BREAKING A FEISTEL-TYPE BLOCK CIPHER BY BACTERIA ENGINEERING

Arash Karimi[*] and Hadi Shahriar Shahhoseini[**1]

Electrical Engineering Department,
Iran University of Science and Technology, Tehran, Iran
[*]ar_karimi@elec.iust.ac.ir, [**]hshsh@iust.ac.ir

ABSTRACT

In this paper we propose a theoretical method for breaking a block cipher based on a Feistel structure which is a variant of DES (S-DES) using one pair of (plaintext, ciphertext). Our scheme utilizes bacterial computing for the first time in cryptanalysis. For this reason, we design and simulate an engineered gene regulatory network to break S-DES which possesses a superior performance than the other methods based on DNA computing because it utilizes the power of massive parallelism of DNA molecules as well as capability of cellular division of bacterial cells which gives our proposed S-DES cracker system the flavour of massive parallel processing.

KEYWORDS

Feistel ciphers, brute-force cryptanalysis, DNA computing, genetic engineering, Mathematical modelling

1. INTRODUCTION

Computer science community has always looked for innovative approaches to facilitate solving problems which need a significant burden of computation. In this regard, genetic engineering techniques or recombinant DNA technology has made tremendous progress during the last decade to design genetically engineered machines to do simple computations inside a living organism [1]. The inherent cell division process that takes place in living cells of bacteria make bacteria valuable tools in constructing machines to solve computationally hard problems that require exponential steps in the size of the input. A practical problem that is considered computationally hard is *Cryptanalysis* which is the art and science of breaking ciphers and is assumed to be a practical problem and usually takes a long time to be accomplished using single silicon computers because of exponential computation steps in the size of the input it demands and also for impractical and unusual side information that is mandatory for applying non-naïve cryptanalytic techniques. For this reason, in order for a computationally hard problem to be solved within an appropriate period of time, we should make use of the computational power offered by a group of silicon computers working in a parallel or distributed fashion. This approach needs those computers to be synchronized which is a demanding task and also it utilizes a big burden of computational resources which is expensive and scarce. In this respect, the idea of using DNA molecules to solve computationally hard problems which was first proposed by Adleman [2] opened new horizons in biologically-inspired computation. After this evolutionary breakthrough, the idea of utilizing bacteria to solve NP-complete problems was introduced that considered the process of cell division which takes place naturally in bacteria cells as generative power required for doing computations.

In [3] M. Elowitz and S. Leibler constructed an artificial genetic oscillator in cells using a synthetic network of repressors and they called it the *repressilator*. In another issue of Nature,

[1] Corresponding author: H.S.Shahhoseini, email :hshsh@iust.ac.ir

T. Gardner et al. constructed a toggle switch made of genes. The toggle in the second article forms the basis for genetic applets- self-contained, programmable, synthetic gene circuits for the control of cell function [4]. Weiss in [5] showed how to use the quorum-based cell-to-cell communication mechanism between bacteria to provide a framework to construct bacterial devices. In [6] a proof-of-concept experiment was conducted that utilized genetic engineering methods to solve the burnt pancake problem (BPP) which is a permutation sorting problem and is in general, NP-complete and in [7] a bacterial computer was proposed and built with similar methods of [6] for solving the Hamiltonian path problem which is another NP-complete problem.

Boneh et al. in [8] gave the first proposal for a DNA computer to break a cipher that they chose to be the data encryption standard (DES) [9]. Roweis in [10] proposed a model for molecular computation. In [11] an algorithm for breaking DES was proposed using P-systems. In [12] a method utilizing networks of evolutionary processors with parallel string rewriting rules (NEPPS) has been proposed to break the data encryption standard (DES). In this paper we aim to propose and simulate a gene regulatory network to break simplified data encryption standard (S-DES). The rest of the paper is organized as follows. In section 2, we introduce preliminary backgrounds which are mandatory for understanding the concepts of the proposed S-DES cracker system. The proposed model is presented in section 3. In section 4 simulations of the paper are presented. Section 5 is devoted to the performance evaluation of the proposed scheme; and finally in section 6 conclusions are drawn.

2. PRELIMINARY BACKGROUNDS

In this section, the background materials which are mandatory for conceiving the proposed method of are explained. In the first part, the process of quorum sensing in E. coli is explained and then, a brief note on the considered cipher which we aim to cryptanalyze is presented.

2.1. Quorum sensing in bacteria

The LuxR quorum sensing system is common among the Vibrio family. The LuxR system is underpinned by three quorum sensing molecules which are those that are the signalling molecule HSL, the HSL synthase LuxI and the transcriptional regulator LuxR [13]. HSL is generated in the cell by either diffusion through the cell membrane from the external environment or within the cell. Whilst in the cell it may bind to LuxR. Two HSL molecules form a tetramer with two complementary LuxR molecules that form a complex which can activate the lux operon. This operon contains the genes that encode LuxI plus the luxR gene that is located upstream and is constitutively produced as well as those that are responsible for production of GFP [5].

2.2. The considered cipher

We have chosen the reduced version of the DES algorithm (S-DES) in order to mount and simulate our cryptanalytic attack. The above-mentioned cipher has been introduced and described in [14]. It has similar properties of DES but deals with a smaller block and key size (it uses a 10-bit key and operates on 8-bit message blocks).

3. THE PROPOSED MODEL

Our approach to break the S-DES cipher is a brute force one in which possessing a pre-specified (plain-text, cipher-text) pair, we wish to find the secret key. To achieve this goal we use the graph shown in Figure 1 which is similar to the NEPPS graph of [12] with some modifications.

We represent the underlying graph of Figure 1 with Equation (1) and similar to [12], we aim to design a network of processors with parallel string rewriting rules. In each node of the graph of Figure 2 a specific part of the attack takes place such that in node Y_{out} the secret key can be found. In node Y_{in} we produce all possible keys that are shown by a_i ($1 \le i \le 10$) in which i shows the position of the key bits and a_i encodes zeros and ones. The procedure for generating all possible keys is demonstrated through the below algorithm.

Algorithm 1. Generation of all possible initial keys:

In order to generate all possible keys in node Y_{in} we use the graph of Figure 2 in which as proposed in [15], all possible paths that initiate from a_1 and terminate at a_n show all n-bit binary numbers such that $a_i b_i a_{i+1}$ and $a_i b'_i a_{i+1}$ encode logical *zero* and *one*, respectively, in the i^{th} bit of the key.

All in all, there exists 3n-2 vertices in graph of Figure 2 that from now on in this paper we call it Lipton graph and each vertex of Lipton graph is encoded by DNA molecules such that it represents a gene and furthermore, without loss of generality, we assume that vertices a_1 and a_n encode input and output vertices of the graph, respectively. The genes that encode vertices except a_n are divided in two halves of $3'$ and $5'$ and each edge that starts from x_i and ends with y_j in which $j = i$ or $j = i+1$, which we demonstrate by $x_i y_j$, is defined as follows: the second half ($3'$) of the gene that encodes x_i if placed in the left hand side of the first half ($5'$) of the gene that encodes y_j, define the edge that encodes $x_i y_j$. Just like [16-18], we utilize the Hin/hixC recombinase systems of Salmonella typhimurium bacterium in order to flip between different combinations and therefore make all possible keys. Indeed, in nature, Salmonella typhimurium bacterium is capable of inverting one segment of its genome DNA so as to switch between one gene to another and in this way it can make its intracellular surface covered with another protein. This phenomenon can be utilized to invert a gene that is placed between two hixC sites such that under exposure of hin proteins, the DNA segment between a pair of hix sites is cut, inverted and reattached. Once we encode Lipton graph with the abovementioned method, the most significant advantage of bacterial computing can be utilized in which knowing the fact that an E. coli bacterium divides every 20-30 minutes, a billion E. coli can grow overnight and therefore, a billion equal and independent bio-processors can be made ready in a single culture. Now, if each cell of E. coli is exposed to hin protein, each cell inverts its DNA segment that is between two hix sites. In this way, the DNA edges that are placed between hix edges frequently reverse their direction and hence, random directions and configurations of the graph edges are produced. To make sure that all represented paths of Lipton graph encode all possible n-bit numbers, the plasmid that is going to encode Lipton graph begins with 5' half of the gene that encodes a_1 and a transcription terminator (TT) is used to encode a_n. By generating all possible paths of Lipton graph that begins with a_1 and ends in a_n, a bacterial population is produced and therefore in this stage, all possible keys have been produced autonomously from only a single initial key string and through bacterial cell proliferation.

