Journal of Theoretical and Applied Information Technology
31st October 2017. Vol.95. No 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SECURE DATA SHARING MODEL FOR PRESERVING PRIVACY

[1]SANAA SARAHNEH, [2]RADWAN TAHBOUB

[1]Researcher, Deanship of Scientific Research, Palestine Polytechnic University, PS

[2]Asstt Prof., Department of Computer Engineering, Palestine Polytechnic University, PS

Networking Research Group (PPU)

E-mail: [1]sanaa.sarahneh@gmail.com, [2]radwant@ppu.edu

ABSTRACT

Data sharing is the process of interchanging, analyzing, retrieving and integrating data. Although data sharing facilitates the way that data can be exchanged, security concerns arise as a challenge for conducting data sharing. Many policies include confidentiality, integrity, availability and privacy must be taken into consideration. In this research, preserving privacy subject will be considered. This research 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 provide privacy to shared files also it reduces large files size since the proposed model uses Huffman coding and it is 87% faster on speed of 1MB/sec and on speed of 16MB/sec it is 38% faster than existing models.

Keywords: *Security, Data Sharing, Privacy, cryptography, compression.*

1. INTRODUCTION

Many policies include confidentiality and privacy must be taken into consideration [32]. However data sharing and integration are prevented from being widespread because of privacy concerns [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 [3–5].

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 [6, 7, 32]. Conducting business transactions is a basic reason for sharing data in e-commerce it is mainly used in Electronic Data Interchange (EDI) [4, 6]. This increases the need for data sharing management and data integration [1]. However data sharing and integration are prevented from being widespread because of privacy concerns. Privacy is defined as the process to protect information from unauthorized access [2]. 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 is an overview for OneSwarm data sharing model structure; it shows a P2P network with secure connection using SSL [2, 13].

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 research, an attempt to study and analyze existing data sharing models. The idea of using data compression such as Huffman coding to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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. This research aims to practice secure data sharing among different organizations focusing on preserving privacy.

Two basic encryption techniques are used to get secure system, symmetric key encryption and asymmetric key encryption [14, 15]. Rivest-Shamir-Adleman (RSA) cryptosystem is asymmetric key encryption algorithm [16].

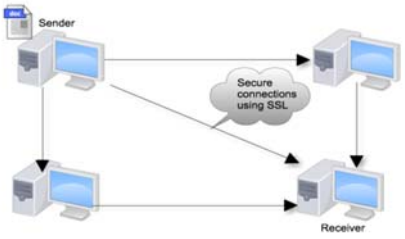

Figure 1: OneSwarm Structure [2].

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 are 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 decodes original file to get smaller file size [20–22].

2. BACKGROUND

A. 2.1 Data Sharing Concepts

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 [6, 7, 32]. 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, 32]. They clarify; e-government involves sharing data for transactions with citizens, other agencies and outside vendors and businesses [32]. The proposed data sharing model is a model that preserves privacy by using symmetric key encryption algorithm and data compression. The proposed model provides privacy to shared files also it reduces transmission time, and the size of the shared data. Figure 2 shows the main components for the proposed model.

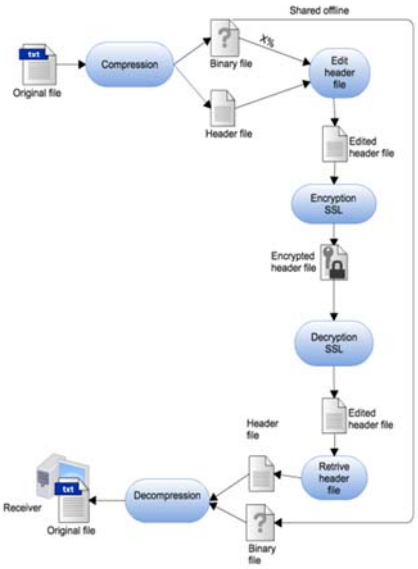

Figure 2: The Proposed Model Components.

2.2 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 lists different data sharing models [32].

2.2.1 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, 25]. as shown in figure 3.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

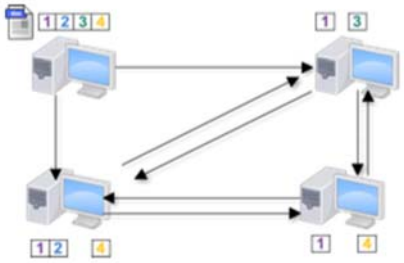

Figure 3: BitTorrent Overview.

2.2.2 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 in anonymizing overlays such as Tor [2, 13, 32].

OneSwarm made a tradeoff between privacy and performance, also it provides better privacy than BitTorrent [2, 13, 32]. 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].

