PROPOSALS OF GRAPH BASED CIPHERS, THEORY AND IMPLEMENTATIONS

Andrzej Paszkiewicz
Institute of Telecommunications, Technical University
Nowowiejska 15/19, 00-665 Warsaw, Poland
E-mail: anpa@tele.pw.edu.pl
Anna Górska, Karol Górski,
ENIGMA Systemy Ochrony Informacji Sp.z.o.o.
Cietrzewia 8, 02-492 Warsaw, Poland
E-mail: ania, karol@enigma.com.pl
Zbigniew Kotulski, Kamil Kulesza, Janusz Szczepański
Institute of Fundamental Technological Research, Polish Academy of Sciences
Świętokrzyska 21, 00-049 Warsaw, Poland
E-mail: zkotulsk, kkulesza, jszczepa@ippt.gov.pl

ABSTRACT

Graphs may be used for the design of stream ciphers, block ciphers or public-key ciphers. This paper presents a method of using paths between a pair of graph vertice for designing effective polyalphabetic substitution ciphers. A similar method can be developed basing on cut-sets or spanning trees. The period of alphabet changeovers is equal to the number of paths between a pair of selected vertices on the graph, the number of spanning trees, cut-sets or is a multiple of these values.
The results of statistical tests and an assessment of the cryptographic strength of these ciphers are presented. Also proposed are modifications of these ciphers based on modifying the labels of vertices or edges (arcs) of the graph. These modifications influence the statistical properties and period lengths.

1. INTRODUCTION

A significant demand exists for new non-standard cryptographic methods (see, e.g., [1], [2]). In addition to traditionally employed for this purpose branches of mathematics such as algebra, Boolean function theory, information and coding theory [3] and classical probability theory, other branches, particularly discrete mathematics and chaos theory [4] may also be used. In discrete mathematics, graph theory appears to be a good source of non-standard methods. Graph theory provides examples of problems characterized by large computational complexity [5]. Problems such as the search for the shortest (weighted) Hamiltonian cycle are known to be NP-complete. Also, the number of various structures which may be created on a given graph, e.g. the number of paths or the number of cut sets between a pair of graph vertices or the number of spanning trees grows much faster than exponentially [6]. This may form a good foundation for the design of efficient and secure ciphers, more secure than the ciphers proposed in the past.

Graphs may be used for the design of stream ciphers, block ciphers or public-key ciphers [7]. This paper presents a method of using paths between a pair of graph vertices and spanning trees for designing effective polyalphabetic substitution ciphers. The period of alphabet changeovers is equal to the number of paths between a pair of selected vertices on the graph, the number of spanning trees or is a multiple of these values.

The results of statistical tests and an assessment of the cryptographic strength of these ciphers are presented. Also proposed are modifications of these ciphers based on modifying the labels of vertices or edges (arcs) of the graph. These modifications influence the statistical properties and period lengths.

The statistical tests carried out by the authors show that promising results may be obtained already for relatively small graph sizes, e.g., 24 vertices. This translates to a

large efficiency of ciphers designed on the basis of graph theory.

Graphs can be also used for other cryptographic algorithms like: secret sharing, and authentication of streams of bits. In the paper we present an idea of the graph-based secret sharing scheme. We indicate the possibility of application of the graphs to some simple authentication method which makes possible to identify the origin and to confirm contents of the message.

Before we present the examples of the graph-based algorithms applied to all the listed cryptographic tools, we introduce the fundamental facts concerning graphs.

2. MATHEMATICAL NOTATION

As we mentioned, graphs, as mathematically complicated objects, can be good candidates for coding cryptographic elements, especially secret keys. However, to apply a certain graph for cryptographic purpose we must know if it is connective and its structure is rich enough to guarantee security of the graph-based cryptosystem. In this section we present some theoretical results, that later are used for in construction of algorithms. The goal is to make them sufficiently general in order to provide theoretical foundation for various cryptological applications. Hence once theorems and proofs are stated, remarks are provided showing options for possible modifications.

We start the presentation of results from introduction of basic notations. Let *G* be the graph with *e* edges and *v* vertices. Then we have:

X is the set of vertices of the graph *G*,

$X = \{x_1, x_2, ..., x_v\}$;

U is the set of edges of the graph *G*;

$U \subset X \times X$, $U = \{u_1, u_2, ..., u_e\}$;

$u(x, y)$ is the edge linking the vertices *x* and *y* and $\vec{u}(x, y)$ is the directed edge starting at *x* and ending at *y;*

x^S is a tagged vertex, called the initial one;

x^T is another tagged vertex, called the final one;

μ is the path, that is the sequence of vertices, such that every neighbors have an edge linking them (alternatively: a sequences of edges such that every neighbors have a common vertex);

Γ is a function mapping the set of vertices to the set of subsets of vertices,

$\Gamma : X \to 2^X$

in such a way, that

$\Gamma(x) = \{y \in X : u(x, y) \in U\}$,

$\Gamma(x)$ is the set of vertices following the vertex *x* (for the non-directed graphs: the set of neighbors of the vertex *x*);

Γ^{-1} is the function inverse to Γ;

$\Gamma^{-1}(x)$ is the set of vertices preceding the vertex *x* (for the non-directed graphs: $\Gamma^{-1}(x) = \Gamma(x)$);

$K(v)$ is the complete graph with *v* vertices;

$\deg(x)$ denotes the degree of the vertex *x* and

$DEG(G)$ is the degree of the graph *G* (the sum of degrees of all vertices in the graph *G*).

