TUGAS 1
RESUME ARTIKEL

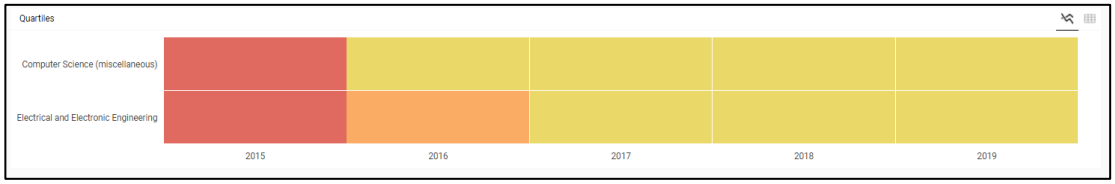
Files cryptography based on one-time pad algorithm

Disusun Oleh:
Dyan Azka Ingkafi
24060118130139

PROGRAM STUDI S1 INFORMATIKA
DEPARTEMEN ILMU KOMPUTER/INFORMATIKA
UNIVERSITAS DIPONEGORO
2021

INFORMASI ARTIKEL

1. Judul Artikel : Files cryptography based on one-time pad algorithm
2. Jurnal : International Journal of Electrical and Computer Engineering (IJECE)
3. Volume / Issue (Nomor) : 11 / 3
4. Halaman : 8
5. Scopus Quartile :



RESUME

Dewasa ini, enkripsi berperan penting di berbagai aspek, seperti militer, rahasia dagang, dan citra satelit. Enkripsi memungkinkan seseorang untuk menyembunyikan makna informasi atau pesan sehingga hanya mereka yang mengetahui metode rahasia yang dapat membacanya. Enkripsi saat ini dicapai dengan memanfaatkan algoritma yang memiliki kunci untuk mengenkripsi dan mendekripsi data. Secara prinsip, semakin panjang kunci maka semakin kompleks untuk memecahkan kode tersebut. Sebagai contoh, setiap unit biner data memiliki estimasi 0 atau 1. Sehingga, kunci 8-bit akan memiliki 256 potensial kunci sedangkan kunci dengan 56-bit akan memiliki 72 kuadriliun potensial kunci. Dengan semakin berkembangnya teknologi, sandi yang menggunakan kunci dengan panjang tersebut semakin mudah dipecahkan. Salah satu peningkatan luar biasa dalam penelitian kriptografi adalah penyajian kunci sandi asimetris yaitu algoritma yang memanfaatkan dua kunci secara matematis untuk mengenkripsi pesan yang sama. Oleh karena itu, pembentukan protokol baru yang dikenal sebagai *secure socket layer* (SSL), membuka jalan baru untuk transaksi *online* seperti proses pembelian, pembayaran tagihan *online,* dan perbankan*.*

One-time pad algorithm berasal dari sandi sebelumnya yang disebut *Vernam Cipher*. *Vernam Cipher* adalah *cipher* yang menggabungkan pesan dengan *keystream* yang dibaca dari pita kertas atau *pad*. *Pad* satu kali biasanya diterapkan dengan menggunakan tambahan modular (XOR) untuk menggabungkan elemen teks biasa dengan elemen aliran utama. Manfaat menggunakan operasi XOR adalah dapat dikembalikan hanya dengan mengimplementasikan operasi yang sama. Berikut adalah formula yang mengilustrasikan proses enkripsi dan dekripsi pada algoritma Vernam.

Enkripsi: $P \oplus k = C$;
Dekripsi: $C \oplus K = P$,

Dimana $\oplus$ menunjukkan operasi XOR, P mewakili teks biasa atau *plain text*, K mewakili kunci, dan C mewakili teks sandi atau *cipher text* . *Advanced Encryption Standard* (AES) adalah cabang dari *block cipher* Rijndael yang dibuat oleh dua kriptografer Belgia. Rijndael adalah sekelompok *cipher* dengan berbagai ukuran kunci dan blok. Algoritma yang digambarkan oleh AES adalah algoritma kunci-simetris, yang mana kunci serupa digunakan untuk mengenkripsi dan mendekripsi data. Kelemahan utama dalam menggunakan *one-time pad* adalah kunci enkripsi memiliki panjang yang sama dengan pesan yang ingin dienkripsi. Dengan demikian, pada artikel ini, kunci acak dibuat sebelum menerapkan algoritma Vernam kemudian dikompresi dengan pesan yang akan dienkripsi.