To break S-DES we assume the following initial configuration for nodes: initially, all nodes are empty except node Y_{in} that contain $\{a_0 a_1 \ldots a_9 m_0 m_1 \ldots m_7 e''_0 e''_1 \ldots e''_7\}$ in which m_i and e''_i ($0 \le i \le 7$) demonstrate 8-bits of plaintext and cipher-text respectively and a_i ($0 \le i \le 9$)

show key bits. Additionally, the solution that contains all possible keys is kept in node Y_{in} and therefore, initially in all strings of node Y_{in} only the substring $m_0m_1...m_7e''_0e''_1...e''_7$ is fixed and the number of strings which is in node Y_{in} equals the number of all combinations of the substring $a_0a_1...a_9$. In addition to generation of all possible keys in node Y_{in}, the initial permutation operation takes place in Y_{in} according to Equation (1).

$$m_0m_1m_2m_3m_4m_5m_6m_7 \to m_7m_6m_4m_0m_2m_5m_1m_3 \quad (1)$$

The permutation mentioned in Equation (1) can be accomplished by the Hin/hix recombinant systems

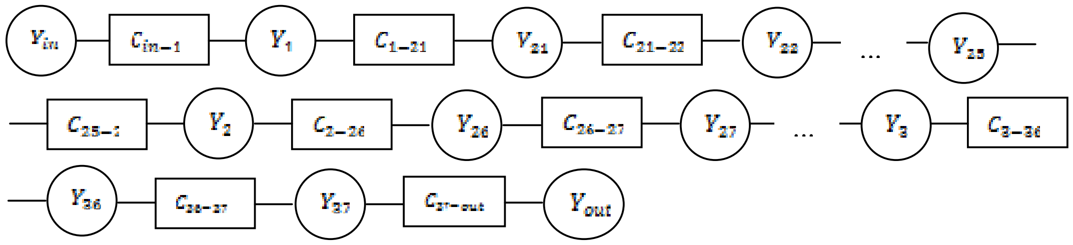

Figure 1. Attack graph for S-DES

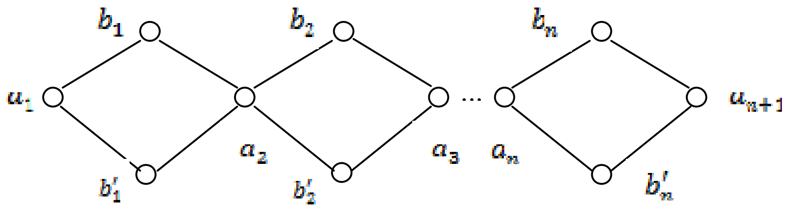

Figure 2. The graph of generation of all possible initial keys (Lipton graph)

and engineering bacteria to solve the burnt pancake problem (BPP) with two inputs [6] in a series of consecutive operations of flipping the bits as shown in the proposed algorithm.

Algorithm 2. Design and engineering bacteria to accomplish initial permutation (IP).

a) Reorder the numbers of the left hand side of Equation (1) that have a difference equal to “1” so that they are arranged in their correct order dictated by the right hand side of Equation (1). i.e. if initially in the left hand side of Equation (1) “2” is placed before “1”, these numbers will be reordered and otherwise, the order of numbers of “2” and “3” must be investigated and this step goes on in this way until numbers of “7” and “8” are taken into consideration.

b) Repeat step I for numbers with difference 2, 3 and 4, respectively and rearrange the numbers if necessary.

c) After analyzing numbers with difference 4, if the right hand side of Equation (1) is not yet equal to the derived sequence, repeat from step (a) until we get to the sequence dictated by the right hand side of Equation (1).

Using algorithm 1, we can achieve the permutation of Equation (1) in 16 steps of ordering. All in all, we demonstrate sequence of the required flips in Equation (2).

$$01234567 \mapsto^{(1,2)} \mapsto 02134567 \mapsto^{(3,4)} \mapsto 0214357 \mapsto^{(5,6)} \mapsto 02143657 \mapsto^{(6,7)} \mapsto 02143756 \mapsto^{(2,4)}$$
$$\mapsto 04123756 \mapsto^{(3,5)} \mapsto 04123756 \mapsto^{(4,6)} \mapsto 06125734 \mapsto^{(5,7)} \mapsto 06127534 \mapsto^{(1,4)} \mapsto 06427531$$
$$\mapsto^{(4,7)} \mapsto 06724531 \mapsto^{(0,4)} \mapsto 46720531 \mapsto^{(6,7)} \mapsto 47620531 \mapsto^{(2,0)} \mapsto 47602531 \mapsto^{(3,1)} \mapsto 4760251$$
$$\mapsto^{(6,7)} \mapsto 76402513$$

(2)

The result of the last derivation of Equation (2) equals to right hand sequence of Equation (1). And since in accordance with [6] to reorder two numbers we need three steps of flipping, using Hin/hix recombinant systems of Salmonella bacteria, in order to accomplish the initial permutation of S-DES, totally, we need 16×3=48 flipping operations. After applying initial permutation to the given plaintext, we achieve L_0 and R_0 that are higher and lower order four bits of the attained plaintext in the previous step, respectively. After the abovementioned operations in Y_{in} , the strings existing in this node, leave Y_{in} and enter into Y_1. The operation of communication of strings from one node to its neighbourhood and its practical realization and simulation is justifiable with the help of communication between bacterial communities and the concept of quorum sensing explained in section 2.1. In node Y_1 by applying permutation PC1 which is depicted in table 1 to the 10-bit key, we get to C_0D_0. In table 1 the upper line demonstrates C_0 bits and the lower bits are D_0 bits and numbers of 0 to 9 belong to key bits. After applying a single circular left shift to C_0 and D_0 we achieve C_1 and D_1, respectively. By concatenating C_1 and D_1, we get C_1D_1. Therefore, the first bit of K_1 is the third bit of C_1D_1. PC1 and circular left shifts can be accomplished using Hin/hix recombinant systems as shown in Equations (3) and (4).

Table 1. PC1 permutation for S-DES

9	7	3	8	0
2	6	5	1	4

$$m_0m_1m_2m_3m_4m_5m_6m_7m_8m_9 \rightarrow m_9m_7m_3m_8m_0m_2m_6m_5m_1m_4 \qquad (3)$$

$$0123456789 \mapsto^{(1,2)} \mapsto 0213456789 \mapsto^{(2,3)} \mapsto 0312456789 \mapsto^{(4,5)} \mapsto 0312546789 \mapsto^{(5,6)}$$
$$\mapsto 0312645789 \mapsto^{(6,7)} \mapsto 0312745689 \mapsto^{(8,9)} \mapsto 0312745698 \mapsto^{(4,6)} \mapsto 0312765498$$
$$\mapsto^{(6,8)} \mapsto 0312785496 \mapsto^{(7,9)} \mapsto 0312985476 \mapsto^{(0,3)} \mapsto 3012985476 \mapsto^{(4,7)} \mapsto$$
$$3012985746 \mapsto^{(1,5)} \mapsto 3052981746 \mapsto^{(3,7)} \mapsto 7052981346 \mapsto^{(1,6)} \mapsto 7092586341$$
$$\mapsto^{(8,3)} \mapsto 7092536841 \mapsto^{(2,8)} \mapsto 7098536241 \mapsto^{(0,8)} \mapsto 7980635241 \mapsto^{(7,9)} \mapsto$$
$$9780635241 \mapsto^{(0,3)} \mapsto 9783605241 \mapsto^{(4,1)} \mapsto 9783605214 \mapsto^{(5,2)} \mapsto 9783602514$$
$$\mapsto^{(6,2)} \mapsto 9783206514 \mapsto^{(8,3)} \mapsto 9738206514 \mapsto^{(2,0)} \mapsto 9738026514$$

(4)

In accordance with Equation (4), permutation PC1 is realizable in 25×3=75 flip operations. Circular left shift can be accomplished utilizing Hin/hixC recombinant systems, as well.

In node Y_1 other than generation of C_1D_1 from the initial key, R_0 which contains four bits should be converted to eight bits. This operation is applicable using E function which is shown in table 2.

Table 2. E selection table of S-DES

3	0	1	2	1	2	3	0

E-function can be accomplished by permutation shown in Equation (5) which is implementable by Hin/hix systems.

$$01230123 \to 30121230 \qquad (5)$$

$$0123 \mapsto^{(2,3)} \mapsto 0132 \mapsto^{(1,3)} \mapsto 0312 \mapsto^{(0,3)} \mapsto 3012 \quad (6\text{-}1)$$

$$0123 \mapsto^{(0,1)} \mapsto 1023 \mapsto^{(0,2)} \mapsto 1203 \mapsto^{(0,3)} \mapsto 1230 \quad (6\text{-}2)$$

Equations (6-1) and (6-2) can be done in 6×3=18 flips.

After generation of intended strings in Y_1 and passing through block $C_{1\text{-}21}$, strings will get to Y_{21} in which 10-bits of C_1D_1 which were generated in the last step, reduce to 8-bits. This permutation (so-called PC2) is shown in Equation (7).

$$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7 \to 3\ 1\ 7\ 5\ 0\ 6\ 4\ 2 \qquad (7)$$

The resulting strings in node Y_{22} are forwarded to node Y_{23} in which 8-bits of k_1 which were previously generated in node Y_{22} are XORed with 8-bits of d_i ($1 \le i \le 8$) which were produced in node Y_1. We propose to do XOR operation using communication between strains of Vibrio Fischieri bacteria as shown in Equation (8) and depicted in Figure 3.

$$A \oplus B = \text{AND}[\text{OR}(A,B), \text{NAND}(A,B)] \qquad (8)$$

To implement XOR gate using the concept of quorum sensing, as demonstrated in [19], we assume two inputs of the XOR gate (i.e. A and B) as two inducers of appropriate promoters that activate the cascade that implements XOR gate. These inputs are read by bacteria of Figure 3 that according to presence or absence of each molecule produce a different response. These outputs are AHL molecules and by generation of them by NAND and OR bacterial gates of Equation (8), the produced AHL molecules are written in agar plate solution which are then received by AND bacterial gate that accomplishes the intended task by activation of its internal state. The operations leading to doing XOR can be represented as follows: the OR gate, expresses luxIa by activation of a promoter that needs an activator. i.e. assuming that Arabinose and IPTG are inputs to the XOR gate, if any of these sugars are existing, luxIa is expressed and output of OR gate becomes 'True'. Output of Figure 3 gives the output of XOR function.

It is noteworthy that by expression of mucAB gene, all activities of transcription factor AlgT is repressed and if output of the XOR gate is 'One', luxIout gene in bacteria strain is activated. After doing XOR on strings existing in Y_1 and Y_{22} that flow into Y_{23}, resulting strings are then sent to node Y_{24} in which output of S-Boxes S_0 and S_1 are calculated. S-Boxes of S-DES, map 4-bit binary strings to 2-bit binary strings in accordance with a predefined look-up table. In order to realize S-Box, S_0, at first, we must be able to recognize left hand side bits of the corresponding lookup table and then, the operation of substitution should take place in two bits of it. We provide a scheme for encoding the left 4-bits of S-Box inputs as follows: the ith-bit from the left hand side is encoded by p_i (p_{iRev}) ($1 \le i \le 4$) promoter if this bit is 'one' ('zero').

