Use of Cryptographic Ideas to Interpret Biological Phenomena (*and Vice Versa*)

Massimo Regoli

1 Introduction

The RNA-Crypto System (shortly RCS) is a symmetric key algorithm to cipher data. This algorithm, as shown below, has the peculiarity to expand the message to be encrypted hiding the ciphered message itself within a set of *garbage* and *control information*.

The idea for this new algorithm starts from the observation of nature. In particular from the observation of RNA behavior and some of its properties.

In particular the RNA sequences has some sections called *Introns*. Introns, derived from the term "intragenic regions", are non-coding sections of precursor mRNA (*pre–m*RNA) or other RNAs, that are removed (spliced out of the RNA) before the mature RNA is formed. Once the introns have been spliced out of a *pre–m*RNA, the resulting mRNA sequence is ready to be translated into a protein. The corresponding parts of a gene are known as *introns* as well.

The nature and the role of Introns in the *pre–m*RNA is not clear and it is under ponderous researches by Biologists but, in our case, we will use the presence of Introns, in the RNA-Crypto System output, as a strong method to add only apparently chaotic and non coding information with an unnecessary behavior in the access to the secret key to code the messages.

In the RNA-Crypto System algorithm the *introns* are sections of the ciphered message with non–coding information as well as in the precursor mRNA. But the term "non–coding" does not necessarily mean "junk data".

In this text a new cryptographic algorithm is described starting from a mathematical point of view.

1.1 Interpretation

This algorithm can be used to code clear messages (and the main scope of this job regard this application), but it can be used as well as to suggest to biologists to consider redundancy in the *pre–m*RNA sequences as a mechanism used by nature as a sort of protection against a possible decoding attack from the outside or, from another point of view, they can give to introns an important role in the mechanism for coding the resulting *mRNA* for example the one to achieve future functionalities or to archive old ones.

2 Ingredients

To introduce the RCS algorithm let us start to give some important ingredients.

2.1 Spaces

We introduce for the first the spaces:

- $\mathcal{K} := \{\kappa_i | \kappa_i \in K\}$ as the space of all finite sequences in a given space K, of length less or equal to N_K where $\infty > N_K \in \mathbb{N}$;

- $\mathcal{M} := \{\sigma_i | \sigma_i \in M\}$, as the space of all finite sequences in M of length less or equal to N_M, where $\infty > N_M \in \mathbb{N}$ (could be $M = K$)
- $S = S_1 \cup S_2$ with $S_1 \cap S_2 = \emptyset$ with $|S| < \infty$.

Namely $\mathcal{K}$ is the set of *Secret Keys of length* $\leq N_K$, $\mathcal{M}$ is the set of *Clear Messages of length* $\leq N_M$, K and M are finite spaces of *symbols* (just to fix the ideas should be: $K = M = \{0, 1\}$ the standard binary digits). Also S is a finite set of symbols (or functions) and we can consider it as the *set of all possible actions* that act on the clear message.

At least we introduce the following spaces:

- $\mathcal{O} = \mathcal{A} \bigcup \mathcal{B}$ where $\mathcal{A} = A^{N_A}$ and $\mathcal{B} = \bigcup_i \mathcal{B}_i$ where $\mathcal{B}_i = A^{N_A} \times B_i^{N_{B_i}}$. Also in this case A and B_i are finite sets of symbols as above.

From a biological point of view the space $\mathcal{A}$ is the *exons* space and $\mathcal{B}$ is the *introns* space.

$\mathcal{O}$ is the base space for *Coded Messages* (a coded messages will be a finite sequences of elements of $\mathcal{O}$).

After this preliminary description of necessary spaces, now we can introduce some useful definitions:

Definition 1 *Be $\kappa \in \mathcal{K}$. We define for $m \in \mathbb{N}$ and for each $i \in \{1, \ldots, N_K\}$,*

$$\tilde{\kappa}_{i,m} := \left\{\kappa_i, \kappa_{i\dot{+}1}, \ldots, \kappa_{i\dot{+}(m-1)}\right\}$$

with $0 \leq m \leq N_K$ as a subsequence of κ

($m = 0$ represent a subsequence of length 0).

