ICIC Express Letters
Volume **18**, Number **10**, October **2024**

ICIC International ©2024 ISSN 1881-803X
pp. **1109–1116**

ENHANCING DATA SECURITY BY COMBINING DATA COMPRESSION AND ENCRYPTION

Narut Butploy[1,*], Kanokwan Khiewwan[1], Silnarong Chaviphat[2] and Jindaporn Ongate[2]

[1]Faculty of Industrial Technology
[2]Faculty of Science and Technology
Kamphaeng Phet Rajabhat University
69 M.1, Nakorn Chum Muang, Kamphaeng Phet 62000, Thailand
{ kanokwan_kh; silnarong; jindaporn_o }@kpru.ac.th
*Corresponding author: narut@kpru.ac.th

Received January 2024; accepted March 2024

Abstract. *This research investigates the integration of the Huffman algorithm for data compression with the Data Encryption Standard (DES) in order to enhance data security while minimizing the size of encrypted files. Specifically, the study focuses on the application of these techniques to student data within Kamphaeng Phet Rajabhat University in Thailand. The Huffman algorithm, renowned for its lossless compression method, effectively reduces data size by assigning shorter codes to frequently occurring characters, ensuring efficient compression without compromising accuracy. In parallel, the DES algorithm, widely recognized as an encryption technique, employs a 64-bit encryption key to transform plaintext into ciphertext. Experimental results reveal the successful encryption and decryption of student data, wherein the encrypted files demonstrate reduced sizes while maintaining data integrity. Through the combination of data compression and encryption techniques, data security is significantly enhanced, while also offering the potential for streamlined data transmission. This research contributes to the field by showcasing the effectiveness of integrating the Huffman algorithm and DES encryption, resulting in heightened data security coupled with reduced data size. Furthermore, experimentation was conducted on six distinct datasets. The results indicate that the size of encrypted files exhibited smaller file sizes, ranging from 36.11% to 36.88%, in comparison to uncompressed files. Specifically, the proposed approach not only contributes to data security but also effectively reduces the size of encrypted files.*

Keywords: Data compression, Data encryption, Data security, Student data protection

1. **Introduction.** The rapid advancement of computer technology has led to the integration of various technologies into our daily lives, increasing the risk of data breaches and theft. The widespread use of portable devices like smartphones, coupled with high-speed Internet networks, has facilitated various work and personal activities. Cloud storage systems have also allowed users to store and access large amounts of information without the need for physical storage devices [1,2]. However, this interconnectedness and data transfer through the Internet present significant security and privacy challenges in communication [3-5]. Cybercrime, which costs billions of dollars annually, is a major challenge in this digital landscape. To address these challenges, encryption has emerged as a common method used to protect information. Encryption converts plaintext into ciphertext, making it difficult for unauthorized parties to interpret stolen data. The encryption process consists of two stages: encryption, which transforms plaintext into ciphertext, and decryption, which reverses the process to retrieve the original plaintext [7,8]. One drawback of data encryption is the increase in file size, as each character requires an 8-bit binary representation. To overcome this challenge, data compression techniques have been explored

DOI: 10.24507/icicel.18.10.1109

to reduce data size [9]. Data compression has proven effective in reducing storage costs and optimizing data transmission. Combining data compression with encryption offers interoperability and reversibility, emphasizing the importance of comprehensive data security [10]. Research has shown that the integration of the Huffman algorithm, which alternates encryption cycles and data compression, results in smaller data sizes and higher encryption reliability [11]. Other encryption algorithms like Advanced Encryption Standard (AES), Blowfish, and Data Encryption Standard (DES) can also be combined with data compression to further reduce data size [12]. This research investigates the integration of the Huffman algorithm for data compression with the Data Encryption Standard (DES) using a 64-bit encryption key. The objective is to enhance data security while reducing the size of encrypted data. The research focuses on data security within the context of a digital university, where the university's information is stored in the cloud system. Given the sensitive nature of student data, it is crucial to study encryption techniques before bringing such data online to ensure confidentiality and protection. The integration of data compression and encryption techniques offers several advantages. Firstly, converting data into binary numbers makes it more challenging for unauthorized individuals to decipher the information, resulting in smaller encrypted data sizes. Secondly, data encryption opens up avenues for further research, such as exploring techniques for faster data transmission through reduced data sizes. To evaluate the effectiveness of the proposed approach, comprehensive tests were conducted, yielding satisfactory results. The data retrieval process demonstrated the complete and accurate restoration of the original data, effectively strengthening data security. This research contributes to the field by showcasing the effectiveness of integrating the Huffman algorithm for data compression with the Data Encryption Standard (DES) and providing insights into its potential benefits. By focusing on the security protection of students' confidential data within Kamphaeng Phet Rajabhat University in Thailand, this research fills a gap in the existing literature and advances knowledge in the field of data security. The structure of the paper is as follows. Section 2 provides an overview of the literature review, followed by Section 3 outlining the method and experiment. Section 4 presents a comprehensive analysis of the results and includes a discussion of the findings. Finally, the conclusion summarizes the research and its contributions to the field.

