Cifrado fuertemente bioinspirado implementable en hardware

Strongly bioinspired, hardware-implementable ciphering

Jaime Parra P.

jparra(AT)admon.uniajc.edu.co

Institución Universitaria Antonio José Camacho. Cali – Colombia

Artículo de Investigación

Resumen

El cifrado se relaciona ya sea con la encriptación o desencriptación de un mensaje. Es parte de un área más amplia, la criptografía, el campo científico que se dedica al estudio y desarrollo de métodos para mantener información secreta, es decir, para ocultarla de intrusos indeseados. El concepto clave es transformar la información deseada, o mensaje, en una cadena no relacionada, o cifra. Se presenta aquí una forma de realizar tal labor empleando un enfoque fuertemente bioinspirado, citocomputación, que tiene por ventajas una robustez muy alta, que hace en extremo difícil romper la cifra, funcionamiento en tiempo real y simplicidad en la matemática, que le hacen apropiado para ser implementado en hardware reconfigurable.

Palabras clave: Bioinspiración, cifrado, hardware reconfigurable.

Abstract

Ciphering is intended to either encrypting or decrypting a message. It is part of a broader area, cryptography, which is the scientific field dedicated to the study and development of methods to keep information secret, i.e., hidden to unwanted intruders. The main concept is transforming the desired information, or message, into an unrelated string, the cipher. Here is presented a way of performing such a task employing a strongly bioinspired approach, cytocomputation, which has as advantages a very high robustness, making hacking the cipher extremely difficult; real-time behavior and simplicity of mathematics, making it appropriate for implementation in reconfigurable hardware.

Keywords: Bioinspiration, ciphering, reconfigurable hardware.

© 2016. IAI All rights reserved
1. Introduction

In this information age, data are omnipresent, and keep them away from undesired eyes is an important concern, hence the need for algorithms and tools that permit people to hide their private information in a secure, yet easily recoverable way. Ciphering is either encrypting or decrypting a message. Cryptography is the scientific field dedicated to the study and development of methods to keep information secret, i.e., hidden to unwanted intruders by transforming the desired information (the message) into an unrelated string (the cipher). Another associated area is cryptoanalysis, the science of attacking ciphers and/or finding weaknesses in the methods suggested. Together, they comprise the wider area of cryptography [1].

There is an agreement that messages contain textual information, since numeric information is on its own indecipherable without knowledge of the context it refers to. Two differentiable epochs are recognized in cryptography: the classical epoch, prior to the World War I, and the modern epoch, after that. In the former, cryptology was synonymous of encryption, i.e., transforming the messages into nonsense. In the latter, the focus is on hiding the messages by developing computational hardness assumptions, considered hard to break by an adversary. Among the main assumptions employed are: factorization of composite numbers having as factors only prime numbers [2], and computing logarithms in finite fields [3].

Cryptography is mainly dedicated to keep secrets. This implies the existence of an adversary, who wants to gain unauthorized access to the contents of a message that some sender wants to transmit to a receiver. To do that, the sender does not transmit the message, but a cipher, a modified version of the message. Encrypting is good when three properties are guaranteed:

1. If the adversary captures the message, he must have great difficulty in understanding the content of the message (i.e., it must be difficult to decode the cipher, even if the adversary knows what the method of encryption is).

2. The authorized receiver must be able to figure out, with relative ease, the content of the message by decoding the cipher. This can be done because both the sender and the receiver know a private key that is used to perform the decoding.

3. Its strength must lie in maintaining the key private, whereas the method employed should be public, to allow standardization of algorithms and their large-scale diffusion. Of course, the key must be managed and distributed properly. In this way, the cryptosystem is practical (this viewpoint is also known as Kerckhoff's principle [4]).