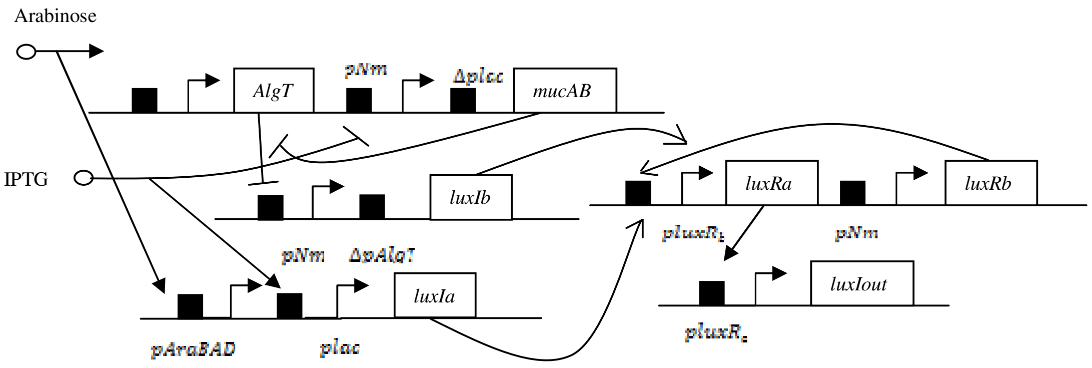

Figure 3. Genetic circuit for implementation of logical XOR [19]

p_{iRev} is promoter p_i in the reverse order. It is noteworthy that all promoters are flanked by hixC sequences. By activation of promoter p_i in the straight order which encodes 'one', $gp_i^{(1)}$ is activated which encodes 'one' in the ith-bit. And by activation of promoter p_{iRev} which is p_i in the reverse order, $gp_i^{(0)}$ is activated that encodes 'zero' for in the ith-bit of the input of S-Box. After doing logical AND operation on the received signals for expression of all $gp_i^{(j)}$s ($1 \le i \le 4, j=0,1$), a unique 4-bit sequence is defined. The HinLVA gene which belongs to Salmonella typhimurium should also be cloned in a different plasmid and by co-transformation of these plasmids into the bacterial cells, it is possible to do the inversion operation. We should note that after recognition of the left hand side bits, only their first two bits will be substituted and this is possible by Hin-mediated inversion of its corresponding promoter whenever necessary. After this operation, the HinLVA plasmid will be removed from the resulting solution. For instance, in order to substitute '1111' with '00' first, we should recognize '1111' uniquely. To this end, we encode '1111' as shown in Equation (9) and the plasmid encoding all possible inputs of S- Box is demonstrated in Figure 4.

$$\begin{aligned}&hixC - p_1 - hixC - RBS - gp_1^{(1)} - hixC - gp_2^{(0)} - hixC - p_2 - hixC - gp_2^{(1)} - hixC - \\ &gp_3^{(0)} - hixC - p_3 - hixC - gp_3^{(1)} - hixC - gp_4^{(0)} - hixC - p_4 - hixC - gp_4^{(1)} - hixC - \\ &plasmid_{CutSite} - repApSC101 - AmpR - plasmid_{CutSite} - gp_1^{(0)}\end{aligned} \tag{9}$$

$plasmid_{CutSite}$ is the cut site point of the plasmid which encodes inputs of S-Box. RBS stands for Ribosome binding site. AmpR is the Ampicillin resistance coding gene and repA pSC101 is the origin of replication. Therefore, Y_{24} contains all plasmids that are waiting to receive output of Y_{23} and by receiving appropriate signals from Y_{23}, appropriate promoters are induced in them and different genes (different luxI-luxR pairs) are expressed. By expressing all genes that are cloned in the input plasmid, HinLVA gene which has been cloned in another plasmid that corresponds to the input plasmid, is expressed and then both plasmids are transformed into a common cell and Hin acts on two first bits of plasmid and reverses direction of the genes between two hixC sites and therefore, when input bits are to be changed, gene direction is reversed and when there is no need to change direction of genes, HinLVA remains functionally inert. For instance, in substitution of '1111' with '00', the first 2-bits of input and therefore, direction of promoters p_1 and p_2 must change. We should note that Hin protein works only on the gene between two hixC sites and may not reorder two different genes in a plasmid and therefore, only the substitution operation is possible.

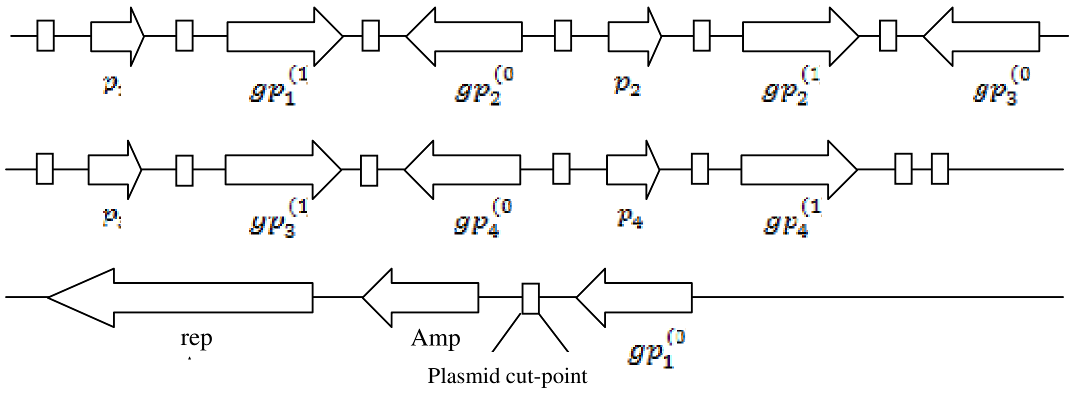

Figure 4. The proposed encoding scheme of inputs of S-box for S-DES

After doing S-Box substitutions, strings that exist in Y_{24} enter into Y_{25} in which a permutation operation takes place as Equation (10) demonstrates. This permutation is realizable through Hin-mediated flipping mechanism as Equation (11) suggests.

$$0123 \mapsto^{p} \mapsto 1032 \qquad (10)$$

$$0123 \mapsto^{(0,1)} \mapsto 1023 \mapsto^{(2,3)} \mapsto 1032 \qquad (11)$$

Permutation, p, needs 2×3=6 inversion operations in total. Strings of Y_{25} are forwarded to Y_2 after generation. In Y_2, outputs of Y_{25} are XORed with l_{0i} ($1 \le i \le 4$) which are outputs of node Y_{in}. Output of Y_2 is R_1 or the output of the first round of S-DES. Next, these strings are entered into node Y_{26} in which R_0 which contains 4-bits by itself is converted into 8-bits. This transformation looks like the function that takes place in node Y_1 and utilizes the same E-function (Equation (5)). The generated strings of Y_{26} are then received by Y_{27} in which generated bits of Y_{25} as well as output of node Y_{22} are removed because they are not useful anymore. Next, in output of node Y_{27}, L_1 and R_1 are calculated and L_1R_1 is treated like L_0R_0. In Y_{31} we convert the 10-bit key to 8 bits and in Y_{32}, permutation of Equation (3) is applied. In Y_{33}, output of Y_{31} is XORed with output of Y_{26} and in Y_{34}, S-Box tables are applied to the output of Y_{33}. In Y_{35}, output of Y_{34} goes to P-Box of Equation (7) and in Y_3, R_2 (right half of the last round) is assumed equal to R_1 and L_2 (left half of the last round) is assumed equal to XOR of R_0 with output of Y_{35}. In node Y_{36}, no change is made to strings and nodes and in Y_{37}, output of Y_{35} and Y_{32} are removed and ultimately, in Y_{37} we get to R_1 and R_2 and strings of Y_{37} are received by Y_4 after departure from Y_{37}. In Y_4, inverse of initial permutation (called IIP) is applied to R_1R_2 string which is in Y_4. The last mentioned operation is similar to applying initial permutation from genetic engineering point of view. Eventually, in Y_4 we have achieved cipher-text which is shown by $h_1h_2...h_8$. In this stage, we should examine that whether in each one of 2^{10} strings existing in Y_4 the calculated cipher-text ($h_1h_2...h_8$) equals with the given cipher-text or not. This operation takes place in Y_{out}.

4. SIMULATIONS

In this section, mathematical modelling for gene and protein interactions that we have used in our simulations will be described.

Binding of a ligand to a molecule has been modelled by Hill equation as follows. If we assume that the fraction of a molecule saturated by a ligand (i.e. the probability of binding the molecule to the ligand) is shown with $\Pr ob_{bi}$, then Equation (12) holds in which $[L]$ is concentration of the ligand; K_{50} is dissociation constant which is the concentration which produces Prob equal to 0.5 and n is the Hill coefficient which describes binding cooperativity [20]. Transcription of an mRNA molecule is regulated by transcription factors in active form which are either activators or inhibitors and can increase or decrease the probability of binding RNA polymerase to the promoter, respectively. The equation that models regulated transcription is modelled with Equation (13).

