	85	88
6	100	111	103	104
7	127	121	120	122
8	135	134	131	133
9	152	148	158	152
10	168	166	163	165

TABLE A.3: Execution Time for the decryption process of AES(File).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	27	23	33	27
2	43	44	41	42
3	66	60	62	62
4	80	79	86	81
5	97	97	103	99
6	121	117	122	120
7	138	136	146	140
8	158	158	160	158
9	197	175	179	183
10	190	194	189	191

TABLE A.4: Execution Time for the encryption process of DES(File).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	68	68	69	68
2	134	136	132	134
3	202	198	197	199
4	262	270	259	263
5	323	309	322	318
6	383	385	385	384
7	446	452	446	448
8	510	507	521	512
9	560	576	567	567
10	628	639	617	628

TABLE A.5: Execution Time for the decryption process of DES(file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	71	71	72	71
2	127	133	152	137
3	190	167	162	173
4	218	259	215	230
5	262	313	258	277
6	360	359	315	344
7	401	360	414	391
8	463	424	441	442
9	497	450	451	466
10	534	491	492	505

TABLE A.6: Execution Time for the encryption process of AES(H file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	190	189	189	189
2	362	356	354	357
3	526	521	511	519
4	691	681	681	684
5	871	848	850	856
6	1013	1008	1012	1011
7	1168	1175	1169	1170
8	1341	1349	1331	1340
9	1496	1491	1501	1496
10	1679	1664	1669	1670

TABLE A.7: Execution Time for the decryption process of AES (H file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	304	294	284	294
2	874	864	864	867
3	1324	1314	1334	1324
4	1884	1864	1904	1884
5	2334	2324	2354	2337
6	2974	3004	2974	2984
7	3524	3534	3514	3524
8	4074	4234	4134	4147
9	4654	4584	4574	4604
10	5274	5214	5254	5247

TABLE A.8: Execution Time for the encryption process of DES(H file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	190	190	191	190
2	359	358	358	358
3	521	520	520	520
4	685	686	685	685
5	857	858	857	857
6	1012	1013	1012	1012
7	1171	1171	1172	1171
8	1342	1341	1341	1341
9	1497	1497	1498	1497
10	1671	1671	1672	1671

TABLE A.9: Execution Time for the decryption process of DES (H file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	298	294	295	295
2	868	867	871	868
3	1324	1325	1328	1325
4	1888	1884	1885	1885
5	2341	2338	2337	2338
6	2985	2984	2988	2985
7	3525	3528	3524	3525
8	4151	4148	4147	4148
9	4608	4605	4604	4605
10	5251	5248	5247	5248

TABLE A.10: Execution Time for the encryption process of AES(H file + 10%B file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	194	191	190	191
2	363	360	359	360
3	526	522	522	523
4	693	689	689	690
5	866	861	861	862
6	1017	1021	1018	1018
7	1181	1178	1177	1178
8	1352	1349	1348	1349
9	1509	1505	1505	1506
10	1685	1683	1678	1682

TABLE A.11: Execution Time for the decryption process of AES(H file + 10%B file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	295	297	297	296
2	869	870	870	869
3	1328	1328	1328	1328
4	1889	1889	1970	1916
5	2344	2343	2343	2343
6	2791	2791	2791	2791
7	3532	3532	3532	3532
8	4156	4157	4156	4156
9	4615	4614	4614	4614
10	5259	5258	5258	5258

TABLE A.12: Execution Time for the encryption process of DES(H file + 10%B file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	191	192	191	191
2	365	364	355	364
3	528	530	527	528
4	696	695	699	696
5	873	875	871	873
6	1027	1028	1029	1028
7	1189	1191	1190	1190
8	1362	1371	1362	1365
9	1521	1526	1521	1522
10	1697	1698	1702	1699

TABLE A.13: Execution Time for the Decryption Process of DES(H file + 10%B file).

File Size(MB)	Execution Time (milliseconds)			
	First Reading	**Second Reading**	**Third Reading**	**Average**
1	299	298	298	298
2	1074	1074	1073	1073
3	1334	1334	1313	1333
4	1896	1897	1897	1896
5	2352	2358	2356	2355
6	3000	3001	3004	3001
7	3547	3543	3545	3545
8	4170	4168	4176	4171
9	4629	4630	4629	4629
10	5276	5273	5275	5274

Bibliography

[1] Doan A. Elmagarmid A. Clifton, C. Privacy-preserving data integration and sharing. *ACM-Research issues in data mining and knowledge discovery*, 45(1):19–26, 2004.

[2] Piatek M. Krishnamurthy A. Anderson T. Isdal, T. Privacy-preserving P2P data sharing with oneswarm. *ACM SIGCOMM Computer Communication Review*, 40 (4):111–112, 2010.

