The ElevenHR Block Cipher : Combination of Feistel and Substitution Algorithm

Habibie Faried
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
habibiefaried@gmail.com

Ridho Akbarisanto
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
ridho.akbarisanto@yahoo.com

This paper describes a new block cipher algorithm that is found by these authors of papers. This algorithm is called ElevenHR block cipher algorithm. The algorithm will use 256-bit key and 256-bit block and combine Feistel algorithm with Substitution Algorithm.

Keywords: cryptography; block cipher; feistel network; substitution

I. INTRODUCTION

Everyday, the technology is developed exponentially, including the communication. The communication ways keep increasing and developing everyday. A few years ago we should use the letter for communication, but nowadays we can use electronic mail or even messenger for communication. This easiness make people forget about the security of the communication. The communication can be hijacked by anyone with many ways. Communication also often be used for transmitting data. To protect the transmitted data from hijacking, the plaintext of the data should be encrypted using cipher algorithm before being sent to the receiver.

ElevenHR is a new block cipher created based on the DES and AES / Rijndael Algorithm. This algorithm combine the Feistel algorithm with some substitution algorithm.

II. THEORY

A. Feistel Cipher

Feistel Cipher is a well known algorithm in cryptography. It has symmetric structure and often used in the construction of the block cipher algorithm. It is also well-known as Feistel Network. Many block cipher algorithm use Feistel Network, including DES and Blowfish. The advantage of the Feistel structure is that the encryption operation and decryption operation is very similar, even identical in some cases.

The structure of the Feistel Cipher is using F function, a kind of customizable round function. There is no standard function for F function. Let K_0, K_1, …. K_n be the key for round 0, 1, … n. Then split the plaintext into two equal pieces (L_0,R_0). For each round compute $L_{i+1} = R_i$ and $R_{i+1} = L_i$ XOR $F(R_i,K_i)$. Then the ciphertext is (R_{n+1},L_{n+1}). The decryption is very similar as the encryption process.

B. Substitution Algorithm

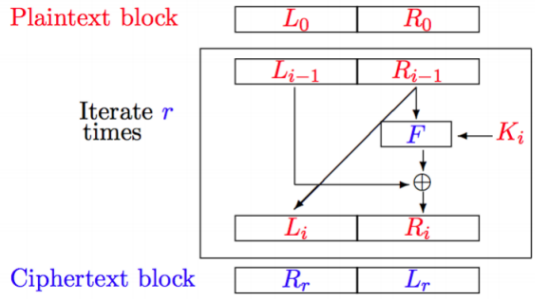
There is no standard for the substitution algorithm. The main goal of the substitution algorithm is to make the output of the cipher harder to solve and random. Many block cipher algorithm use this algorithm, including AES/Rijndael. In AES, the substitution used are SubBytes, ShiftRows, MixColumns, and AddRoundKey. SubBytes will map every byte of the array state using S-Box. ShiftRows will slide every row using in a cyclic method. MixColumns will multiply every columns in array state by polynom a(x) mod $(x^4 + 1)$. AddRoundKey will do XOR operation using a round key to array state.

The algorithms used in AES are some examples of substitution algorithm. There are still many kinds of this algorithm. Because there is no standard for this, the creator of the algorithm can create many ways to make the output harder to solve.

III. CRYPTOSYSTEM ALGORITHM

Our Algorithm consist of two parts. First part is feistel network algorithm with F function. Algorithm of F function will be explained in first part. Second part is subtitution algorithm. Firstly, the overall of this encryption algorithm will be explained.

1. Plain text will be converted into binary numbers. Those binary numbers will be separated as N binary block.. Each block has 512-bit. They have also an integer number to represent the block sequential ID for each block.
2. Then, divide block into 2 sub-block. Left block and Right block, each block has 256-bit length
3. Run the feistel network for 11 rounds. Feistel network algorithm like this



4. Properties:
   a. L0 = Left block
   b. R0 = Right Block
   c. F = Customizable encryption function (will be explained in section II.A below)
   d. K = key, each round of feistel network has different key
5. See that F function. We will explain F function in section II.A below
6. Then, run *substitution algorithm* for the block matrix for 11 rounds. This algorithm will be explain below in section II.B
7. Join all block matrix into one big block. Then, write that to file.