In which $\Pr ob_{tr}$ is the probability of transcription of the gene; $[T]$ is concentration of the transcription activator (TF) in active form and n and K_{50} are defined as above.

$$\Pr ob_{bi} = \frac{[L]^n}{K_{50}^n + [L]^n} \tag{12}$$

$$\Pr ob_{tr} = \frac{[T]^n}{K_{50}^n + [T]^n} \tag{13}$$

For unbound promoter, Equation (14) holds if TF inhibits transcription. Note that in Equation (14) [T] depicts concentration of the transcription inhibitor in active form.

$$\Pr ob_{in} = \frac{K_{50}^n}{K_{50}^n + [T]^n} \tag{14}$$

The ODE of Equation (15) can be written to describe activated transcription in which $V_{\max}$ is the maximum transcription rate, $[m]$ and $[T]$ are concentrations of mRNA molecule and transcription activator in active form, respectively. delta is mRNA degradation constant and 'a' is the leakage factor that can model promoter basic activity and is a percentage of $V_{\max}$. For modelling inhibited transcription we may write Equation (19) and constitutive transcription can be described by Equation (17) [21].

$$\frac{d[m]}{dt} = V_{\max}\{a + (1-a)\frac{[T]^n}{K_{50}^n + [T]^n} - \delta[m]\} \tag{15}$$

$$\frac{d[m]}{dt} = V_{\max}\{a + (1-a)\frac{K_{50}^n}{K_{50}^n + [T]^n} - \delta[m]\} \tag{16}$$

$$\frac{d[m]}{dt} = c - \delta[m] \tag{17}$$

Assuming that the TF can be activated or inhibited by an inducer factor, which is considered as an external input, using Equation (18) concentration of bound and unbound TFs can be expressed as a function of the inducer factor.

$$[T]_b = \frac{[I]^n}{K_{50}^n + [T]^n}[T]_t \tag{18}$$

$$[T]_u = 1 - [T]_b = \frac{K_{50}^n}{K_{50}^n + [T]^n}[T]_t \tag{19}$$

In Equations (18) and (19), $[T]_b$, $[T]_u$, $[T]_t$ and $[I]$ are concentrations of bound TF, unbound TF, total concentration of the TF and concentration of the inducer, respectively.

ODE for protein production can be written as Equation (20) suggests in which protein production and degradation are linear.

$$\frac{d[P]}{dt} = \alpha[m] - \beta[P] \tag{20}$$

In Equation (20) $[P]$ and $[m]$ are concentrations of protein and mRNA molecule, respectively and α and β are translation and protein degradation rates, respectively.

Cell population growth in generation of all possible keys is modelled by ODE of Equation (21) when we have one type of cell. In Equation (21), N denotes the number of cells, N_m denotes the maximum allowed number of cells due to nutrient limitation, μ is a constant coefficient, γ is the rate constant, $[AntiBio]$ is concentration of antibiotics, η is constant for specific genes and $[AntiBioR]$ is concentration of antibiotic resistance enzymes. While two types of cells are used, Equation (22) can be written to describe dynamics of the number of cells.

$$\frac{dN}{dt} = \mu_A N(1 - \frac{N}{N_m}) - \gamma_{AntiBio} N(AntiBio - \eta_{AntiBio} AntiBioR) \tag{21}$$

$$\frac{dN_A}{dt} = \mu_A N_A(1 - \frac{N_A + N_B}{N_m}) - \gamma_{AntiBio} N_A(AntiBio - \eta_{AntiBio} AntiBioR) \tag{22}$$

Expression level of resistance gene can be regulated by AHL-R protein complex. These R proteins are LuxR proteins that are synthesized sufficiently and therefore, we assume that the concentration of complex depends only on the AHL concentration. In this regard, we model expression of resistance genes using Equation (23) [22].

$$\frac{dAntiBioR}{dt} = k_{AntiBioR}\frac{3OC6HSL^n}{3OC6HSL^n + m_{AntiBioR}^n} + leakyAntiBioR - d_{AntiBioR}AntiBioR \tag{23}$$

Note that in Equation (23) $leakyAntiBioR$ denotes leaky production of antibiotic resistance enzyme, $k_{AntiBioR}$ denotes the maximum synthesis rate of antibiotic resistance enzyme, n is Hill coefficient and $m_{AntiBio}$ denotes the AHL concentration which is mandatory to produce half level of antibiotic resistance enzymes.

Synthesis and degradation of AHL has been modelled in the course of our simulations using Equation (24). Note that AHL is enzymatically synthesized by I protein from some substrates that are assumed to be sufficient so that we can estimate AHL synthesis rate to be proportional to the cognate I protein. The degradation rate of AHL is also assumed to decay with first-order kinetic which is included in Equation (24).

$$\frac{d3OC6HSL}{dt} = k_{3OC6HSL}.LuxI.N - d_{3OC6HSL}.3OC6HSL \qquad (24)$$

Production of protein under control of AHL dependent promoter can be modelled by Equation (25).

$$\frac{dLuxI}{dt} = \frac{k_{LuxI}}{(\frac{3OC6HSL}{m_{LuxI}})^n + 1} - d_{LuxI}.LuxI \qquad (25)$$

In Equation (25), k_{LuxI} denotes maximum production rate of $LuxI$ which is achieved when the concentration of AHL is zero. n is Hill coefficient and m_{LuxI} denotes the AHL concentration which is required to produce half level of I proteins and d_{LuxI} is constant coefficient. Based on the parameters which have been adopted from [22-27], we simulated different parts of our cipher cracking system.

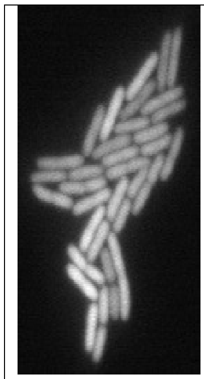	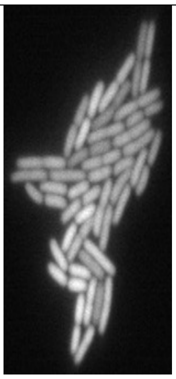	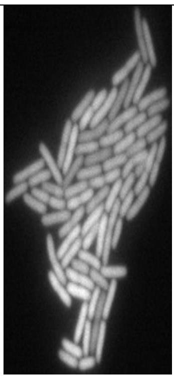	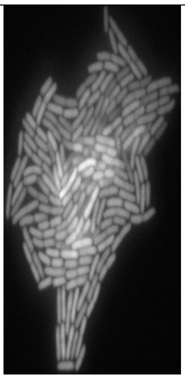	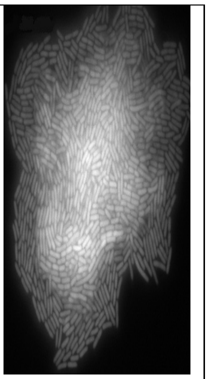
250 min	280 min	310 min	340 min	370 min

Figure 5. Simulation of growth of E. coli in different times

In order to generate all initial keys and growing the culture using cellular division on agar plate, a simulation has been conducted to model proliferation and mortality rate of bacteria which basically utilizes differential equations to provide necessary means for modelling. Results of simulations are depicted in Figure 9 in which half-life of the initial plate containing all possible keys can be estimated. We assume that bacteria culture grows on a radially symmetric agar plate with diameter of $9\,cm$ and we simulate diffusion and distribution of bacterial culture as well as nutrient distribution in Figure 6 [28]. The rest of parameters for simulations are as follows.

Table 3. Parameters of simulations

Parameter	Description	Value
N_m	The maximum value of cell density	2.8 [OD600]
μ_A	Cell growth rate coefficient for cell A	1.8 [1/hour]
$\gamma_{AntiBio}$	Antibiotic resistance coefficient	2 [L/g.hour]
$\eta_{AntiBio}$	Antibiotic resistance gene coefficient	0.005 [g/L.nM]
$m_{AntiBio}$	Hill coefficient for antibiotic	8 [1/hour]
d	The diameter of culture	9cm
Threshold	Threshold concentration for communication between layers of circuit	$10^{-8}M$

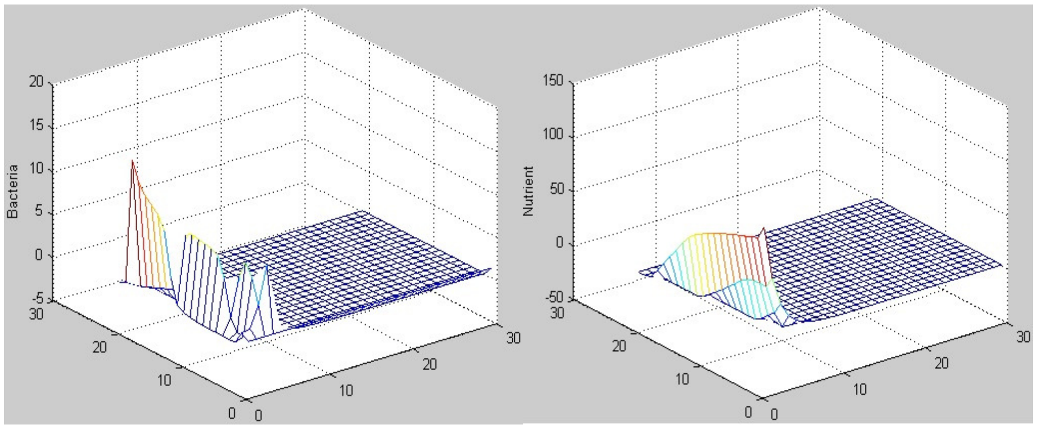

Figure 6. Diffusion of bacteria culture (left) and nutrient (right) on agar plate

For encoding all possible combinations of a 10-bit binary number which is to be used as the key, genes are cloned in a plasmid as shown in Figure A.1.

