BDG48 - Design and Concept of Genetic Cryptography

Satria Ady Pradana (13510030)
Informatics Engineering
School of Electrical Engineering and Informatics (SEEI)
Bandung Institute of Technology, ITB
Bandung, Indonesia
satria@arc.itb.ac.id

***Abstract*—This paper presents design and concept of Genetic Cryptography and an example algorithm to satisfy this purpose. This papers will propose a new and unique way for encrypting and decrypting message compared to conventional concept. The concept will purely based on genetic concept and the will be focused into block cipher. The fundamental concept used will be cross over and mutation, without any restriction for further developed concept later.**

***Index Terms*—cryptography, block cipher, genetic, cross over, mutation, bdg48.**

I. INTRODUCTION

Nowadays in digital era where time and space is not limitation, information is highly available. Any kind of information can be obtained using various method. The information might vary from general knowledge to sensitive information including private information and confidential information. Internet as one of communication media has proved how information can be obtained easily.

Protecting confidential and private information is a human's natural behavior. There are many ways to do it, but the common method is using cryptography.

Cryptography has evolved from ancient time, where Julius Caesar use . Cryptography is also widely used nowadays, where originally it is used only for military works. With aid of computer, more cryptography algorithms have been proposed.

Although dozens and hundreds cryptography algorithms available, most of them used similar concept and methods. The difference is only at design and implementation of the concepts. The methods mention before are mostly: XOR operations, substitution, permutation, and feistel network.

Although we do not said it is good or bad, a new concept can be used for create cryptography algorithm.

A new concept will be proposed, a cryptography algorithm using genetic concept.

Genetic concept is a concept that mimics the process of natural evolution. The concept can be described as combination of cross over concept and mutation concept - which is also described on Genetic Algorithm.

This paper will utilize BDG48 - algorithm developed for this paper. The algorithm is categorized as Block Cipher algorithm and will follows some fundamental of cipher block algorithm design. Although there are some aspect that might not be completely fit as is.

II. CIPHER BLOCK DESIGN PRINCIPLES

Designing a block cipher algorithm should follow some "de facto" rules. There are five fundamental principles that acknowledge as basis for creating a new cryptography algorithm. These principles are:

1. Shannon's Confusion and Diffusion Principle
2. Iterated Cipher
3. Feistel Network
4. Weak key
5. S-box

Confusion and diffusion are two properties proposed by C.E. Shannon for secure cryptography algorithm [1].
In Shannon's original definitions, confusion refers to making the relationship between the ciphertext and symmetric key as complex and involved as possible. This principle also includes obscuration of statistic pattern of ciphertext.
The diffusion refers to making the relationship between the ciphertext and the plaintext complex as complex and involved as possible. A single and slight modification might result in unpredicted outcome.
The iterated cipher is a iteration or repeated operation of an invertible transformation on fixed-size block. The transformation function is known as round function. For each iteration or each round the round function will be operated on fixed block data and transform it to another form with same size. For each iteration, an internal / self generated key is used on function.
The key for each round on iterated cipher might be self generated key using user-supplied key as feed, or using external key directly. The most common method used are generating key rather than used key directly.
Feistel network is defined as a symmetric structure in the construction of block ciphers. It has advantage of similarity between enciphering and deciphering process, even identical. The smallest structure is called feistel block and consists of some operation. The operation might vary from one algorithm to other algorithm.

Weak key is a key, which used with a specific cipher makes the cipher behave in some undesirable way. It is a key in which makes a series of operation of enciphering will result back to plaintext. The good cryptographic algorithms doesn't have this properties.

S-box or substitution box is a simple substitution matrices which maps one or more bits to other one or more bit. The matrice can be viewed as a lookup table. The table might consists of some rows and some columns. When a block of bit comes, it can be broken down with some way. The substitution value is obtain by looking up to intersection of row and column. This methods is used widely in DES (Data Encryption Standard) but will not applicable in this paper.

II. PROPOSED ALGORITHM

BDG48 or Bit Diffusion Genetically 48, is an algorithm using Genetic Cryptography concept. The algorithm utilize 4 (four) systems and 8 (eight) internal keys with external keys supplied by user.