Definition 2 *Be $\sigma \in \mathcal{M}$. We define for $n \in \mathbb{N}$ and for each $i \in \{1, \ldots, N_M\}$,*

$$\tilde{\sigma}_{i,n} := \left\{\sigma_i, \sigma_{i\dot{+}1}, \ldots, \sigma_{i\dot{+}(n-1)}\right\}$$

with $0 \leq n \leq N_M$ as a subsequence of σ.

($n = 0$ represent a subsequence of length 0).

In the above definitions the operator $\dot{+}$ is the sum with appropriate module depending on the length of the sequence and, for simplicity, in the following we will use the form $\tilde{\kappa}_{i,m} \equiv \tilde{\kappa}_i$ and $\tilde{\sigma}_{i,n} \equiv \tilde{\sigma}_i$ moreover holds: $\tilde{\kappa}_i \in \mathcal{K}$ and $\tilde{\sigma}_i \in \mathcal{M}$

Definition 3 *Be $\mathcal{C} = \{\Sigma_i | \Sigma_i \in \mathcal{O}\}$ the space of all finite sequences in $\mathcal{O}$.*

$\mathcal{C}$ is the set of *Coded Messages*.

2.2 Functions

To reach our goal we must introduce two kinds of functions:

- **Coding functions**: these functions are used to:
  - transform a portion of the clear message in a portion of the coded message (*exons*)
  - insert some *apparently* redundant information in the coded message (*introns*)
- **Operational functions**: these functions are used to:
  - modify the local state of the coding system
  - modify the global state of the variables of the system (see below)

For the first we start with the definition of a family of *coding functions* $f_\alpha : \mathcal{M} \times \mathcal{K} \to \mathcal{O}$ with $\alpha \in S$:

$$f_\alpha(\sigma, \kappa) = \Sigma = \begin{cases} \varepsilon & \text{if } \alpha \in S_1 \\ \iota & \text{if } \alpha \in S_2 \end{cases} \tag{1}$$

where $\varepsilon \in \mathcal{A}$ and $\iota \in \mathcal{B}$.

The functions f_α must be chosen in such way to guarantee the existence of a function $\bar{f} : \mathcal{O} \times \mathcal{K} \to \mathcal{M}$ such that

$$\bar{f}(\Sigma, \kappa) = \bar{f}(f_\alpha(\sigma, \kappa), \kappa) = \begin{cases} \sigma & \text{if } \alpha \in S_1 \\ \emptyset & \text{if } \alpha \in S_2 \end{cases} \tag{2}$$

Now we introduce the definition of some operational functions: be $g_\alpha : \mathcal{K} \times \mathcal{O} \to \mathcal{K}$ as follow:

$$g_\alpha(\kappa, \Sigma) = \bar{g}(\kappa, \Sigma) = \kappa^* \tag{3}$$

the existence of a function $\bar{g}$ that does not depend from α ensures the chaoticity of the system.

For the last, we need, as a member of the set of the operational functions, the trivial characteristic function $\chi_{S_1} : S \to \{0, 1\}$

2.3 Global variables and further definitions

Now let us introduce some further definitions:

Definition 4 *Be $\kappa \in \mathcal{K}$ a Pre Shared Key (PSK) of length N_K.*

In cryptography, a pre–shared key or PSK is a shared secret which was previously shared between the two parties using some secure channel before it needs to be used.

Definition 5 *Be $\{\alpha\} = \{\alpha_i | \alpha_i \in S\}$ a random sequence of arbitrary length.*

A random sequence is a kind of stochastic process. In short, a random sequence is a sequence of random variables. In computer science it comes from a random number generator, often abbreviated as RNG, that is a computational device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. **appear random**.

Definition 6 *Be $\sigma \in \mathcal{M}$ a message of length N_M.*

In communications science, a message is information which is sent from a source to a receiver.

Definition 7 *$\Sigma \in \mathcal{C}$ will be the coded message of length that will depend on the system and the variables state.*

In cryptography, a coded message is a clear message transformed into an obscured form, preventing those who do not possess special information, or *key*, required to apply the transform from understanding what is actually transmitted.