The classic methods for cryptography produce ciphers either by substitution or transposition [5]. A substitution or shift cipher is obtained by systematically replacing letters (or groups of letters) with other letters (or groups of letters). For a given letter in the message, its replacement corresponds to the letter present k positions forward in the common alphabet. These ciphers are also known as Caesar ciphers since they were used by Julius Caesar to communicate with his generals during his military campaigns, especially in his battles against the Gauls. Supposing a shift of three positions, ciphering will proceed as follows (messages are presented in lowercase, and ciphers in uppercase):

Message: all of gaul is divided into three parts
Cipher: DOO RI JDXO LV GLYLGHG LQWR WKUHH SDUWV

Of course, one way to harden decoding is by allowing the amount of shift to be variable, becoming a shared, private key between the sender and the receiver. Nonetheless, the fact that the length of the original words is preserved makes possible that a brute force attack can easily discover the message content. Its weakness derives from the fact of being monoalphabetic in the sense that, given a key, it always encrypts a letter the same way.

A transposition cipher appears when the letters of a message are rearranged, i.e., moved to other positions in the message. The concept can also be expressed by saying that the cipher is a permutation of the message. Another name for these ciphers is anagrams. Several methods have been proposed using this idea. One example of this method is the route cipher (Simmonds). It is based on the construction of a matrix using the message letters. The length and width of such matrix constitute the key. Consider obtaining a route cipher using as key a 3x7 matrix for the following text:

Message: this is a secret message

First, the text is put on the matrix:

\[ \begin{array}{ccccccc}
TS & AC & T & S & G & \hline
HI & SR & M & SE & I & S & E & E & A \\
\end{array} \]

Second, the cipher is created by reading out the matrix row by row:

Cipher: TSACTSGE|AEESIHISRMSE

Transposition ciphers are harder to decode than substitution ciphers. Nonetheless, since the original letters remain, applying frequency count it is possible to break the code, providing there is some knowledge about how frequent letters are in the language used to write the message. Moreover, the cipher is created as a monolithic entity and requires the entire content of the message to be present.

As classical encryption has important weaknesses, modern cryptography is based on hiding the message into a cipher created by applying some transformation manifesting a very hard to compute property. Consider the case of the RSA cipher [6], which bases its power on the following fact about prime numbers: Given two prime numbers \( p, q \), calculating the product \( n = pq \) is not hard, but factoring \( n \) into its prime factors is essentially impossible if \( p, q \geq 10^{100} \). RSA ciphering/deciphering is performed following these steps:

1. Select two private keys \( p \) and \( q \).
2. Select a public key \( e \).
3. Calculate \( n \) and \( d \) using the formulas:
   \[
   n = pq \\
   d = (1/e) \mod ((p-1)*(q-1))
   \]
4. Encode each message letter \((l)\) as a positive integer to obtain encoded message \(x\).
5. Encrypt the message by applying the formula:
   \[ y = E(x,n,e) = x^e \mod n \]
6. Decrypt the cipher by applying the formula:
   \[ x = D(y,n,d) = y^d \mod n \]

Consider the following numerical example, which, for simplicity, uses small numbers (strong ciphering requires using very large numbers). Suppose \(p=71\), \(q=59\), hence \(n=4189\). Suppose also \(e=3\), hence \(d=2707\). \(n\) and \(e\) are publishable, whereas \(p\) and \(q\) must be kept secret. Suppose the sender wants to send message 1234. He will calculate the corresponding cipher as:

\[
y = E(1234,4189,3) = 1234^3 \mod 4189 = 229
\]

And transmit it to the receiver, who can recover the hidden message by applying:

\[
x = D(229,4189,2707) = 229^{2707} \mod 4189 = 1234
\]

2. Biociphering

The idea behind modern cryptographers is very powerful. However, they can only produce hard-to-break ciphers at the cost of managing extremely large numbers, which requires sophisticated arithmetic computation and important memory resources. On the other hand, classical cryptographers are very simple, but usually impractical because of their evident weakness. It is because of such extreme characteristics that it was decided to propose a strategy that takes the good aspects of each approach and discourages the bad ones. Another desire was to have an approach that allows a serial, character-based processing taking into account that such a property permits real-time processing and also facilitates its implementation in reconfigurable hardware. To do that, a bioinspired approach was considered.