OneSwarm protocol are managed by the DHT which contain of hashed IP and port, entries for a client encrypted with the public key. Naming and locating data in OneSwarm used Secure Sockets Layer (SSL) for connection [2, 32].

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 [26, 27]. It is important to use cryptography to secure communications between users or to transmit and store sensitive data [26].

2.3.1 Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) was developed by Netscape in 1994 to provide a secure transmission between applications [26]. SSL provides confidentiality by using symmetric key encryption such as AES and DES also it provides message integrity by using a Message Authentication Code (MAC), figure 4 shows the general processes for SSL.

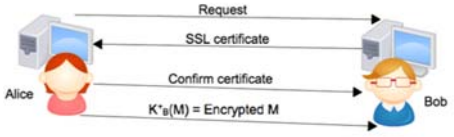

Figure 4: SSL Overview.

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 9, data are encrypted in 64-bit per block using short key 56-bit [28].

The Advanced Encryption Standard (AES) is a symmetric key encryption published by (NIST) in 2001 as shown in figure 10, it was a replacement for Data Encryption Standard (DES) [19].

2.3.2 RSA and A-RSA cryptosystems

In 1977, Ron Rivest, Adi Shamir and Leonard Adleman developed RSA as an encryption technique [17]. Although RSA is the most popular encryption method, it takes long time for encrypting and decrypting large data, also it is not semantically secure because encrypting the same message more than once gives the same ciphertext, which make it vulnerable to some attacks [17, 29].

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 5.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

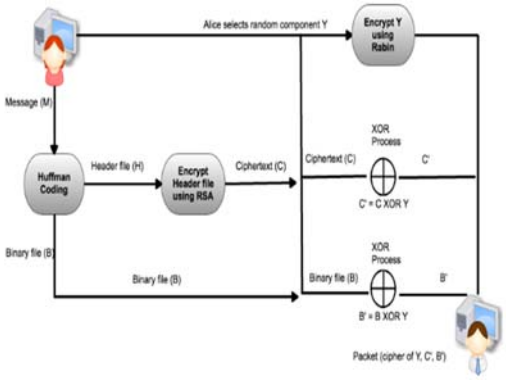

Figure 5: A-RSA Cryptosystem [17].

2.4 Data Compression and Coding

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 and reduce transmission time. Also data compression is used in security areas because compressed data is different than the original data [30]. It 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.

2.4.1 Huffman coding

Huffman coding is the most used of lossless data compression algorithm, which 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, 30]. 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. The header file contains a list of all characters and their frequency counts. Binary file and header file depend on each other's, if the header file does not exist we cannot recover the origin data [30].

3. PROPOSED DATA SHARING MODEL FOR PRESERVING PRIVACY

The generic proposed data sharing model was developed to provide privacy to the shared files without affecting the performance using compression and encryption. 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 6.

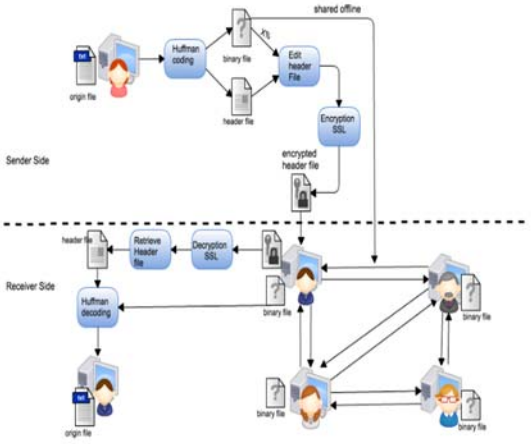

Figure 6: The Proposed Model Overview.

3.1 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,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

decryption algorithms. Also X% of data have been added to header file before encrypt 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.

3.1.1 Compression using Huffman coding

The compression algorithm used in this study was the Huffman code compression. It used to reduce size of data, transmission time and bandwidth [20]. Huffman coding generates two file binary file and header file. Binary file and header file depend on each other's, if the header file does not exist we cannot recover the original data. The header file size is fixed for all files sizes, it is 1KB.

3.1.2 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 such as direct 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 [31]. 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.

