Super-Encryption Implementation Using Monoalphabetic Algorithm and XOR Algorithm for Data Security

Dian Rachmawati¹, Mohammad Andri Budiman¹, Indra Aulia²

¹ Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia.
² Departemen Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia.

Email: dian.rachmawati@usu.ac.id

Abstract. The exchange of data that occurs offline and online is very vulnerable to the threat of data theft. In general, cryptography is a science and art to maintain data secrecy. An encryption is a cryptography algorithm in which data is transformed into cipher text, which is something that is unreadable and meaningless so it cannot be read or understood by other parties. In super-encryption, two or more encryption algorithms are combined to make it more secure. In this work, Monoalphabetic algorithm and XOR algorithm are combined to form a super-encryption. Monoalphabetic algorithm works by changing a particular letter into a new letter based on existing keywords while the XOR algorithm works by using logic operation XOR. Since Monoalphabetic algorithm is a classical cryptographic algorithm and XOR algorithm is a modern cryptographic algorithm, this scheme is expected to be both easy-to-implement and more secure. The combination of the two algorithms is capable of securing the data and restoring it back to its original form (plaintext), so the data integrity is still ensured.

1. Introduction

The technology of internet and multimedia industry rapidly develop, which makes people realize the importance of data security [1]. Protecting confidentiality and authenticity of some transactions by the use of cryptography method has become standard practice [2]. Cryptography is the art and science of using mathematical methods to secure messages from unauthorized parties [3]. To preserve security with cryptography, messages in analog forms or symbols (which are called “plaintext”) are generally transformed using tables (for example, ASCII table) into numbers. A secret key or password is then generated by the sender and by using an encryption function (which usually includes some mathematical formulas), these numbers are transformed into meaningless information which are called “cipher text”. The cipher text is then sent via a channel to the recipient. Using different (and preferably more secure) channel, the secret key is also sent to the recipient. Having got the cipher text and the key, the recipient can use the decryption function to transform the cipher text back to the original plaintext. Mathematically, the decryption function is simply the inverse of the encryption function.

Monoalphabetic algorithm is a classical cryptographic algorithm that works by switching each letter in the plaintext into another letter based on some secret keys or keywords. Since there are 26 letters in the alphabet, a cryptanalyst (a person who wants to convert the cipher text into the plaintext without knowing the secret keys) can try a brute-force attack for at most 26 times and it is guaranteed that he will be able to succeed. Therefore, the security of Monoalphabetic algorithm is weak and it
should not be used for encrypting very sensitive and classified information such as those in banking and military.

In modern cryptography, XOR function plays an important role since it is very easy to perform [4]: to encrypt a message (or plaintext), one should transform it using an encoding table (such as ASCII) to become decimal numbers, and convert these numbers into binaries. Then, some random bits of the same length of those binaries are generated as the secret key. The binaries and the secret key are then XOR-ed together, and we get the cipher text. The cipher text is then sent to the recipient via a channel and the secret key is sent via another channel (which is preferably more secure than the previous channel). In order to convert the cipher-text back to the plaintext, the recipient just can just XOR the cipher-text with the secret key, convert the resulted binaries to decimal numbers, and by using the same encoding table, the decimal numbers can be converted back to the plaintext. This whole encryption and decryption system is called XOR algorithm.

In order to create a stronger cryptosystem, two or more cryptography algorithms can be combined together. This new cryptosystem is called super-encryption. The purpose of this research is to combine Monoalphabetic algorithm with XOR algorithm in a super-encryption scheme in order to make the whole cryptosystem more secure yet still easy-to-implement.

This super-encryption scheme is unique, since it conjoins a classical cryptographic algorithm (Monoalphabetic algorithm) and a modern cryptographic algorithm (XOR algorithm). The proposed method may be used to encrypt data or password that needs to be transmitted across a public network, to ensure its confidentiality and integrity against the attackers [5]. A Python source-code for the new crypto-system is provided. Python was designed to be a highly readable language. Python aims to be simple and consistent in the design of its syntax [6].

2. Methods
In this section, we explain and show an example of how Monoalphabetic algorithm works an example of how XOR algorithm works the model of the new crypto-system, and the Python source-code.

2.1 Monoalphabetic algorithm
Suppose we have a plaintext = “KOMPUTER” and a secret key = “DIAN RACHMAWATI”. The secret key should have no repeating letters and there should be nothing inside except letters. Thus, we omit the repeating letters and space, and we have the keyword = “DIANRCHMWT”.