A. *F function in Feistel Algorithm*

The main goal of F function is to randomize the cipher block output. See that F function needs different keys for each blocks and rounds. So, this is the algorithm:

1. F function *key* is generated from *seed key*. The *seed key* has N *shift-left bits*. N is equal to (block_id + round_number). Then we have a key

   Figure for shift-left *seedkey*:

Round	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6
1	1	0	1	0	1	1
2	0	1	0	1	1	0
3	1	0	1	1	0	0
4	0	1	1	0	0	1

2. Then, the blocks that come in to F function will be *shift-righted* for N bits. N is equal to (block_id + round_number). Then we have shifted blocks

   Figure for shift-right bits:

Round	Bit-1	Bit-2	Bit-3	Bit-4	Bit-5	Bit-6
1	1	0	1	0	1	1
2	0	1	0	1	0	1
3	0	0	1	0	1	0
4	1	0	0	1	0	1

3. Do the XOR operation between key and shifted blocks. Then return

B. *Substitution Algorithm*

The main goal of this algorithm is to randomize and harden the cipher block output. This is the algorithm:

1. Do the XOR operation between blocks and *seed key*

   Block (0) ^ Seed Key (0)

   Block (1) ^ Seed Key (1)

   ……

   Block (n-1) ^ Seed Key (n-1)

   Block (n) ^ Seed Key (n)

2. Then, build (Nx4) matrix blocks from that block. Each block cell has 256-bit length. Matrix will look like this

B1	B2	B3	B4
B5	B6	B7	B8
B9	B10	B11	B12
B13	B14	B15	B16
B17	B18	B19	B20

3. For each I row will be shift-lefted for (2I-1) rounds. For example in row 3 will be 5 shift-left bits.

B1	B2	B3	B4	
B5	B6	B7	B8	
B9	B10	B11	B12	Shift left 5(2*3-1) times
B13	B14	B15	B16	
B17	B18	B19	B20	

4. For each block (I , J) will be swapped with (I+1 , 3J-1). For example in the table below two cells with the same color will be swapped

B1	B2	B3	B4
B5	B6	B7	B8
B9	B10	B11	B12
B13	B14	B15	B16
B17	B18	B19	B20

5. Then return the matrix as a block

IV. KEY GENERATOR

Key generator always generates 256-bit *seed key*. This *seed key* is used to generate the real-key in cyptosystem algorithm above.These are steps how to generate *seed key*:

1. Program will ask for *user password*. This password is used to *decrypt* and *encrypt* the plain text/file
2. Then, that password will be converted to binary numbers {0,1}. We call that *seed key*
3. If the length of the *seed key* == 256. Then approve
4. Else if the length of the *seed key* > 256. Then truncate the *seed key* until it's length == 256.
5. Else, we must generate additional binary numbers until it's length == 256.

6. When we generate additional binary numbers. Our algorithm just like OFB (Output Feedback Cipher) algorithm.
7. Let H = *seed key (length < 256)* and Let n = length of current *seed key*. In example,

$$H(n+1) = H(1) \text{ XOR } H(n)$$

$$H(n+2) = H(2) \text{ XOR } H(n+1)$$

…

$$H(n+k) = H(k) \text{ XOR } h(n+k-1)$$

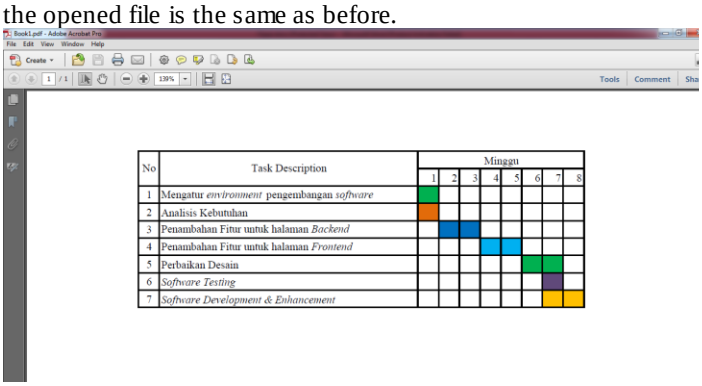
Continue until (n+k == 256).

