Citation: Ling, MHT. 2018. **A Cryptography Method Inspired by Jigsaw Puzzles.** In Current STEM, Volume 1, pp 129-142. Nova Science Publishers, Inc. ISBN 978-1-53613-416-2.

A CRYPTOGRAPHY METHOD INSPIRED BY JIGSAW PUZZLES

Maurice HT Ling *

Colossus Technologies LLP, Singapore

School of BioSciences, The University of Melbourne, Australia

ABSTRACT

Cryptography 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 can be seen as a transformation process. This is a deficiency as all information is present, though encrypted, in the cipher text. Inspired by Jigsaw puzzles, a new cryptography system, Jigsaw Encryption System (JES) is proposed where a single plain text file results in many cipher text files, resembling jigsaw pieces from a single image; thus, the loss of a small number of cipher text files may not compromise the entire contents in plain text. Each cipher text can be further processed for added security. This can result in larger permutations needed to decipher by brute force. Reference implementations of preliminary JES versions had been deposited into COPADS (Collection of Python Algorithms and Data Structures) repository (https://github.com/mauriceling/copads; File name: copads/jigsaw.py).

Keywords: Encryption for Transportation, File Splitting, Jigsaw, Python.

Date Submitted: August 4, 2017. **Date Accepted:** September 7, 2017.

* Corresponding Author: mauriceling@acm.org, mauriceling@colossus-tech.com

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 (Whitman and Mattord, 2005). 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 (Salomaa, 2013). The loss of secured information may have dire consequences, as exemplified in Babington Plot leading to the trial and public beheading of Mary Queen of Scots by her cousin, Queen Elizabeth of England, in 1587 (Lewis, 1999).

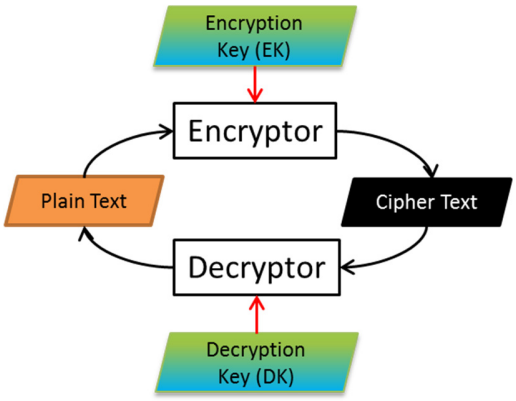

Figure 1. Encryption-Decryption Cycle. Unencrypted data is denoted as plain text, which is converted to encrypted data denoted as cipher text, using an encryptor. The encryptor requires an encryption key. The cipher text can then be decrypted by a decryptor, requiring a decryption key, and produces the plain text. For the cipher text to be correctly decrypted, the correct decryption key to the encryption key must be used.

There are four components to any cryptography system (Figure 1). 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), an encryption mechanism (known as encryptor) is needed. The encryptor takes the plain text and

an encryption key as input to produce the cipher text. Conversion from cipher text back to plain text will require the cipher text and decryption key as input to the decryptor. Hence, the encryptor and decryptor for a cryptographic system exist as a pair, and 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 (Daemen and Rijmen, 2013) and BlowFish (Schneier, 1994). However, the encryption key and decryption key are different in some cryptographic method. This is known as asymmetric-key cryptography or public-key cryptography, such as RSA (Rivest et al., 1978) and ElGamer (ElGamal, 1985). The strength of the system lies in the difficulty in obtaining the encryption-decryption key-pair (Schneier, 1994), which can be translated to the number of possible encryption-decryption key-pairs. Hence, the larger the possible number of key-pairs, the stronger the encryption (Schneier, 1994). As the number of possible key-pairs is proportional to the computing time required obtaining the correct key-pairs using brute force methods (Bhanot and Hans, 2015) and 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 (Bhanot and Hans, 2015; Mane, 2015). In addition, the length of keys is largely dependent on the nature of the cryptographic algorithm (Bhanot and Hans, 2015; Mane, 2015); hence, difficult to expand without crucial changes to the algorithm.