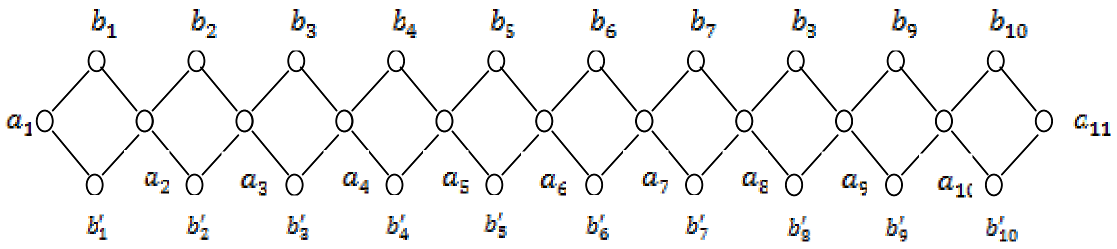
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_3 b_9 b_{10}
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11}
b'_1 b'_2 b'_3 b'_4 b'_5 b'_6 b'_7 b'_8 b'_9 b'_{10}

Figure 7. Graph for representation of all keys

In Figure A.1, gene halves are demonstrated by and . And hixC sites are depicted with ▲ .. Also transcription terminator gene is shown wit . By exposure to Hin protein We can produce all combinations of a_i, b_i and b_j $(1 \le i \le 11, 1 \le j \le 10)$ in plasmid of Figure A.1 which is depictive of all possible combinations of the key. Note that a gene can be expressed if and only if a promoter, an RBS and a coding sequence exist for it. In 3' end, a transcription terminator (TT) exists to prevent transcription mechanism of transcription machine and has been put in the last node of the graph. We have simulated the process of generation of all possible keys from an initial key under exposure of Hin protein using BioBrick library with MATLAB software [29]. Therefore, by putting together 31 different genes each of which is representative of a specific node, and by finding appropriate cut points for insertion of 26bp-hix sites as well as RBS sequences between gene halves and by putting promoter PBad at the beginning of genes and cloning all these sequences into a BioBrick standard plasmid [28], we can get to Lipton graph of Figure 7. Furthermore, another plasmid is to be made that contains HinLVA gene. And after co-transformation of these plasmids into the cell and exposure of Hin protein to the genes located between $hixC$ sites in the plasmid containing Lipton graph, we can achieve to all possible combinations of the key. The secondary plasmid containing HinLVA is also simulated using MATLAB simulink as shown in Figure 8.

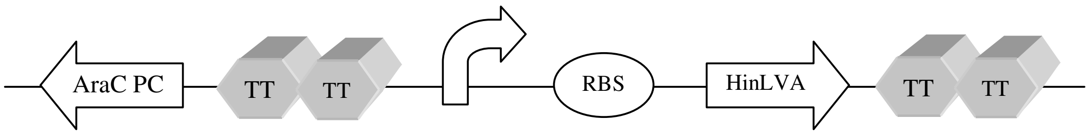

Figure 8. The secondary plasmid containing HinLVA gene (Kanamycin vector)

In Figure 8, AraC is repressor of PBad promoter and Hin gene is characterized with resistance against Kanamycin antibiotic. In this stage, cells that carry primary and secondary plasmids are appropriate for usage in a plate containing Ampicillin and Kanamycin after transformation to E. coli cells. The resulting agar plate contains cells that have two plasmids, one of which is the vector of initial genes in an scrambled order (the Lipton graph encoding plasmid or primary plasmid) that from now on in this paper, is called Amp vector and the other one (secondary plasmid) is vector of Hin gene that we name Kan vector. The operation of Hin-mediated recombination is simulated by inverting the DNA sequence between two sites which is similar to Equation (26).

$$hix_{ACGAT}^{TGCTA}hix \rightarrow hix_{ATCGT}^{TAGCA}hix \tag{26}$$

In this way, all possible paths of Lipton graph that begin from a_1 and end with a_n and introduce all possible keys are represented. By simulation of cutting the BioBrick standard plasmid using appropriate enzymes and then simulation of ligase enzyme to paste different gene halves in the abovementioned plasmid [29] using MATLAB and finding cut points and then insertion of hix DNA sequences between genes, we simulated Amp plasmid and by simulation of cutting for Kan plasmid and cloning HinLVA gene in it, Kan plasmid is also simulated. Afterwards, by inversion of genes existing between two hix sites between any two consecutive nodes, we can get to all possible combinations of the key. The graph of Figure 9 demonstrates all naturally-generated bacteria in different times.

After generation of all strings in Y_{in} since there are a multitude of meaningless combinations that are generated and do not encode Lipton graph of Figure A.1, we must design and simulate a filter to select molecules that represent all possible key combinations. Therefore, only those combinations are desired for us for whom the gene that is activated by signal generated from Equation (27) is expressed.

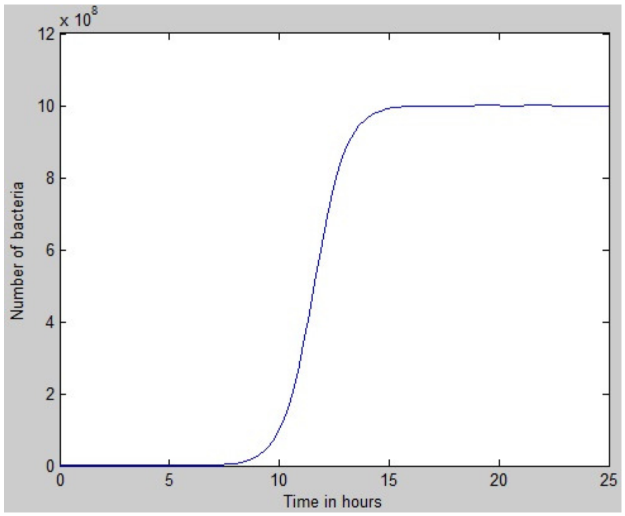

Figure 9. Natural growth of E. coli bacteria in different times

$$(a_1 \wedge \overbrace{(b_1 \vee b'_1)}^{b''_1} \wedge \ldots \wedge \overbrace{(b_{n-1} \vee b'_{n-1})}^{b''_{n-1}} \wedge a_n \wedge \overbrace{(b_n \vee b'_n)}^{b''_n} \wedge a_{n+1}) \qquad (27)$$

In Equation (27), $a_i\,(1 \le i \le n+1), b_j, b'_j\,(1 \le j \le n)$ are signals that represent expression of genes a_i, b_j and b'_j respectively. All possible combinations of the key are generated if and only if the logical expression of Equation (27) becomes $True$ and then all bacterial colonies containing all possible combinations are sent to the next node. We propose the genetic circuit depicted in Figure A.2 to implement Equation (27).

Therefore, by expression of genes of Figure A.2 that contain all possible combinations which demonstrate cipher key, these strings are sent to the next node. The communication interface node can be simulated as shown in Section 4.2.

4.1. Simulation of XOR operation

We simulated the XOR operation in MATLAB and to evaluate our XOR scheme, we utilized GFP protein as the output of our genetic circuit. The graph for expression of GFP in different times has been shown in Figure 10 and behaviour of molecules containing proteins and mRNAs is demonstrated in Figure 11.

Note that in Figure 10, the dotted curve depicts expression of GFP which belongs to logical one (01 or 10 inputs) and the other curve marked by $\circ$ demonstrates logical zero (00 or 11 inputs).

4.2. Designing genetic interface circuit between string generating nodes

After generation of strings in generating nodes of Figure 1, through genetic circuit of Figure 12, they are transferred to the next node. It is worthwhile that in Figure 12 S_i, $1 \le i \le 18$, demonstrate generated strings in the i^{th} node.

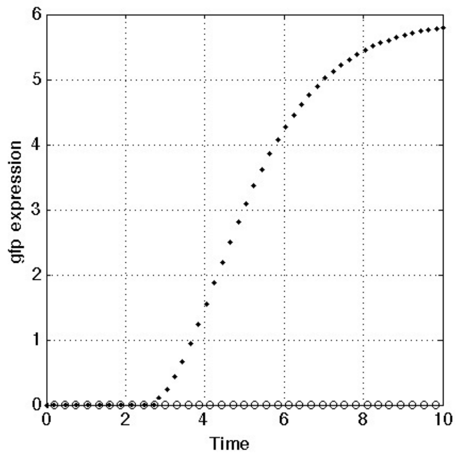

Figure 10. Expression of GFP vs. time in evaluation of XOR

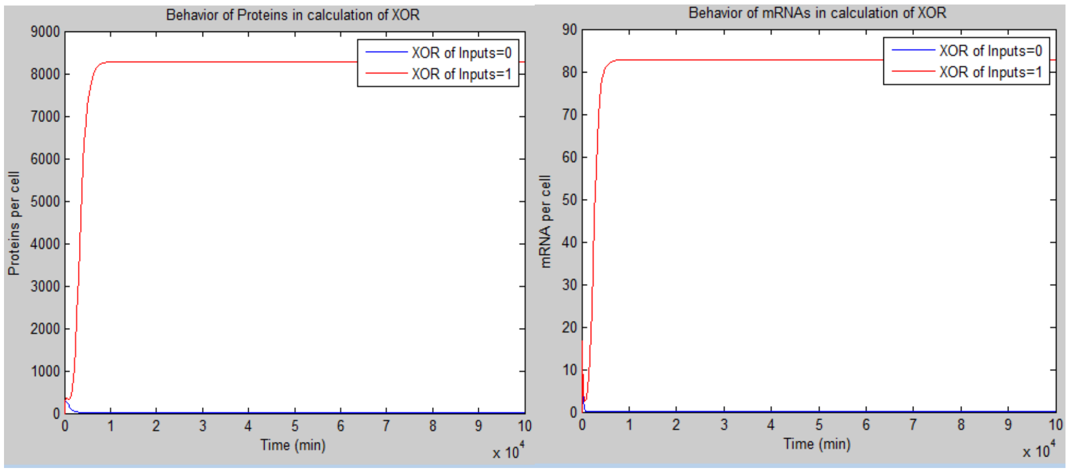

Figure 11. Behaviour of proteins (left) and mRNAs (right) in calculation of XOR

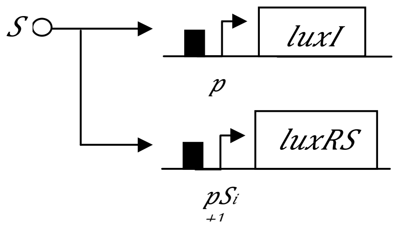

Figure 12. The proposed genetic interface circuit

The genetic interface circuit between nodes of Figure 1 is simulated and HSL and *luxI* protein concentrations over time are shown in Figure 13.