Pada tahap implementasi, sistem dibuat menggunakan bahasa pemrograman VB.Net. Langkah pertama dalam metodologi penelitian ini yaitu kunci acak enkripsi dibuat untuk mengenkripsi teks biasa menggunakan algoritma Vernam, kemudian kunci dengan versi terenkripsi disimpan dalam satu *file.* Kedua, algoritma AES digunakan untuk mengenkripsi

kata sandi, yang dianggap sebagai titik pemisah antara teks terenkripsi dan kunci enkripsi. Fase enkripsi mencakup dua algoritma (Vernam dan AES) dan menggunakan teknik steganografi sederhana untuk menyembunyikan data kunci kriptografi. Ketiga, ukuran *file* menjadi dua kali lipat karena kunci enkripsi dan data terenkripsi disimpan dalam file yang sama sehingga diimplementasikan kompresi data menggunakan algoritma Huffman.

Dengan demikian, metodologi ini telah mengatasi manajemen kunci enkripsi pada algoritma Vernam dengan mengontrol tipe data pada kunci enkripsi. Algoritma Huffman digunakan untuk mengurangi ukuran *file* keluaran. *File* keluaran dilindungi dengan kata sandi yang dienkripsi oleh algoritma AES, sehingga meningkatkan kesulitan dalam memecahkan *file* keluaran yang dienkripsi. Berbagai jenis *file*, seperti (.txt, .pdf, .doc, .bmp, .mp4, .exe), dilakukan untuk percobaan ini dan berhasil tanpa kehilangan informasi apapun. Lebih lanjut, hasil dari penelitian ini mendemonstrasikan bahwa waktu yang dihabiskan untuk enkripsi dan dekripsi *file,* yang mana dikompresi dengan kunci kriptografi yang dihasilkan dari *integer* memakan waktu lebih kecil dibandingkan kunci kriptografi yang dihasilkan dari tabel ASCII. Sementara itu, ukuran *file* berpengaruh kecil terhadap waktu enkripsi data tanpa kompresi. Sejak kriptosistem berperan penting dalam banyak aplikasi, tugas di masa yang akan datang adalah bagaimana mengeksplorasi perancangan sistem kriptografi yang aman pada kunci panjang enkripsi *one-time-pad.*

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 3, June 2021, pp. 2335~2342
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i3.pp2335-2342

Files cryptography based on one-time pad algorithm

Ahmad Mohamad Al-Smadi[1], Ahmad Al-Smadi[2], Roba Mahmoud Ali Aloglah[3], Nisrein Abu-Darwish[4], Ahed Abugabah[5]
[1,3,4]Department of Computer Science, Al-Balqa Applied University, Ajloun University College, Jordan
[2]School of Artificial Intelligence, Xidian University, China
[5]College of Technological Innovation, Zayed University, Abu Dhabi Campus, United Arab Emirates

Article Info

Article history:

Received Sep 5, 2020
Revised Sep 26, 2020
Accepted Dec 5, 2020

Keywords:

AES algorithm
Encryption
Huffman
Steganography
Vernam

ABSTRACT

The Vernam-Cipher is known as a one-time pad of algorithm that is an unbreakable algorithm because it uses a typically random key equal to the length of data to be coded, and a component of the text is encrypted with an element of the encryption key. In this paper, we propose a novel technique to overcome the obstacles that hinder the use of the Vernam algorithm. First, the Vernam and advance encryption standard AES algorithms are used to encrypt the data as well as to hide the encryption key; Second, a password is placed on the file because of the use of the AES algorithm; thus, the protection record becomes very high. The Huffman algorithm is then used for data compression to reduce the size of the output file. A set of files are encrypted and decrypted using our methodology. The experiments demonstrate the flexibility of our method, and it is successful without losing any information.

Corresponding Author:

Ahmad Mohamad Al-Smadi
Department of Computer Science, Al-Balqa Applied University
Ajloun University College,
Ajloun, Jordan
Email: amhs1966@bau.edu.jo

1. INTRODUCTION

In recent years, encryption is playing an essential role in many aspects, such as military, trade secrets and satellite imagery [1-6]. Encoding is the process of converting something in the physical world into a representation that can be stored or shared. Letters and words encode ideas and what is heard into a format that can be stored or shared. The goal of encoding is to deliver these ideas to their intended recipients. A person who does not understand the language, or does not know how to read it, will be unable to decode the information [7]. Encryption allows a person to hide the meaning of information or messages in such a way that only those who know the secret method may read them. For a very long time, people have had many different reasons for wanting to hide information from others. The earliest historical examples were for hiding trade secrets, military secrets, and secret correspondences between spies and lovers [8]. There were many ways that encryption tools were used in ancient times until the modern era saw the revolution of technology, and the concept of contemporary encryption emerged [9]. These same encryption principles are now used to safeguard your Internet communications [8].