The four systems refers to: internal key generation and scheduling; modified feistel network; cross over; and mutation. These keys are then used for encryption, rather than user-supplied key.

BDG48 is a symmetric key algorithm. To encrypt a message, a 64-bit key must be supplied by user. The key generation process will then create 8 subkeys for used internally.

BDG48 also works on bit level. This block cipher will encrypt 64-bit data for each block. If data block doesn't have sufficient size, the zeroes will be padded to block until the block is filled.

For each 64-bit block 32 iterations process will be used. In each iteration, a special "permutation" system applied. The system in this context refer to cross over and mutation process.

BDG48 does not used substitution method. It will only relies on "permutation", especially described as crossover and mutation.

III. KEY GENERATION AND SCHEDULING

Key generation is a process of generating key for internal usage from *external key*. The external key is a key given by user. The key used for encryption and decryption is same or identic. This process is not obligatory thing. But mostly the key generation will make the encryption stronger.

The external key are only used once in the key generation process and won't be used on encryption. But the user should always remember the key as the symmetric key cryptography only permits data encrypted and decrypted by same and identic key.

The resulting key might be vary from one key to not-limited amount of keys. These keys then called as subkeys.

The key generation might involving some function and the key generation itself can be considered as a function. The design of key generation must following confusing and diffusion principle. The generated key must ensure the encrypted key is diffused. The level of diffusion might be hard to be measured, but this principle can be measure by the pattern and mapping from plaintext character to ciphertext character.

While the key generation can used various ideas, three subjects are proposed. These three are: logical shifting, permutation, and substitution.

Logical shifts are bitwise operation, operation which works on bit-level. Logical shift operation shifts all the bits of its operand to a specific direction. There are two direction which are left and right. The left shift is operation which shifting is done to left direction while the right shift is to right direction. The number of shifting is given as an unsigned integer n.

The logical shifting should use wrapping method to fill the vacant bit-position. The wrapping method suggest that the out-of-bound bit are positioned back to vacant slot instead of filling them with predetermined value. The sequence must be kept therefore the first thrown bit will be the first bit padded to.

Permutation is a process of rearranging objects / bit values. The permutation process can be divided into three category: pure permutation, compression permutation, and expansion permutation.

A pure permutation or simply a permutation is a permutation which only do rearrangement of bits. This is the basic permutation known.

A compression permutation is a special permutation. While also permutate the bits, this method also select which bit should be produced as result. In this type of permutation, the operand size will always be greater than the product size. Therefore, some bits might not be included a production. The remaining bits are either discarded or used as parity to check bits.

An expansion permutation is the reciprocal to compression permutation. Expansion permutation is a process of permutation which duplicate some or all bits of operand. In this type of permutation the size of product will always be greater than the operand size. A bit on som position might be duplicate two or more times, depends on the design.

The substitution is a process for substitute or exchange a value to other value. The substitution rules might defines what values should be substitute and what values as replacement.

In BDG48, key generation will use left shift operation, pure permutation and compression permutation.

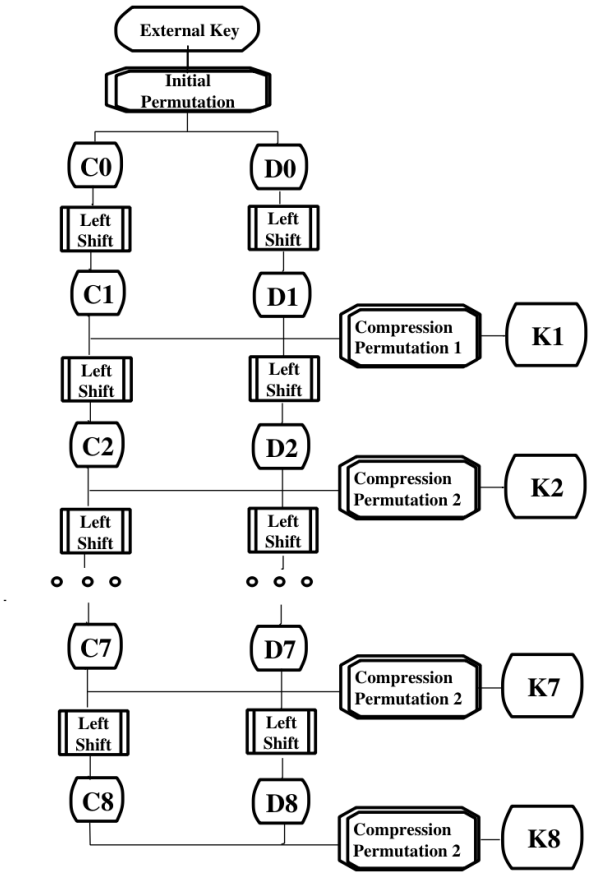
To encrypt a message, BDG48 use 64-bit key which is equivalent to 8 ASCII characters, no less and no more.