4.3. Simulation of permutation operation by engineering gene networks

We simulated the permutation operation by modelling the problem of string permutation as solving an instance of the burnt pancake problem using bacteria [6]. A mathematical modelling based on markov chain model is presented. To this end, we consider random walk on the graph of Figure 14. In [6] it is shown that from point A one may arrive at point B by applying three steps of flipping operations. Figure 14 is the basis for all permutations of S-DES cipher and as can be seen in Figure 15, the percentage of plasmids that depart from A to B is converged and therefore, we can engineer bacteria to flip two numbers.

4.4. Design and simulation of the S-box operation

In node $Y_{24,}$ there are plasmids for which promoters need inducers for their activity and are identified, transformed and extracted as shown below: consider that four bits of input that enter into S-Box are represented as a_i ($1 \leq i \leq 4$) and if we received "1", direction of the promoters are not changed and otherwise, their directions are reversed. (Considering that promoters are between two hixC sequences). This flipping operation takes place by exposing Hin protein to proper points in plasmid. Therefore, either one of downstream or upstream genes of each promoter is always expressed and so, *zero/one* information of node Y_{23} will be intelligible for node Y_{24}. Now, to apply S-Box tables, we assume that input strings of S-Box are of the form $a_1a_2a_3a_4$ in which, a_i ($1 \leq i \leq 4$) depicts generated signals in Y_{23} that have either *one* or *zero* meaning. As a result, the genetic model of Figure15 demonstrates all possible combinations for S-Box. Due to lack of space, we have shown the S-Box circuit only for substitution of "0000" to "01" in S_0.

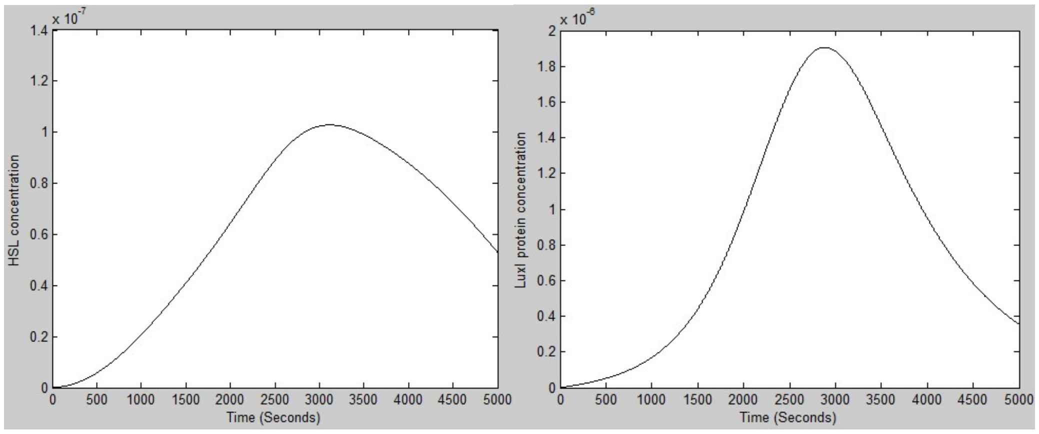

Figure 13. Concentration of HSL molecules (left) and protein (right) in genetic interface circuit

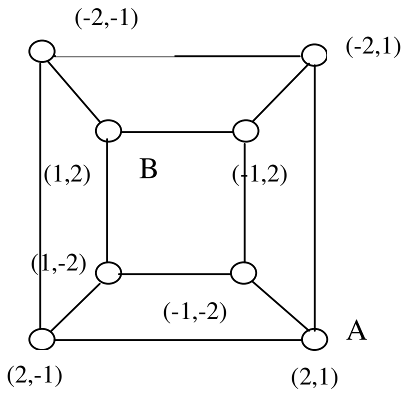

Figure 14. The graph of sorting two signed numbers

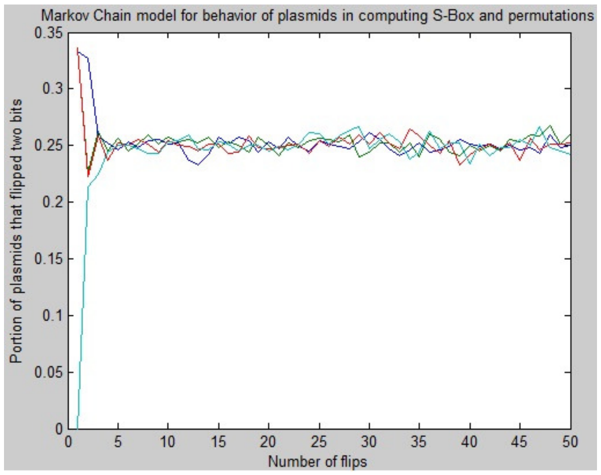

Figure 15. Mathematical modelling of behaviour of plasmids in computing P-box and S-boxes

Genetic circuit of Figure A.3 operates as follows. As soon as a specific combination from all 16 possible combinations of four input bits has been identified, a signal is generated that induces promoter of the plasmid that contains Hin gene which is supposed to convert output plasmid of S-Box (*Erith* plasmid that is characterized by resistance against Erythromycin antibiotic) to a specific combination that is dictated by S-Box table. For instance, in Figure A.3, if logical AND of output of genes, $luxIP_1^0$, $luxIP_2^0$, $luxIP_3^0$, $luxIP_4^0$ equals to '*one*', then, input bits of S-Box are of the form "0000" and by activation of the plasmid carrying HinLVA (that we call Chloramphenicol), ps_1 promoter flanked by hixC sites is reversed, $luxIps_1^{(0)}$ gene is expressed that encodes "*zero*" and promoter ps_2 is not reversed. Therefore, $luxIps_2^{(1)}$ is expressed that encodes "*one*" in the output of the S-Box. Hence, chloramphenicol encodes output sequence of "01". Genetic circuit functionality of the other 15 combinations of the S-Box table are justified and designed just similarly.

5. Performance evaluation of the proposed scheme

In order to accomplish the initial permutation operation, we need 48 flipping steps. Realization of permutation PC1 needs 75 flipping operation as well. Implementation of the E- table in the cipher system needs 18 flipping operation. Applying S-Box table on each input string needs 16 flip operation that generates 32 flipping operation in total. Permutation operation of Equation (10) also needs 6 flipping operations. PC2 permutation operation needs $15\times3=45$ Hin-mediated flipping operations. One and two-unit circular left shift operations also need $7\times3=21$ and $10\times3=30$ flipping operations, respectively. IIP operation needs 48 flipping operations. In each round, 12 XOR operations are mandatory which makes a total number of 24 for our cipher system. In general, 24 XOR operations and 323 flipping operations are required.

6. Conclusions

In spite of growing interest in developing novel approaches for the state of the art block cipher design, Feistel schemes and DES-like ciphers remain widely used in practice. Efforts to break ciphers based on this structure in a naïve fashion yield no much better result than the brute force attack which exhibits an exponential complexity by increasing the key size [31]. Linear attacks that are considered as one of the most effective attack on these structures [32] also need impractical information on adversary's side. In this paper we proposed a gene regulatory network to break Feistel ciphers which utilizes quorum sensing mechanism of Vibrio Fischeri bacteria and we simulated it to break S-DES cipher. Our proposed scheme exhibits a better result than previous results on breaking block ciphers using DNA computing as it combines the advantages gained by inherent cellular division of bacteria as well as massive parallelism offered by DNA computing.

This main idea of this paper considers some phenomena which take place during *in-vivo* biological natural processes or *in-vitro* operations in genetic engineering laboratories. These operations are Hin-mediated recombination and quorum sensing. On the other hand, the required operations for breaking the considered cipher are: generation of all keys from an initial key; the permutation operation (P-box); the substitution operation (S-box) and the logical XOR operation. Inspired by these biological operations, in the course of this paper as Figure 16 demonstrates, we proposed to do the computation operations of breaking the cipher according to the following classification: the operation of generation of all possible keys from an initial key and P-box and S-box operations are accomplished by Hin-mediated recombination of DNA strings and also, the logical XOR operation and the communication of strings between string generating nodes of graph of Figure 1 are done by the quorum sensing mechanism explained in section 2.1.

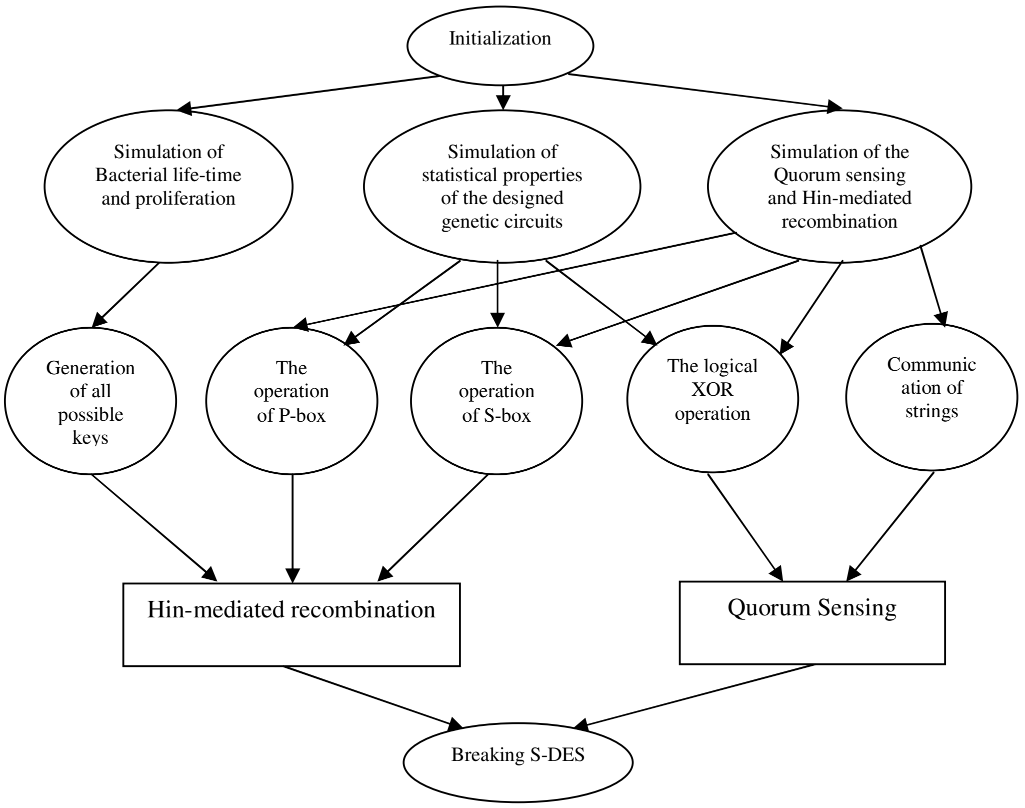
Initialization
Simulation of Bacterial life-time and proliferation
Simulation of statistical properties of the designed genetic circuits
Simulation of the Quorum sensing and Hin-mediated recombination
Generation of all possible keys
The operation of P-box
The operation of S-box
The logical XOR operation
Communic ation of strings
Hin-mediated recombination
Quorum Sensing
Breaking S-DES