[3] Khan L. Paul R. Thuraisingham B. Harris, D. Standards for secure data sharing across organizations. *ACM-Computer Standards and Interfaces*, 29(1):86–96, 2007.

[4] Kohno T. Krishnamurthy A. Piatek, M. Challenges and directions for monitoring P2P file sharing networks. *Proceeding HOTSEC'08 Proceedings of the 3rd conference on Hot topics in security*, 12(12):1–6, 2008.

[5] Kim H. Kim D. Son, J. On secure data sharing in cloud environment. *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, 6(1):1–6, 2014.

[6] Muralidhar K. Sarathy, R. Secure and useful data sharing. *Elsevier*, 42(1):204–220, 2004.

[7] Chun S. Kim D. Keromytis A. Choi, J. Securegov: Secure data sharing for government services. *The Proceedings of the 14th Annual International Conference on Digital Government Research*, 41(1):1–6, 2013.

[8] Yang J. Hu, Y. A semantic privacy-preserving model for data sharing and integration. *ACM-Web Intelligence, Mining and Semantics*, 9(9):1–12, 2011.

[9] Balazinska M. Gribble S. Levy H. Geambasu, R. Homeviews: P2P middleware for personal data sharing applications. *ACM*, 42(9):235–246, 2007.

[10] B. Cohen. Incentives build robustness in BitTorrent. *IPTPS*, 43(9):1–5, 2003.

[11] Johnson M. Appari, A. Information security and privacy in Healthcare: Current state of research. *International Journal of Internet and Enterprise Management*, 6 (4):1–36, 2008.

[12] Zaiane O. Oliveira, S. Achieving privacy preservation when sharing data for clustering. *Secure Data Management*, 3178(9):67–82, 2004.

[13] Levine B. Liberatore M. Prusty, S. Forensic investigation of the OneSwarm anonymous filesharing system. *Proceedings of the 18th ACM conference on Computer and communications security*, 44(9):201–214, 2011.

[14] Dixon D. Wilson J. Sasi, S. A general comparison of symmetric and asymmetric cryptosystems for WSNs and an overview of location based encryption technique for improving security. *IOSR Journal of Engineering*, 4(3):01–04, 2014.

[15] Philemon D. Falaki O. Alese, K. Comparative analysis of public-key encryption schemes. *International Journal of Engineering and Technology*, 2(9):1552–1568, 2012.

[16] Shamir A. Adleman L. Rivest, R. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[17] Alsadeh A. Karakra, A. A-RSA: Augmented RSA. *SAI Computing Conference*, 45 (9):1016–1023, 2016.

[18] Supriya Singh, G. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applicationsy*, 67(19): 33–38, 2013.

[19] The National Institute of Standards and Technology. Announcing the advanced encryption standard (AES). *Federal Information Processing Standards Publication 197*, 5(1):1–4, 2001.

[20] G. Blelloch. *Introduction to Data Compression*. Carnegie Mellon University, 2013.

[21] Amaeasinghe U. Kodituwakku, S. Comprison of lossless data compression algorithms for text data. *Indian Journal of Computer Science and Engineering*, 1(4): 416–425, 2006.

[22] Itwala U. Rana R. Patel, H. Survey of lossless data compression algorithms. *International Journal of Engineering Research and Technology*, 4(4):926–929, 2015.

[23] Aouad G. Kagioglou M. Bakis, N. Towards distributed product data sharing environments progress so far and future challenges. *Elsevier-Automation in Construction*, 16(5):586–595, 2007.

[24] Sarin S. Greif, I. Data sharing in group work. *ACM*, 5(2):187–211, 1987.

[25] Bugrara K. Mannai, D. Enhancing inter-operability and data sharing in medical information systems. *ACM*, 22(2):495–498, 1993.

[26] Grunwald D. Sicker D. Bauer, K. The arms race in P2P. *37th Research Conference on Communication, Information and Internet Policy*, 7(1):1–4, 2009.

[27] Begg C. Connolly, T. *Darabase System*. Addison Wiley, 1996.

[28] Tahboub R. Sarahneh, S. Secure data sharing polices and architecture preserving privacy. *The 7th International Conference on Information Technology*, 40(1):308–314, 2015.

[29] Manohar T. Surendra, B. Advance secure multi-owner data sharing for dynamic groups in the untrusted cloud. *International Journal of advanced technology and Innovative Research*, 10(6):1214–1219, 2014.

[30] C. Mackay. Big data could benefit Healthcare but raises privacy concerns, Dec 2015. URL www.theparliamentmagazine.eu.

