Kristik

A Sewing Block Cipher Algorithm

Hayyu' Luthfi Hanifah (*Author*)
Informatics/Computer Science Departement
Institut Teknologi Bandung
Bandung, Indonesia
hayyuhanifah52@gmail.com

Choirunnisa Fatima (*Author*)
Informatics/Computer Science Departement
Institut Teknologi Bandung
Bandung, Indonesia
choirunnisa.fatima@gmail.com

***Abstract*—The variation of information form (text, audio, image, or audiovisual) evokes the need of new cryptography algorithms which can encrypt/decrypt any form of information. These new algorithms (modern cryptography algorithms) do encryption or decryption per bit (stream cipher) or per block of bits (block cipher). In this paper, we propose a block cipher algorithm use Feistel Network. Plain text is transformed to and proceeded in hexadecimal (but in the implementation we use integer 0-15 to simplify the computation). (*Abstract*)**

Keywords—modern cryptography, block cipher, Feistel network, hexadecimal

I. INTRODUCTION

The main purpose of cryptography is to ensure the security of any transmitted information. So far there are several cryptography algorithms.

Early in cryptography history, people substituted each character in the plain text with another character based on a key. The result is an unreadable string known as cipher text. The receiver has to decrypt this cipher text based on the same key to get the information from the sender.

Nowadays, the kind of information is varied. Not only text, information may has a form of audio, image, or audiovisual. These kinds of information can't be encrypted/decrypted by classic cryptography algorithms. Here we need an algorithm to perform encryption/decryption in the order of bits because any kind of information can be represented in bits. Besides, by performing encryption in the order of bits, the difficulty to do cryptanalysis on cipher text can be increased.

II. THEORY OF MODERN CRYPTOGRAPHY

A. Block Cipher

In classic cryptography, plain text is proceeded per character or two character. In block cipher algorithm, plain text is proceeded block per block data. The size of block data is equal to the size of key used in the algorithm. Block cipher algorithm maps the block data in plain text into the same size of cipher text.

There are 4 operation modes for block cipher:

1. Electronic Code Book (ECB), a block of plain text always be encrypted to a block of cipher text. For example, given a block of plain text 1101, we will always get 1001 as the cipher text.
2. Cipher Block Chaining (CBC), a block of plain text is encrypted based on the given key and the result (cipher text) from a block before it. The first block of plain text is encrypted based on the given key and an initialization vector (given by user or generated by program).
3. Cipher Feedback (CFB), encrypt data in the smaller unit to handle incomplete blocks. For example, given the size of a block is 64 bits, block cipher using CFB mode can still proceed a block with only 56 bits.
4. Output Feedback (OFB), similar with CFB mode. The difference is in this method, the bits resulted from encryption are appended to the block plain text (to complete the block) while in CFB method the bits resulted from encryption need to be proceeded in a function before they are appended to the block plain text.

B. Feistel Network

Feistel network is a reversible algorithm which is used to construct a block cipher algorithm. The algorithm works by splitting the input into two blocks, left (L) and right (R), then perform a function and XOR to the blocks. It is said to be reversible because the encryption and decryption operations are similar.

III. THE PROPOSED METHOD

The proposed method is based on feistel network algorithm in which it is operated for a certain number of rounds. The difference between this method and other methods is in generating inner-key feistel network. And that it uses 8-bit plain text per operation, the inner function in feistel network will use 4-bit text. The detailed encryption steps of the proposed method are as follows.

1) Transformation

Take 8-bit of plaintext and divide into two binary blocks of 4-bit. Let the first 4-bit be referred as *L*, and the last 4-bit be referred as *R*. Then convert, both of them into decimal representation or 10-base integer. The two integers will then be inputs of feistel network.

2) Feistel network

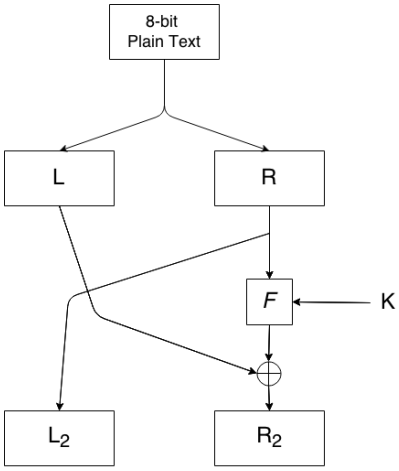

Figure 1 Feistel Network

As we see in above picture, *L* and *R* are placed in the top of the network. *R* will be processed in a function with key *K*, then XOR'ed with *L*. The result is an integer and referred as R_2. We then assign the value of old *R* into L_2. Again, we've got two integers R_2 and L_2.