Cyto computation is a strongly bio-inspired computational intelligence strategy, which uses as inspiration the biomolecular dynamics occurring within biological cells [7]. It has been employed successfully for a variety of applications, including optimization, data discovery, unsupervised learning, and constraint-based processing [8]. There is a variety of cells and cell components, however all cells share two common and universal characteristics: a genetic machinery and a set of proteins [9]. The purpose of a particular gene is the creation of a molecule called polypeptide. When a group of polypeptides are linked in a certain way, a protein is created (Figure 1).

Proteins serve for very different aspects inside a cell, among which is the control of genes themselves. Through them genes can regulate each other, establishing a differentiation between genes that only produce proteins (structural genes), and genes that produce special proteins (transcription factors) that regulate other genes (regulatory genes) [10]. Besides that, basic behavior, proteins are also used as a means of carrying information for the outside of the cell. A general view of these components and their relationships is shown in Figure 2.

Within a cell there is a huge amount of proteins, with very different configurations and functionality, but, for the sake of computational purposes, three of them are the most important: enzymes, ribozymes and transcription factors (TF). They are in charge of building, analyzing, and decoding information present in standard proteins, which can be seen as data at this respect [11].

An enzyme is responsible for creating and/or modifying the structure of a protein, a ribozyme analyzes the structure of a protein and generates molecules that account for that analysis, a TF’s mission is to join a particular gene and modulate the production rate of its polypeptide, either promoting or inhibiting generation (see Figure 3). It should be noted that formally a ribozyme is not a protein but an RNA, but functionally acts as if it were an enzyme [12].

These two processes, the genomic one and the proteomic one, cross-control each other, allowing different computational schemes exploiting the various possibilities of interaction. The diagram presented in Figure 4 shows the computational model obtained from this biological behavior. In it, several processes exist. Although they run concurrently, for the purpose of explanation they are described as sequential.

The first activity is the creation of features by genes, the second is the assembly of compounds joining the features available, the third is the modification of such compounds to insert/adjust them into the current computational space, the fourth is event detection, its association with existing features and activation of promotion and/or inhibition modulators, the fifth is the
application of such modulators for the next iteration. Not all these processes are mandatory, and the designer is free to select those more appropriated for the task in hand. In the case of ciphering and deciphering, they are straightforward processes that can be optimized for speed of resource utilization, moreover taken into account that the system is intended to be run on a basic hardware at full speed for real-time processing.

With that consideration, the adaptive machinery of Cyto is optional and concentration will be devoted to the transformation processes. The two main processes will be employed in order to maximize the two aspects abovementioned. Genes will produce fragments of the cipher and proteins will encourage maximal distortion of patterns in original message so as to prevent any attempt to detect possible relationships between both.

The cipher to be created will be a protein. Each letter in it will be a polypeptide. A given gene will produce one or more polypeptides. Transcription factor will force genes to modify their behavior in accordance to a given rule. The secret key in Cyto Crypto is constituted by the original affinity and expressivity of the genes. Affinity has to do to how a gene relates to a transcription factor, and expressivity indicates the number of polypeptides a gene can generate. The secret key is given as a structure indicating the affinity and expressivity of the set of genes. Suppose a simple example in which the input alphabet is comprised of digits from 1 to 5, and output alphabet is the numbers from 1 to 9. Table 1 presents such a structure. Consider the following input: Message: 315311. The system proceeds to read the first letter (3) and to check genes’ affinity to it. In this case, that affinity is with gene 3 (fifth data column in Table 1). This gene in turn expresses the substring comprised of polypeptides 5, 1, and 3 (expression row in Table 1). After that, the affinity of the transcription factor used is modified to become favorable to another gene. In this simple case that change is done by “moving” the affinity value to a new location, as far as the first polypeptide expressed indicates (5 in this case, corresponding to the entry in fifth data column and expression row). Again, that moving action is given in that simple way for this example, not as a general rule.

Table 1. Initial representation for the simplest Cyto Crypto

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expression</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Assuming that movement will always proceed rightward, the new location will be position 1. Observe how positions must be considered as continuous, with the position at the right extreme contiguous to the first one. The current position for such transcription factor is 5, and the moving value is also 5, that will give a new position of 10. Since there are only 9 columns, values are 9-modulus. That is the reason why new location is 1. Table 2 shows the new state of the system, when, for the message=315311, the current cipher is 513.