2.4 Biological parallelism

From the biological point of view the PSK and the random sequence $\{\alpha\}$ represent the whole mechanism of *splicing* (this is the process by which *pre–m*RNA is modified to remove certain stretches of non-coding sequences called exactly *introns*), while the coded message is the *pre–m*RNA itself and the clear message is the final *mRNA*.

3 The algorithm

3.1 Coding

Suppose to have a $\kappa \in \mathcal{K}$ as a Pre Shared Key (PSK) of length N_K and a message $\sigma \in \mathcal{M}$ of length N_M and that we have fixed two numbers $n, m \in \mathbb{N}$. (For simplicity and without affecting the generality of the system, we can suppose that N_M is a multiple of n).

Using the above definitions for α_i, $\tilde{\sigma}_{l_i}$ and $\tilde{\kappa}_{j_i}$ we can define the following algorithm:

$$\begin{cases} f_{\alpha_i}(\tilde{\sigma}_{l_i}, \tilde{\kappa}_{j_i}) & = \Sigma_i = \begin{cases} \varepsilon_i & \text{if } \alpha_i \in S_1 \\ \iota_i & \text{if } \alpha_i \in S_2 \end{cases} \\ l_{i+1} & = l_i + \chi_{S_1} n \\ \tilde{\kappa}_{j_{i+1}} & = g_{\alpha_i}(\kappa, \Sigma_i) \end{cases} \tag{4}$$

where $\varepsilon_i \in \mathcal{A}$ and $\iota_i \in \mathcal{B}$.

Definition 8 *The sequence $\Sigma = \{\Sigma_1, \ldots, \Sigma_N\} \in \mathcal{C}$ is the coded message.*

3.2 Decoding

For the decoding phase suppose to have $\kappa \in \mathcal{K}$ as a Pre Shared Key (PSK) of length N_K and a coded message $\Sigma \in \mathcal{C}$ of arbitrary length, and suppose that we have fixed two numbers $n, m \in \mathbb{N}$.

Using the above definitions for $\tilde{\sigma}_{l_i}$ and $\tilde{\kappa}_{j_i}$ we can define the following algorithm:

$$\begin{cases} \begin{cases} \bar{f}(\Sigma_i, \tilde{\kappa}_{j_i}) & = \tilde{\sigma}_i \\ \tilde{\kappa}_{j_{i+1}} & = \bar{g}(\kappa, \Sigma_i) \end{cases} & \text{if } \Sigma_i \in \mathcal{A} \\ \begin{cases} \bar{f}(\Sigma_i, \tilde{\kappa}_{j_i}) & = \emptyset \\ \tilde{\kappa}_{j_{i+1}} & = \bar{g}(\kappa, \Sigma_i) \end{cases} & \text{if } \Sigma_i \in \mathcal{B}_* \subset \mathcal{B} \end{cases} \tag{5}$$

Note that, thanks to the definition of $\bar{g}$, in the decoding phase is not necessary to know the random $\{\alpha_i\}$ sequence.

3.3 Preliminary observations

The coded message depends on:

- The clear message;
- The secret Key (PSK);
- The random sequence $\{\alpha\}$;

and it important to understand that the algorithm during the coding phase using the same message and the same PSK can supply different coded messages simply using different $\{\alpha\}$.

It is also important to underline the fact that it is no necessary, during the decoding phase, to know the sequence $\{\alpha\}$ used in the coding phase. This because the information about the $\{\alpha\}$ sequence are intrinsic in the couple $(\Sigma_i, \tilde{\kappa}_{j_i})$.

In this circumstance we want to underline the possible role of *introns* to carry important information for the decoding phase, for example the function $\bar{g}(\kappa, \Sigma)$ can use the information present in $\Sigma \in \mathcal{B}$ to produces a change in the PSK.

Moreover, during the coding phase there is an expansion of the original messages into the coded message, and this expansion comes from several factors, for example it comes from the dimensions of the $\mathcal{B}_i$ and from the number of elements of S_2 in the $\{\alpha\}$ sequence.

4 Early implementations

4.1 The environment

Now let us introduce the first realization of the above algorithm. Step by step we will substitute each ingredient of the algorithm, with its representative object in the implementation.