2. **Literature Review.**

2.1. **Definition of data compression.**

Data compression is an encryption process that aims to reduce data size for more convenient storage and efficient transmission across networks [13]. There are two types of data compression: lossless and lossy. Lossless compression ensures no data loss during encryption, is suitable for accurate data, and utilizes algorithms like Huffman and Lempel-Ziv (LZ) [14]. Lossy compression allows some data loss while preserving essential information, used in formats like MPEG, JPEG, and MP3. Compression is achieved by identifying data occurrences and combinations through information theory, formal grammars, and artificial intelligence [15].

2.2. **The Huffman algorithm.**

The Huffman algorithm is an efficient lossless data compression technique that assigns shorter binary codes to frequently occurring symbols in a dataset [16]. It constructs an optimal prefix code based on symbol frequencies, ensuring unambiguous decoding. The process involves analyzing the dataset's symbols, constructing a Huffman tree, and mapping symbols to shorter binary codes for effective compression. Despite some limitations related to character frequencies, the combination of Huffman coding and the stream cipher algorithm is a suitable choice for the Joint Compression and Encryption (JCAE) scheme due to low computational cost and preserving essential data characteristics [17].

2.3. **Definition of data encryption.** Data encryption is an effective method that converts data into an unreadable form, ensuring only authorized users at the destination can access it [18]. It involves transforming plaintext into ciphertext during encryption and decoding it back to plaintext during decryption [18]. If the ciphertext is compromised, it remains inaccessible and indecipherable, represented in binary numbers [19].

2.4. **The DES algorithm.** The Data Encryption Standard (DES) is a widely recognized symmetric encryption algorithm operating on 64-bit data blocks using a 56-bit key for encryption and decryption [20-22]. Key generation is crucial, starting with a 64-bit key, of which 56 bits are used, and 8 bits are for parity checks [23]. The 56-bit key provides a balance between security and efficiency. It generates 16 round keys through a process called key scheduling, involving permutation and rotation operations. The initial permutation rearranges the input data based on the Initial Permutation (IP) table, distributing data for robust mixing. The DES algorithm, despite its widespread adoption, has limitations due to its key length, making it unsuitable for modern security requirements. Nevertheless, understanding DES is essential for appreciating contemporary encryption techniques' development.

3. Method and Experiment.

3.1. **Dataset.** This research used students' data of the Computer Technology Program, Faculty of Industrial Technology, Kamphaeng Phet Rajabhat University, Thailand. The data was taken from the registrar's website, including personal information, enrolment, and academic records. The students' data retrieved from the registrar's site were tabulated, and the students' identification numbers were used as a referential index. The original data were converted to a comma file format, as presented in Table 1.

Table 1. Dataset

std_id	std_name	Semester	...	grad6	grad7
601321001	...	1/2019	...	C+	C
601321002	...	1/2019	...	D+	D+
...	...	...	...	...	...
631321025	...	2/2020	...	B	B+

3.2. **Run time.** Besides the encryption, run time is also a factor to consider. Hence, it is crucial to test the run time of the DES algorithm. The researcher tested the encryption and decryption run time using a computer with Intel (R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 16 GByte main memory, and Windows 10 operating system. The program was developed by the C and java programming language. The encryption and decryption of different file sizes were timed. The original files were in a CSV format of 10 KByte, 12 KByte, 14 KByte, 16 KByte, 18 KByte, and 20 KByte.