Table 2. Cyto Crypto state after processing the first letter

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expression</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Now the current letter to process is 1, which shows affinity for gene 4, whose expression is polypeptide 4. After affinity moves to its next position, the system is on the state shown in Table 3, when, for message=315311, the current cipher is 5134.

Table 3. Cyto Crypto state after processing the second letter

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expression</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

The next letter on the message is 5, affine to gene 2. After the gene expresses polypeptide 7 and affinity moves to its next position, the system is on the state shown in Table 4 (message=315311; current cipher=51347). Observe here how the movement is performed. Affinity must move 7 places, but this location

319
is occupied by another affinity. The searching for an empty place finishes at position 5.

Table 4. Cyto Crypto state after processing the third letter

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

The fourth letter to be processed is 3, which matches gene 1 and promotes creation of polypeptides 9 and 7. Movement places affinity at position 2 because the domain is finite, which implies that moving M places is the same as moving M mod DomainLength, which in this case is 5. This action reports a new position that equals the current one, movement proceeds from that point until finding an empty place. Table 5 (Table shows the new state of the system (message=315311; current cipher=51347972).

Table 5. Cyto Crypto state after processing the fourth letter

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

The next value on the message is letter 1, which promotes gene 2 to express. The product of the expression, along with the current distribution of genes, motivates affinity to jump to position 6. Table 6 (Table 6 summarizes the current state of the crypto system (message=315311; current cipher=51347972).

Table 6. Cyto Crypto state after processing the fifth letter

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

The last letter to be processed is letter 1 again. After expression of the gene, three new polypeptides are added to the cipher and the process is suspended. The state of the system after suspension is shown in Table 7 (message=315311; current cipher=51347972134). The cipher obtained is: Cipher: 51347972134.

Table 7. Cyto Crypto state after processing the sixth letter

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Affinity</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Deciphering is performed symmetrically. The components of the cipher can be seen in one of two ways: if their status as polypeptides is preserved, deciphering is a kind of reverse transcription, process in which the products expressed by a gene activate it, or they can be considered as TFs, case in which the original TFs become new polypeptides. Irrespective of the way the process is seen, it runs in only one way. The machinery reads a cipher one by one, detects the gene related to it, obtains the corresponding message element and updates cipher and gene positions. Table 8 depicts the whole process (cipher=51347972134; expected message=315311).

Table 8. Deciphering a Cyto cipher

```
<table>
<thead>
<tr>
<th>Cipher</th>
<th>51347972134</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>315311</td>
</tr>
</tbody>
</table>
```

3. Hardciphering

Taking into account that the approach considered employs very simple mathematics (just operations such as addition of shifting), a hardware implementation is viable. Moreover, one in custom hardware where the designer can decide what resources to use. Reconfigurable hardware is one of such possibilities. It is a kind of hardware that is initially in an omnipotent state, i.e., all its resources are available for any functionality. An FPGA (field-programmable gate array) is one kind of reconfigurable hardware; it allows prototyping different architectures thanks to its capability of universality. By default, the device comes with a sea of digital resources in the form of logic blocks, all of them with the possibility of being interconnected in a very liberal way [14], see Figure 5.

![Figure 5. FPGA architecture](image)

Each logic block has a mixture of combinatorial blocks (or LUTs, for look-up tables) and sequential, storing elements (flip-flops), in which the implementation is made. Depending of the complexity of the logic blocks, the device is classed as fine-, medium- or coarse-grain. Figure 6 shows an example of the first case and Figure 7 of the second case.

![Figure 6. Fine-grain logic block](image)

![Figure 7. Medium-grain logic block](image)
To define a particular architecture, the designer describes it by means of a Hardware Description Language (or HDL). Through it, a rather high-level description can be presented to the synthesis tool which takes it and generates a configuration file that contain information for the device indicating which fuses must be kept and which ones should be blocked [15]. Figure 8 shows an example of an HDL code intended to describe in a high level fashion a resettable, loadable, ascending counter.

