# New families of cryptographic systems

**Maryna NESTERENKO [†§], Jiri PATERA [†] and Dmytro ZHAVROTSKYI [‡]**

[†] Centre de Recherches Mathématiques , Université de Montréal, 6128, succursale Centre-ville, Montréal, QC, H3C3J7, Canada
E-mail: patera@crm.umontreal.ca

[§] Institute of Mathematics of NAS of Ukraine, 3 Tereshchenkivs'ka Str., Kyiv-4, 01601 Ukraine
E-mail: maryna@imath.kiev.ua

[‡] Interzvyazok, 15 Novokostyantynivska str., Kyiv, 04080, Ukraine
E-mail: dima_zh@isv.com.ua



**Abstract**

A symmetric encryption method based on properties of quasicrystals is proposed. The advantages of the cipher are strict aperiodicity and everywhere discontinuous property as well as the speed of computation, simplicity of implementation and a straightforward possibility of extending the method to encryption of higher dimensional data.

## 1 Introduction

Here we address the 'classical' cryptographic problem, i.e. protection of transmitted and stored data from divulgence and distortion. Such problem frequently arise when data creation or/and data usage are disjoint in time or/and space. There exist many approaches to this problem in modern cryptography, which can be divided into three main groups in accordance with their key primitives, i.e., unkeyed, symmetryc-key and public-key encodings, see [2, 4, 9].

In the present work we propose a new generic encoding procedure based on the aperiodic point sets, called quasicrystals in the physics literature and model sets in the mathematics literature [1, 3]. Generally speaking, such an algorithm is a symmetric stream cipher endowed with strict aperiodicity (no periodic subsets) and everywhere discontinuous properties.

The paper starts by preliminaries and general encoding idea, which is followed by the precise statement of pertinent mathematical ingredients, Sections II and III. In Section 4 applications of the proposed cipher to encoding of bitmap pictures are presented and discussed. The advantages of the approach are in the speed of computation and a straightforward possibility of extending the method to encryption of 3 and higher dimensional data. Possible range of private keys and some other applications of quasicrystals are discussed in Conclusions.

## 2 Preliminaries and encryption procedure

We call cut-and-project quasicrystal a discrete deterministic aperiodic point set $\Lambda$, in a finite-dimensional real Euclidean space. In many ways such quasicrystals resemble lattices in all but the translation symmetry. Sometimes they are even called aperiodic lattices or aperiodic crystals. In this work we make use of two remarkable properties of the quasicrystals: (i) no periodic subsets of any kind are contained in a quasicrystal, and (ii) the discontinuity of the 'star map' between a quasicrystal $\Lambda$ and its 'acceptance window' $\Omega$.

General idea of the proposed approach is to take given digital data, set a one-to-one correspondence between information bites and integer numbers $N$, and then to map points of $N$ on a quasicrystal fragment $\Lambda$ (see the first two lines of Fig. 1). At this stage the data is mapped to the quasicrystal points. It is followed by the application of the star map to the quasicrystal points (see the second and the third lines of Fig. 1), which is a crucial step of the encryption. As a result, we get the quasicrystal points, carrying our data, 'tossed up' in the acceptance window

Ω. It remains to map the content of Ω on a desirable lattice for the output (see the last two lines of Fig. 1).

Decryption proceeds in the opposite direction.

The set of private keys in our encryption method is infinite and is discussed in Conclusions. In addition, one may choose to use any common substitution scheme.

Generation of quasicrystal points is very flexible. It can either be done as a single stream or split into several blocks. Moreover, calculation of the points of each block is easily amenable to parallel computing.

Using quasicrystals, all calculations can be carried out in integers. That guarantees absolute accuracy of coding and decoding procedures, assures the stability of the algorithm.

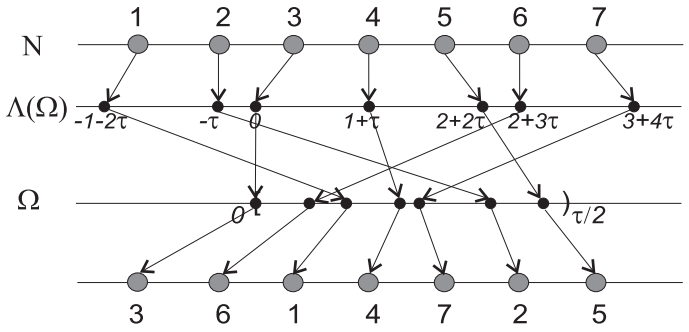


Figure 1: Scheme of the simplest quasicrystal-based cipher

# 3 Mathematical ingredients

## 3.1 Theoretical background

Theoretical background of the work can be found in [1, 3, 5, 7, 8, 10].

Our aim is to consider two-dimensional data. We start with a choice of two straight lines $V_1 \colon y = \tau x$ and $V_2 \colon y = \tau' x$ on a square lattice $\mathbb{Z}_2$ (see Fig. 2), where $\tau$ and $\tau'$ are two irrational numbers to be specified later. The simplest choice is $\tau = \frac{1}{2}(1+\sqrt{5})$ and $\tau' = \frac{1}{2}(1-\sqrt{5})$. One of the lines, say $V_1$, plays the role of the quasicrystal space onto which a 'cut' of the lattice $\mathbb{Z}_2$ is projected. In the other direction $V_2$, we choose a finite interval $\Omega$. Point of the lattice $\mathbb{Z}_2$ becomes a point of the quasicrystal, after projection on $V_1$, provided its projection on $V_2$ falls within $\Omega$, i.e. it is within the 'cut'.