3.3. **Encryption without data compression.** The program read files in a CSV format. For example, the data in the file contained the following information: "617040xxx, Mr. Narut, Programming, Database, B+, B". Then the program converted all of the information into the hexadecimal ASCII code to obtain referential numbers for the new binary number conversion as 2c4d722e4e617275742c50726f6772616d6d6973652c422b2c42. An ASCII sample code for the number 6 is 36, 1 is 31, M is 4d, and r is 72. Next, eight characters were read per reading time to convert them into binary numbers of 64 bits. For instance, the data '61704xxx' was converted to 36 31 37 30 34 30 78 78 78. After that, these numbers were encrypted to binary numbers as 1001 1001 1001 1010 1011 1011 1011 0010 1011 0000 1001 0010 1001 1100 1011 1110 0010 1011 1010 1001 1111 1110 0111 0111 0000 1101 1100 1110 1011 1000 0110 0110.

The total data generated from encryption were 384 bits. The binary-number set was downloaded for the DES algorithm. The processing was repeated until all alphabets in each row and every row in the file were encrypted. Finally, all of the binary numbers were written in the file. Figure 1 illustrates the proposed method, wherein the researcher queries the database to retrieve the necessary data for creating a dataset to be used in the experiment. The information is stored in a Comma-Separated Values (CSV) file. The experiment consists of two distinct phases. The first phase involves applying uncompressed DES encryption. In the second phase, the dataset is compressed using the Huffman technique, resulting in a new binary-formatted dataset. This dataset is subsequently examined to ensure that when modulated with 64, the size of the data produces a fraction of 0. If not, the necessary number of zeros must be added to complete the bits. For instance, when compressing the data to obtain a dataset of 570 bits, an additional 6 zeros must be appended to achieve a total of 576 bits, and so on. Following compression, the resulting file is encrypted using the DES technique. Both phases of the experiment collect relevant data, including encoding time, decoding time, as well as the sizes of the compressed and uncompressed datasets.