3) Inner function

This is the function inside the feistel network. The idea is to perform addition between function parameters, these are *R* and *K*. The result of addition, however, could be more or equal than 16. So after addition, the function will perform modulo 16 so that it keeps 4-bit operation. We already know that *R* is an integer which we've got from transformation, but we don't know about *K* yet. *K*, we say in this proposed method, as inner key. The inner function can be represented by the following equation.

$$f(R,K) = R + K \ (mod\ 16) \qquad (1)$$

4) Inner keys

We will need an array of integers which keeps the inner keys. The number of inner keys is equal to the number of 4-bit texts in key, we would rather say it is equal to two times of number of characters in key.

In the feistel network, we've got L_2 and R_2. We then keep L_2 in the array of integer such that it will be the inner key for the next feistel network of the next 8-bit plain text. So on, the inner key of current feistel network will be get from left result of feistel network before. For the inner key of feistel network of first 8-bit plain text, we get it from converting 4-bit key to integer. We use this technique to improve the diffusion of proposed method. You could look at the picture below to see the flow of the inner keys.

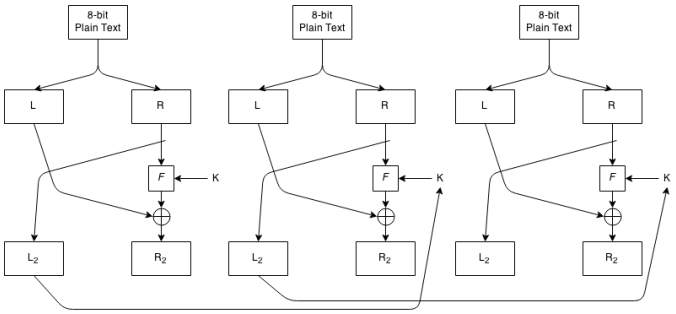

Figure 2 Inner Keys Flow

5) Round

In order to increase the complexity of proposed method, the feistel network is performed in a certain number of rounds. The number of rounds is equal to the number of 4-bit text in the key. So, if the key is 64 bits then the number of rounds will be 16. Then the feistel network is performed 16 times. The two integers result from the first round of feistel network then will be used in second round of feistel network, the next result will be used in third round, and so on.

After 16 rounds, the last result will be the 8-bit ciphertext. As the result is in the form of two integers, we convert it into binary representation, concatenate it such that *L* is in the left-hand-side and *R* is in the right-hand-side. We know that the integer result will always be maximum 16, that is a 4-bit integer. At last, we've got a 8-bit binary representation of cipher text.

The number of rounds, however, will affect inner keys. We said that we need array with size of a certain number. That certain number is actually also the number of rounds. So the initial inner keys are all 4-bit text of the key, after converting it into integer. And the inner keys of others are determined from corresponding order of rounds of feistel network. You can see the illustration of overall proposed method in the picture below.

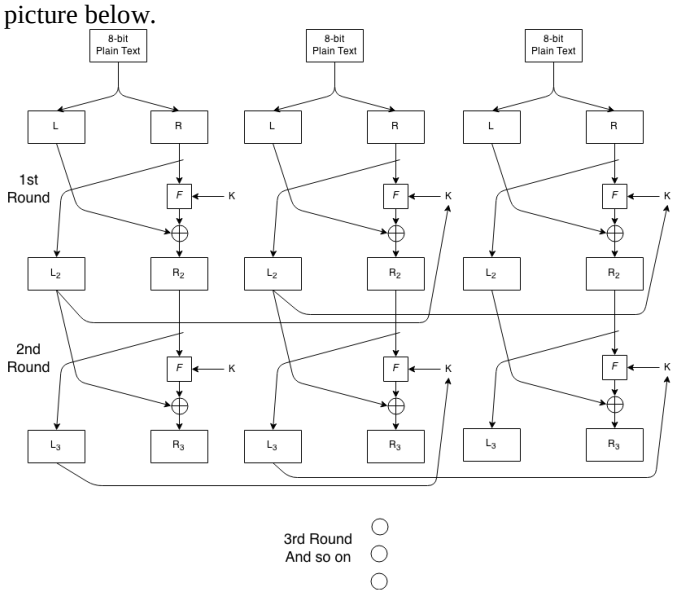

Figure 3 Overall Illustration of Proposed Method

The decryption steps of the proposed method is done in similar way of encryption steps. Inner keys are generated in the reverse order.

IV. Experimental Result and Analyses

We're doing several experiment on every mode of operations. The experimental results are as follows.