3.1.3 Privacy for the proposed model

The proposed data sharing model provides privacy to the shared data. 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]. Encryption algorithm to encrypt the header file. Also to improve security Cipher Block Chaining (CBC) is used. CBC makes the plain- text analysis difficult for the attacker comparing with other operation modes due to every block of the plaintext is dependent on the previous block [26]. 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 Computation and Transfer Time Parameters for Proposed Model

This section shows the total sharing time for the proposed model that uses Huffman coding as compression algorithm and encrypts the header file. The section shows the enhancement for the sharing time for the 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 the proposed model that using compression and encryption is expressed in equation 1.

$$T_{sharingHuff} = T_{computationHuff} + T_{trabsferHuff} \quad (1)$$

Where $T_{sharingHuff}$ is the sharing time when using Huffman coding as compression algorithm, $T_{computationHuff}$ is the computation time and $T_{trabsferHuff}$ is the transfer time. The computation time for compression, decompression using Huffman coding and encryption, decryption is:

$$T_{computationHuff} = T_{Huffcom} + T_{computation}(Binary) + T_{Enc}(Header) + T_{Dec}(Header) + T_{Huffdecom} + T_{Processing} \quad (2)$$

Where $T_{Huffcom}$ is the compression time using Huffman coding, $T_{computation}(Binary)$ is the computation time for the binary file, $T_{Enc}(Header)$ is the encryption time for the header file, $T_{Dec}(Header)$ is the decryption time for the header file at the receiver side, $T_{Huffdecom}$ is the decompression time at the receiver side using Huffman coding and $T_{Processing}$ is a processing time.

4. PROPOSED SYSTEM IMPLEMENTATION

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. The scenarios were tested on macOS Sierra version 10.12.3 with: processor: Intel Core i5, CPU 2.90 GHz, memory 8 GB and the system type is 64-bit operating system. Sharing files was tested between two users on two different networks with two different speeds (1MB/sec and 16MB/sec). The experiments were performed three times, average values are used to express the results.

5. EXPERIMENTS AND RESULTS

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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- ✓ 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, this research assumed that the X% of data is equal to 10%.
- ✓ 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, this research assumed that the X% of data is equal to 10%.

The computation time for AES (file), DES (file) and A-RSA(file) is:

$$T_{computation} = T_{Enc}(File) + T_{Dec}(File) + T_{Processing} \quad (3)$$

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

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

$$T_{computationHuff} = T_{Huffcom} + T_{Enc}(Header) + T_{Dec}(Header) + T_{Huffdecom} + T_{Processing} \quad (4)$$

Where $T_{Huffcom}$ is the compression time using Huffman coding, $T_{Enc}(Header)$ is the encryption time for the header file, $T_{Dec}(Header)$ is the decryption time for the header file at the receiver side, $T_{Huffdecom}$ is the decompression time at the receiver side using Huffman coding and $T_{Processing}$ is a processing time.

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

$$T_{computationEditHuff} = T_{Huffcom} + T_{computation}(Binary) + T_{Enc}(Header) + T_{Dec}(Header) + T_{Huffdecom} + T_{Processing} \quad (5)$$

Where $T_{computation}(Binary)$ is the computation time for the binary file.

Based on the previous equations, $T_{computationHuff}$ and $T_{computationEditHuff}$ will consumes more time than the $T_{computation}$ because of the compression time. Table 1 shows the average execution time for encrypting 10 files. Each file is encrypted 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 compression process. And Table 2 shows the decrypted and decompression process for each file in the mentioned scenarios. 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 1: Computation Time for compression and Encryption Process.

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
		Huffman	AES	Sum			Huffman	DES	Sum		
1MB	22	185	4	189	191	68	185	5	190	191	1043
2MB	38	353	4	357	360	134	353	5	358	364	1211
3MB	61	515	4	519	523	199	515	5	520	528	1373
4MB	70	680	4	684	690	263	680	5	685	696	1538
5MB	88	852	4	856	862	318	852	5	857	873	1710
6MB	104	1007	4	1011	1018	384	1007	5	1012	1028	1865
7MB	122	1166	4	1170	1178	448	1166	5	1171	1190	2024
8MB	133	1336	4	1340	1349	512	1336	5	1341	1365	2194
9MB	152	1492	4	1496	1506	567	1492	5	1497	1522	2350
10MB	165	1666	4	1670	1682	628	1666	5	1671	1699	2524

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Computation Time for Decompression and Decryption Process.

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
		Huffman	AES	Sum			Huffman	DES	Sum		
1MB	27	290	4	294	296	71	290	5	295	298	1620
2MB	42	863	4	867	869	137	863	5	868	1073	2393
3MB	62	1320	4	1324	1328	173	1320	5	1325	1333	3150
4MB	81	1880	4	1884	1916	230	1880	5	1885	1896	3910
5MB	99	2333	4	2337	2343	277	2333	5	2338	2355	4663
6MB	120	2980	4	2984	2791	344	2980	5	2985	3001	5410
7MB	140	3520	4	3524	3532	391	3520	5	3525	3545	6150
8MB	158	4143	4	4147	4156	442	4143	5	4148	4171	6973
9MB	183	4600	4	4604	4614	466	4600	5	4605	4629	7730
10MB	191	5243	4	5247	5258	505	5243	5	5248	5274	8473

Table 3: 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

Table 4: 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

The sharing time for the mentioned scenarios where is:

$$T_{sharing} = T_{computation} + T_{transfer} \quad (6)$$

Where $T_{sharing}$ is the sharing time, $T_{computation}$ is the computation time and $T_{transfer}$ is the transfer time.