In generation process, BDG48 will generates 8 (eight) internal keys. The keys will be labeled as K1, K2, ..., K8. The internal keys also generated based on external keys with the help of three matrices for permutating.

The matrices used in key generation are one pure permutation matrix and two compression permutation matrices.

The permutation matrix is a 1x64 *row matrix*. This matrix will be used at initial permutation process. The permutation matrix is an ordinary integer matrix where each column represent a bit position in which the bit on original message will be permutated to. Thus a bit on nth position of source will be rearrange to position pointed by value of M_{1xn}.

Compression matrices is similar to permutation matrix but differs in size. Each Compression Matrix is a 1x32 *row matrix*. The Compression Permutation then will reduce output into only 32-bit block. The remaining 32 bits are then discarded.



Picture 1: Key Generation for BDG48

External key is processed with Permutation matrix (initial permutation). The result is a 64-bits key. The result then divided into two halves: C0 and D0. C0 is the first 32-bits of the result key (upper bits) and the remaining 32-bits is D0 (lower bits).

The C0 then left-shifted by 1 and similar thing done on D0. The shifts are circular so the the most significant bit will be the least significant bit after the process. The process will generate a pair C1 and D1 respectively.

The ith internal key is obtained by concatenation of Ci and Di and later used as operand of Compression Permutation matrices.

There are two Compression Matrices, denoted by CP-1 and CP-2. CP-1 is used for generating key K1, K3, K5, and K6 while CP-2 is used for generating key K2, K4, K7, and K8.

Formally, the generation process can be formulated as:

K1 = CP1 (C1 . D1)
K2 = CP2 (C2 . D2)
K3 = CP1 (C3 . D3)
K4 = CP2 (C4 . D4)
K5 = CP1 (C5 . D5)
K6 = CP1 (C6 . D6)
K7 = CP2 (C7 . D7)
K8 = CP2 (C8 . D8)

$C_i = C_{i-1} << 1$

$D_i = D_{i-1} << 1$

The "." refer to concatenation of bits string and << refers to left shift operation.

BDG48 has 32 iterations. For each iteration, a key is required. As there are only 8 subkeys available, a scheduling is required to make sure every iteration will use one key.

Scheduling is an act of ordering the key usage for each iteration.

Following is the scheduling used in BDG48 for selecting subkey.