We denote by $\mathbb{Z}[\tau]$ all the real numbers of the form $(a + b\tau)$, with integers $a$ and $b$.

The *star map* between a quasicrystal point $(a + b\tau)$ and the point $(a + b\tau')$ of $\Omega$ is given by

$$\begin{array}{rccl} \star : & \mathbb{Z}[\tau] & \longrightarrow & \mathbb{Z}[\tau'], \\ & (a+b\tau) & \mapsto & (a+b\tau'), \qquad a, b \in \mathbb{Z}. \end{array}$$

There is an unlimited number of choices of the irrational pairs to use in our construction. For example, one may choose as $\tau$ and $\tau'$ the solutions of the two infinite series of the quadratic equations $x^2 = mx + 1,\ m = 1, 2, \ldots$ and $x^2 = mx - 1,\ m = 3, 4, \ldots$.

The one-dimensional *quasicrystal* or *cut and project set* $\Lambda_\tau(\Omega)$ is described as follows,

**Definition 1.** Let $\Omega$ be a finite interval and $\tau$, $\tau'$ be irrational numbers, then

$$\Lambda_\tau(\Omega) = \{a + b\tau \mid a, b \in \mathbb{Z},\ a + b\tau' \in \Omega\},$$

where $\Omega$ is called the *acceptance window* of $\Lambda_\tau(\Omega)$.

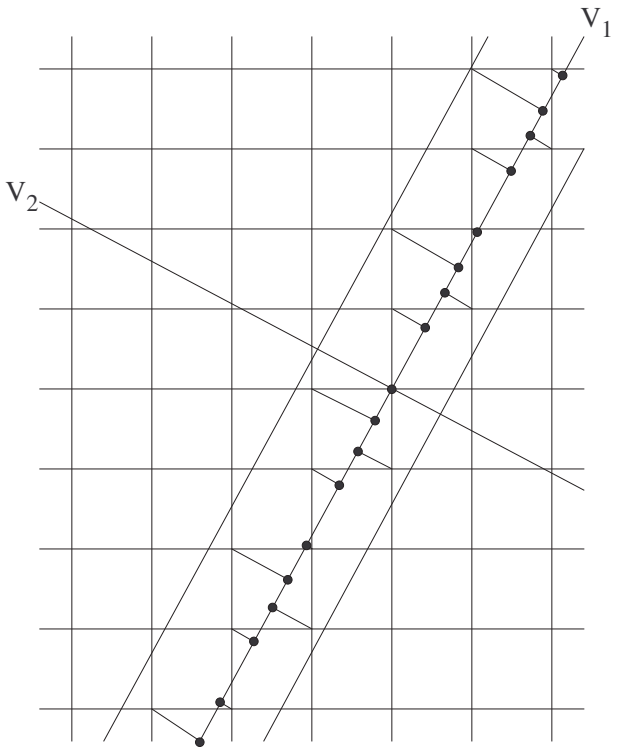


Figure 2: Construction of one-dimensional quasicrystal

Let us underline, that $\Lambda_\tau(\Omega)$ defined here is an infinite, uniformly dense, and uniformly discrete point set, while its window $\Omega$ is a finite interval densely covered by the star map of the points of $\Lambda_\tau(\Omega)$.

From the definition of $\Lambda_\tau(\Omega)$ a number of properties of $\Lambda_\tau(\Omega)$ can be shown [3, 10]. In particular, the aperiodicity, discontinuity of the star map and the existence of only two or three different distances between adjacent points of $\Lambda_\tau(\Omega)$. More precisely, we recall the following theorems.

**Theorem 1.** *Let $\Omega$ be a semi-closed interval. For every $\Lambda_\tau(\Omega)$ there exist positive numbers $\Delta_1, \Delta_2 \in \mathbb{Z}[\tau]$, such that the distances between adjacent points in $\Lambda_\tau(\Omega)$ take values in $\{\Delta_1, \Delta_2, \Delta_1 + \Delta_2\}$. The distances depend only on $\tau, \tau'$ and length of the acceptance window.*

More can be said about the distances between adjacent points, namely

$$\{x_{n+1} - x_n | n \in \mathbb{Z}\} = \begin{cases} \{\Delta_1,\ \Delta_2,\ \Delta_1 + \Delta_2\}, \\ \text{when} \quad \Delta_1^\star - \Delta_2^\star > |\Omega| \\ \{\Delta_1,\ \Delta_2\}, \\ \text{when} \quad \Delta_1^\star - \Delta_2^\star = |\Omega|. \end{cases}$$

One has a general scaling property of quasicrystals:

**Theorem 2.** *For every irrational numbers $\tau, \tau'$, $\tau \neq \tau'$ and a bounded interval $\Omega$, there exist $\tilde{\tau}' \in (-1, 0)$, $\tilde{\tau} > 0$, $s \in \mathbb{R}$ and $\tilde{\Omega}$ (satisfying $\max\{1 + \tilde{\tau}', -\tilde{\tau}'\} < |\tilde{\Omega}| \leq 1$), such that*

$$\Lambda_\tau(\Omega) = s\Lambda_{\tilde{\tau}}(\tilde{\Omega}).$$

*Moreover, the distances between adjacent points in $\Lambda_{\tilde{\tau}}(\tilde{\Omega})$ are equal to $\tilde{\tau}$, $1 + \tilde{\tau}$, $1 + 2\tilde{\tau}$, when $|\tilde{\Omega}| \neq 1$ and to $\tilde{\tau}$, $1 + \tilde{\tau}$, when $|\tilde{\Omega}| = 1$.*

Finally we underline that the 1-dimensional quasicrystal $\Lambda_\tau(\Omega)$ is an infinite discrete aperiodic set of points uniquely determined by the choice of $\tau$, $\tau'$ and of acceptance window $\Omega = [c, c+d)$, where $c$ and $d$ are any real numbers. Due to the theorem 2, without loss of generality, we can consider $\tau' \in (-1, 0)$, $\tau > 0$, $c = 0$ and $d \in (\max\{1 + \tau', -\tau'\}, 1]$.

In higher dimensions, it is adequate for our purposes to take for the quasicrystals a straightforward concatenation of one-dimensional ones in pairwise orthogonal directions.

### 3.2 Computational aspects

First let us introduce two specific examples of one-dimensional quasicrystals $\Lambda_\tau(\Omega)$.

**Example 1.** Take $\tau = \frac{1+\sqrt{5}}{2}$, $\tau' = \frac{1-\sqrt{5}}{2}$, the two roots of the quadratic equation $x^2 = x + 1$. Then $\tau+\tau' = 1$, $\tau\tau' = -1$. Choosing $\Omega = [0, d)$, $d = \frac{\tau}{2}$ the conditions of Theorem 2 are satisfied: $\tau' \approx -0.62 \in (-1, 0)$ and $\tau \approx 1.62 > 0$.

Due to Theorem 2, the distances between adjacent points of the quasicrystal $\Lambda_\tau([0, d))$ are $\Delta_1 = \tau$, $\Delta_2 = 1 + \tau$ and $\Delta_3 = 1 + 2\tau$.

As the seed point of the quasicrystal $\Lambda_\tau([0, d))$, we can choose any point $x = a + b\tau$, such that $x' = a + b\tau' \in [0, d)$. For our example we put $x = 0$.

Unlike Definition 1, which is not constructive, here we have the information for fast generation of quasicrystal points. We can move right or left from the seed point by adding or subtracting one-by-one of the distances $\Delta_1$, $\Delta_2$ or $\Delta_3$.

Suppose we have already established, that $x$ is a point of $\Lambda_\tau([0, d))$, i.e. $x' \in [0, d)$. The point adjacent to $x$ from the right is one of the three

$$x + \tau, \quad x + 1 + \tau \quad \text{or} \quad x + 1 + 2\tau.$$

In order to decide which one is the case, we verify one-by-one the inclusions $x + \tau' \in [0, d)$, $x + 1 + \tau' \in [0, d)$ or $x + 1 + 2\tau' \in [0, d)$. The first confirmed inclusion determines the new quasicrystal point.

In such a way the following two subsets of the quasicrystal $\Lambda_\tau([0, d))$ were obtained

$$\begin{aligned}
&\{\ldots, -1-2\tau,\ -\tau,\ 0,\ 1+\tau,\ 2+2\tau,\ 2+3\tau,\ 3+4\tau, \ldots\} \leftrightarrow \\
&\qquad \leftrightarrow \{\ldots,\ \Delta_2,\ \Delta_1,\ \Delta_2,\ \Delta_2,\ \Delta_1,\ \Delta_2,\ \ldots\}; \\
&\{\ldots, -\tau,\ 0,\ 1+\tau,\ 2+3\tau,\ 3+4\tau, \ldots\} \leftrightarrow \\
&\qquad \leftrightarrow \{\ldots, \Delta_1,\ \Delta_2,\ \Delta_3,\ \Delta_2, \ldots\}.
\end{aligned}$$

Here the second line of each example contains the corresponding sequence of distances between quasicrystal points. The two quasicrystals differ by the sequence in which the three distances $\Delta_1$, $\Delta_2$ and $\Delta_3$ were tried in the construction.

The first case is was shown as the second line on Fig. 1.

**Example 2.** Next quasicrystal $\Lambda_\tau(\Omega)$ is built using the irrationalities $\tau = 1 + \sqrt{2}$, $\tau' = 1 - \sqrt{2}$, roots of the quadratic equation $x^2 = 2x + 1$.

The conditions of Theorem 2 are again satisfied: $\tau' \approx -0.42 \in (-1, 0)$ and $\tau \approx 2.42 > 0$. Putting $\Omega = [0, 1)$, there are only two distances $\Delta_1 = \tau$ and $\Delta_2 = 1 + \tau$ between adjacent points of the quasicrystal $\Lambda_\tau([0, 1))$.