Table 3 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 . While table 4 shows the sharing time for the ten files on 16MB/sec speed.

Figure 7 shows that sharing time on 1MB/sec speed when using compression and add X% data is 87% faster than sharing file without compression assuming that the binary file is shared offline.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 8 shows that sharing time on 16MB/sec speed when using compression and add X% is 38% faster than sharing file without compression assuming that the binary file is shared offline.

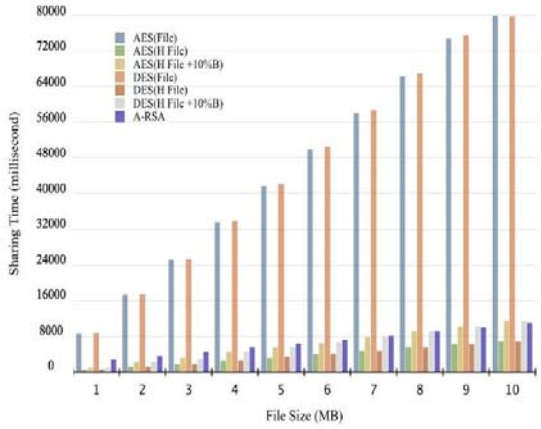

Figure 7: Sharing Time on 1MB/sec Speed.

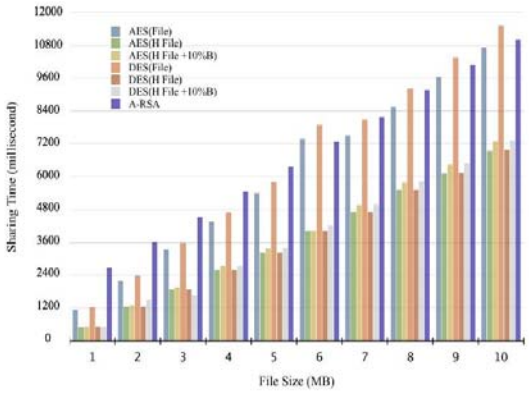

Figure8 : Sharing Time on 16MB/sec Speed.

6. DISCUSSION

Experiments result shows that the proposed model saves storage by 45% comparing 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. 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. In addition, the proposed model provides 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 plain text analysis difficult for the attacker comparing with other operation modes due to every block of the plaintext is dependent on the previous block [26].

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.
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.

7. CONCLUSION

In this work, preserving privacy subject will be considered. This research proposed a new privacy preserving data sharing model that combines compression techniques such as Huffman coding

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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 provide privacy to shared files also it reduces large file size since the proposed model use Huffman coding and it is 87% faster on speed of 1MB/sec and on speed of 16MB/sec it is 38% faster than existing models

REFERENCES:

[1] Doan A. Elmagarmid A. Clifton, C. Privacy-preserving data integration and sharing. ACM-Research issues in data mining and knowledge discovery, pages 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), 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, 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, 2013.

[8] Yang J. Hu, Y. A semantic privacy-preserving model for data sharing and integra- tion. 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, pages 235–246, 2007.

[10] B. Cohen. Incentives build robustness in BitTorrent. IPTPS, pages 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, 3178:67–82, 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, pages 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, pages 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.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[19] The National Institute of Standards and Technology. Announcing the advanced encryption standard (AES). Federal Information Processing Standards Publication 197, 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 Construc- tion, 16(5):586–595, 2007.

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

[25] Grunwald D. Sicker D. Bauer, K. The arms race in P2P. 37th Research Conference on Communication, Information and Internet Policy, 2009.

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

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

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

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

[30] D. Huffman. A method for the construction of minimum-redundancy codes. Proceedings of the I.R.E., pages 1098–1101, 1952.

[31] Shacham H. Modadugu N. Boneh D. Goh, E. Sirius: Securing remote untrusted storage. Proc. Network and Distributed Systems Security Symp, pages 131–145, 2003.

[32] Sarahneh S. Tahboub R. Secure Data Sharing Model Using New Technique for Preserving Privacy Journal of Internet Technology and Secured Transactions, 4(4) ,2015