Figure 16. Our proposed bacterial computer for breaking S-DES in a nutshell

Although in some papers such as [33], novel and robust ideas for enhancing practicality and at the same time security of unconventional cryptographic schemes are proposed, but our proposed DNA computer for breaking the S-DES cipher has shown to be more practical than any other unconventional computer for doing encryption and decryption such as the quantum computer for cryptography [34] which is just a potential and hypothetical computer. The practical experiments of [6-7] show how near we are to building a computer for cryptanalysis of a sophisticated cipher.

ACKNOWLEDGEMENTS

The work of this paper is done in Iran University of Science and Technology as a part of MSc Project which is supported by a grant from Iran Telecommunication Research Center (ITRC). Authors would like to thank ITRC for their supports.

REFERENCES

[1] Drew Endy (2005) "Foundations for engineering biology," Nature, 436:449–453, 2005.

[2] Adleman LM (1994): Molecular computation of solutions to combinatorial problems**.** *Science*, 266:1021-1024.

[3] M. Elowitz and S. Leibler (2000) "A synthetic oscillatory network of transcriptional regulators," Nature, 403:335–338.

[4] Amos M. (2009) "Bacterial computing," Encyclopedia of complexity and systems science. pp. 417-426.

[5] Lingchong You, Robert Sidney Cox III, Ron Weiss, and Frances H. Arnold, (2004), "Programmed population control by cell-cell communication and regulated killing," Nature, 428, pp. 868–871.

[6] Harden WL, Heard LH, Jessen EL, Malloy KJ, Ogden BJ, Rosemond S, Simpson S, Zwack E, Campbell AM, Eckdahl TT, Heyer LJ, Poet JL: Engineering bacteria to solve the burnt pancake problem. *Journal Biol En* 2008, 2:8.

[7] Baumgardner J, Acker K, Adefuye O, Crowley ST, DeLoache W, Dickson JO, Heard L, Martens AT, Morton N, Ritter M, Shoecraft A, Treece J, Unzicker M, Valencia A, Waters M, Cmpbell AM, Heyer LJ, Poet JL, Eckdahl TT. "Solving a Hamiltonian Path Problem with a bacterial computer," Journal of Biological Engineering. 3:11. (2009).

[8] Boneh, D., Dunworth, C., Lipton, R., "Breaking DES Using a Molecular Computer," Princeton CS Tech-Report CS-TR-489-95.

[9] National Bureau of Standards: "Data Encryption Standard," U.S. Department of Commerce, FIPS, pub. 46, January 1977.

[10] Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N. V., Goodman, M.F., Rothemund, P. W. K. Adleman, L.M. "A Sticker Based Model for DNA Computation,"

[11] Krishna, S. N., and Rama, R., Breaking DES Using P System. Theoretical Computer Science, 299, 495-508, 2003.

[12] Ashish Choudhary, Kamala Krithivasan, "Breaking DES Using Networks of Evolutionary Processors with Parallel String Rewriting Rules," International Journal of Computer Mathematics, Vol. 86, No. 4, 2009, pp. 567-576.

[13] Alberghini S, Polone E, Corich V, Carlot M, Seno F, Trovato A, Squartini A. "Consequences of relative cellular positioning on quorum sensing and bacterial cell-to-cell communication," FEMS Microbiol. Lett. 292: 149-161. (2009).

[14] Edward Schaefer (1996), "*A Simplified Data Encryption Standard," Algorithm,*Cryptologia 96.

[15] R. Lipton, "Using DNA to solve NP-Complete Problems," Science 268, pp. 542-545, April 1995.

[16] Nanassy OZ, Hughes KT: In vivo identification of intermediates stages of the DNA inversion reaction catalyzed by the *Salmonella* Hin recombinase. *Genetics* 1998, 149(4):1649-1663.

[17] Zieg J, Silverman M, Hilmen M, Simon M: Recombinational switch for gene expression. *Science* 1977, 196**:**170-172.

[18] Zieg J, Simon M: Analysis of the nucleotide sequence of an invertible controlling element. *Proc Natl Acad Sci USA* 1980, 77:4196-4200.

[19] Ángel Goñi Moreno, "Arquitecturas de comunicaciones para la computación algor□tmica en poblaciones de bacterias multi-cepa," Tesis Doctoral, Universidad Polit_ecnica de Madrid ,Año 2010.

[20] Subhayu Basu; "A synthetic multicellular system for programmed pattern formation." Nature April 2005: 434, 1130-1134

[21] Bintu, Lacramioara, Terence Hwa. "Transcriptional regulation by the numbers: applications." Current Opinion in Genetics & Development 2005, 15:125–135.

[22] Goryachev, A.B., D.J. Toh and T. Lee. "System analysis of a quorum sensing network: Design constraints imposed by the functional requirements, network topology and kinetic constant." BioSystems 2006: 83, 178-187.

[23] Alon, Uri. "An Introduction to Systems Biology Design Principles of Biological Circiuts," (2007) London: Chapman & Hall/CRC.

[24] Kepes, A., "Etudes cinetiques sur la galactoside-permease D'Escherichia coli. Biochim," (1960), Biophys. Acta 40, 70-84.

[25] Andersen JB, Sternberg C, Poulsen LK, Bjorn SP, Givskov M, Molin S. (1998) "New Unstable Variants of Green Fluorescent Protein for Studies of Transient Gene Expression in Bacteria." Appl Environ Microbiol, 64(6):2240-6.

[26] ETHZ 2007. "Engineering" iGem wiki. 13th August 2009.

[27] Bo Hu, Jin Du, Rui-yang Zou, Ying-jin Yuan, (2010) "An Environment-Sensitive Synthetic Microbial Ecosystem," PLoS ONE, volume 5 issue 5.

[28] Frank C. Hoppensteadt, Charles S. Peskin, (2001) "Modeling and Simulation in Medicine and the Life Sciences," 2nd Edition, Springer, pp. 321-324.

[29] Knight T, Rettberg R, Chan L, Endy D, Shetty R, Che A: "Idempotent Vector Design for Standard Assembly of Biobricks"

[31] M. Wiener, (1993) "Efficient DES Key Search," Crypto 93 rump session.

[32] M. Matsui, (1994) "The first experimental cryptanalysis of the Data Encryption Standard," Proceedings Crypto 1994, pp. 1-11.

[33] A. Singh, N. Sharma (2011) "Development of Mechanism for Enhancing Data Security in Quantum Cryptography," Advanced Computing: An International Journal (ACIJ), Vol.2, No.3, May 2011.

[34] Shubhra Mittal, "Quantum Cryptography", Department of Computer Science, Project report, Montclair State University, New Jersey, 2008.

Arash Karimi Received the B.S. and M.S. degrees in the Dep. Of Electrical Engineering from Amirkabir University of Technology (Polytechnic of Tehran) and Iran University of Science and Technology (IUST), Tehran, Iran, in 2008 and 2011, respectively. His research interests include cryptography, unconventional methods in computation with a focus on cryptanalysis, Biochemical computing, and formal languages and automata.

Hadi Shahriar Shahhoseini received B.S. degree in electrical engineering from University of Tehran, in 1990, M.S. degree in electrical engineering from Azad University of Tehran in 1994, and Ph.D. degree in electrical engineering from Iran University of Science and Technology, in 1999. He is an assistant professor of the electrical engineering department in Iran University of Science and Technology. His areas of research include networking, supercomputing and reconfigurable computing. More than 130 papers have been published from his research works in scientific journals and conference proceedings. He is an executive committee member of IEEE TCSC and serves IEEE TCSC as regional coordinator in middle-East Countries.

APPENDIX

In this section, the genetic circuits designed for different parts of the cryptanalysis are shown.