Encryption in present-day has been achieving by utilizing algorithms that have a key to encrypt and decrypt data. These keys convert the data into "digital gibberish" via means of encryption and afterwards return them to the first structure through decryption [10]. Principally, the more extended the key is, the complex it is to break the code. An example of this would be, every binary unit of data has an estimation of 0 or 1. Therefore, an 8-bit key would then have 256 potential keys; a 56-bit key would have 72 quadrillion

prospective keys to attempt to decipher the message. With present-day innovation, cyphers utilizing keys with these lengths are getting simpler to break. DES, an early US Government, endorsed cypher, has a sufficient key length of 56 bits, and test messages utilizing that cypher, were broken. Further, as technology advances, so reflect on the aspect of encryption, one of the most remarkable improvements in the research of cryptography is the presentation of the asymmetric key cyphers, these are algorithms which utilize two mathematically related keys to encrypt the same message. Before the presentation of the advanced encryption standard (AES), most ordinarily, the data sent over the Internet, for example, financial information, were encrypted by utilizing the data encryption standard (DES) which was endorsed for a brief period, even though, witnessed extensive use [11, 12].

Therefore, the establishment of a new protocol known as the secure socket layer (SSL) [13], drew the way for online transactions to pass. Transactions were extending from the purchasing process to online bill pay and banking utilized SSL [14]. Besides, as wireless Internet connections turned out to be progressively essential among people, the necessity for encryption raised, as a level of security was needed in everyday situations. Data compression includes encoding information utilizing fewer bits than the original representation [15]. Compression can be grouped into two classifications, lossy or lossless. Lossless compression decreases bits by recognizing and disposing of measurable excess; consequently, no data are lost in lossless pressure. Conversely, Lossy compression diminishes bits by evacuating redundant or less significant data. Data compression is dependent upon a space-time complexity trade-off [16, 17].

The one-time pad algorithm is derived from a previous cipher called Vernam Cipher, named after Gilbert Vernam [17]. The Vernam Cipher was a cipher that combined a message with a key-stream read from paper tape or pad [18]. The unbreakable aspect of the one-time pad comes from two assumptions; the key-stream used is entirely random, and the key cannot be utilized more than once [18]. The security of the one-time pad depends on keeping the key 100% secret. The one-time pad is typically implemented by using a modular addition (XOR) to combine plain text elements with key stream elements. The key used for encryption is also used for decryption, applying the same key to the cipher text results back to the plain text. The cipher text is normally executed by utilizing the logical XOR operation to the individual bits of plain text and the key stream. The benefit of utilizing the XOR operation for this is that it can be reverted, simply via implementing the same operation again. The formulas (1) illustrate the encryption and decryption processes in Vernam algorithm:

$$\text{Encryption: } P \oplus k = C; \text{Decryption: } C \oplus K = P \quad (1)$$

where $\oplus$ indicates to XOR operation, P , K, and C represent the plain text, the key-stream, and the cipher text, respectively. The Advanced Encryption Standard AES is a branch of the Rijndael block cipher created by two Belgian cryptographers [19]. Rijndael is a group of ciphers with various key and block sizes. The algorithm portrayed by AES is a symmetric-key algorithm, which means a similar key is utilized for both encrypting and decrypting the data [20]. In software engineering and information theory, a Huffman code is a specific kind of ideal prefix code that is usually utilized for lossless data compression [21]. The Huffman's algorithm result may be adopted as a variable-length code table for encoding a source symbol. The algorithm derives this table from the estimated probability or recurrence of the event for each potential value of the source symbol, therefore, Huffman's technique can be proficiently executed [22].

Recently, Zaeniah *et al.* [21] presented an examination of encryption and decryption application by utilizing the one-time pad algorithm, to guarantee the information of the individuals who don't have the power to fill in the data, in which they exploited the statistical analysis utilized in the compression algorithms to acquire all the more efficient encryption key. Rishav Ray *et al.* introduced a scheme to encrypt texts using randomized data hiding algorithm with modified generalized Cipher Method [22]. Miyano *et al.* [23] proposed a one-time cushion cryptographic strategy utilizing a star system of N Lorenz subsystems, alluded to as expanded Lorenz conditions, which produces messy time arrangement as pseudo random numbers to be utilized for concealing a plain text. Abiodun *et al.* discussed the problem of the encryption key and the process of moving safely [24].

The primary disadvantage of encryption using the one-time pad is that the encryption key has a similar length as the message to be encrypted [24]. Thus, this paper, a random key is generated before applying the Vernam algorithm and is then compressed with the message to be encrypted. The biggest obstacle to the application and circulation of the Vernam algorithm on the data of considerable size, for instance, we assume that we want to encrypt 1 MB of the data; therefore, we need to 1 MB for the key, Key data between the sending person and the recipient. The problems of using this algorithm can be summarized as follows:

- Keys should not be reused.
- Keys sequences should not be repeated.