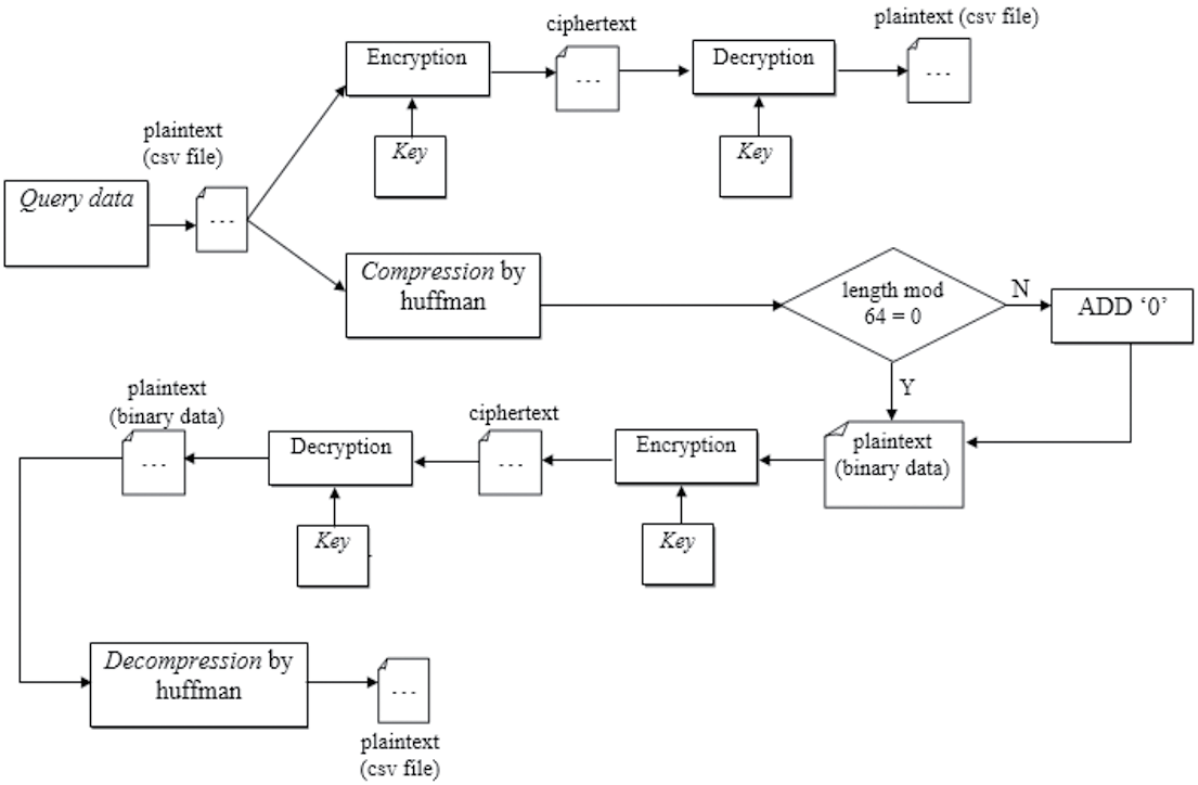

FIGURE 1. The proposed method

3.4. **Encryption with pre-data compression.** The program read a CSV file format before encryption. For example, the file contained the following information: "617040027, Mr. Narut, Programming, Database, B+, B". Then, the program counted the frequency of each character and generated the binary tree. According to the example file, the frequency codes were as follows: 0=1010 1=101111 2=111010 4=111000 6=01010 P=111011 r=000. The total compressed codes were 0101 0101 1111 1000 1010 1110 0010 1010 1011 1010 1100 0011 1011 1000 1010 0011 0111 1111 0110 0000 1111 0010 1100 1111 1011 0000 1001 1100 1000 1001 1010 1101 0001 1111 0110 1100 1011 0010 0100 1011 0100 1110 0110 0010 0001 0110 1111 1110 0110 0111 1111, which were 204 bits. Then the 204-bit length was moded with the 64 bits to find the amount of bit loss and added 0 to fill up the 64 bits

for the DES algorithm. In this case, an addition of 0 for 52 bits was therefore required. In total, the file had 256 bits for the DES algorithm.

4. Main Results.

4.1. Experimental results of encryption without data compression.

The accuracy of the data was assessed by the researcher through the decryption of the encrypted data. The decryption process demonstrated precise results, confirming that the decrypted data remained identical to the original data. Figure 2(a) depicts the outcome of the encryption, highlighting the generation of 64-bit data in each processing round. Subsequently, each bit was recorded in a file to produce the encryption result file. Additionally, Figure 2(b) illustrates the decryption result represented as binary numbers. Notably, the final permutation of 64 bits was generated, and subsequently, each group of eight bits was divided for hexadecimal conversion prior to transformation into characters.

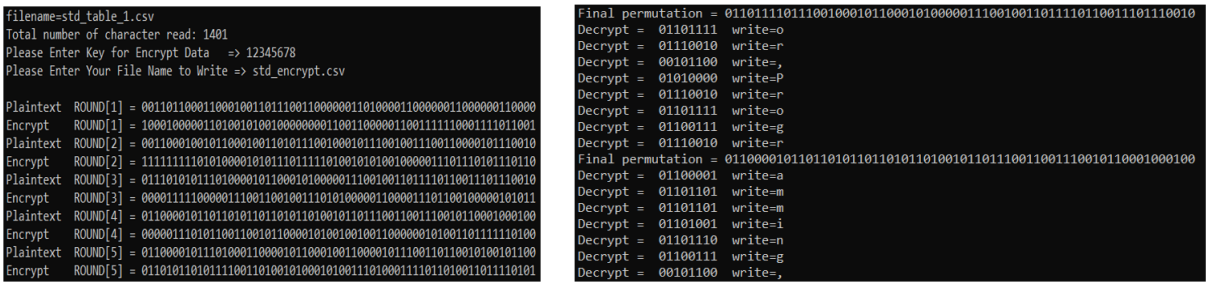

FIGURE 2. Results of encryption without data compression

4.2. Experimental results of the DES algorithm with data compression.

Table 2 and Table 3 provide a comparison of file sizes and processing times for encrypted files using the Data Encryption Standard (DES) algorithm, with and without the application of Huffman data compression (EWC and EWO, respectively). The utilization of Huffman data compression resulted in smaller binary-number files across all experimental file sizes. Figure 2 presents an illustrative example of the data compression process, wherein each character is assigned a corresponding code, subsequently linked together based on their respective positions. During the data compression stage, the data is converted into binary numbers, which subsequently serve as plaintext for the DES algorithm, without any mutual interference.