Another way to view the encryptor and decryptor is that they are data transposition algorithms between plain text and cipher text; where plain text can be any data stream (Jothi, 2013), including images (Bai and Wu, 2016). In this point of view, cipher text is essentially a transposed text containing all the information and data in the plain text. In addition, the number of possible key-pairs can represent the total number of ways in which a cipher text can be decrypted. Thus, the entire weight of the cryptographic method rests on the strength of the encryption-decryption key-pairs. In Babington Plot, Sir Francis Walsingham, Queen Elizabeth's secretary, knew of the plot to assassinate Queen Elizabeth for some time but lacked the crucial evidence to incriminate Mary Queen of Scots to the assassination plot; hence, unable to convince Queen Elizabeth of the plot; until a single letter from Mary Queen of Scots approving the plot that 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.

In this study, a new cryptographic method based on Jigsaw puzzle is proposed; hence, named as Jigsaw Encryption System (JES); to address the weakness of full information correspondence between a single plain text and a single cipher text. In addition, Jigsaw encryption has no inherent key length requirement, which allows 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

Inspired by jigsaw puzzles 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. Using the analogy of Babington plot, Jigsaw Encryption System (JES) will be similar to shredding Mary Queen of Scots' approval letter and send each labeled shred separately to Anthony Babington, the chosen assassin. 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 JES before transmission, which adds another layer of complexity as each intercepted shred will not contain the entire combat order.

In this study, three versions of JES with increasing complexity have been implemented. As the possible number of key-pairs (also known as permutations) is proportional to the strength of the encryption [6], the number of permutations needed decrypt a 1 megabyte file by brute force method will be emphasis, on basis that each Jigsaw-ed cipher file (hereinafter known as Jigsaw file) is set to 64 kilobytes.

JES version 1 (JES1) encryptor takes a plain text file and split the file into a set of Jigsaw files based on a specified size for each Jigsaw file (Figure 2), and generates a log file. There are 2 modes for each JES version, even slicing and uneven slicing. In even slicing, each Jigsaw file will be of the same length as specified except the last Jigsaw file. In uneven slicing, the size of each Jigsaw file will be 1 to 2 times of the specified length. The generated log file will be used as input, together with the set of Jigsaw files, 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.

Assuming that JES1 is used to encrypt a 1 MB (1,048,576 bytes) file and split into 16 64-KB Jigsaw files, there are 2 x 10^{13} permutations to assemble 16 64-KB files into the original 1 MB file.

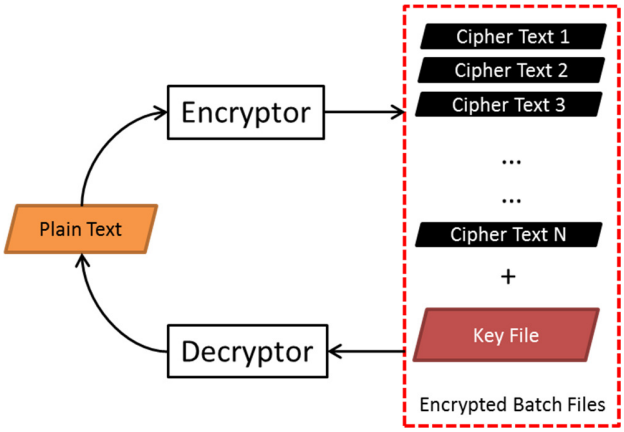

Figure 2. Schema of Jigsaw Cryptography.

However, JES1 cannot be considered to be true cryptographic method as each Jigsaw file are essentially unencrypted sections of the original plain text. This is likened to cutting a book into pages, remove all page numbers, and jumbling the pages up. The cipher text files are really sectioned plain text files. Despite so, the complexity in reassembly can be exponentially complex in event whereby the number of plain text files increases – instead of splitting a book into pages, the entire library of books is now split into pages.