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all; -- for the unsigned type
type generic;
width : in natural := 32;
port:
  rst : in std_logic;
  clk : in std_logic;
  load : in std_logic;
  data : in std_logic_vector(width-1 downto 0);
  q : out std_logic_vector(width-1 downto 0);
end entity COUNTER;
architecture RTL of COUNTER is
  signal cnt : unsigned(width-1 downto 0);
beg
  process(rst, clk) is
    begin
      if rst = '1' then
        cnt <= (others => '0');
      elsif rising_edge(clk) then
        if load = '1' then
          cnt <= unsigned(data); -- converted to unsigned
        else
          cnt <= cnt + 1;
        end if;
      end if;
    end process;
    q <= std_logic_vector(cnt); -- converted back to slv
  end architecture RTL;

Figure 8. HDL code example

Figure 9 shows the main aspects of the implementation of Cyto Crypto in an FPGA. It was implemented on an Altera EP3SL50F484C2. It includes two Register files (RF1 and RF3) to store the initial values for genes and proteins respectively. A special register file (RF2) is used to store intermediate values obtained during the shifting stages, case in which signal Flag is observed when the value present in the register is valid. The different moves performed on genes are managed with the help of three counters (COUNT-Address, COUNT1, and COUNT2). Finally, some comparators are used to compare the input data MSG with the RF2 output in encryption mode (CMP1) or with the RF3 outputs in decryption mode (CMP2).

Table 9 presents the resources used by CytoCrypto for different alphabet lengths. Observe how, in spite of the changes in the number of proteins, the maximum frequency for the implementation, which is an indication of how fast the processing can be done, only decreases very slightly, even for a drastic change in the alphabet length such as that from eight to 256. The amount of resources are only a small percentage of those present in the device, allowing the insertion of other systems which can interact with Cyto Crypto in the same device, providing a convenient way of adding a service.

Figure 10 shows a comparison between combinatorial and sequential resources required for different lengths in alphabet for Cyto Crypto. It is observed how the amount of resources increases linearly with a slope of less than the unit. When the number of proteins is duplicated, the increment in the amount of resources is inferior to the double. This is particularly important for the case of LUTs, which are a very precious resource in reconfigurable hardware, since registers can be provided externally given a situation.

Table 9. Physical resources required by CytoCrypto

<table>
<thead>
<tr>
<th># Proteins</th>
<th>LUTs</th>
<th>Registers</th>
<th>Fmax(MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>208</td>
<td>254</td>
<td>244.62</td>
</tr>
<tr>
<td>16</td>
<td>278</td>
<td>384</td>
<td>209.03</td>
</tr>
<tr>
<td>32</td>
<td>431</td>
<td>643</td>
<td>216.26</td>
</tr>
<tr>
<td>64</td>
<td>720</td>
<td>1159</td>
<td>198.22</td>
</tr>
<tr>
<td>128</td>
<td>1069</td>
<td>2186</td>
<td>167.36</td>
</tr>
<tr>
<td>256</td>
<td>1873</td>
<td>4236</td>
<td>160.93</td>
</tr>
</tbody>
</table>

Figure 10. CytoCrypto Area for different alphabet sizes

Figure 11 presents a snapshot of the simulation of the system during the encryption process. The circles show the current letter of the input message and the corresponding produced ciphers. Figure 12 presents another snapshot in this case during the decryption process. The circles again show the input cipher and the corresponding decrypted letter.
4. Tests and results

In order to determine how difficult breaking a cytoly cipher could be, several tests were run. The scenario offers sets with combinations of:

- Genome length: from seven to 12 positions.
- Message length: from ten to 15 characters.
- Alphabet length: from five to seven characters.

The hacker is allowed to know this information. Moreover, he was given to know the language the message is written in. This information is not mandatory, since users are allowed to exchange information in the language they desire, however, without this knowledge; hackers were unable to detect messages ciphered by Cyto Crypto, even using alphabets with only some characters. This evidences the strength of the proposal.