Iteration	1	2	3	4	5	6	7	8
key	1	5	3	4	7	6	2	8

Iteration	9	10	11	12	13	14	15	16
key	6	1	2	8	7	3	5	4

Iteration	17	18	19	20	21	22	23	24
key	2	5	4	6	8	3	7	1

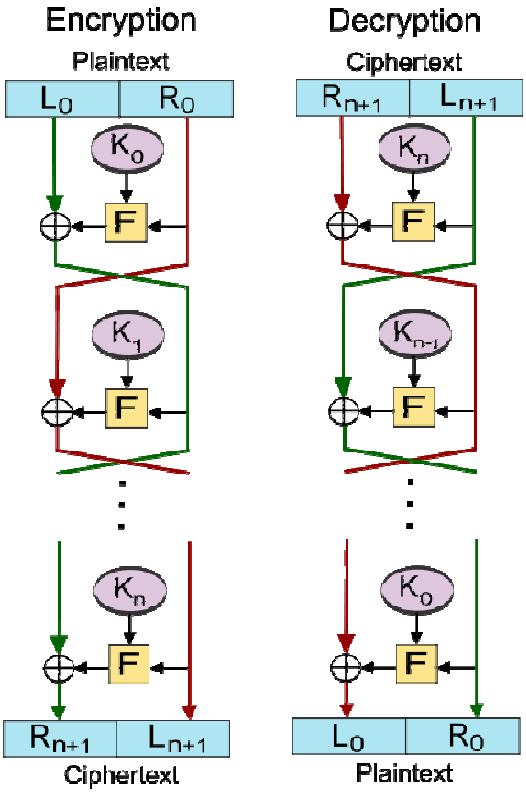
Iteration	25	26	27	28	29	30	31	32
key	8	7	2	1	3	5	4	6

Using the table, the 9th iteration / round will use K6 as its key.

IV. FEISTEL NETWORK

Feistel Network is symmetric structure and a basic for construction of block ciphers. The main advantage of feistal network is the feistal network offers similar process to do encryption and decryption.

In feistel network, data block is divided into two equal-length block, L and R. For each round, a round function is defined.



Picture 2: Common Feistel Network

The round function is a global function works on each round. The design of round function might vary, but it should implement confusion and diffusion principles.

The round function should need at least a key as its argument. The function should behave differently when different key is applied and produce different product when a combination of operand and key is given.

Mostly, a round function operates on one side / one part only. The resulting part then will be XORed with other part. Later, both part will be swapped. In other word L' is the product of R and R' is the product of L.

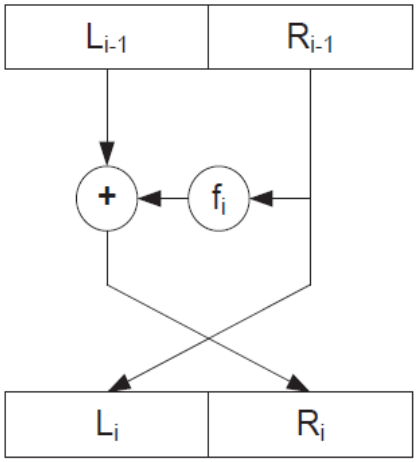


Figure 3: One round of Feistel Network

Let F be the round function and K0, K1, ... , Kn be the subkeys for rounds 0, 1, ..., n respectively. We can formulate the operation as follow:

$L_i = R_{i-1}$

$R_i = L_{i-1}$ XOR $F(R_{i-1}, K_i)$

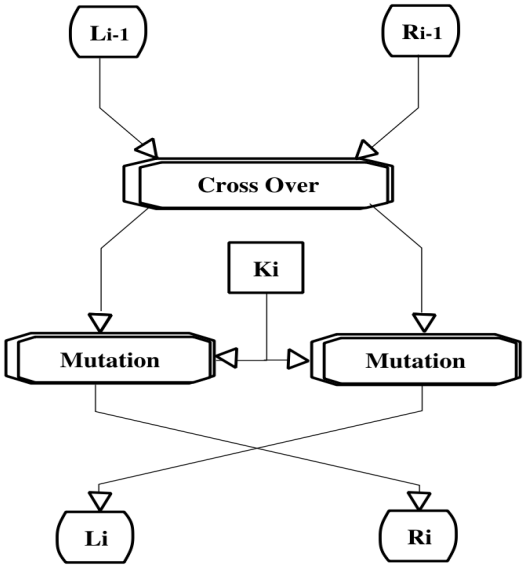
However genetic cryptography does not fully fit this definition. Genetic cryptography relies upon cross over and mutation. Those methods works on both part of data chunks. Therefore we can not follow the original definition of round function and structure proposed by feistel network.

The design of feistel round proposed here is a modified version of original feistel round.

The data block is divided into two equal-length block L and R. The L is the first halve of data block and R is the remaining halve.

A crossover operation will be applied to both part. The L and R can be viewed as arguments for cross over system. This function does not need any key to operated. More explanation of cross over will be explained briefly on next section.