JES version 2 (JES2) builds on JES1 by provides 63 randomly assigned variations for each Jigsaw file. Each variation is the reversion of a portion of the Jigsaw file (Figure 3). The difference between each reversion scheme is the start and end byte positions of the inversion. Bytes in between the start and stop positions are reversed. As a result, each Jigsaw file is no longer sectioned plain text as in JES1. Including the permutations on each JES1 Jigsaw file, the full permutations possible for 16 JES2 64-KB Jigsaw files is $(2 \times 10^{13})^{63}$ or 9×10^{837}.

JES version 3 (JES3) adds an additional step of transposition from the Jigsaw file provided by JES2 by applying one of the 1,350 randomly assigned transpositions to each Jigsaw file (Figure 3). The transposition function considers a Jigsaw file into two equal halves before randomly assigning one of the 1,350 start

position and length in the first half of the Jigsaw file. It then takes the bytes as identified in the first half of the Jigsaw file and transpose it with the equal number of bytes extracted from the second half of the Jigsaw file starting from the beginning of the second half (the midpoint of the Jigsaw file). As a result, there can be a section of the Jigsaw file that is both reversed and transposed, adding to the complexity for deciphering. This means that there are possibly $63^{1,350}$ or 5 x $10^{2,428}$ permutations for each of the 16 JES3 64-KB Jigsaw files generated for a 1 MB file, Hence, the full permutations possible for 16 JES3 64-KB Jigsaw files is (9 x 10^{837})^{1,350} or 6 x $10^{1,131,240}$.

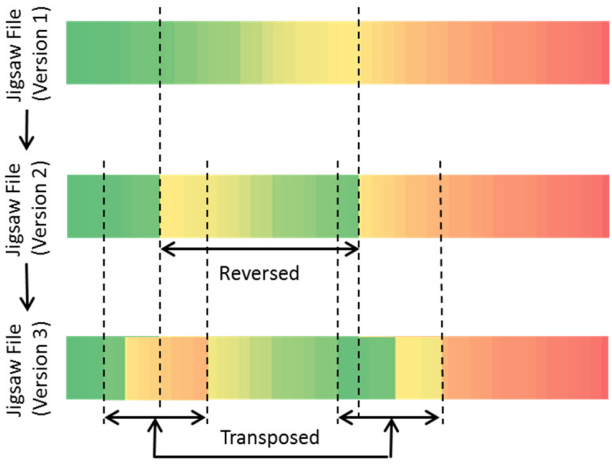

Figure 3. Relationship between Jigsaw Files of 3 Different Versions of Encryption.

IMPLEMENTATION AND TESTING

JES versions 1 to 3 have been implemented in Python programming language and deposited in COPADS (Collection of Python Algorithms and Data Structures) repository (https://github.com/mauriceling/copads; File name: copads/jigsaw.py) under Python Software Foundation License version 2. Besides setting the version of JES to use and the size of Jigsaw file (as blocksize), the following operational parameters are available:

- Type of slicer (`setting = 'slicer'`), where "even" means that each Jigsaw file will be of the same size except the final Jigsaw file, and

"uneven" means that each Jigsaw file size will be between block size and twice of block size

- Length of Jigsaw file name (`setting = 'filenamelength'`), and it is usually better to set a longer file name to prevent name collision
- Hash length of each Jigsaw file (`setting = 'hashlength'`), which will be generated for each Jigsaw file for debugging purposes

The following code exemplifies using JES to encrypt and decrypt a MP4 file:

```
import jigsaw

j = jigsaw.JigsawFile()

j.setting('slicer', 'even')
j.setting('blocksize', 32768) # 32KB Jigsaw file
j.setting('filenamelength', 30)
j.setting('hashlength', 16)
j.setting('version', 3)
j.setting('verbose', 2)

# Encrypt file: A.mp4
keyFileName = j.encrypt('A.mp4')

# Decrypt file to A_1.mp4
j.decrypt(keyFileName, 'A_1.mp4')
```