[31] Tahboub R. Sarahneh, S. Secure data sharing model using new technique for preserving privacy. *Journal of Internet Technology and Secured Transactions*, 4(4): 427–434, 2015.

[32] M. Rhee. *Internet Security: Cryptographic Principles, Algorithms and Protocols*. John Wiley and Sons Ltd, 2003. ISBN 0-470-852285-2.

[33] W. Stallings. *Cryptography and network security principles and practice*. Pearson Education, 2011. ISBN 978-0-13-609704-4.

[34] Davie B. Peterson, L. *Computer Networks a systems approach*. Elsevier, Inc, 2012.

[35] W. Diffie. The first ten years of public-key cryptography. *IEEE*, 76(5):560–577, 1988.

[36] J. Daemen and V. Rijmen. AES proposal: Rijndael. 48, 1999.

[37] D. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, 40(1):1098–1101, 1952.

[38] Mustafa M. Ibrhim, A. Comparison between (RLE and Huffman) algorithms for lossless data compression. *International journal of innovative technology and research*, 20(1):1808–1812, 2015.

[39] Shacham H. Modadugu N. Boneh D. Goh, E. Sirius: Securing remote untrusted storage. *Proc. Network and Distributed Systems Security Symp*, 48(1):131–145, 2003.

[40] Fu K. Green M. Hohenberger S. Ateniese, G. Improved proxy re-encryption schemes with applications to secure distributed storage. *Proc. Network and Distributed Systems Security Symp*, 57(1):29–43, 2005.

[41] Hossain B. Uddin S. Imtiaz S. Hossain, A. Performance analysis of different cryptography algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering*, 6(3):659–665, 2016.

[42] Maakar S. Kumar S. Singh, S. A performance analysis of DES and RSA cryptography. *International Journal of Emerging Trends and Technology in Computer Science*, 2(3):418–423, 2013.

[43] Khalid A. Osman S. Rihan, S. A performance comparison of encryption algorithms AES and DES. *International Journal of Engineering Research and Technology*, 4 (12):151–154, 2015.

[44] Rane S. Brito A. Uzun E. Freudiger, J. Privacy preserving data quality assessment for high-fidelity data sharing. *Proceedings of the 2014 ACM Workshop on Information Sharing and Collaborative Securit*, 10(1):21–29, 2014.

[45] Sarin S. reif, I. Data sharing in group work. *ACM*, 5(2):187–211, 1987.

[46] M. Sharma. Compression using huffman coding. *IJCSNS International Journal of Computer Science and Network Security*, 10(5):133–141, 2010.

[47] Vazirgiannis M. Varlamis, I. Bridging xml-schema and relational databases a system for generating and manipulating relational databases using valid xml documents. *ACM*, 30(1):105–114, 2001.

[48] McCoy D. Grunwald D. Sicker D. Bauer, K. Bitblender: Light-weight anonymity for bittorrent. *ACM*, 21(1):1–4, 2008.

[49] University of Miami. Confidentiality, integrity and availability (CIA), April 2008. URL http://it.med.miami.edu/x904.xml.

[50] M. Fonville. Confidential peer-to-peer file-sharing using social-network sites. *13th Twente Student Conference on IT*, 52(1):1–4, 2010.

[51] M. Rabin. Digitalized signatures and public key functions as intractable as factorization. *Massachusetts Institute of Technology Laboratory for Computer Science*, 38(1):1–4, 1979.

[52] M. Kakish. Enhancing the security of the rsa cryptosystem. *IJCSNS*, 8(2):239–246, 2011.

[53] Garbacki P. Epema D. Sips H. Pouwelse, J. The bittorrent p2p file-sharing system: measurement and analysis. *Proceeding IPTPS'05 Proceedings of the 4th international conference on Peer-to-Peer Systems*, 22(1):205–216, 2005.

[54] Isdal T. Anderson T. Krishnamurthy A. Venkataramani A. Piatek, M. Do incentives build robustness in bit torrent. *Proceeding NSDI'07 Proceedings of the 4th USENIX conference on Networked systems design and implementation*, 23(1):1–4, 2007.

[55] Agarwal S. Prasad R. Goel, I. A linear representation of huffman tree. *Proceedings of the International MultiConference of Engineers*, 1(1):246–248, 2005.

[56] Vasudevan S. Zhang, H. Coalitions improve performance in data swarming systems. *IEEE/ACM Transactions on networking*, 23(6):1790–1804, 2015.

[57] Rocha A. Li B. Towsley D. Venkataramani A. Menasche, D. Content availability and bundling in swarming systems. *IEEE/ACM Transactions on networking*, 21 (2):580–593, 2013.