A. ECB Mode

Plaintext	Game from scratch
Key	experiment
Cipher text	-å2í=èjÔJ¨z£¹i
Size of Key/Blocks	80 bits
Size of Plaintext	160 bits
Encryption Time	569 us
Decryption Time	614 us

Table 1 Experiment 1

Plaintext	You may prefer to use System.nanoTime() if you are looking for extremely precise measurements of elapsed time.
Key	Another option would be
Cipher text	ÛENOô0ZFÄé¬-¿ÐXYÁDC4` ´]Õu DC2º è8EOTéË1syïIÔöØENOå DC1îhî&ÿETX¦2ýqé9 £DC3Ý× ¯ ® ENOSYNxËBEL¤Q£¦ð
Size of Key/Blocks	184 bits
Size of Plaintext	920 bits
Encryption Time	2755 us
Decryption Time	2589 us

Table 2 Experiment 2

B. CBC Mode

Plaintext	Game from scratch
Key	experiment
Cipher text	ÞRSÍóPÕ§Y*EbÔq®M×W¶DLE
Size of Key/Blocks	80 bits
Size of Plaintext	160 bits
Encryption Time	664 us
Decryption Time	735 us

Table 3 Experiment 3

Plaintext	You may prefer to use System.nanoTime() if you are looking for extremely precise measurements of elapsed time.
Key	Another option would be
Cipher text	¥ö#So#â(LÂ¢SIÖÓ.ã¼ÃSOACK©ÍïZ ÅVDC4MàÊà FÌÕ@ò*Ù!Ô:W$iw¬{§F 1ö*FFp"oÎþþl îEOTLw"¼iâP\z1TG ½fÁ{cù÷RSÝ>6»v¥NÍ7®Bõõwê Ó@ ETXSUB
Size of Key/Blocks	2987 bits
Size of Plaintext	2755 bits
Encryption Time	2755 us
Decryption Time	2589 us

Table 4 Experiment 4

C. CFB Mode

Plaintext	Game from scratch
Key	Experiment
Cipher text	¾Ç=È´dC¨_«WENOÆ²Ü
Size of Key/Blocks	80 bits
Size of Plaintext	136 bits
Encryption Time	3740 us
Decryption Time	4421 us

Table 5 Experiment 5

Plaintext	You may prefer to use System.nanoTime() if you are looking for extremely precise measurements of elapsed time.
Key	Another option would be
Cipher text	ETBênåCAN¼»«PÏ»^ÂøSOdmÂACKÎÝ AC¯ó¦±móMðn´,0ûH¼é3îp-j¸®¶YÚG· k¥n¤é<FFõDC4ðETBhÃ¨±Ä¸"_°fÎ- DC3{GSÐFëBEL© ¸VTðÙ\ L$SYNH(R BELÖÿ
Size of Key/Blocks	184 bits
Size of Plaintext	880 bits
Encryption Time	44813 us
Decryption Time	12303 us

Table 6 Experiment 6

From the tables above, you can see that the size of plaintext are different although using the same plaintext. It happened because when we are using ECB and CBC mode, we have to add some bits in the end of plaintext to make the block size equal to key size. The execution time is likely to be the same between encryption time and decryption time in each mode of operation. It is because the proposed algorithm uses feistel network as main operation which is reversible. So the proposed algorithm works in similar operations between encryption and decryption. The execution time in ECB mode is likely to be shorter than CBC mode, and execution time in CBC mode is shorter than CFB mode. It is because the CBC mode is more complex than ECB mode and CFB mode is more complex than CBC mode.

V. Security Analysis

One of attack against cryptography is ciphertext-only attack. This attack try to decrypt the cipher text without knowing the key.

A popular method to do this attack is frequency analysis. Every emersion of character in cipher text is counted. Then, the character which has the highest number of emersion is translated to the character with the highest number of emersion in a language (the language that is used in plain text, for example in English "E" is the character with the highest number of emersion).

By using Kristik, a character in plain text may be encrypted to many characters depends on the key. So, cipher

text resulted by Kristik algorithm can't be easily decrypted by doing frequency analysis.

Another attack is exhaustive key search. This is a brute force way to find the key. Here is a table of time needed to decrypt a cipher text using exhaustive key search:

Key size	**Number of probable key**	**Time (10^6 attempts per second)**
16 bits	$2^{16} = 65536$	32.7 milliseconds
32 bits	$2^{32} = 4.3 \times 10^9$	35.8 minutes
56 bits	$2^{56} = 7.2 \times 10^{16}$	1142 years
128 bits	$2^{128} = 4.3 \times 10^{38}$	5.4×10^{24} years

Table 7 Table of estimated time for exhaustive key search

VI. Conclusion

In this paper, we have proposed a modern cryptography algorithm named Kristik. This algorithm does encryption/decryption in order of hexadecimal (but in the implementation we use integer 0-15). From the security analysis, we conclude that this algorithm is secure enough.

References

[1] A. Menezes, P. van Oorschot, and S. Vanston, "Handbook of Applied Cryptography" CRC Press. 1996.

[2] William Stalings, "Data and Computer Communication Fourth Edition" Prentice Halll. 2004.