During encryption, the following summary will be generated.

```
Encrypting file: A.mp4
... Sorting out input (source file) directory
...... Attempt to get input directory from input file path
...... Failed: Input file is relative file name
...... Set input directory to currect working directory
... Sorting out output (encrypted files output) directory
...... Output directory not given
...... Attempt to find output directory from input file
...... Failed: Input file is relative file name
...... Set output directory to currect working directory
... in input directory: C:\Users\Maurice Ling\Desktop\jigsaw
... onto output directory: C:\Users\Maurice
Ling\Desktop\jigsaw
```

```
... using Jigsaw version: JigsawFileTHREE
... using slicer: even
... using block size: 32768
Writing key file: C:\Users\Maurice
Ling\Desktop\jigsaw\A.mp4.jgk
Processing using even slicer
0 blocks processed
1000 blocks processed
2000 blocks processed
3000 blocks processed
3920 blocks processed
Encrypting file, A.mp4, completed
```

The key file, which represents the decryption key, is generated with the file name format as `<plain text file name>.jgk`. In this case, the key file is `A.mp4.jgk` as the plain text file is `A.mp4`. This key file is required for decryption as it contains the assembly scheme. During decryption, the following summary will be generated. A series of file hashes will be generated from the decrypted file and compared with the corresponding file hashes generated from the original plain text file during encryption process and stored in the key file.

```
Decrypting file using keyfile: C:\Users\Maurice
Ling\Desktop\jigsaw\A.mp4.jgk
... Processing key file
... Sorting out input (encrypted files) directory
...... Input directory is not given
...... Attempt to get input directory from keyfile name
...... Keyfile name is absolute path: C:\Users\Maurice
Ling\Desktop\jigsaw\A.mp4.jgk
...... Set input directory to directory of keyfile
... Sorting out output file (unencrypted file) name
...... Given output file name is relative path: A_1.mp4
...... Attempt to generate absolute output file path
...... Set folder to write output file as input directory
... Directory of encrypted files (input): C:\Users\Maurice
Ling\Desktop\jigsaw
... Uncrypted file name (output): C:\Users\Maurice
Ling\Desktop\jigsaw\A_1.mp4
Decrypting file ......
0 blocks processed
1000 blocks processed
2000 blocks processed
```

```
3000 blocks processed
3920 Jigsaw files processed
Expected number of bytes: 128388534
Actual number of bytes  : 128388534
File Hashs (Decrypted File vs Original Unencrypted File)
md5: 49c978e1e1124a3eb245eb7430843152
  vs 49c978e1e1124a3eb245eb7430843152
sha1: 43355d06548dee9027afaf4825b1f82226eb298e
  vs  43355d06548dee9027afaf4825b1f82226eb298e
sha224:
      54ea7a0e419ce7d28a68514815ceab811131b4923f671973dbaa11
      9c
  vs
      54ea7a0e419ce7d28a68514815ceab811131b4923f671973dbaa11
      9c
sha256:
      d2dc6e12f4f9f7d7ccf4a8e1ea41566a1a8041c3ad0863194daa08
      1d2507d068
  vs
      d2dc6e12f4f9f7d7ccf4a8e1ea41566a1a8041c3ad0863194daa08
      1d2507d068
sha384:
      17f9e4fbddb316ed3c87ed83afd1db31f73bb351d4d48e3af17e21
      b2f13fe9aa6de3bbe2d4c36ef47dad92058fde2926
  vs
      17f9e4fbddb316ed3c87ed83afd1db31f73bb351d4d48e3af17e21
      b2f13fe9aa6de3bbe2d4c36ef47dad92058fde2926
sha512:
      b0d64e8a12c25103de65562d22e38a8a89a6b95be38471bcf376a6
      13e9e158abb97ee3f87848192b35ee005434713a4e712db2353912
      f091b4835478b80d095b
  vs
      b0d64e8a12c25103de65562d22e38a8a89a6b95be38471bcf376a6
      13e9e158abb97ee3f87848192b35ee005434713a4e712db2353912
      f091b4835478b80d095b
Decryption completed
```

In the example above, all 6 file hashes (md5, sha1, sha224, sha 256, sha384, and sha512) are identical for both before (original plain text) and after (decrypted cipher text) an encryption-decryption cycle, indicating that the JES encryptor and decryptor are functioning as expected.