- Keys need to be shared somehow.

The modern cryptosystem concerns security criteria for data integrity, confidentiality, authentication, Non-Repudiation and reliability [25]. The main contributions of the proposed secure cryptosystem that overcome the problems that stand in the way of using the Vernam algorithm can be outlined as:

- A random key is generated with the option to be the key from the integers or ASCII table symbols.
- The encryption key is compressed with encrypted text to overcome the encrypted data size problem.
- The key is hidden with the same encrypted message to become a single encrypted file, to overcome the problem of transferring the encryption key to the recipient.
- Protect the encrypted file with a password encrypted with the AES algorithm to be used during the decryption process.
- Set the password hide point to reduce the decryption time.

The rest of this paper is organized as: The proposed methdology is presented in section 2. The experimental results and analysis are introduced in section 3. Finally, the main conclusions are presented in section 4.

2. METHODOLOGY

The flowchart of the proposed methodology is illustrated in Figure 1. In the stage of the implementation part, the VB.Net programming language is used to create the system. Figure 2 shows the Vernam system. Therefore, the main steps of our methodology, Firstly, a random encryption key is generated, to en-crypt the plain text with the Vernam algorithm, then the key with the encrypted version is stored in one file. Secondly, the AES algorithm is utilized to encrypt the password, which is considered the point of separation between the encrypted text and the encryption key. The encryption phase includes two algorithms (Vernam and AES) and using a simple steganography technique that hid the cryptographic key data. Thirdly, data compression is implemented since the file size has doubled due to the encryption key mode with encrypted data in the same file. Therefore, Huffman algorithm is used. The outlines of the encryption phase and the decryption phase are shown in Algorithm 1 and Algorithm 2, respectively.