Starting from the definition of $K = M = A = B_1 = \{0, 1\}$, with $N_K, N_M \in \mathbb{N}$, $N_A = 3$, $N_{B_1} = 2$ and $S = S_1 \cup S_2$ with $S_1 = \{a, b\}$ and $S_2 = \{c, d\}$. $\mathcal{O} = \{0, 1\}^{N_A} \times \left(\{0, 1\}^{N_A} \times B_1^{N_{B_1}}\right)$. Finally $n = 1$ and $m = 3$.

4.2 Functions and implementation

To introduce the functions f_α the table 1 $T : \mathcal{A} \times K^n \to S$ will be useful using the following rules:

1) if $\alpha \in S_1$
   - be $c = \tilde{\kappa}_j$
   - if $\tilde{\sigma}_i = \sigma_i = 0$ then localize a binary number $000 \leq r \leq 111$ such that $T_{r,c} = a$ (or $T_{r,c} = b$)
   - if $\tilde{\sigma}_i = \sigma_i = 1$ then localize a binary number $000 \leq r \leq 111$ such that $T_{r,c} = b$ (or $T_{r,c} = a$)
   - now let $\Sigma_i = (\Sigma_{i,1}, \Sigma_{i,2}, \Sigma_{i,3}) = (r_1, r_2, r_3)$ (i.e. an exon $\Sigma_i \in \mathcal{A}$).
2) if $\alpha \in S_2$
   - be $c = \tilde{\kappa}_j$
   - localize a binary number $000 \leq r \leq 111$ such that $T_{r,c} = \alpha$
   - now let $\Sigma_i = ((\Sigma_{i,1}, \Sigma_{i,2}, \Sigma_{i,3}), (\Sigma_{i,4}, \Sigma_{i,5})) = ((r_1, r_2, r_3), (\Sigma_{i,4}, \Sigma_{i,5}))$, (i.e. an intron $\Sigma_i \in \mathcal{B}$)
   - the other 2 components of the vector (i.e. $\Sigma_{i,4}, \Sigma_{i,5}$) will be given using a random choice in the binary range $\{00, \ldots, 11\}$

		$\tilde{\kappa}_j$							
		000	001	000	011	100	101	110	111
$(\Sigma_{i,1}, \Sigma_{i,2}, \Sigma_{i,3})$	000	a	b	c	d	a	b	c	d
	001	b	c	d	a	b	c	d	a
	010	c	d	a	b	c	d	a	b
	011	d	a	b	c	d	a	b	c
	100	a	b	c	d	a	b	c	d
	101	b	c	d	a	b	c	d	a
	110	c	d	a	b	c	d	a	b
	111	d	a	b	c	d	a	b	c

Table 1: The seek table $T_{r,c}$

Now for the $\bar{g}$ function we will use the following rules:

- obtain r, c as above
- if $T_{r,c} \in \{a, b\}$ then $\bar{g}$ will return $\tilde{\kappa}_{j_i+m}$
- if $T_{r,c} = c$ then $\bar{g}$ will return $\tilde{\kappa}_{j_i+(\Sigma_{i,4}\Sigma_{i,5})_{10}}$
- if $T_{r,c} = d$ then $\bar{g}$ will return $\tilde{\kappa}_{j_i-(\Sigma_{i,4}\Sigma_{i,5})_{10}}$

where the subscript 10 means the decimal representation of the binary number in parenthesis.

The last function we must define is $\bar{f} : \mathcal{O} \times \mathcal{K} \to \mathcal{M}$. Remember that conditions in equation (2) must hold.

The function $\bar{f}(\Sigma_i, \tilde{\kappa}_j)$ could be able to understand if $\Sigma_i \in \mathcal{A}$ or $\Sigma_i \in \mathcal{B}$. Starting from the fact that Σ (i.e. the entire coded message) is a sequence of 0 and 1, we can extract the single element in terms of bits (i.e. $\tilde{\Sigma}_{l_i}, \tilde{\Sigma}_{l_i+1}, \ldots$). Following this strategy we can define:

$$\bar{f}(\Sigma_i, \tilde{\kappa}_{j_i}) := \bar{f}((\tilde{\Sigma}_{l_i}, \tilde{\Sigma}_{l_i+1}, \ldots), \tilde{\kappa}_{j_i}) \tag{6}$$

the new function act in this way:

- be $r = (\tilde{\Sigma}_{l_i}, \tilde{\Sigma}_{l_i+1}, \tilde{\Sigma}_{l_i+2})$
- be $c = \tilde{\kappa}_{j_i}$
- if $T_{r,c} = a$ then $\bar{f}$ returns $\sigma_i = \{0\}$ (or $\sigma_i = \{1\}$)
- if $T_{r,c} = b$ then $\bar{f}$ returns $\sigma_i = \{1\}$ (or $\sigma_i = \{0\}$)
- else $\bar{f}$ return $\sigma_i = \emptyset$

The function $\bar{g}$ acts as above but using $(\tilde{\Sigma}_{l_i}, \tilde{\Sigma}_{l_i+1}, \tilde{\Sigma}_{l_i+2})$ if $T_{r,c} = \{a, b\}$ or $(\tilde{\Sigma}_{l_i}, \tilde{\Sigma}_{l_i+1}, \tilde{\Sigma}_{l_i+2}, \tilde{\Sigma}_{l_i+3}, \tilde{\Sigma}_{l_i+4})$ if $T_{r,c} = \{c, d\}$.

4.3 Observation

In this simple implementation of the RNA algorithm, redundancy is given by the following considerations:

- for each processed bit in the clear message 3 bits will be inserted in the coded one
- each *intron* will insert in the coded message 5 bits
- we use an uniform distribution for the generation of the sequence $\{\alpha_i\}$
- then we can suppose that the length of the sequence $\{\alpha_i\}$ is $\|\{\alpha_i\}\| = \|\{\alpha_i | \alpha_i \in S_1\}\| + \|\{\alpha_i | \alpha_i \in S_2\}\|$
- of course, in this implementation, must be $\|\{\alpha_i | \alpha_i \in S_1\}\| = N_M$

Then, if the original message has a length of N_M bits, the length of coded one is $l_c \approx 3N_M + 5N_M = 8N_M$.

But it is important to underline that this implementation uses a redundant table $T_{r,c}$, in fact for each c there are two r that satisfy the condition $T_{r,c} = \alpha$.

But of course if we want to reduce the expansion of the message we can replace it with the one in table 2 (in this version must be $N_A = 2$ and $m = 2$).

		$\tilde{\kappa}_j$			
		00	01	10	11
$(\Sigma_{i,1}, \Sigma_{i,2})$	00	a	b	c	d
	01	b	c	d	a
	10	c	d	a	b
	11	d	a	b	c

Table 2: An alternative seek table $T_{r,c}$

in this case the message explosion will be $l_c \approx 2N_M + 4N_M = 6N_M$.

Another solution in order to diminish the explosion of the coded message is that to use an asymmetric probability function for the generation of the $\{\alpha_i\}$ for example:

α	p
a	3/8
b	3/8
c	1/8
d	1/8

Table 3:

Using these last two optimizations the approximate length of the coded message will be reduced to $l_c \approx 3N_M$.

4.4 A biological implementation

Now, as an example, we want to create an implementation of our model nearer the biology. For this let us introduce again the ingredients for the new model:

Starting from the definition of $K = \mathbb{N}$, $M = A = \{a, b, c, d\}$, with $N_K, N_M \in \mathbb{N}$, and $S = S_1 \cup S_2$ with $S_1 = \{c\}$ and $S_2 = \{nc\}$. $\mathcal{O} = \{A^{N_A} \backslash (a, a, a)\} \cup \left(\{a, a, a\} \times \left(\bigcup_i A^{N_{B_i}}\right)\right)$, $N_A = n = 3$, $N_{B_i} \in \mathbb{N}$ and $m = 1$.

The function f will be:

$$\Sigma_{i^*} = f_\alpha(\tilde{\sigma}_i, \tilde{\kappa}_{j_i}) = \begin{cases} \tilde{\sigma}_i & \text{if } \alpha \in S_1 \\ (a, a, a, B) \text{ with } B \in A^{\tilde{\kappa}_{j_i}} & \text{if } \alpha \in S_2 \end{cases} \tag{7}$$

in this case the element $(a, a, a) \in A^3$ is the codon to localize the *beginning of the introns* in the sequence (it will never appear in the clear code) and the function $\bar{f}$ will be:

$$\sigma_{i^*} = \bar{f}(\Sigma_i, \tilde{\kappa}_{j_i}) = \begin{cases} (\Sigma_{i,1}, \Sigma_{i,2}, \Sigma_{i,3}) & \text{if } (\Sigma_{i,1}, \Sigma_{i,2}, \Sigma_{i,3}) \neq (a, a, a) \\ \emptyset & \text{otherwise} \end{cases} \tag{8}$$

In both cases holds $\tilde{\kappa}_{j_{i+1}} = \tilde{\kappa}_{j_i+1}$.