The resulting data part, L' and R' will then mutated independently. The mutation need at least one subkey. The mutation operation can be viewed as round function with supplied key. The mutation will be explained more on next section.



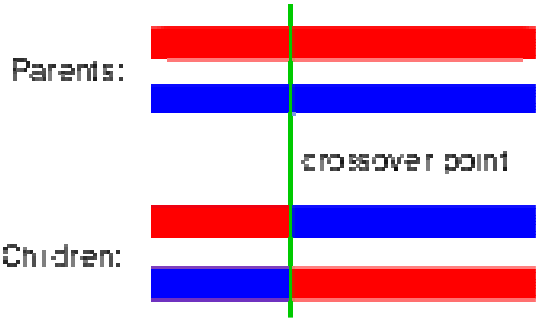
Picture 4: Modified Feistel

BDG48 use standard modified feistel network proposed above. The key used for mutation is obtained from key scheduling mention on previous section.

In formal notation, we can formulate the process as:

V. CROSS OVER

In Genetic, crossover or crossing over is the exchange of genetic material between homologous chromosomes that results in recombinant chromosomes. Cross over is one of the final phases of genetic recombination.



Picture 5: Crossover in Genetic

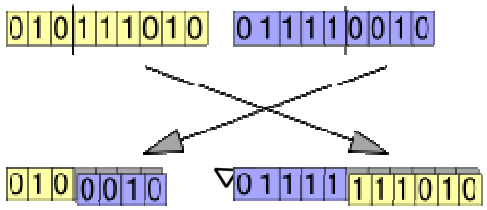
The origin or source chromosomes denoted as parent. The parent chromosome might be and should be distinct.

A crossover occur on a point. This point is referred as crossover point. At each parent, chromosomes are divided into some parts, relative to the crossover point. The crossover might not divide chromosome to equal length part, and there is no guarantee that it will always do so.

In picture 5, there are two parents denoted by their color (red and blue). A crossover point occur which divide parent into two. The crossover point is not precisely on middle of chromosome thus cut the chromosome into two different size part. The first half is shorter than the second half.

In generation of children, both parent exchange their part. The result of this operation is two children chromosome. The children parent now has both their parent part.

The crossover point might occur in more than one position. But this paper will limit into only one crossover point.



Picture 6: Crossover on Bit Array

Using crossover concept on Genetic Cryptographic is simply implementing the above concept.

The two equal-length block L and R can denoted as parent chromosomes. The product of crossover is two children or two equal-length block L' and R'. The childrend L' and R' has both parent parts.

Crossover point is defined as an index of bit position, denote by i. The division is not more than substringing the bits array. The data must be divided into n+1 part where n is the amount of crossover point.

C = Cross Over
M = Mutation

Li = M(C(Li-1, Ri-1, i), Ki)
Ri = M(C(Li-1, Ri-1, i), Ki)

Following is the formalized steps for crossing over with one crossover point:

1. For data block with length m bits, choose i where $0 < i < m$
2. Get substring of L from 1 to i, named it as L1
3. Get substring of R form 1 to i, named it as R1
4. Get substring of L from i+1 to m, named it as L2
5. Get substring of R from i+1 to m, named it as R2
6. L' is obtained from concatenation of L1 and R2
   L' = L1 . R2
7. R1 is obtained from concatenation of R1 and L2
   R' = R1 . L2

The selection of value of i must not be truly random. The value i should be able to be replicated or the value must be obtained using identical series of operation.

A rule should be defined to schedule the values of i.

BDG48 use basic crossover concept. The alternating value of i is defined using the following rules / schedule:

Iteration	1	2	3	4	5	6	7	8
i	1	2	3	4	7	16	18	20

Iteration	9	10	11	12	13	14	15	16
i	1	2	3	4	7	16	18	20

Iteration	17	18	19	20	21	22	23	24
i	1	2	3	4	7	16	18	20

Iteration	25	26	27	28	29	30	31	32
i	1	2	3	4	7	16	18	20

VI. MUTATION

In genetic term, mutation is change of the nucleotide sequence of the genome of an organism, virus, or extrachromosomal genetic element. The change might resulted form unrepaired damage to DNA or RNA genomes, from errors in the process of replication, or from insertion or deletion of segments of DNA by mobile genetic elements.