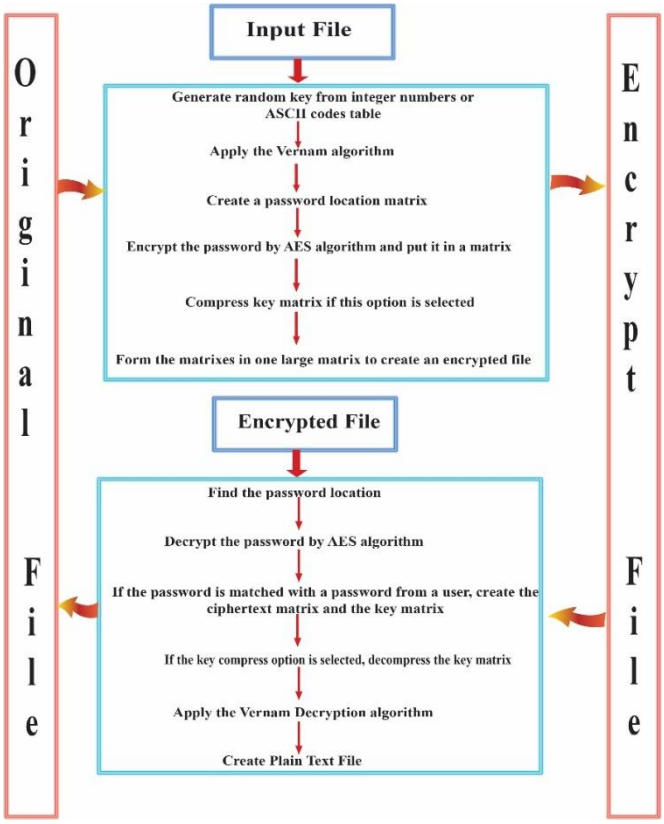


Figure 1. The flowchart of the proposed methodology

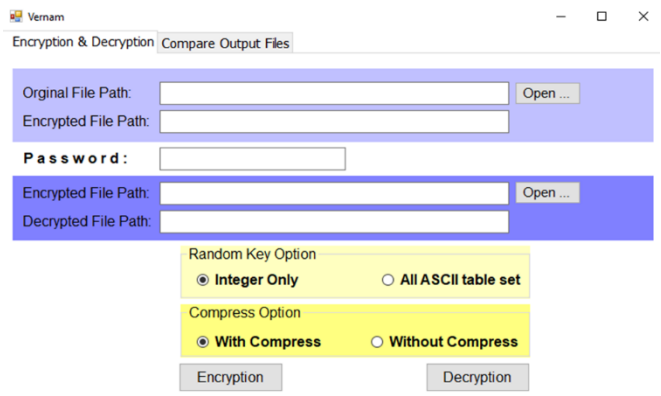


Figure 2. The Vernam system interface

Algorithm 1: The encryption process outlines
Input : Input file
Output : Encrypted file
1 Upload the file
2 Choose the encryption process options,
(i) The password to protect the file.
(ii) The option of processing (Random encryption key from the integer numbers only or the symbols of the ASCII table), and the option for compressing data or not.
3 Convert the input file into an array of bytes,
4 Coding with the Vernam algorithm, which is based on a XOR Operator in vb.net,
5 Create sub-matrices to be assembled into one large matrix that includes these matrices and be in the following order: (i) Matrix Location of the password, (ii) Matrix the encrypted text, (iii) Matrix the password (encrypted by AES algorithm), (iv) Matrix the random key which is used in Vernam encryption (compressed via Huffman algorithm),
6 Write the encrypted output file has the same extension of an input file.

Algorithm 2: The decryption process outlines
Input: Encrypted file
Output: Original file
1 Upload the encrypted file
2 Check the decryption options (Password, Random key option, Compress option) that should be the same options used in the encryption,
3 Convert the decrypted file into an array of bytes,
4 Obtain the first ten elements of the matrix containing the password location
5 Extract the password from its location and decrypt it with the AES algorithm
6 Compare the password extracted with the password that was typed by the user to follow the decryption process if they match,
(i) If the passwords match, then the encrypted text is divided into two parts, the encrypted text which is located at the left of the password, whereas, the right of the password indicates the encryption key which can be pressed according to the selected compression options.
(ii) Else; the password does not match.
7 If the encryption key is compressed, it is decompressed using the Huffman decompression algorithm,
8 After the random key matrix created, start a process of decryption with the Vernam algorithm,
9 Write the output file that contains the decoded text.

3. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, the program and proposed methodology have been conducted on a set of files of varying sizes and types; thus, the changing of the file size concerning the option of compression (with compression, or without) and the selected random keys (Integers, or ASCII table) was being studied. Figure 3 displays the implementation interface of our system. Further, the comparison of encryption time and decryption time regarding encryption key options and compression options were introduced for some files.

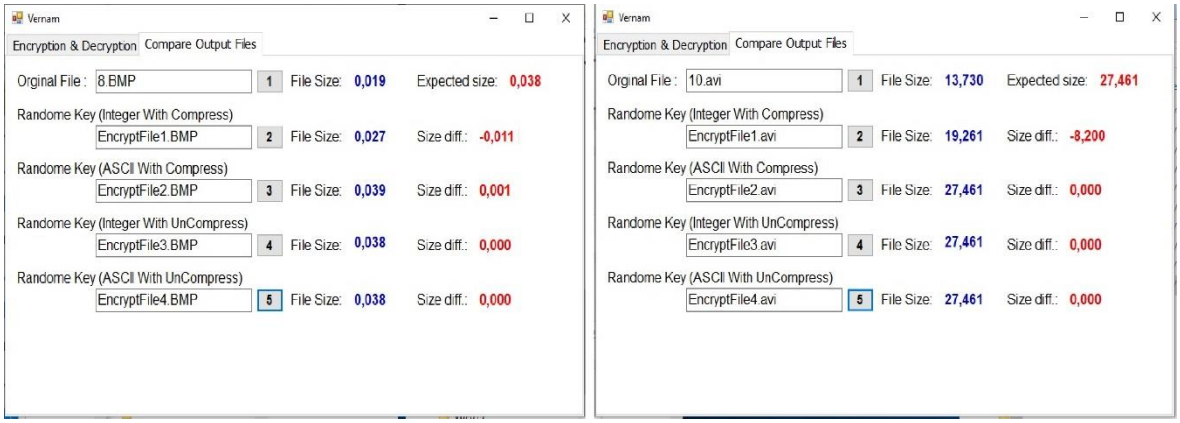


Figure 3. Screen-shots of the program interface for two selected files for testing

3.1. Experimental results via integer's random key

Here, we were able to achieve the highest compression rate of data, and the proportion of this compression 30% have exceeded this proportion in files, the results of this experiment are reported in Table 1. Besides, Figure 4. Illustrates the size difference between the expected file and the encrypted file after compression, concerning some various file types. Therefore, without utilizing data compressing, the output file size was equal to the expected file size of the encoder; thus, there is no difference between the expected file size and encrypted file size. The numerical results of these experiments are reported in Table 2.