5 The role of the Coding Functions

In section 2.2 we introduce the definition of Coding functions.

These functions play an important role in the complexity and in the lengths of the cipher text.

It is also very important to understand that the number of functions involved in the process of encryption can be huge. In fact, besides the canonical functions introduced in section 4.1 we can add functions that have the role of:

- Change the public / secret keys
  - For example in the cipher text we can insert a sequence of bits of arbitrary length that will replace the public (private) key to decode the rest of the message.
- Insert a long sequences of random bits
  - As before but the inserted bits are ignored.
- Move forward or backward the key pointer position

  - The presence of a disruptive effect on the sequencing access to the secret key can only add more noise in the attacks

- Reset global parameters like message pointer position, key pointer position, secret and public data
  - See before

is easy to see that the number of Coding functions that can be inserted in the encoding mechanism is great and may become part of the shared secret information.

In the same way it is obvious that some Coding functions can greatly increase the size of the cipher text as it is also obvious that the inclusion of random bits within the text could create an increase in the randomness of the cipher text.

6 Statistical Analysis

6.1 Cryptanalysis

Cryptanalysis (from the greek krypts, "hidden" and analein, "break") is the study of methods for obtaining the meaning of encrypted information without having access to secret information which is usually required to perform the operation. Typically it comes to finding a secret key.

Cryptanalysis is the "counterpart" of cryptography, namely the study of techniques for concealing a message, and together form the cryptology, the science of writing hidden.

Cryptanalysis refers also to any attempt to circumvent the security of other cryptographic algorithms and cryptographic protocols.

Although the methods of cryptanalysis usually excludes attacks that are directed to the inherent weaknesses of the method to violate, such as bribery, physical coercion, theft, social engineering, such attacks are often the most productive of cryptanalysis traditional, they are still an important component.

The first cryptanalytic tool used to analyze the RNA-Crypto System is based on the Statistical analysis. For this purpose we used a famous battery of tests called *Diehard* (or *Dieharder* in the new version).

- The *Diehard tests* is a battery of statistical tests for measuring the quality of a set of random numbers.
- It is cited by NIST as one of the best statistical suite for testing *randomness*
- It was developed by **George Marsaglia** over several years and first published in 1995 and then maintained and improved by **Robert Brown** at the Duke University.
- We focused our attention to the following tests:
  1. Birthday spacings
  2. Overlapping permutations
  3. Ranks of matrices
  4. Monkey tests
  5. Count the 1s
  6. Parking lot test
  7. Minimum distance test
  8. Random spheres test
  9. The squeeze test
  10. Overlapping sums test
  11. Runs test
  12. The craps test

all these tests are well described in the software package and in literature.

base	binary
a	00
c	01
g	10
t|u	11

Table 4: Translation table

6.2 Our Idea

In nature a nucleotides ribbon is a sequence of (almost always) 4 symbols (a, c, g, t|u).

In computer science a 'string' is a sequence of 2 symbols (0, 1).

Now, suppose to translate the 4 symbols as in table 4: then we have a correspondence between bits and nucleotides.

With this assumption the *Diehard tests* can be used to assess the randomness of this *binary conversion* of a sequence of *nucleotides* as well as the sequence of an *encrypted message*.

Our idea is based on the following questions:

1. Have RNA ribbons a good random behavior according to DieHard(er) tests?
2. Are RNA-Crypto System sequences a good random behavior according to DieHard(er) tests?

The answers can be given by looking at test results.

6.3 BIO results

6.3.1 The Experiment

We used only some very long sequences of nucleotides from on line standard free databases, this because the tests run well on more than 12 M bytes of data.