Nonetheless, tests were run in order to establish a reference to provide data for comparison and parameter analysis. For each set, 10 runs, each using up to 12 machine hours were allowed for the hacker to do his attacks, using a desktop computer based on an Intel Core i7 microprocessor, running with a frequency of 2.8 GHz and an L3 Cache of 8 MB. The test consisted of trying to enter to an email account by providing a known username and guessing the password, ciphered by Cyto Crypto. After the ten runs, the one with the lowest time was chosen. Previous to show the results, let us consider how probable would be that an attacker can successfully decode a cytoly cipher. Since there is no way to establish any kind of pattern to predict how substitution of translation proceeds, the only possibility is to try a brute force approach. Hence, it is important to dimension the space the attacker must explore. In its simplest version, Cyto Crypto manages three independent degrees of variability:

- Genome size
- Gene distribution
- Gene expression

Genome size is the total number of genes, including both coding and non-coding genes. Gene distribution has to do with how genes are initially located. Gene expression is related to what the relationship between coding genes and transcription factors is. Combining these three parameters, the number of possible scenarios the attacker must explore is calculated as:

\[
\frac{P!}{(P-G)!} \times G! \times P!
\]  

Equation 1 has three parts, each one corresponding to one of the variability degrees. The first term calculates the possible permutations with respect to the positions coding genes can occupy in the genome, the second one is the possible orderings gene distribution can have with respect to their affinity to transcription factors, and the third one determines the number of possible associations between genes and polypeptides. For the example presented before, with \( P=9 \) and \( G=5 \), the number of possible scenarios is an astonishing \( 6.58\times10^{11} \). This value is on its own excessive and increases very fast with little increments in parameters, getting unmanageable figures even using massive computation.

For the smallest set, with \( P=7 \) and \( G=5 \), the search space is superior to 1524 million combinations. This does not consider the number of combinations that must be explored later when the candidate deciphered message is obtained, number that depends on both the length of the message and the interpretation of it as sense information, an aspect that depends on many other variables. The fact is that the search exploration is only a small value in the number of possible combinations the attack may employ and nonetheless they are astonishing even for the simplest case.

Table 10 shows the results for some sets. As seen, the time necessary for a successful attack increases exponentially and is only meaningful for very small sizes both for alphabet and genome. With values as small as nine positions and an alphabet length of five characters, the time required is near the limit time allowed. With just a very little increase in such a value (by increasing the number of positions to 12) the attack completely fails, even reducing the length of the message. A theoretical calculation for the last two rows of Table 10 gives values of seven and 19260 years respectively, certainly not to be considered in any practical attack. In a real environment, sizes for genome and alphabet will be greater, which guarantees the highest security for messages ciphered with Cyto Crypto. The fact that ciphering is performed in real time is another characteristic that drastically reduces the possibility for a successful attack.

Table 10. Tests results for Cyto Crypto attacking

<table>
<thead>
<tr>
<th>Genome</th>
<th>Msg Len</th>
<th>Alpha Len</th>
<th>Hack Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>10</td>
<td>5</td>
<td>57 segs</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>7</td>
<td>2.2 hours</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>6</td>
<td>7 hours</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>5</td>
<td>11.5 hours</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>5</td>
<td>Unhacked</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>7</td>
<td>Unhacked</td>
</tr>
</tbody>
</table>

It is interesting to consider in more detail the behavior of Cyto Crypto, to determine what characteristics makes it so strong:

1. The strategy is highly polyalphabetic. Each time a message letter is processed, its representation on the cipher is different. In the example presented before, the message was chosen to have several letters repeated. It has two 3s and three 1s and all of them have different representations. The first 3 was ciphered as 513, whereas the second one was deciphered as 97. The three 1s were deciphered as 4, 2, and 134.

2. The strategy is also highly polysymbolic. The number of ciphers associated to a message letter...
is variable. In the example in question, there are six message letters, and the ciphers created for each one of them are 3, 1, 1, 2, 1, and 3 respectively. Moreover, this fact is optional and depends on the spaces left between genes. If the lengths of the message and the cipher are required to be equal, it is simply a matter of eliminating the intergenic spaces or of only expressing the first possible polypeptide.