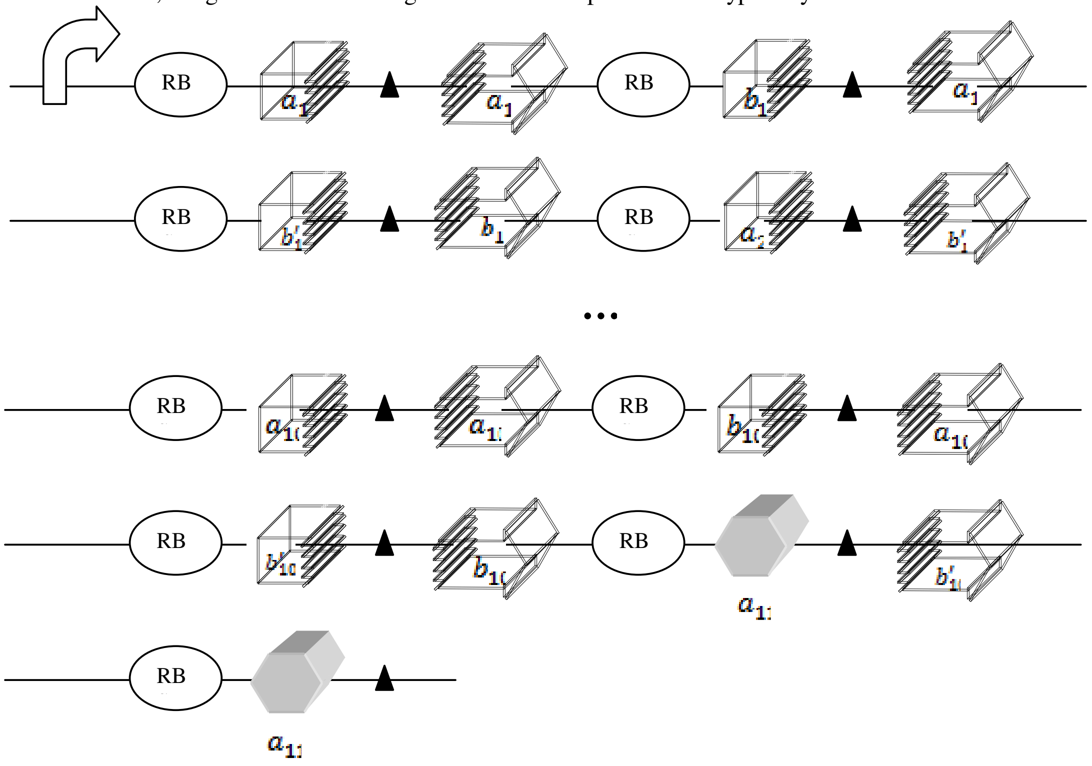

Figure A.1. The plasmid containing all initial keys of the cipher

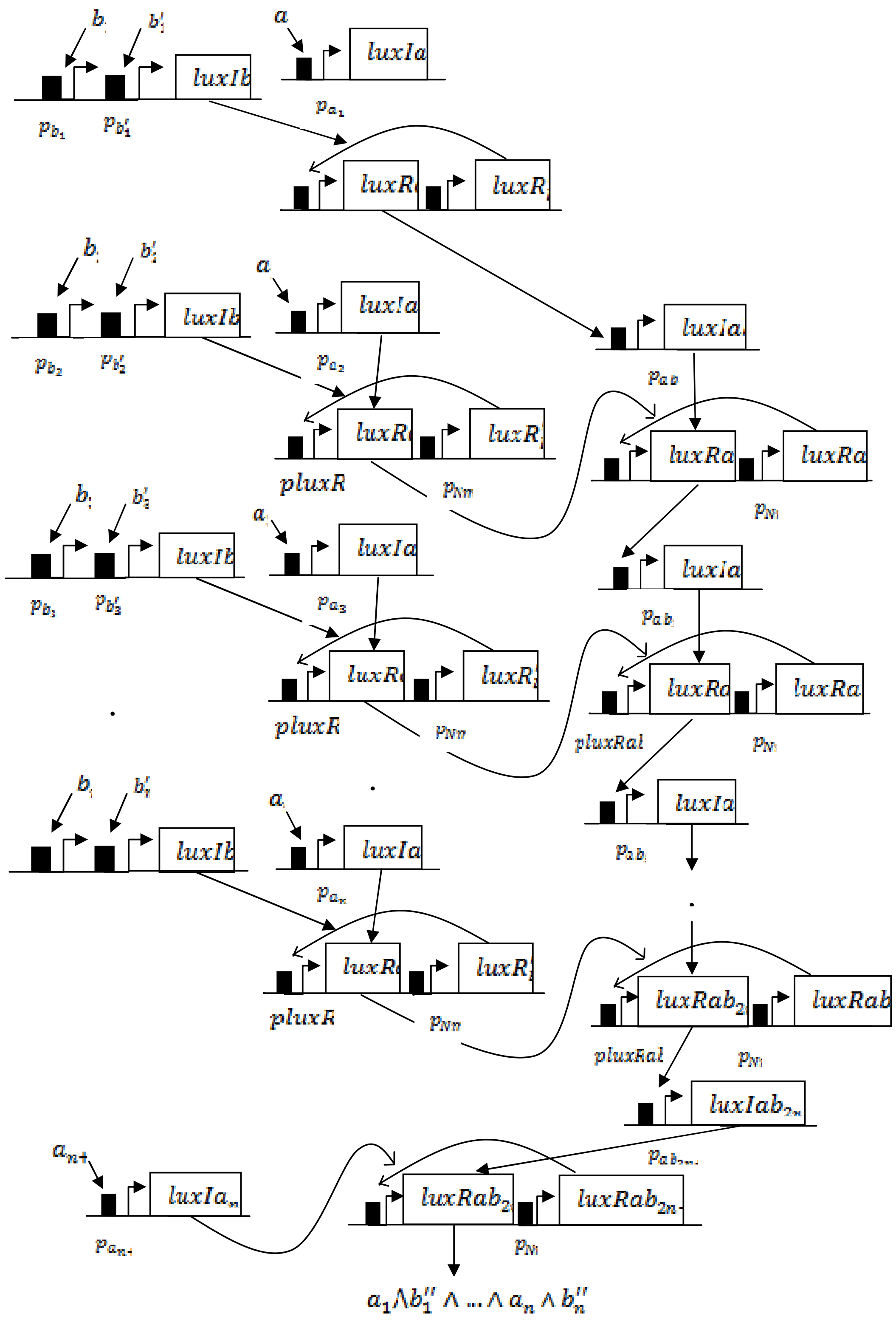

Figure A.2. Genetic filter to generate all possible combinations of the key

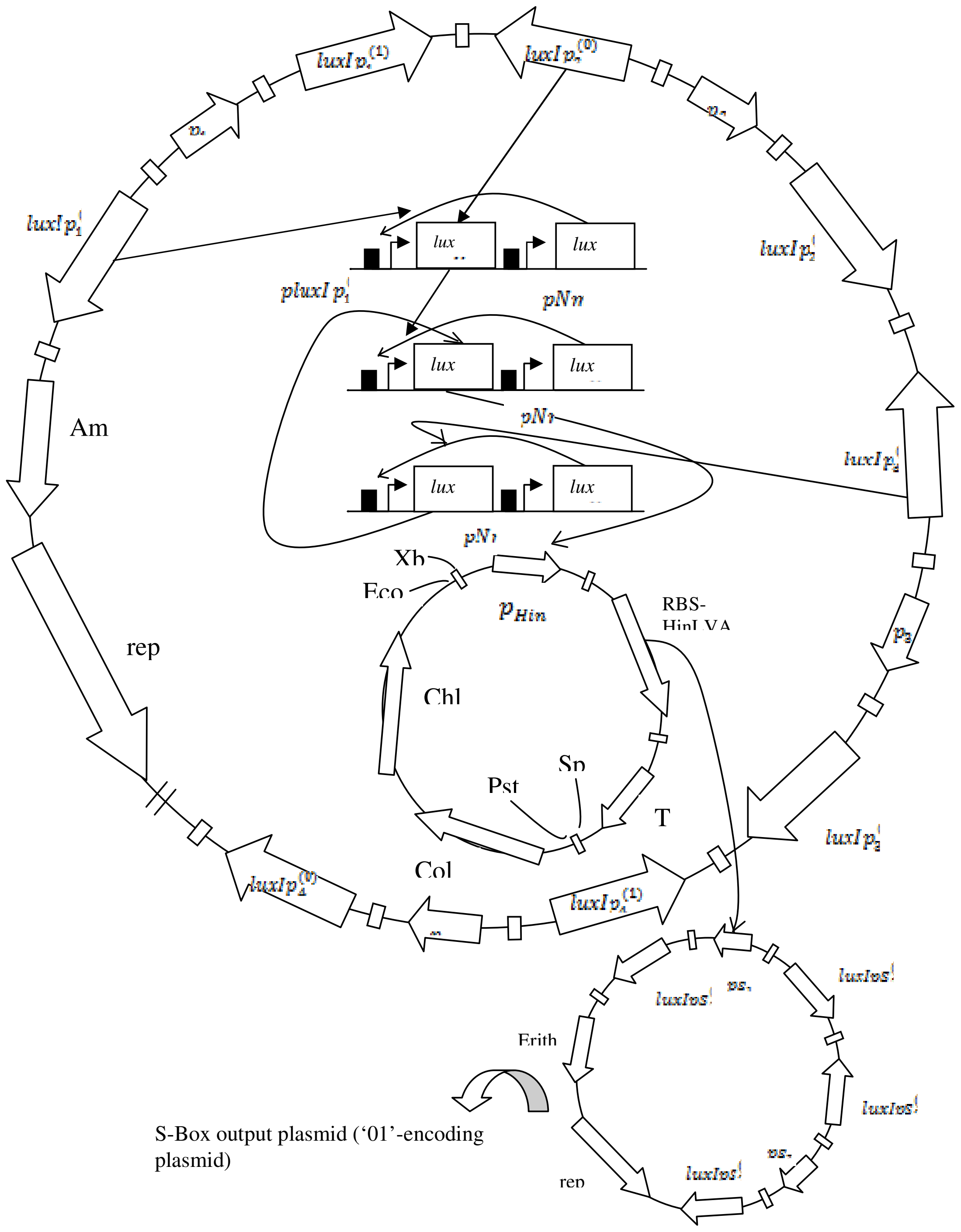

Figure A.3. Genetic circuit for implementation of S-box, S_0 in S-DES for substitution of '0000' with '01'