V. TESTING

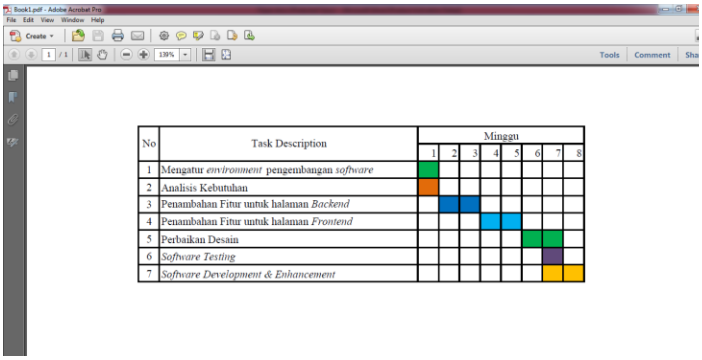
We have to measure our cryptosystem algorithm with 3 steps. First step is measuring the compression rate and the second is measuring computational time for doing this algorithm

a) Encryption and Decryption Result

Before we measure anything, we should assure that the encryption and the decryption is success. By trying to open the decrypted file we can make sure of that the algorithm is good. Using 2 files, the decrypted document can be opened again and the opened file is the same as before.



Before Encryption



After Decryption

b) Compression Rate

We have to measure the size of compressed cipher file. In this paper we use ZIP to compress cipher file. If compression rate of cipher file is nearly 0%, then we can conclude that our cipher file has very random blocks and prove that our algorithm is good enough. So, we have 2 files to measure the compression rate after being encrypted

1. PDF File (7 KB): 0.002 %
2. ZIP File (289 KB): 0.01 %

c) Computational Time

Our new algorithm must be proven fast and efficient computation for ciphering and deciphering process. Slow computation will give indication this algorithm is bad and useless. So, we have 2 files to test the computational time.

1. PDF File (7 KB): 794 ms for encryption and 779 ms for decryption
2. ZIP File (289 KB): 401362 ms for encryption 401377 ms for decryption

VI. SECURITY ANALYSIS

There are many methods of cryptanalyst that can be used to find the key. The security analysis will be based on the brute force attack.

a. *Brute Force Attack*

The duration needed for finding the key depends on the length of the key. The length of ElevenHR algorithm key is 256-bit, so there will be 2^{256} or about 10^{77} possible key combinations. The table below will show the time needed for finding the key if all the key should be tried.

Rate	Time
1 key/sec	**3.17×10^{69} years**
10 key/sec	**3.17×10^{68} years**
1000000 key/sec	**3.17×10^{64} years**
1000000000000 key/sec	**3.17×10^{58} years**

From the table above the time shows that it will be impossible to find the key by using the brute force attack and the algorithm is secured from the brute force attack

b. *Uniqueness*

The generated cipher block is almost unique cipher. It's guaranteed that all blocks almost have different cipher blocks. The difference makes this cipher is very hard to be broken by frequency analysis attack.

VII. CONCLUSIONS

There are many block cipher algorithms created. ElevenHR is one of the algorithms. Many of the algorithms is good, including ElevenHR. The analysis shows that ElevenHR can be considered a good algorithm, by the security or the performance. In the future, additional research and development of cipher block algorithm is must. Fast and efficient cipher block algorithm is needed because of larger and bigger data we will have.

References

[1] Menezes, Alfred J.; Oorschot, Paul C. van; Vanstone, Scott A. (2001). *Handbook of Applied Cryptography*