TABLE 2. Sizes of data encrypted and sizes of compression (Kbyte)

	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6
Original	10	12	14	16	18	20
EWC	46	57	65	77	89	100
EWO	72	91	104	122	141	159

TABLE 3. Time of encryption and decryption (Sec)

	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6
EWC	0.358	0.397	0.438	0.486	0.522	0.576
DWC	0.359	0.397	0.438	0.491	0.525	0.596
EWO	0.483	0.538	0.674	0.729	0.834	0.522
DWO	0.309	0.315	0.362	0.401	0.421	0.455

Figure 3 presents the results obtained from encryption with data compression. The utilization of the Huffman code facilitates a reduction in file size by assigning codes to characters based on statistical data. Consequently, repeated characters are assigned the same code, eliminating the need for generating new codes. Conversely, the DES algorithm without data compression necessitates assigning a new code to each character, even in cases of character repetition. Figure 4 demonstrates the generation of smaller file sizes, namely 36.11%, 37.36%, 37.50%, 36.89%, and 36.88%. In contrast, encrypted files generated by the DES algorithm without data compression exhibit larger sizes due to the encryption and conversion of each character into a 64-bit binary number. During the data writing process, the program converts the binary numbers back into characters. Consequently, each character is converted, encrypted, and written in the form of 0 s or 1 s for a total of 64 characters. Considering a file comprising 10,000 characters, this would result in the encryption and writing of 80,000 characters in the 0 or 1 format.