Accordingly with some simple calculations, holds the formula 1M bytes = $\frac{1}{4}$M bases, then we need sequences of at least 48 M bases, so our attention was focused also on the human genome which can gave very long sequences.

6.3.2 The protocol

1. Get a sequence from the database (longer than 48 M bases)
2. Translate it in binary mode
3. Run the DH test on it

6.3.3 The Data

For the biological sequences we uses the following:

- *Caenorhabditis elegans* chromosomes I–V, complete sequences.
- *Wallaby*, whole genome.
- *Human* chromosome 14 complete sequence
- *Drosophila melanogaster* some chromosomes, complete sequences

all encoded use table 4.

6.3.4 The Results

In table 5 we can see the results of our experiment using the Biological data. It is obvious that Bio–sequences are not random at all.

The reason for this fact could be found, of course, in:

- Some parts of a *pre–m*RNA sequence could be highly repetitive (Satellite, Minisatellite, ...)
- Some parts of a *pre–m*RNA sequence could be made up a very long sequence of the same nucleotide (Polyadenylation tail, ...)

n	Test name	Status
1	Birthday Spacings	FAIL
2	Overlapping Permutations	FAIL
3	Ranks of 31x31 and 32x32 matrices	FAIL
4	Ranks of 6x8 Matrices	FAIL
5	Monkey Tests on 20–bit Words	FAIL
6	Monkey Tests OPSO,OQSO,DNA	FAIL
7	Count the 1's in a Stream of Bytes	FAIL
8	Count the 1's in Specific Bytes	FAIL
9	Parking Lot Test	FAIL
10	Minimum Distance Test	FAIL
11	Random Spheres Test	FAIL
12	The Squeeze Test	FAIL
13	Overlapping Sums Test	FAIL
14	Runs Test	FAIL
15	The Craps Test	FAIL

Table 5: Bio results

6.4 RNA-Crypto System results

6.4.1 The Experiment

In the RNA-Crypto System experiment we long sequences of binary data encrypted with our protocol (approximately 100 MBytes of data for each experiment).

6.4.2 The Protocol

The protocol used to estimate the randomness of RNA-Crypto System is:

1. Run the DNACrypto program on a message of opportune length
2. Save the encrypted message
3. Run the DH test considering it as a *random* sequence.

6.4.3 The Data

For the cryptographic sequences we uses:

- A 10 M bytes file filled with ASCII char 'A'
- A 10 M bytes file filled with random binary numbers

- One of the above biological sequence

encoded with the Algorithm

6.4.4 The Results

In table 6 we can see the results of our experiment using the RNA-Crypto System data. It is obvious that RNA-Crypto System sequences are random following the DH tests.

n	Test name	Passed
1	Birthday Spacings	PASS
2	Overlapping Permutations	PASS
3	Ranks of 31x31 and 32x32 matrices	PASS
4	Ranks of 6x8 Matrices	PASS
5	Monkey Tests on 20–bit Words	PASS
6	Monkey Tests OPSO,OQSO,DNA	PASS
7	Count the 1's in a Stream of Bytes	PASS
8	Count the 1's in Specific Bytes	PASS
9	Parking Lot Test	PASS
10	Minimum Distance Test	PASS
11	Random Spheres Test	PASS
12	The Squeeze Test	PASS
13	Overlapping Sums Test	PASS
14	Runs Test	PASS
15	The Craps Test	PASS

Table 6: Crypto results

6.5 First Results – Differences

As expected the results correspond to our ideas on the sequences of nucleotides and maybe also to the ones about the Crypto System. Of course natural phenomena is not *really random* like a cryptographic system Some obvious questions are:

- Why *pre–m*RNA sequences do not pass the statistical tests?
  - Of course they are not random (life is not random)
- But some of the motivations, from the statistical point of view, may be the follows
  - Some parts of a *pre–m*RNA sequence could be highly repetitive (Satellite, Minisatellite, ...)
  - Some parts of a *pre–m*RNA sequence could be made up a very long sequence of the same nucleotide (Polyadenylation tail, ...)
- Is is possible to manipulate the RNA-Crypto System protocol to obtain the same results?

