ACTA SCIENTIFIC COMPUTER SCIENCES

Volume 1 Issue 2 December 2019

Short Communication

# Draft Implementation of a Method to Secure Data by File Fragmentation

**Maurice HT Ling***

*Colossus Technologies LLP, HOHY PTE LTD, Republic of Singapore*

***Corresponding Author:** Maurice HT Ling, Colossus Technologies LLP, HOHY PTE LTD, Republic of Singapore.

**Received:** November 05, 2019; **Published:** November 18, 2019

## Abstract

Cryptography is fundamental in data security and is a critical tool in safeguarding information from "unauthorized" view during the storage and transportation of data. Due to the one-to-one correspondence between plain text and cipher text, encryption algorithms are transformation processes. This implies that all information is present, though encrypted, in the cipher text. Inspired by Jigsaw puzzles, a new cryptography system, Jigsaw Cryptography System (JCS) is proposed where a single plain text file results in many cipher text files, resembling jigsaw pieces from a single image. Thus, the interception of a small number of cipher text files may not compromise the entire contents in plain text. This can result in larger permutations needed to decipher by brute force, which is not easily achievable in most cryptographic methods.

**Keywords:** Cryptography; Jigsaw Encryption System (JCS)

## Introduction

Cryptography is synonymous with code writing and has its roots dated to at least 1900 BC when Egyptians used non-standard hieroglyphs to conceal the meaning of their writings [1]. Salomaa cleverly explained the two-world concept of cryptography – the sunlit world of secured information exchange, and the dark world of intercepting and decoding this information [2]. Data security and hiding is an important aspect of modern-day life [3,4]. The loss of secured information may have detrimental consequences, exemplified in the Babington Plot in 1587 leading to the public beheading of Mary Queen of Scots by her cousin, Queen Elizabeth of England [5]. This led to the development of cryptography systems, such as Honey Encryption [6,7], which provides a valid-looking but fake message upon decryption with a wrong key.

There are four components to any cryptography system. An encryption mechanism (known as encryptor) uses an encryption key to convert an unsecured and human-readable text (usually known as plain text) into a secured and human-unreadable text (usually known as cipher text) and produces a decryption key. A corresponding decryption mechanism (known as decryptor) uses the decryption key converts the cipher text back to plain text. Hence, the encryption key and decryption key exist as a pair. When the encryption key is the same as the decryption key, the cryptographic method is known as symmetric-key cryptography, such as AES [8] and BlowFish [9]. However, or public-key cryptography uses different encryption key and decryption key. Examples of asymmetric-key cryptography are RSA [10] and ElGamer [11]. The strength of the system lies in the difficulty in obtaining the encryption-decryption key-pair [9], which is reflected by the number of possible encryption-decryption key-pairs. The larger the possible number of key-pairs, which is proportional to the computing time required obtaining the correct key-pairs using brute force methods [12]; the stronger the encryption [9]. As the number of possible key-pairs is largely correlated to the length of the key, the length of the key is commonly used to compare the strength of the cryptographic method [12,13]. In addition, the length of keys is largely dependent on the nature of the cryptographic algorithm [12,13]; hence, difficult to expand without crucial changes to the algorithm.

The encryptor and decryptor can be considered as data transposition algorithms between plain text and cipher text; where plain text can be any data stream [14], including images [15]. From this point of view, the cipher text is a transposed text containing all the information in the plain text, and the number of possible key-pairs represents the total number of ways in which a cipher text can be decrypted. Thus, the strength of the cryptographic method rests on the strength of the encryption-decryption key-pairs. Sir Francis Walsingham, Queen Elizabeth's secretary in the Babington Plot, was aware of the plot to assassinate Queen Elizabeth for some time but lacked the crucial evidence to incriminate Mary Queen of Scots; hence, unable to convince Queen Elizabeth of the plot; until a single approval letter from Mary Queen of Scots finally convinced Queen Elizabeth and sealed the fate for Mary Queen of Scots. Therefore, the possible weakest link in any cryptographic system may be the fact that all the information in the plain text is found within a single cipher text, inviting and awaiting deciphering. The same can be said for each intercepted message sent to Bletchley Park for decryption during World War II [16,17].

In this study, a draft/prototype of a new cryptographic method based on file splitting is proposed. This is inspired by jigsaw puzzles; hence, named as Jigsaw Cryptography System (JCS); to address the weakness of full information correspondence between a single plain text and a single cipher text. Jigsaw cryptography has no inherent key length requirement, allowing for key length expansion with relative ease. Calculation of the number of possible means to revert the cipher text back to plain text, which corresponds to the possible number of key-pairs, demonstrates that Jigsaw cryptography outperforms current cryptographic systems.