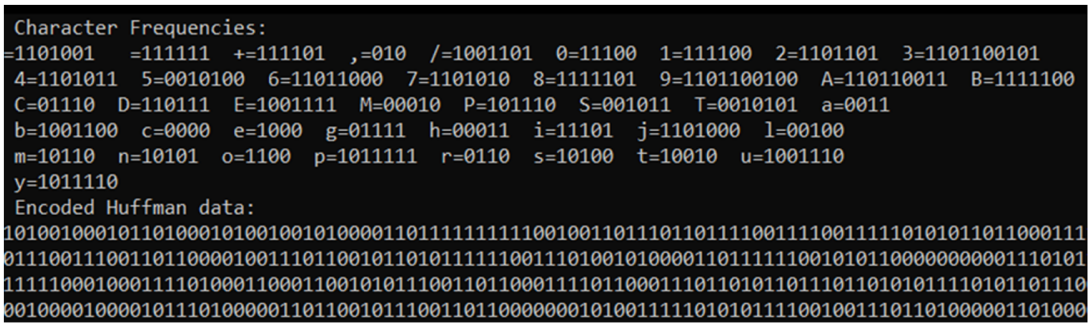

FIGURE 3. Results of encryption with data compression

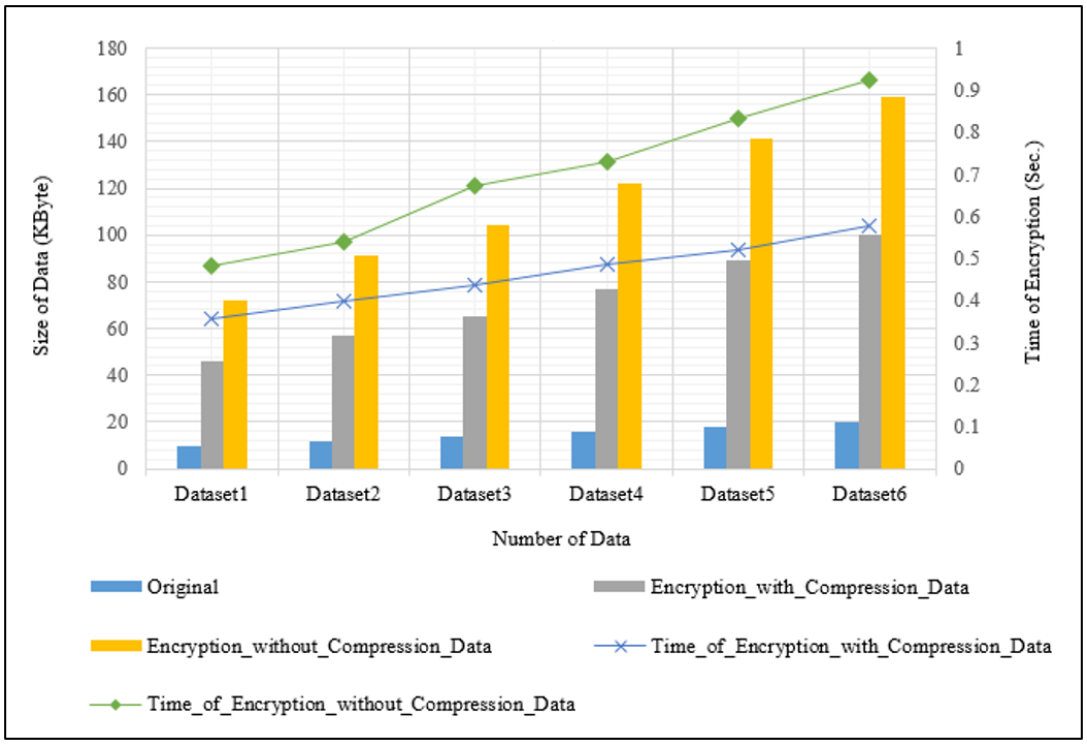

FIGURE 4. Data size and time of encryption with compression and without compression

5. **Conclusions.** The research aims to contribute to the field by showcasing the effectiveness of integrating the Huffman algorithm for data compression with the Data Encryption Standard (DES) and providing insights into its potential benefits. By focusing on enhancing the security protection of students' confidential data within Kamphaeng Phet Rajabhat University in Thailand, this research fills an existing gap in the literature and advances knowledge in the field of data security. By considering the findings from [24,25] alongside the results discussed in this study, a comprehensive understanding of the advantages of combining encryption with compression can be gained. This highlights the significance of the proposed method in enhancing both data security and usability. The study suggests that the combination of encryption with compression not only significantly enhances data security but also effectively reduces file size compared to using the DES algorithm without data compression. The present research applies the Huffman algorithm to compressing the original data before encryption with the DES algorithm, resulting in a file size reduction of approximately half in every experimental file. Despite the need to add an additional "0" to meet the 64-bit requirement for the DES algorithm, the encrypted files generated with data compression are still smaller than those encrypted solely with the DES algorithm. These findings contribute to the ongoing research in the field of data encryption and compression, providing valuable insights into the benefits of the proposed method. By combining encryption with compression, data security is strengthened while maintaining usability. It is important to note that the data receiver must possess a frequency table, which presents a challenge requiring further investigation and solutions. Future research should focus on addressing this challenge to fully exploit the potential of combining encryption with compression. In summary, this research contributes to the field by demonstrating the effectiveness of integrating the Huffman algorithm with the DES algorithm for data compression and encryption. It fills a gap in the literature and advances knowledge in data security, particularly in protecting students' confidential data. The findings highlight the advantages of combining encryption with compression, including enhanced security and reduced file sizes. However, further research is needed to address challenges related to the frequency table requirement for data receivers, thus fully realizing the potential benefits of this approach.

Acknowledgment. This work is partially supported by Kamphaeng Phet Rajabhat University, Thailand.

REFERENCES

[1] D. KR Shukla, Vijay K.R. Dwivedi and M. C Trivedi, Encryption algorithm in cloud computing, *Materials Today: Proceedings*, vol.37, pp.1869-1875, 2021.

[2] A. Yousefi and S. M. Jameii, Improving the security of Internet of Things using encryption algorithms, *International Conference on IoT and Application (ICIOT)*, 2017.

[3] R. Vijayan and R. S. Sreedivya, Cryptographic-steganography network communication, *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, 2017.

[4] R. A. Zitar and H. Hussar, A mirroring neural network approach for encryption/decryption of data, *ICIC Express Letters*, vol.13, no.11, pp.1057-1064, 2019.

[5] T. Adamski and W. Nowakowski, Security of Nyberg-Rueppel digital signatures without message recovery, *Bull. Pol. Acad. Sci. Tech.*, vol.62, no.4, pp.817-825, 2014.