Table 1. Numerical results of changes in the file size while using integers random key with compression

File Type	Size (KB)	Expected Size (KB)	Encrypted Size (KB)	Size Difference (KB)	Difference %
gif	6	13	9	4	31%
jpg	16	33	23	10	30%
bmp	19	38	27	11	29%
tiff	30	60	42	18	30%
png	56	112	78	34	30%
txt	6	12	8	3.6	30%
doc	99	199	139	60	30%
xls	386	772	541	231	30%
pdf	539	1,077	765	312	29%
mp3	2,996	5,992	4,203	1,789	30%
mp4	3,443	6,886	4,829	2,057	30%
webm	6,228	12,456	8,737	3,719	30%
m4v	9,553	19,106	13,401	5,705	30%
wmv	12,874	25,747	18,058	7,689	30%
avi	13,730	27,461	19,261	8,200	30%

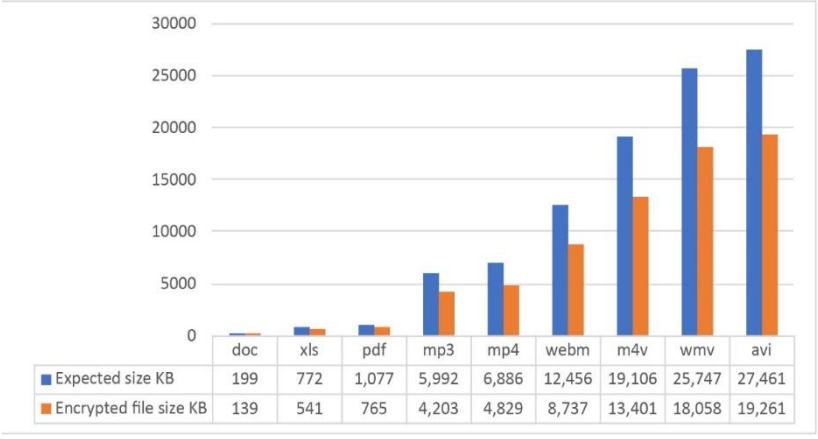


Figure 4. The size difference between the expected file and the encrypted file after compression, concerning some various file types; where the horizontal axis is the file's type and the vertical axis is the size difference

Table 2. Numerical results of changes in the file size while using integers random key without compression

File Type	Size (KB)	Expected Size (KB)	Encrypted Size (KB)	Difference %
gif	6	13	13	0
jpg	16	33	33	0
bmp	19	38	38	0
tiff	30	60	60	0
png	56	112	112	0
txt	6	12	12	0
doc	99	199	199	0
xls	386	772	772	0
pdf	539	1,077	1,077	0
mp3	2,996	5,992	5,992	0
mp4	3,443	6,886	6,886	0
webm	6,228	12,456	12,456	0
m4v	9,553	19,106	19,106	0
wmv	12,874	25,747	25,747	0
avi	13,730	27,461	27,461	0

3.2. Experimental results via ASCII table random key

In the same manner, we used the same files which were used in the previous experiments, but the ASCII table random key was used with data compression. This experiment did not achieve the compression of the data that we were expected. An example of this would be, some file types reached 0% at the size difference between expected and encrypted file sizes, and in others, reached under 0%. That means, the performance of the designed solution affected by a dictionary of the compressed file, the compressed data volume exceeded the expected size, albeit a small percentage. The reason behind is the primary function of compression algorithms is to create a dictionary of compressed file symbols and included it in the same file as well as the encryption key generated from the symbols of ASCII table, therefore, increased the size of the dictionary. Although we used data compression, we noted there are some file types, such as image files and text files that have a bigger size than the expected output files, as shown in Table 3. Moreover, we studied the effect on file size while using ASCII table random key without compressing the data, the numerical results of this experiment are shown in Table 4.

Table 3. Numerical results of changes in the file size while using ASCII table random key with compression

File Type	Size (KB)	Expected Size (KB)	Encrypted Size (KB)	Size Difference (KB)	Difference %
gif	6	13	13	0	0.000%
jpg	16	33	33	0	0.000%
bmp	19	38	39	-1	-2.632%
tiff	30	60	61	-1	-1.667%
png	56	112	112	0	0.000%
txt	6	12	12	0	0.000%
doc	99	199	199	0	0.000%
xls	386	772	772	0	0.000%
pdf	539	1,077	1,078	-1	-0.093%
mp3	2,996	5,992	5,993	-1	-0.017%
mp4	3,443	6,886	6,886	0	0.000%
webm	6,228	12,456	12,457	-1	-0.008%
m4v	9,553	19,106	19,107	-1	-0.005%
wmv	12,874	25,747	25,748	-1	-0.004%
avi	13,730	27,461	27,461	0	0.000%

Table 4. Numerical results of changes in the file size while using ASCII table random key without compression