Using the zero point as the seed point, we obtain the cut and project set

$$\begin{aligned}
&\{\ldots, -1-3\tau, -2\tau, -\tau, 0, 1+\tau, 1+2\tau, 2+3\tau, 2+4\tau, 3+5\tau, \ldots\} \\
&\qquad \leftrightarrow \{\ldots,\ \Delta_2,\ \Delta_1,\ \Delta_1,\ \Delta_2,\ \Delta_1,\ \Delta_2,\ \Delta_1,\ \Delta_2,\ \ldots\}.
\end{aligned}$$

It may happen that some point $x'$ is arbitrary close to an end point of $\Omega$. Then it would be a time demanding computational task to decide whether $x' \in \Omega$ or $x' \notin \Omega$. Such a difficulty is simply avoided by disqualifying from our consideration any point $x'$ which comes closer to the boundary of $\Omega$ than the distance $\varepsilon$ agreed in advance.

The real numbers of the form $a + b\tau$ can be understood as given by two integer components $(a, b)$. Since our operations are only addition or subtraction, the arithmetics of such numbers is elementary introduced. Consequently all transformations are performed with absolute precision.

The transformations between the original and encrypted data go through the several stages. At each stage we need only the previous point. Therefore we have only minimal requirements to RAM.

# 4 Application to image coding

In this section fragments of quasicrystal point sets are used for encryption/decryption of 2-dimensional digital data.

Assuming that the data is given on the rectangular grid containing final number $n_1$, $n_2$ of points in each direction, we deal with final fragments $\Lambda^{(i)}(\Omega_i)$, $i = 1, 2$ of the quasicrystal $\Lambda(\Omega) = \Lambda^{(1)}(\Omega_1) \otimes \Lambda^{(2)}(\Omega_2)$. Number of points in each fragment is given by digital data size $n_i = |\Lambda^{(i)}(\Omega_i)|$.

Each $\Omega_i$ is an interval on real axis in mutually orthogonal directions. They can be chosen for both fragments independently.

Our main tool is the one-to-one map (star map) between the finite point sets $\Lambda_i(\Omega_i)$ and its image in $\Omega_i$

$$\Lambda_i(\Omega_i) \quad \longleftrightarrow \quad \Omega_i, \qquad i = 1, 2\,.$$

Note, that roles of $\Lambda_i(\Omega_i)$ and its image in $\Omega_i$ can be interchanged, because the fragments $\Lambda_i(\Omega_i)$ are finite sets of points.

Fast generation of $\Lambda_i(\Omega_i)$ was described in previous section. It requires that one is given the values of $n_1$, $n_2$, $\tau_i$, $\tau_i'$ and $\Omega_i$, and two seed points of $\Lambda_i(\Omega_i)$, $i = 1, 2$.

Suppose one is given a bitmap picture $P(L)$, sampled on the points $(x, y)$ of a fragment $L$ of a rectangular lattice.

Then we treat independently each coordinate of the data using one fragment of the quasicrystal as illustrated in Fig. 1.

## 4.1 Basic encryption algorithm

- Construction of two different 1-dimensional quasicrystals $\Lambda^{(1)}$ and $\Lambda^{(2)}$ of sizes $n_1$ and $n_2$ appropriate for the given data $P(L)$. (Each fragment, as well as its acceptance window–interval can be situated anywhere on the corresponding real axes.)

- Matching one-by-one the sequence of coordinates $x$ with points of $\Lambda^{(1)}$, similarly matching the coordinates $y$ with points of $\Lambda^{(2)}$. (In this way $x$ becomes point $x_1 + x_2\tau$ of the $\Lambda^{(1)}$ and $y$ becomes point $y_1 + y_2\tau$ of the $\Lambda^{(2)}$, where $x_1$, $x_2$, $y_1$, $y_2$ are integers.)

- The main step of the encoding is the star map of $\Lambda^{(1)} \otimes \Lambda^{(2)}$ into the corresponding window $\Omega_1 \otimes \Omega_2$

$$\begin{array}{cc} \Lambda^{(1)}(\Omega_1) \longrightarrow \Omega_1, & \Lambda^{(2)}(\Omega_2) \longrightarrow \Omega_2; \\ x_1 + x_2\tau_1 \mapsto x_1 + x_2\tau_1', & y_1 + y_2\tau_2 \mapsto y_1 + y_2\tau_2'. \end{array}$$

(The sequence of the original data coordinates is dramatically changed due to the discontinuity of the star map.)

- It remains to map the points of encrypted quasicrystal fragments, which are now in $\Omega_1$ and $\Omega_2$, to a sequence of integers along $x$ and $y$ directions.

The present algorithm permutes the data points without changing the value of the data at each point. It is straightforward to consider the possible values of the data function as another discrete point set and to apply the same algorithm to it. Effectively we are thus dealing with a three-dimensional problem, where the data function takes only two values at 3-dimensional points.

All the maps we have used in the encryption algorithm are one-to-one, therefore they can be used in the inverse order for the decryption. For the same reason, nothing prevent us from iterative using our algorithm.

In certain types of data the encryption algorithm leaves residue of the orthogonal directions, see Fig. 3b. It can be avoided in a many different ways. One of them was mentioned above, when the problem was viewed as a three-dimensional one. Another possibility is illustrated on Fig. 3c. The two-dimensional data is taken as an ordered one-dimensional sequence (modification 2). The particularly fast possibility is to introduce the additional cyclic permutations of the points (modification 1), see Fig. 3d.