The keyword is placed underneath the alphabet. The other letters that are not within the keyword is appended to the keyword, and the whole is called cipher alphabet.

\[
\begin{align*}
\text{alphabet} & = \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
\text{cipheralphabet} & = \text{DIANRCHMWTBEFGJKLOPQSUVXYZ}
\end{align*}
\]

Through the encryption, key mapping can be done by searching the couple’s alphabet to cipher alphabet so to plaintext = “KOMPUTER” cipher text obtained “BJFKSQRO” as follows:

\[
\begin{align*}
\text{alphabet} & = \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\
\text{cipheralphabet} & = \text{DIANRCHMWTBEFGJKLOPQSUVXYZ}
\end{align*}
\]

To restore the cipher text into the plain text then use the form of the mapping as follow:

\[
\begin{align*}
\text{cipheralphabet} & = \text{DIANRCHMWTBEFGJKLOPQSUVXYZ} \\
\text{alphabet} & = \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
\end{align*}
\]

So the ciphertext “BJFKSQRO” back into “KOMPUTER”.

2
2.2 XOR
In cryptography, making a cipher (text encryption results) through the operation of XOR, which is an encryption of algorithm, are relatively simple. This technique operates in accordance with the principle of:
A XOR 0 = A,
A XOR A = 0,
(B XOR A) XOR A = B XOR 0 = B,

With this logic, a string text could be encrypted by applying XOR operation based of bit (binary digit), for each character using the specific key. To decrypt the cipher text is applied the operation XOR on the cipher text, so that plain text back to its original form. For example: if we have D message and the key is a so the encryption process is as follows:

\[
\begin{align*}
D &= 68 = 01000100 \\
a &= 97 = 01100001 \\
\text{XOR} &\rightarrow 37 = \% \\
\end{align*}
\]

The resulting cipher text is %, as for the decryption process then performed the operation XOR between cipher text with a key so that it returns to the original message.

\[
\begin{align*}
\% &= 37 = 00100101 \\
a &= 97 = 01100001 \\
\text{XOR} &\rightarrow 68 = D \\
\end{align*}
\]

2.3. The model of the new crypto-system
This research aims to improve security in the process of encoding a message by using a combination of Monoalphabetic and XOR algorithms, so this coded messages increasingly difficult to solve by any third party due to use of the concept Super-encryption. Super-encryption is a cryptographic concept that combines two or more symmetric encryption algorithms to ensure stronger confidentiality. The illustration of the super -encryption process in this research can be seen in Figure 1.

![Figure 1. Super-encryption process with algorithm mono-alphabetic and XOR](image_url)
The super-encryption in Figure 1 is explained as follows. Firstly, the sender encrypts the plaintext with Monoalphabetic encryption algorithm, resulting in the first cipher text. Secondly, the sender uses XOR encryption algorithm to encrypt the first cipher text into ciphertext2. The nature of the XOR algorithm that uses random bits as its key makes the ciphertext2 appears more random than the first cipher text. Thirdly, the ciphertext2 is sent to the recipient. Fourthly, the recipient decrypts the ciphertext2 into cipher text using XOR algorithm. Lastly, the recipient decrypts the cipher text with Monoalphabetic algorithm and gets the original plaintext.

2.4. The Python source code
The code below is used to perform the key generation based on user input string:

```python
for c in key:
    if c not in alphabet:
        continue
    if c not in keyword:
        keyword += c

cipheralphabet = keyword
for c in alphabet:
    if c not in cipheralphabet:
        cipheralphabet += c
```

Source code below are used to encrypt and decrypt messages:

```python
for p in plaintext:
    if p not in alphabet:
        continue
    i = alphabet.index(p)
    ciphertext1 += cipheralphabet[i]

decryptedtext = ""
for c in ciphertext1:
    i = cipheralphabet.index(c)
    decryptedtext += alphabet[i]
```

Source code below are used to encrypt and decrypt messages using XOR operation:

```python
def XOR(bits1, bits2):
    #assert len(bits1) == len(bits2)
    result = ""
    for i in range(len(bits1)):
        if bits1[i] == bits2[i]:
            result += "0"
        else:
            result += "1"
    return result
```

3. Results and Discussions
By using super-encryption algorithm Monoalphabetic and XOR then obtained the following results:

*First Encryption (Mono-alphabetic Cipher)*

*Key Initialization*
key = DIAN RACHMAWATI
keyword = DIANRCHMWT

*Table Initialization*
alphabet = ABCDEFGHIJKLMNOPQRSTUVWXYZ
ciphertext1 = DIANRCHMWTBEFGJKLOPQSUVXYZ

*Encryption*
plaintext = KOMPUTER
ciphertext1 = BJFKSQRO

*Second Encryption (XOR Cipher)*
*Initialization*
ciphertext1 = BJFKSQRO
plainbit = 01000010010010100100011001001011010100110101001010100100001010001000101000001

*Encryption*
random key = 00100000000110100000001111010100111010001111010001101000101010001010010101010100010001111
plainbit = 01000010010010100100011001001011010100110101001010100100001010001000101000001
cipherbit = 011000100100100001001110101001100111101100101010001010010011111111011010100101010001000100011

*ciphertext* = bPEž hl÷e

*First Decryption (XOR Cipher)*

*Decryption*
Ciphertext = bPEž hl÷e
Cipherbit = 01100010010010000100111010100110011110110010101000101010010001111
random key = 00100000000110100000001111010100111010001111010001101000101010001010010101010100010001111
decryptedbit = 010000100100101001000110010010110101001101010010101001000110001000101000001
decryptedtext1 = BJFKSQRO

*Second Decryption (Monoalphabetic Cipher)*

*Key Initialization*
Key = DIAN RACHMAWATI
Keyword = DIANRCHMW

*Table Initialization*
Alphabet = ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext = DIANRCHMWTBEFGJKLOPQSUVXYZ

*Decryption*
decryptedtext1 = BJFKSQRO
decryptedtext2 = KOMPUTER

4. Conclusions
It can be seen that the obtained encryption scheme which uses two encryption algorithms (i.e., Monoalphabetic and XOR algorithms) is more secure than the scheme that only uses single algorithm (i.e., Monoalphabetic algorithm) alone. This is because the XOR algorithm converts the cipher text resulted by the Monoalphabetic algorithm into its corresponding ASCII values (in binary form) then performs operation on bit by XOR-ing it with randomly generated bits as the keyword. The cipher text resulted by the combination of these two algorithms appears random, thus it is completely dissimilar the original message. Therefore, without knowing the passwords of both algorithms, a hacker or a cryptanalyst will have a much longer computational time in order to derive the original message from
the cipher text. It is also noticed that encrypted message can return to its original plaintext, and, therefore, the data integrity of this scheme is maintained.

Although it can be noticed that the resulting cipher text appears more random, the cryptanalyst attacks are beyond the scope of this study. Therefore, in order to get more information on how well this scheme performs against a considerable number of different attacks, more studies are required.

Since XOR algorithm requires random bits, which are as long as the bits of its input, it is expected that in the future works, these random bits are generated by cryptographically secure pseudo-random number generator algorithms to ensure that the bits are sufficiently random to be used as a secure key for the XOR algorithm. Moreover, it is also considered more secure and more appropriate if in the future works the XOR algorithm can be broken-down into two stages: key scheduling algorithm (KSA) and pseudo-random number generator algorithm (PRGA). By using the newest KSA and PRGA available, the XOR algorithm can be transformed into the state-of-the-art stream cipher.

Acknowledgments
The authors gratefully acknowledge that the present research is supported by Fund Dissemination IPTEKS Research Results for Lecturers / Researchers University Sumatera Utara.

References
[1] Zhang D, Zhang J, Wang Y, Xu S and Li Y 2010 *International Conference on Web Information Systems and Mining* (Sanya) p 71-73. doi: 10.1109/WISM.2010.74
[2] Baelde D, Delaune S, Gazeau I and Kremer S 2017 *IEEE 30th Computer Security Foundations Symposium (CSF)* (Santa Barbara: CA) p 234-248 doi: 10.1109/CSF.2017.22
[3] El-Aziz A E A A A and Kannan A 2012 *International Conference on Computer Communication and Informatics* (Coimbatore) p 1-3. doi: 10.1109/ICCCI.2012.6158815
[4] Paar C and Pelzl J 2010 *Understanding Cryptography* (Berlin: Springer-Verlag)
[5] Roy A and Nath A 2016 *8th International Conference on Computational Intelligence and Communication Networks (CICN)* (India: Tehri) p 390-393 doi: 10.1109/CICN.2016.83
[6] Python Software Foundation *PEP 20 - The Zen of Python* 2004-08-23 Retrieved 2008-11-24.