File Type	Size (KB)	Expected Size (KB)	Encrypted Size (KB)	Size Difference (KB)	Difference%
gif	6	13	13	0	0%
jpg	16	33	33	0	0%
bmp	19	38	38	0	0%
tiff	30	60	60	0	0%
png	56	112	112	0	0%
txt	6	12	12	0	0%
doc	99	199	199	0	0%
xls	386	772	772	0	0%
pdf	539	1,077	1,077	0	0%
mp3	2,996	5,992	5,992	0	0%
mp4	3,443	6,886	6,886	0	0%
webm	6,228	12,456	12,456	0	0%
m4v	9,553	19,106	19,106	0	0%
wmv	12,874	25,747	25,747	0	0%
avi	13,730	27,461	27,461	0	0%

3.3. Encryption and decryption time study

In this subsection, we study the encryption time (Et), and decryption time (Dt), thus, the Et is utilized to compute the throughput of encryption (TE) which is calculated as follows:

$$TE = \frac{Tp(Kbytes)}{Et(ms)} \quad (2)$$

where Tp indicates the total plain text (File Size) in KiloBytes (KB) and Et denotes encryption time in milliseconds (ms).

According to the consumed time of the encryption process, as shown in Table 5. We can see that selecting the data encryption option with compression took longer due to the compression algorithm which takes some time; thus, the more the file size will increase the time taken, and vice versa is true. It was noted that the use of cryptographic key generation technology from integers only takes much less time, sometimes reached more than 50% compared with the ASCII table random key generation option. However, in terms of the prospect of encrypting data without compression, the results were very close, and the size of the file had little effect on data encryption time.

Overall, the throughput of encryption for encryption key without compression is higher than the throughput of encryption for integer key with compression. In the stage of the consumed time at decryption process, as shown in Table 6. We found that the data which were encrypted with a data compression algorithm took longer during the decoding process, compared to the data that were encrypted without compression. Data compressed with a cryptographic key generated from integers only took much less time during the decryption process, sometimes reached to 85% of those data encrypted with a cryptographic key generated from ASCII code symbols. In the option to decrypt data without compression, the file size has not a significant impact on data decryption time.

Table 5. The comparison of the Et process and the TE regarding encryption key options and compression

File	File Size (KB)	Et of Integer & with Compress ms.	TE for Integer with Compress (MB/s)	Et of Integer & without Compressed ms.	TE for Integer & without Compressed (MB/s)	Et of ASCII & with Compress ms.	TE for ASCII & with Compressed (MB/s)	Et of ASCII & without Compressed ms.	TE for ASCII & without Compressed (MB/s)
2.txt	4	0.212	18,867	0.215	18,604	0.331	12,084	0.326	12.269
calc.exe	27	0.224	120,535	0.318	84,905	0.425	63,529	0.329	82,066
Arabic.doc	100	0.427	234,192	0.328	304,878	0.646	154,798	0.315	317,460
1.txt	115	0.435	264,367	0.314	366,242	0.762	150,918	0.427	269,320
Report.pdf	539	1.84	292,934	0.328	1643,292	2.295	234,858	0.319	1689,655
1.bmp	583	1.88	310,106	0.332	1756,024	2.726	213,866	0.332	1756,024
VID.mp4	6697	12.354	542,091	0.533	12564,728	38.935	172,004	0.548	12220,802

Table 6. The comparison of the Dt process regarding encryption key options and compression options

File	File Size (KB)	Dt of Integer & with Compress ms.	Dt of Integer & without Compress ms.	Dt of ASCII & with Compress ms.	Dt of ASCII & without Compress ms.
2.txt	4	205	0.219	0,541	0.328
calc.exe	27	648	0.325	2,184	0.323
Arabic.doc	100	1,41	0.328	6,992	0.324
1.txt	115	1,751	0.323	8,843	0.330
Report.pdf	539	6,891	0.327	36,202	0.329
1.bmp	583	7,544	0.331	38,714	0.321
VID.mp4	6697	1,020,394	0.437	7,022,744	0.436

4. CONCLUSION

Cryptography is a multidisciplinary topic, which plays a vital role in many applications such as network security, the privacy of information and communications. A cryptosystem becomes worthless if poorly managed and improperly carried out. Therefore, this paper proposed a practical methodology for files cryptography based on the one-time pad algorithm. Thus, this methodology has overcome the encryption key management of the Vernam algorithm by controlling the data type in the encryption key. The Huffman algorithm reduced the size of the output file. The output file was protected with password encrypted by the AES algorithm, therefore increased the difficulty of breaking the encrypted output file. Various types of files, such as (txt, pdf, doc, bmp, mp4, exe), were carried out for successful experiments without losing any information. Further, the outcome of this research demonstrated that the time consumed for encryption and decryption the file, which compressed with a cryptographic key generated from integers less than when the

cryptographic key generated from the ASCII table. Whereas, the file's size had a little effect on data encryption time without compression. Since the cryptosystem has been playing a vital role in many applications. Future work would be how to explore designing a secure cryptosystem on the long key of encryption one-time-pad.