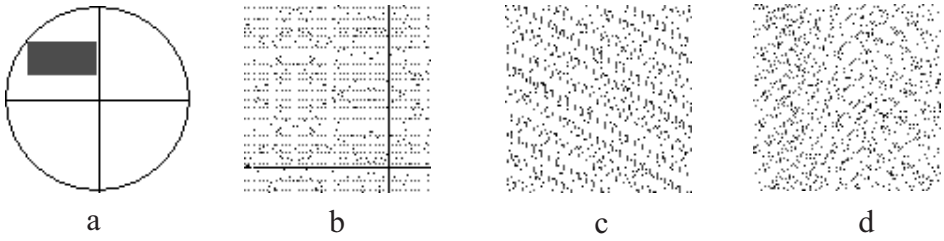


Figure 3: Application of the algorithms proposed in Section 4: a – initial image; b – main algorithm; c – second modification; d – first modification

The computation time of our algorithms is compared in Table 1. Computations were carried out by Pentium M processor 1.70 GHz with 1 Gb RAM. As the test image we took the 512x512-pixel 8 bit grayscale picture Lenna (see Figs. 4 and 5). But the second modification was tested on 200x200-pixel 8 bit grayscale picture Lenna (Fig. 5). Note, that the program does not cut the images into blocks for processing.

Table 1. Program working time.

| | Main algorithm | Modification 1 | Modification 2 |
|---|---|---|---|
| time | 3 seconds | 6 seconds | 192 seconds |

Program realizing these algorithms can be freely downloaded from *http://132.204.90.210/maryna/* and listing of the main coding algorithm is adduced in Appendix A.

## 5 Conclusions

In the paper one of the simplest versions of the quasicrystal-based symmetric cipher is described. Even in this form it appears adequate for many applications. Its complicated key is described

below. It is based on strongly aperiodic sets, includes everywhere discontinuous 'star map', and it is absolutely stable, since it operates with integer numbers only. Similarly, damaging a few data points does not affect the rest of the encryption/decryption process.

Furthermore coordinates of the data can be divided into several non-overlapping intervals, then contents of all intervals could be encrypted in parallel.

### 5.1 Cryptovariables

Application of the algorithm requires that the following variables (cryptovariables of the private key) are fixed

$$\tau\_\tau'\_a\_b\_c\_d\_sml\_X\_\varepsilon\_I, \quad \text{where}$$

$\tau$ and $\tau'$ are irrational numbers, solutions of one of equations of the two infinite series of the quadratic equations $x^2 = mx + 1$, $m = 1, 2, \ldots$ or $x^2 = mx - 1$, $m = 3, 4, \ldots$;

$a$ and $b$ are integers, fixing a seed point of the quasicrystal;

$c$ and $d$ are arbitrary real numbers, fixing the position of the acceptance window $\Omega$;

$X$ takes values {'+', '-'}, defining direction of quasicrystal construction relative to the seed point;

$sml$ sequence of distances of adjacent points during quasicrystal generation;

$\varepsilon$ is the maximal distance from the end points of $\Omega$ at which the star map points are still considered;

$I$ is the positive integer number specifying number of iterations of the certain quasicrystal cipher.

More elaborate encryption system can be built by exploiting properties of 1 or 2-dimensional quasicrystals. Using the present algorithm in several iterations, one may choose new quasicrystal for each iteration, or new position of the quasicrystal fragment, or different size, position of the acceptance window, etc.

Rich scaling properties of the quasicrystal could be involved into encryption scheme [3].

Genuinely two-dimensional quasicrystals can be used for the encryption, some examples of the possibilities are found in [12]. The number of distinct quasicrystals of this kind is unlimited.

One can easily build one-dimensional quasicrystal by projecting cuts of n-dimensional latices on the direction of the quasicrystal.

Intriguing appears to be the possibility to read the initial lattice data as an ordered stream of points and map them into single one-dimensional quasicrystal, perform the star map and reorder transform points into encrypted data. There are also other applications of the quasicrystals e.g. for generation of random numbers [6], even some cryptographic ones [11].

## Acknowledgment

One of us (M.N.) is grateful for the hospitality extended to her at the Center de recherche mathématique, Université de Montréal during the work on this project.

We are grateful for partial support of the work by the Natural Science and Engineering Research Council of Canada, MITACS, Lockheet Martin of Canada and MIND Research Institute of California.

# References


[1] L. Chen, R. Moody and J. Patera, Non-crystallographic root systems. Quasicrystals and discrete geometry (Toronto, ON, 1995), 135–178, *Fields Inst. Monogr.*, **10**, Amer. Math. Soc., Providence, RI, 1998.

[2] W. Diffie and M. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, **IT-22** (1976), 644–654.

[3] J.P. Gazeau, Z. Masakova and E. Pelantova, Nested quasicrystalline discretisations of the line, *Physics and number theory* **10** (2006), 79–131.

[4] O. Goldreich, Foundations of Cryptography, V.1: Basic Tools, Cambridge University Press, 2001.