DISCUSSION

Cryptographic methods are integral to information security during storage and transportation. However, one-plain text/one-cipher text relationship may represent a weak link in most cryptographic systems as all the information in the plain text exist in the cipher text, though encrypted. This study proposes to address this weakness by a 1 to N relationship between plain text and cipher text. As a result, the interception of a small number of cipher texts is unlikely to compromise the security of the plain text.

256-bit Advanced Encryption Standard (AES-256) is currently considered resistant to attacks [13] and suitable for use in many encryption applications [14]. Given that the key size of AES-256 is 32 characters long (32 characters of 8 bytes each = 256 bytes) and given that there are 94 characters in the set of mixed alphanumeric with symbols (that is, `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/`), the total number of possible keys is 94^{32} or 1.38 x 10^{63}. In comparison, the number of permutations for 16 JES1 Jigsaw files is significantly lesser than the total number of possible keys in AES-256 but the number of permutations for 16 JES2 or JES3 Jigsaw files are many magnitudes larger than that of AES-256. It has to be noted that the calculations are based on a 1 megabyte plain text file. The number of 64-KB Jigsaw files will increase linearly with large plain text files, leading to exponential increase in assembly complexity in terms of permutations. Moreover, the number of permutations for JES1 can be easily increased by reducing the size of each Jigsaw file, such as from 64-KB to 8-KB will increase the number of Jigsaw files by 8 times.

Three versions of JES, with increasing complexity, are presented in this study as example implementations; demonstrating the relative ease in extending this concept into increasingly complex cryptographic systems in the future. For example, it may be possible to encrypt sections of the plain text files with existing cryptographic systems, such as randomly choosing AES [5] or BlowFish [6] for each section, before passing through JES. In JES2, only one reversal is carried out; and JES3, only one reversal and one transposition are carried out. Future versions may include multiple rounds of reversals or/and transpositions. Other possible enhancements may include cross-encryption where 2 or more plain text files are encrypted together, or the inclusion of decoys where short sections of novels may be interspaced at random into a business document before encryption.

The main feature of JES is the 1-to-M plain text/cipher text relationship. Although this will allow secured transportation of data by separating the set of Jigsaw files into separate storage devices for transporting, this also meant that JES requires a large volume of disk I/O due to file writing and reading, resulting in significant processing time. Hence, JES may only be suitable for securing data

archives and transportation rather than real-time use (Xie et al., 2016), such as in filesystem-level encryption [16], where encryption/decryption efficiency is critical.

REFERENCES

Bai, K., and Wu, C. (2016). An AES-Like Cipher and Its White-Box Implementation. The Computer Journal *59*, 1054–1065.

Bhanot, R., and Hans, R. (2015). A review and comparative analysis of various encryption algorithms. International Journal of Security and Its Applications *9*, 289–306.

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

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory *31*, 469–472.

Jothi, L. (2013). A Literature Review: Cryptography Algorithms for Wireless sensor networks. In International Journal of Computer Science and Information Technologies, p.

Lewis, J.E. (1999). The trial of Mary Queen of Scots: A Brief history with documents (Macmillan).

Mane, R. (2015). A Review on Cryptography Algorithms, Attacks and Encryption Tools. International Journal of Innovative Research in Computer and Communication Engineering *3*, 8509–8514.

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

Salomaa, A. (2013). Public-key cryptography (Springer Science & Business Media).

Schneier, B. (1994). Description of a new variable-length key, 64-bit block cipher (Blowfish). (Springer), pp. 191–204.

Whitman, M., and Mattord, H. (2005). Principles of information security. Canada, Thomson Learning, Inc. *4*, 2009.

Xie, Y., Ding, W., and Wang, Y. (2016). A More Extensible Transparent Encrypted File System Based on Filter Driver. Journal of Communications *11*, 383–387.