## Jigsaw cryptography

The analogy of jigsaw puzzles is used to address the weakness of full information correspondence between a single plain text and a single cipher text. Jigsaw cryptography proposes to slice one plain text file into multiple cipher text files during encryption process. This is likened to taking a large picture image and chopping it up in many jigsaw pieces; or cutting a book into pages, remove all page numbers, and jumbling the pages up. Using the analogy of Babington plot, Jigsaw Cryptography System (JCS) is likened to cut Mary Queen of Scots' approval letter into pieces and send each labelled piece separately to Anthony Babington, the chosen assassin. Sir Francis Walsingham would have to intercept adequate pieces including pieces containing critical words before the letter could be deciphered. In the case of Enigma, encrypted message sent by Germany during World War II, multiple instructions for various combat units can be concatenated together and shredded by JCS before transmission, which adds another layer of complexity as each intercepted shred will not contain the entire combat order.

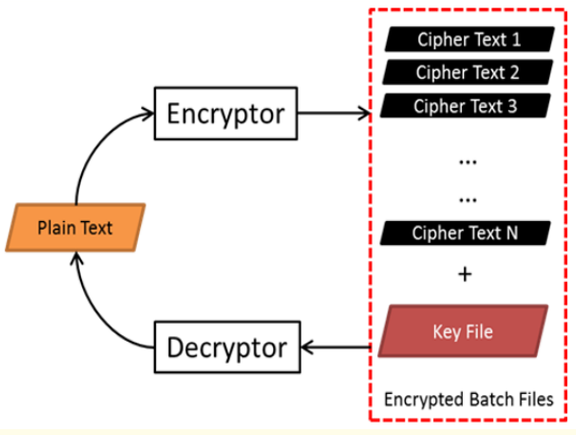


**Figure 1:** Schema of Jigsaw Cryptography.

In this study, a fundamental version of JCS, version 1, has been implemented. The encryptor takes a plain text file and split the file into a set of Jigsaw files of pre-defined size (Figure 1) and generates a log file. There are 2 modes, even slicing and uneven slicing. In even slicing, the size of each Jigsaw file will be the same except for the last Jigsaw file. In uneven slicing, the size of each Jigsaw file will be between 1 and 2 times of the specified length. The generated log file, together with the set of Jigsaw files, will be input to the decryptor for assembly into the original plain text file. To ensure fidelity, file hashes for the original and assembled plain text file, as well as for each Jigsaw file, are generated and compared.

Although JCS1 is not a true cryptographic method as each Jigsaw file are essentially unencrypted sections of the original plain text, the number of assembly permutations can be exponentially increased by reducing the size of each Jigsaw file. Moreover, the complexity in reassembly can also be exponentially increased as the number of plain text files increases – instead of splitting a book into pages, the entire library of books is now split into pages. Given that the key size of AES-256 is 32 characters long and given that there are 94 characters in the set of mixed alphanumeric with symbols, the total number of keys is $94^{32}$ or Hence, JCS1 can be theoretically as strong as AES-256 if the number of assembly permutations exceeds $1.38 \times 10^{63}$; which can be achieved by using at least 50 Jigsaw files, yielding ${}^{50}P_{50}$ or $3 \times 10^{64}$ permutations.

Cryptographic systems can generally be classified into 2 broad categories based on usage; namely, data transport cryptography and data storage cryptography. Data transport cryptography methods emphasize equally on algorithmic efficiency and security [18] while data storage cryptography methods emphasize more on security than algorithmic efficiency [19]. JCS is more suitable for data storage than data transport as its main emphasis is on security rather than efficiency.

## Implementation and Usage

JCS is implemented by adapting the codes from my previous code paper [20] into a command-line tool using Python 3 and Python-Fire module (https://github.com/google/python-fire), which aims to simplify the implementation of command-line interface in Python 3 [21]. Encryption is performed using the `encrypt` command; `python jsc.py encrypt --slicer=<slicer> --blocksize=<blocksize> --filenamelength=<filenamelength> --hashlength=<hashlength> --version=<version> --verbose=<verbose> --filename=<filename> --output_dir=<output_dir>;` where filename is relative or absolute path to the file to be encrypted, slicer sets the file slicing method, blocksize sets the size of a Jigsaw file, filenamelength set the length of file name of a Jigsaw file, hashlength sets the file has length of each Jigsaw file, version sets the Jigsaw version, and verbose sets the verbosity from 1 (most information). For example, `python jsc.py encrypt --slicer=even --blocksize=262144 --filenamelength=30 --hashlength=16 --version=1 --verbose=2 --filename=test_data/DataA.xlsx --output_dir=./test_data` splits (encrypt) test_data/DataA.xlsx into Jigsaw files of 256 kilobytes each.