Mostly the change is permanent unless a reverse operation is used. The changes might also result on change of the system behavior.

In Genetic Cryptography, a mutation is defined as change of data block part as a result of invertible function applied to it. The mutate part might be all part (whole data block) or subset of it.

The invertible function might be vary in design but it should has diffusion and confusion principle. The function should need a key with equal length to the data part. At least one key should be available, however two or more keys are not encouraged.

For a mutation, there should be a pointer used as starting point and another index as end point. The area covered by the start point to end point is called as *mutation area*. The mutation might covered a whole data part or just the subset of it.

When a mutation occurred, a mutation area from data block should be operated with a function together with mutation area. The mutation area should always match the corresponding index (start and end index).

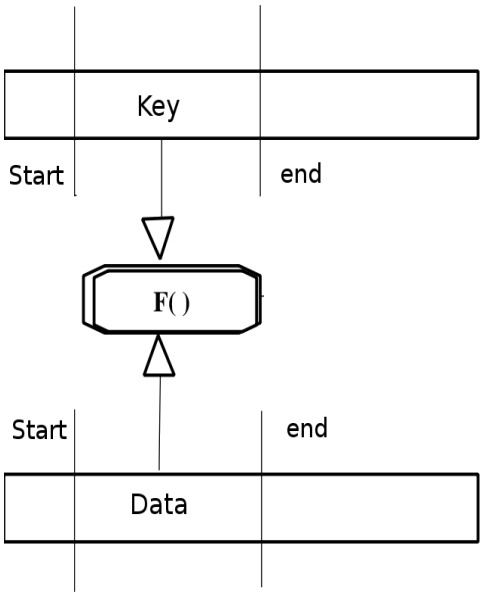


Figure 7: Mutation

Suppose integer i and j is denotes starting point and end point respectively. Index i must always less than index j. Both index are pointing to valid bit position on data and key. If we denote M as message / data block with length m and key having the same length, mutation can be formulated as:

L = substring(M, 0, i)
R = substring(M, j, m-j)
X = substring(M, i, j) XOR substring(Ki, i, j)
M' = L . X . R

Choosing i and j pair should not have any pattern for each round. The pair should be random enough but should be deterministic. This means a replication means should be able to obtain the similar pair of i and j for other block.

In BDG48, a mutation is done by applying XOR function into a mutation area.

The index i and j are summarize on following table:

Iteration	1	2	3	4	5	6	7	8
i	1	1	4	30	5	4	7	1
j	2	3	7	31	28	31	9	8

Iteration	9	10	11	12	13	14	15	16
i	2	5	13	6	4	21	22	13
j	5	6	17	13	23	28	26	23

Iteration	17	18	19	20	21	22	23	24
i	14	19	1	7	13	27	4	4
j	27	31	31	25	29	28	17	24

Iteration	25	26	27	28	29	30	31	32
i	6	9	9	1	4	3	5	6
j	7	13	31	2	17	6	27	28

V. CONCLUSION

In the end, genetic cryptography is an interesting field. This paper has presents a new way for encrypting and decrypting a message using genetic concept.

In the future, if the concept is developed deeper, there would be other variations that might occurs.

VII. ACKNOWLEDGMENT

I wish to thank everyone who helped me complete this paper. Without their continued efforts and supports I would have not be able to bring my work to a successful completion.

Dr. Ir. Rinaldi Munir, MT: for guidance and lecture

Ezra Hizkia Nathanael: for support and ideas

Mufi Yanuar Triputranto: for support and ideas

REFERENCES

[1] Shannon, C.E. "Communication Theory of Secrecy Systems,". 1949.

[2] Knudsen, Lars R. The Block Cipher Companion. Springer. 2011.

[3] John Holland, Adaptation in Natural Search Algorithm Systems, University of Michigan Press. Ann Arbor, Michigan. 1975

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Maret 2013

ttd

Satria Ady Pradana
13510030