[6] M. Strachacki and S. Szczepański, Power equalization of AES FPGA implementation, *Bull. Pol. Acad. Sci. Tech. Sci.*, vol.58, no.1, pp.125-128, 2010.

[7] J. Chen, J. Tang, F. Zhang, H. Ni and Y. Tang, A novel digital color image encryption algorithm based on a new 4-D hyper-chaotic system and an improved S-box, *International Journal of Innovative Computing, Information and Control*, vol.18, no.1, pp.73-92, 2022.

[8] I. Polak and M. Boryczka, Tabu search against permutation based stream ciphers, *International Journal of Electronics and Telecommunications*, vol.64, no.2, pp.137-145, 2018.

[9] K. A. Nor, J. B. Asral and I. C. Rizalafande, Performance evaluation of arithmetic coding data compression for Internet of Things applications, *Indonesian Journal of Electrical Engineering and Computer Science*, vol.13, no.2, 2019.

[10] E. Setyaningsih and R. Wardoyo, Review of image compression and encryption techniques, *International Journal of Advanced Computer Science and Applications*, vol.8, no.2, pp.83-94, 2017.
[11] M. Usama, Q. M. Malluhi, N. Zakaria et al., An efficient secure data compression technique based on chaos and adaptive Huffman coding, *Peer-to-Peer Networking and Applications*, vol.14, no.3, pp.1-14, 2021.
[12] F. Osama, H. Aziza, H. Hesham, K. Hamdy and K. A. M. Ashraf, Efficient combination of RSA cryptography, lossy, and lossless compression steganography techniques to hide data, *Procedia Computer Science*, 2021.
[13] Z. Bohdan, B. Juliy, D. Vladymir, Z. Irina and E. Oleksander, Increasing the efficiency of information transmission in communication channels, *Indonesian Journal of Electrical Engineering and Computer Science*, vol.19, no.3, pp.1306-1315, 2020.
[14] M. Rodziewicz, Joint source-channel coding in dictionary methods of lossless data compression, *International Journal of Electronics and Telecommunications*, vol.56, no.4, pp.351-355, 2010.
[15] R. Boris and S. Nadezhda, Using data compression to build a method for statistically verified attribution of literary texts, *Entropy*, 2021.
[16] L. Yong, L. Bing, Z. Yan and Z. Xia, A Huffman-based joint compression and encryption scheme for secure data storage using physical unclonable functions, *Electronics*, 2021.
[17] E. Erdal and E. Atilla, An efficient encoding algorithm using local path on Huffman encoding algorithm for compression, *Applied Science*, 2019.
[18] K. Algazy, L. K. Babenko, R. Biyashev, E. Ishchukova, R. S. Romaniuk, N. Kapalova, A. Smolarz and S. Nysynbaeva, Differential cryptanalysis of new Qamal encryption algorithm, *International Journal of Electronics and Telecommunications*, vol.66, no.4, pp.647-653, 2020.
[19] P. Priyadarshini, N. Prashant, D. G. Narayan and S. M. Meena, A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish, *Procedia Computer Science*, vol.78, pp.617-624, 2016.
[20] Y. He, N. Ye and R. Zhang, Analysis of data encryption algorithms for telecommunication network-computer network communication security, *Wireless Communications and Mobile Computing*, 2021.
[21] W. Stallings, *Cryptography and Network Security-Principles and Practice*, 7th Edtion, Pearson, 2017.
[22] A. Skander, M. Nadjim and B. Malek, A new accurate quantum cyptography control error reconciliation (QCCER) with XOR Operator in BB84 Protocal, *ICIC Express Letters*, vol.2, no.2, pp.187-192, 2008.
[23] Z. R. M. Subhi, DES encryption and decryption algorithm implementation based on FPGA, *Indonesian Journal of Electrical Engineering and Computer Science*, vol.18, no.2, pp.774-781, 2020.
[24] C. Meixi, Accounting data encryption processing based on data encryption standard algorithm, *Complexity*, 2021.
[25] D. Puchala and M. M. Yatsymirskyy, Joint compression and encryption of visual data using orthogonal parametric transforms, *Bull. Pol. Acad. Sci. Tech.*, vol.64, no.2, pp.373-382, 2016.