Decryption is performed using the `decrypt` command; `python jsc.py decrypt --keyfilename=<keyfilename> --outputfile=<outputfile> --encrypt_dir=<encrypt_dir>`; where keyfilename is the relative or absolute path to the

key file (produced during encryption process), outputfile is the relative or absolute path to the file after decryption, and encrypt_dir is the relative or absolute path to the directory containing the jigsaw files. For example, `python jsc.py decrypt --keyfilename=test_data/DataA.xlsx.jgk --outputfile=test_data/DataA_1.xlsx --encrypt_dir=./test_data` assembles (decrypt) the Jigsaw files in ./test_data folder into DataA_1.xlsx using DataA.xlsx.jgk as the key file.

## Future work

This study presents the first implementation of JCS. Future work on JCS can advance along two inter-related tracks. Firstly, additional file manipulation methods, such as bit swapping, can be implemented to improve the cryptographic complexity of JCS. Each Jigsaw file may also be independently encrypted by existing cryptography systems using randomized keys as long as these randomized keys and their corresponding systems are recorded in the key file. Secondly, JCS can be developed into a cryptographic file system [22] using user-space file systems [23].

## Availability

Jigsaw Cryptography System is available for use with Python 3.7 and forking under GNU General Public License version 3 at https://github.com/mauriceling/jigsaw_cryptography.

## Conflict of Interest

The author declares no conflict of interest.

## Bibliography

1. M Whitman and H Mattord. Principles of information security". Canada, Thomson Learning, Inc., 4 (2005).

2. A Salomaa. "Public-key cryptography". Springer Science and Business Media (2013).

3. AV Duka and B Genge. "Implementation of SIMON and SPECK lightweight block ciphers on programmable logic controllers". in 5th International Symposium on Digital Forensic and Security (ISDFS) (2017): 1-6.

4. Pandey and S Som. "Applications and usage of visual cryptography: A review". presented at the 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) (2016): 375-381.

5. JE Lewis. "The trial of Mary Queen of Scots: A Brief history with documents". Macmillan (1999).

6. A Juels and T Ristenpart. "Honey encryption: Security beyond the brute-force bound". presented at the Annual International Conference on the Theory and Applications of Cryptographic Techniques (2014): 293-310.

7. AE Omolara., *et al*. "A comprehensive review of honey encryption scheme". *Indonesian Journal of Electrical Engineering and Computer Science* 13.2 (2019): 649-656.

8. J Daemen and V Rijmen. "The design of Rijndael: AES-the advanced encryption standard". Springer Science and Business Media (2013).

9. B Schneier. "Description of a new variable-length key, 64-bit block cipher (Blowfish)". presented at the Fast software encryption (1994): 191-204.

10. RL Rivest., *et al*. "A method for obtaining digital signatures and public-key cryptosystems". *Communications of the ACM* 21.2 (1978): 120-126.

11. T ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Transactions on Information Theory* 31.4 (1985): 469-472.

12. R Bhanot and R Hans. "A review and comparative analysis of various encryption algorithms". *International Journal of Security and Its Applications* 9.4 (2015): 289-306.

13. R Mane. "A Review on Cryptography Algorithms, Attacks and Encryption Tools". *International Journal of Innovative Research in Computer and Communication Engineering* 3.9 (2015): 8509-8514.

14. L Jothi. "A Literature Review: Cryptography Algorithms for Wireless Sensor Networks". *International Journal of Computer Science Engineering and Technology* 4.10 (2013): 1277-1282.

15. K Bai and C Wu. "An AES-Like Cipher and Its White-Box Implementation". *The Computer Journal* 59.7 (2016): 1054-1065.

16. C Smith. "Bletchley Park and the Development of the Rockex Cipher Systems: Building a Technocratic Culture, 1941-1945". *War in History* 24.2 (2017): 176-194.

17. M. Smith. "The Wrens of Bletchley Park". *ACM Crossroads* 21.3 (2015): 48-53.

18. S. Verma., *et al*. "An Enhanced Cryptographic System for Fast and Efficient Data Transmission". In International Conference on Advanced Computing Networking and Informatics 870, R. Kamal, M. Henshaw, and P. S. Nair, Eds. Singapore: Springer Singapore (2019): 287-297.

19. M Popli and Gagandeep. "DNA Cryptography: A Novel Approach for Data Security Using Flower Pollination Algorithm". In Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur, India (2019).

20. MH Ling. "A cryptography method inspired by jigsaw puzzles". *Current STEM 1* (2018): 129-142.

21. MH Ling. "RANDOMSEQ: Python command–line random sequence generator". *MOJ Proteomics and Bioinformatics* 7.4 (2018): 206-208.

22. D Burihabwa., *et al*. "SGX-FS: Hardening a File System in User-Space with Intel SGX". In 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Nicosia (2018): 67-72.

23. R Pontes., *et al*. "SafeFS: a modular architecture for secure user-space file systems: one FUSE to rule them all". In Proceedings of the 10th ACM International Systems and Storage Conference on - SYSTOR '17, Haifa, Israel (2017): 1-12.

**Volume 1 Issue 2 December 2019**