[5] L.S. Guimond, Z. Masakova and E. Pelantova, Combinatorial properties of infinite words associated with cut-and-project sequences, *J. Theor. Nombres Bordeaux* **15** (2003), 697–725.

[6] L.S. Guimond, Jan Patera and Jiri Patera, Statistics and implementation of aperiodic pseudorandom number generators, *Applied Numerical Mathematics* **46** (2003), 295–318.

[7] Z. Masakova, J. Patera and E. Pelantova, Inflation centres of the cut and project quasicrystals, *J. Phys. A* **31** (1998), 1443–1453.

[8] Z. Masakova, J. Patera and E. Pelantova, Minimal distances in quasicrystals, *J. Phys. A* **31** (1998), 1539–1552.

[9] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 2001.

[10] R. Moody and J. Patera, Quasicrystals and icosians, *J. Phys. A* **26** (1993), 2829–2853.

[11] Z. Masáková, J. Patera, E. Pelantová, Aperiodic Encryption Method, US patent, September 14, 2004, US 6,792,108 B1

[12] J. Patera and M. Nesterenko, Quasicrystals in cryptography, *E. Kranakis, E. Haroutunian and E. Shahbazian, editors, Aspects of Network and Information Security, Proceedings of NATO Advanced Studies Institute on Network Security and Intrusion Detection, held in Nork, Yerevan, Armenia, October 01-12*, 2006, IOS Press, 2007, to appear.