REFERENCES

[1] A. M. Qadir and N. Varol, "A Review Paper on Cryptography," *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, Barcelos, Portugal, 2019, pp. 1-6.

[2] R. F. Abdel-Kader, et al., "Efficient two-stage cryptography scheme for secure distributed data storage in cloud computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 3295-3306, 2020.

[3] A. J. Abboud, et al., "Balancing compression and encryption of satellite imagery," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 5, pp. 3568-3586, 2018.

[4] A. P. Pljonkin, "Vulnerability of the Synchronization Process in the Quantum Key Distribution System," *International Journal of Cloud Applications and Computing*, vol. 9, no. 1, pp. 50-58, 2019.

[5] A. Pljonkin and P. K. Singh, "The review of the commercial quantum key distribution system," *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan Himachal Pradesh*, India, 2018, pp. 795-799.

[6] A. E. Omolara and A. Jantan, "Modified honey encryption scheme for encoding natural language message," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 3, pp. 1871-1878, 2019.

[7] R. E. Endeley, et al., "End-to-End Encryption in Messaging Services and National Security-Case of WhatsApp Messenger," *Journal of Information Security*, vol. 9, no. 01, pp. 95-99, 2018.

[8] J.F. Dooley, "History of Cryptography and Cryptanalysis," *Springer International Publishing*, 2018.

[9] A. Joseph and V. Sundaram, "Cryptography and steganography-A survey," *International Journal of Computer Technology and Applications*, vol. 2, no. 3, pp. 626-630, 2011.

[10] A. A. Labaichi, et al., "Image steganography using least significant bit and secret map techniques," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 935-946, 2020.

[11] R. Bhandari and V. B. Kirubanand, "Enhanced encryption technique for secure iot data transmission," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3732–3738, 2019.

[12] P. Liu, et al., "Efficient verifiable public key encryption with keyword search based on KP-ABE," *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications, Guangdong*, 2014, pp. 584-589.

[13] S. Duddu, et al., "Secure Socket Layer Stripping Attack Using Address Resolution Protocol Spoofing," *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2020, pp. 973-978.

[14] G. Saranya, "An efficient data hiding method in images," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 4713-4720, 2019.

[15] A. T. Hashim and B. D. Jalil, "Color image encryption based on chaotic shit keying with lossless compression *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 5736-5748, 2020.

[16] S. R. Kodituwakku and U. S. Amarasinghe, "Comparison of lossless data compression algorithms for text data," *Indian Journal of Computer Science and Engineering*, vol. 1, no. 4, pp. 416-425, 2010.

[17] P. Gleichauf, "Method and system for securely storing and trasmitting data by applying a one-time pad," Google Patents, 2003.

[18] G. Renuka, et al., "Comparison of AES and des algorithms implemented on virtex-6 FPGA and Microblaze soft core processor," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 5, pp. 3544-3549, 2018.

[19] A. Kaur, P. Bhardwaj, and N. Kumar, "FPGA implementation of efficient hardware for the advanced encryption standard," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 2, no. 3, pp. 186-189, 2013.

[20] M. E. Hameed, et al., "An enhanced lossless compression with cryptography hybrid mechanism for ECG biomedical signal monitoring," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 3235-3243, 2020.

[21] B. E. P. Zaeniah, "An Analysis of Encryption and Decryption Application by using One Time Pad Algorithm," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 9, 2015.

[22] R. Ray, et al., "A new randomized data hiding algorithm with encrypted secret message using modified generalized Vernam Cipher Method: RAN-SEC algorithm," *2011 World Congress on Information and Communication Technologies*, Mumbai, 2011, pp. 1211-1216.

[23] P. Garg, et al., "An Analysis of Encryption and Decryption Application by using One Time Pad Algorithm," *2014 Ninth Int. Conf. Broadband Wirel. Comput. Commun. Appl.*, vol. 1, no. 4, pp. 6–12, 2011.

[24] A. E. Omolara, et al., "An enhanced practical difficulty of one-time pad algorithm resolving the key management and distribution problem," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2018.

[25] B. Carpentieri, "Efficient compression and encryption for digital data transmission," *Security and Communication Networks*, vol. 2018, 2018.