3. Identical letters happening consecutive in the message are processed as if they were completely different. In the example presented, the last two 1 s were ciphered as 2 and 134, without any evident relationship between these two strings.

4. The next position a gene will occupy is not easily deduced. It depends on the current interaction of three independent values: the input letter, the gene distribution, and the polypeptides expressed.

5. Complexity of the implementation is very low, being in general similar to that of a classical method. For a conventional transposition-based approach, complexity is \( O(N*D) \), with \( N \) being the number of letters in the text message and \( D \) the average distance where to find the replacement symbol. Cyto complexity is, at worst, \( O(N*(G+S)) \), where \( N \) is as before, \( G \) is the number of genes, and \( S \) is the number of spaces, inactive genes, in the genome. Since \( G+S \) is the total length of the genome, it follows that the complexity of Cyto is similar to the conventional transposition approach and, however, the strength of Cyto is much greater.

Suppose an attacker has access to much more machine power, for instance through a grid of computers working in parallel. Cyto crypto can be instructed to increase the complexity of the deciphering in many ways; turning unauthorized deciphering impossible in practice. Moreover, because of the astronomical number of possibilities that arises, a brute-force attack is also prohibitive. The level of sophistication that can be added is only limited by the imagination of the programmer. Following is the explanation of two suggested alternatives: multiple gene levels, and variable jumping command.

**Multiple gene levels** can be added through gene regulatory networks. Consider the case of two levels. Message will be now affine with regulatory genes, which will create TFs affine with structural genes. Besides of making notably difficult the work of an intruder, this alternative adds another interesting feature: the management of an internal alphabet, different from that used by TFs and polypeptides, i.e., messages and ciphers will continue using a common alphabet, but regulatory-gene-created TFs and structural genes can be defined in another alphabet, completely hidden from intruders. In this case, it is evident that, in principle, the possibility of deciphering is zero.

**Variable jumping command** can be built by modifying the gene expression. Instead of calculating the skip length just by using the own polypeptide, a more elaborated alternative can be easily defined, one more robust by providing variable, unpredictable lengths. Suppose one scenario where genes create not only the cipher polypeptide, but also a second polypeptide used to determine the skip length. This second polypeptide can direct length calculation by some method. As an example, consider that these skip polypeptides are created with a random value in the range 0 to 3, where this value will mean indirect addressing for obtaining the skip length. Hence, a possible coding is: 0-use the gene on the left, 1-use the gene on the right, 2-use the message letter, 3-use the polypeptide (which is the simple case explained before). Increasing the range, more coding is possible, for example, to vary not only the skip length, but also the skip direction (left or right).

5. Conclusions

- The inspiration for Cyto Crypto is taken from nature, in particular from the processes happening inside biological cells, more precisely, the way genes produce proteins and the way special kind of proteins modulate genes’ activity.
- This biological inspiration makes Cyto Crypto very simple and yet very strong in the degree of difficulty it imposes for a possible attack. Attacks were only possible for tiny combinations of the parameters: alphabet size, genome size, and message size. For more conventional values, the time required for a successful attack using brute-force techniques is in the range of years, even using state-of-the-art machines.
- It was devised a method, Cyto Crypto, for deciphering and deciphering messages. The approach can work in a character-based, streaming way, allowing real-time behavior.
- Because of the degree of confusion and diffusion Cyto Crypto imposes, the cipher created is highly polyalphabetic and highly polysymbolic, making the attempt to deduce any kind of predictable pattern negligible.
- Although the degree of safety Cyto Crypto generates is extremely high, its complexity is very low, being at the level of a conventional transposition-based approach. This allows processing of large amounts of data in short times.
- Since Cyto Crypto does not required sophisticated mathematics, it can be implemented in simple machines. In fact, an implementation in reconfigurable hardware was carried out, using only a small fraction of a basic reconfigurable hardware device.

Acknowledgments

The author thanks Colciencias, the Colombia Science Foundation, for its support through his doctorate studies, from which this work emanated. He also expresses his gratitude to professors Jaime Velasco and Vladimir Trujillo for their collaboration and advice with the hardware implementation of Cyto Crypto.

References