# A Program listing

Realization of the main algorithm on Delphi 6.0:

```
unit Unit1;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls;
type
   TNode = Record
      a       : integer;
      b       : integer;
      rs, rb  : real;
      ns, nb  : integer;
   end;
   TSequence = Record
      n       : integer;
      Nodes   : array of TNode;
      a,b     : integer;
      s       : string;
   end;
   TForm1 = class(TForm)
   Button1         : TButton;
   Button2: TButton;
   procedure Button1Click(Sender: TObject);
```

```
   procedure FormCreate(Sender: TObject);
   procedure Button2Click(Sender: TObject);
 private
    Private declarations
 public
    Public declarations
 end;
var
 Form1         : TForm1;
 RootS, RootB  : real;
implementation
{$R *.dfm}
function CalcSmall (a, b: real) : real;
begin
   CalcSmall := a + b * RootS;
end;
function CalcBig (a, b: real) : real;
begin
   a := a + 1;
   b := - b;
   CalcBig := CalcSmall (a, b);
   CalcBig := a + b * RootB;
end;
function CheckShort (a, b: integer; var Node : Tnode; dir: integer) : boolean;
var tempB : real;
   res : boolean;
begin
   a := a - dir;
   b := b + dir;
   tempB := CalcBig (a,b);
   res := ((tempB < 0) and (tempB < 2));
   If res then begin
       node.a := a;
       node.b := b;
       node.rb := tempB;
   end;
   CheckShort := res;
end; //CheckShort (a, b: integer; var node : Tnode; dir: integer) : boolean;
function CheckMedium (a, b: integer; var node : Tnode; dir: integer) : boolean;
var tempB : real;
   res : boolean;
begin
   a := a + dir;
   tempB := CalcBig (a,b);
   res := ((tempB > 0) and (tempB < 2));
   If res then begin
       node.a := a;
       node.b := b;
       node.rb := tempB;
   end;
```

```
  CheckMedium := res;
end; //CheckMedium (a, b: integer; var node : Tnode; dir: integer) : boolean;
function CheckLong (a, b: integer; var node : Tnode; dir: integer) : boolean;
var tempB : real;
  res : boolean;
begin
  b := b + dir;
  tempB := CalcBig (a,b);
  res := ((tempB > 0) and (tempB < 2));
  If res then begin
     node.a := a;
     node.b := b;
     node.rb := tempB;
  end;
  CheckLong := res;
end; //CheckLong (a, b: integer; var node : Tnode; dir: integer) : boolean;
function FillSequence (var seq : TSequence; a, b, n, dir: integer): string;
var res            : string;
  aPrev, bPrev    : integer;
  i               : integer;
begin
  res := '';
  seq.n := n;
  seq.a := a;
  seq.b := b;
  SetLength (seq.Nodes, n);
  seq.Nodes[0].a := a;
  seq.Nodes[0].b := b;
  seq.Nodes[0].rb := CalcBig (a, b);
  seq.Nodes[0].rs := CalcSmall (a, b);
  seq.Nodes[0].ns := 0;
  for i := 1 to n - 1 do begin
     aPrev := seq.Nodes [i-1].a;
     bPrev := seq.Nodes [i-1].b;
     seq.Nodes [i].ns := i;
     if dir = 1 then begin
        if CheckShort (aPrev, bPrev, seq.Nodes [i], dir) then
           res := res + 's'
        else if CheckMedium (aPrev, bPrev, seq.Nodes [i], dir) then
           res := res + 'm'
        else if CheckLong (aPrev, bPrev, seq.Nodes [i], dir) then
           res := res + 'l'
     end else if CheckLong (aPrev, bPrev, seq.Nodes [i], dir) then
           res := res + 'l'
        else if CheckMedium (aPrev, bPrev, seq.Nodes [i], dir) then
           res := res + 'm'
        else if CheckShort (aPrev, bPrev, seq.Nodes [i], dir) then
           res := res + 's';
     seq.Nodes[i].rs := CalcSmall (seq.Nodes[i].a, seq.Nodes[i].b);
  end;
```

```
    seq.s := res;
    FillSequence := res;
  end; //FillSequence (var seq : TSequence; a, b, n, dir: integer): string;
  Procedure SaveSequence (seq : TSequence; fname : string);
  var f : TextFile;
  var i : integer;
  begin
    AssignFile (f, fname);
    rewrite (f);
    writeln (f, seq.n,' : ', seq.s);
    for i:= 0 to seq.n - 1 do
        writeln (f, seq.Nodes[i].ns, ', ', seq.Nodes[i].nb, ', ', seq.Nodes[i].a, ', ', seq.Nodes[i].b, ',
', seq.Nodes[i].rs, ', ',  seq.Nodes[i].rb);
    Flush (f);
    CloseFile (f);
  end; //SaveSequence (seq : TSequence; fname : string);
  Procedure SortSequence (var seq : TSequence; dir: integer);
  var node    : TNode;
    i,j     : integer;
    flag    : integer;
  begin
    for i := seq.n-2 downto 0 do
      for j:= 0 to i do begin
        flag := 0;
        if dir = 1 then begin
            if seq.Nodes[j].rb > seq.Nodes[j + 1].rb then
                flag := 1;
        end else
            if seq.Nodes[j].rs > seq.Nodes[j + 1].rs then
                flag := 1;
        if flag = 1 then begin
            node := seq.Nodes[j + 1];
            seq.Nodes[j + 1] := seq.Nodes[j];
            seq.Nodes[j] := node;
        end;
      end;
    if dir = 1 then
        for i := 0 to seq.n do
          seq.Nodes[i].nb := i;
  end; //SorteSequence (var seq : TSequence; dir: integer)
  Procedure CodeImg (dir: integer);
  var fname1, fname2      : string;
    Bitmap1, Bitmap2    : TBitmap;
    SeqX, SeqY          : TSequence;
    i,j                 : integer;
    CurWidth, CurHeight : integer;
    strX, strY          : string;
  begin
    if dir = 1 then begin
        fname1 := 'in.bmp';
```

```
        fname2 := 'out.bmp';
      end else begin
        fname1 := 'out.bmp';
        fname2 := 'back.bmp';
      end;
      Bitmap1 := TBitmap.Create;
      Bitmap1.LoadFromFile (fname1);
      Bitmap2 := TBitmap.Create;
      CurWidth := Bitmap1.Width;
      CurHeight := Bitmap1.Height;
      Bitmap2.Width := CurWidth;
      Bitmap2.Height := CurHeight;
      StrX := FillSequence (SeqX, 1, 0, CurWidth, 1);
      StrY := FillSequence (SeqY, 1, 0, CurHeight, 1);
      SortSequence (SeqX, 1);
      SortSequence (SeqY, 1);
      for i:= 0 to CurWidth - 1 do
        for j:= 0 to CurHeight - 1 do
          if dir = 1 then
              Bitmap2.Canvas.Pixels [i,j] := Bitmap1.Canvas.Pixels [SeqX.Nodes [i].ns,
SeqY.Nodes [j].ns]
          else
            Bitmap2.Canvas.Pixels [SeqX.Nodes [i].ns,SeqY.Nodes [j].ns] := Bitmap1.Canvas.Pixels
[i,j];
      Bitmap2.SaveToFile (fname2);
      Bitmap1.Free;
      Bitmap2.Free;
      if dir = 1 then begin
        SortSequence (SeqX, -1);
        SortSequence (SeqY, -1);
        SaveSequence (SeqX, 'x.txt');
        SaveSequence (SeqY, 'y.txt');
      end;
    end; //CodeImg (dir: integer)
    procedure TForm1.Button1Click(Sender: TObject);
    begin
      Button1.Enabled := False;
      Button2.Enabled := False;
      CodeImg (1);
      Button1.Enabled := True;
      Button2.Enabled := True;
    end;
    procedure TForm1.FormCreate(Sender: TObject);
    begin
      RootS := (1 + sqrt (5)) / 2;
      RootB := (1 - sqrt (5)) / 2;
    end;
    procedure TForm1.Button2Click(Sender: TObject);
    begin
      Button1.Enabled := False;
```

```
    Button2.Enabled := False;
    CodeImg (-1);
    Button1.Enabled := True;
    Button2.Enabled := True;
  end;
  end.
```

# B Application examples

In this section we will show several examples of the application of the main algorithm, its modification 1, and modification 2, as well as its iterations several times. Let us emphasize that the data was encrypted as a single block. Computing time is ranging between two seconds (one iteration) and nine seconds (five iterations) for the main algorithm.