6.6 Changes: Informatics emulates biology

6.6.1 Modify Cryptographic protocol – Phase I

Due to its random nature the cryptographic model has not repeated sequences inside then we introduce a new coding function ρ (the replicator function) that will add these repeated sequences artificially inside the code

These code can be considered:

- active (they act in some way with the system)
- passive (just junk!!!)

6.6.2 Strategies

The replicator function

Definition Be $\rho : \mathcal{M} \times \mathcal{K} \times \mathcal{O} \to \mathcal{O}$ a *junk function*:

$$\rho\left(\sigma, \kappa, o\right) = \Sigma = \iota \tag{9}$$

where $\mathcal{B} \ni \iota = o_{\sigma,\kappa}$ and $o_{\sigma,\kappa}$ is a subsequence of o.

If o is a part of the encoded message, this mechanism implements the repeated sequences phenomenon.

6.6.3 Alter Cryptographic protocol – Phase II

Due to its random nature the cryptographic model has not significant *all–equal* subsequences then we introduce a new coding function τ (stutter function) that will add these *mono–symbols* subsequences artificially.

Also in this case this sub–sequences can be created to be:

- active (they act in some way with the system)
- passive (just junk!!!)

6.6.4 Phase II – Strategies

The *stutter* function

Definition Be $\tau : \mathcal{M} \times \mathcal{K} \to \mathcal{O}$ a *junk function*:

$$\tau\left(\sigma, \kappa\right) = \Sigma = \iota \tag{10}$$

where $\mathcal{B} \ni \iota = \tilde{\iota}^{n_{\sigma,\kappa}}$ and $\tilde{\iota} \in A \cup B_i$, $n_{\sigma,\kappa}$ is an integer.

This mechanism implements the *polyadenylation tail*.

6.7 Results

New results come from statistical analysis on the new cryptographic data.

Note: the result in table 7 holds for all the data in the cryptographic set

6.8 New Results – Differences

In these days we are working hard to prove that the security of the cryptographic system is not affected by the new entrants, nowadays we did not find any attack that can exploit these features, so we are highly confident that the new system has the same level of security of the standard one.

- In fact, even if the spy could isolate the *Introns*, the rest of the message would be exactly equal to the previous version
- Then the safety remains the same (or better)

Moreover adding redundancy we obtained, as a side effect, some properties:

- A certain robustness to error in transmission

n	Test name	Status
1	Birthday Spacings	FAIL
2	Overlapping Permutations	FAIL
3	Ranks of 31x31 and 32x32 matrices	FAIL
4	Ranks of 6x8 Matrices	FAIL
5	Monkey Tests on 20–bit Words	FAIL
6	Monkey Tests OPSO,OQSO,DNA	FAIL
7	Count the 1's in a Stream of Bytes	FAIL
8	Count the 1's in Specific Bytes	FAIL
9	Parking Lot Test	FAIL
10	Minimum Distance Test	FAIL
11	Random Spheres Test	FAIL
12	The Squeeze Test	FAIL
13	Overlapping Sums Test	FAIL
14	Runs Test	FAIL
15	The Craps Test	FAIL

Table 7: New crypto results

  - Indeed, in a very redundant system, the probability that an error in a bit prevents the decoding is very low
  - This also suggests a particular interpretation of redundancy in RNA (protection against excessive mutations as an example)

- Furthermore, the spy, during his attack, does not know whether the piece of code that is trying to attack be an *Exon* or an *Intron*
  - This also suggests another particular interpretation of redundancy in RNA (protection against pathogens ?)

7 Conclusion

In conclusion this job will allow us to take some considerations.

For the first, adding to a good random sequence some artificial noise, we transformed it in a non random sequence but this transformation does not affect the cryptographic security.

Then this means that tests of randomness, while giving a good estimate of safety, are not always infallible and more tests are needed to determine the goodness of an encryption system.

Finally, the similarity between biology and computer science (cryptography) must be investigated further in order to give further interpretation to the biological mechanisms also today still partially obscure.

References

[1] Lewin, B.,"Gene VI" *Oxford University Press*, 1997.

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

[3] Regoli, M., "Bio-Cryptography: A Possible Coding Role for RNA Redundancy", Foundations of Probability and Physics-5 (AIP Conference Proceedings), Accardi EDT, 2008