Figure 4: Main algorithm coding without iterations

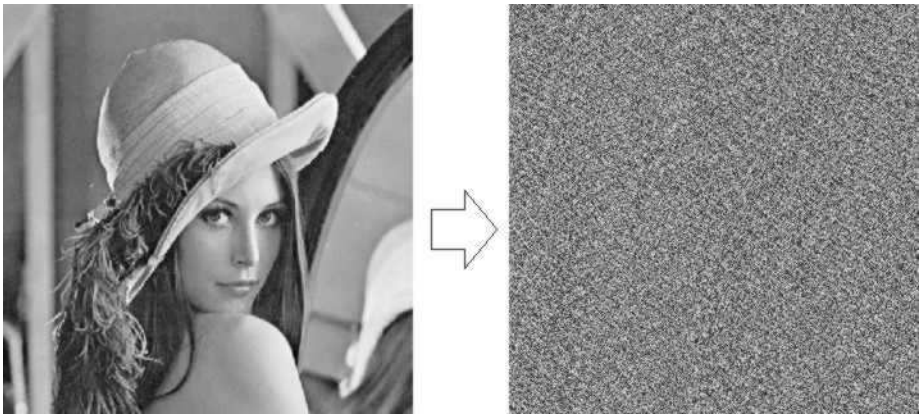

Figure 5: Modification 1 of the algorithm without iterations

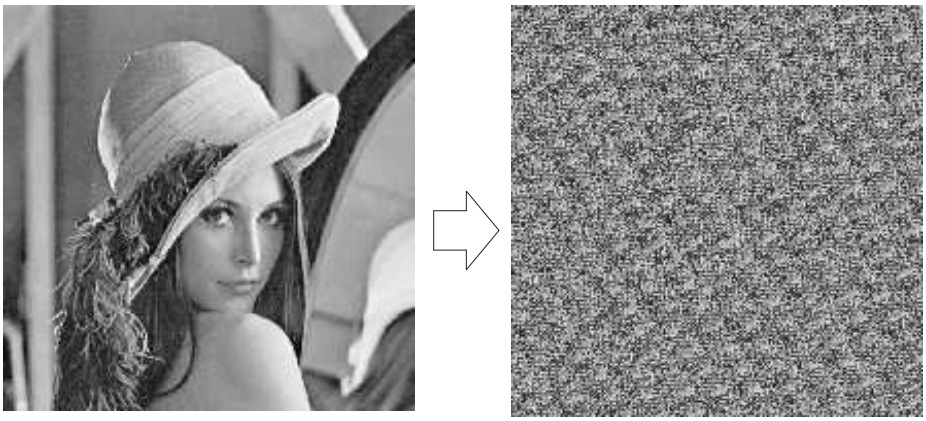

Figure 6: Modification 2 of the algorithm without iterations

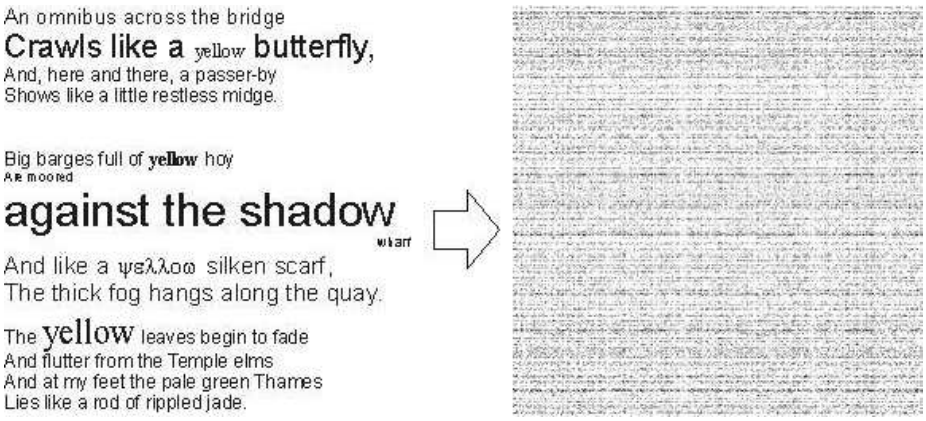


Figure 7: Main algorithm coding without iterations

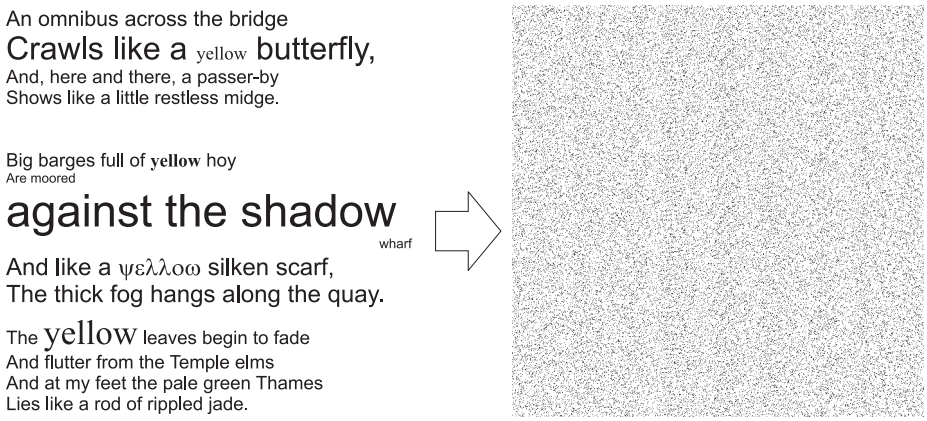


Figure 8: Modification 1 of the algorithm without iterations