We introduce the vector *WAY* as a register of variable length, where we store the sequence of vertices of the path linking the initial vertex x^S and the final vertex x^T.

The conditions that should be satisfied by the graph applied in our cryptosystem are presented in the following series of statements.

Theorem 1

Graph *G* with *v* vertices is connective if

$$e \geq \frac{(v-1)(v-2)}{2} + 1$$

Outline of proof:

One shows for the graph *G* with *v* vertices:

a. The existence of a non-connective graph with $e = \frac{(v-1)(v-2)}{2}$ edges;
b. The property that graph with $e = \frac{(v-1)(v-2)}{2} + 1$ edges cannot be partitioned into disjoined (non-connective) subgraphs.

Corollary 1.1

Maximal edges difference between $K(v)$ and the graph satisfying conditions of Theorem 1 is $v - 2$

Corollary 1.2

In the graph G, $e \geq \frac{(v-1)(v-2)}{2}+1$ implies $v^2-3v+4 \leq DEG(G)$

Theorem 2

Graph G, such that $v^2-3v+4 \leq DEG(G)$, will have minimum $v-2$ vertices of degree at least $v-3$.

Corollary 2.1

The graph G, such that $v^2-3v+4 \leq DEG(G)$, will have maximum 2 vertices of degree lower than $v-3$.

Corollary 2.2

In the graph G satisfying the condition $v^2-3v+4 \leq DEG(G)$, such that, every vertex x of $\deg(x) \geq v-3$ will be connected with minimum $v-5$ vertices of $\deg(x) \geq v-3$.

Theorem 3

In any graph maximal number of vertices of $\deg = v-1$ is limited to the lowest degree of the vertex in the given graph.

Corollary 2.3

Graph G with v vertices and $v^2-3v+4 \leq DEG(G)$ can have only one vertex of degree 1. Moreover:

a. Such a vertex can be connected only with the vertex of degree $v-1$
b. Subgraph G' created by removing the vertex of degree 1 from graph G will be complete.

Outline of proof:

Corollary 2.3 results from Theorem 3, some combinatorial reasoning and a simple computation.

Lemma 1

In the graph G, such that $v^2-3v+4 \leq DEG(G)$, any two vertices $x_i, i=1,2$ of degree $\deg(x_i) \geq v-3$ can be connected by a path of the length of 2 edges.

Outline of proof:

One should consider three cases, where the graph G has:

a. v vertices, $x_i, i=1,2,...,v$, with $\deg(x_i) \geq v-3$,
b. one vertex x_1 with $\deg(x_1) < v-3$ and $v-1$ vertices, $x_i, i=2,3,...,v$, with $\deg(x_i) \geq v-3$,
c. two vertices, x_1, x_2 of $\deg(x_1), \deg(x_2) < v-3$ and $v-2$ vertices, $x_i, i=3,4,...,v$, with $\deg(x_i) \geq v-3$.

Each case should be proven separately using theorems stated above, combinatorial reasoning and simple computation.

Lemma 2

In the graph G such, that $v^2-3v+4 \leq DEG(G)$, any two vertices can be connected by the path of the length of 3 edges.

Outline of proof:

Assume, x^S and x^T are the starting and the ending vertex of the path. The proof should be made for three separate cases:

a. both vertices x^S and x^T have $\deg \geq v-3$,
b. one vertex is with $\deg < v-3$ another with $\deg \geq v-3$,
c. both vertices have $\deg < v-3$.

Each case should be proven separately using theorems stated above and lemma 1, combinatorial reasoning and simple computation.

Theorem 4

In the graph G with v vertices and satisfying the condition $v^2-3v+4 \leq DEG(G)$, the minimal number $N(x^S)$ of paths such that:

a. x^S is the starting point for the path;
b. n is the number of edges in the path and $2 < n < v-3$;
c. each edge is used only once in the path;
d. each vertex is used only once in the path

can be calculated from the formula

$$N(x^S) = (v-5)(v-6)...(v-5-(n-3)) =$$
$$(v-5)(v-6)...(v-2-n) =$$
$$= \prod_{c=0}^{n-3}(v-5-c) = \frac{(v-5)!}{(v-3-n)!}.$$

Outline of proof:

Because theorem states “minimal number of

paths", the worst case scenario (conditions for the lowest numbers of path) should be found. Then it should be proven using theorems stated above and combinatorial reasoning.

Remarks:

1. Theorem and proof were made for theoretical worst case scenario, careful analysis shows that such strict conditions are mutually exclusive, hence cannot hold both. Further reasoning yields:

$$N(x^S) = (v-3)\prod_{c=0}^{n-3}(v-5-c)$$

2. Eliminating condition *d* from Theorem 4 can significantly increase *N*. In such cases it is also possible to create path with length $n > v$ (*n* can even reach magnitude $\frac{v^2}{2}$), further increasing *N*.

Theorem 5

In the graph *G*, with *v* vertices and satisfying the condition $v^2 - 3v + 4 \le DEG(G)$, the minimal number $N(x^S, x^T, m)$ of paths such that:

a. x^S is the starting point and x^T is ending point of the path;
b. *m* is a natural number and the path length is a number form the interval $(m, m+2)$;
c. each edge is used only once in the path;
d. each vertex is used only once in the path, with the exception for last two vertices *x'* and *x''* preceding x^T, *x'* and *x''* can come from vertices previously visited,

can be calculated according to

$$N(x^S, x^T, m) = (v-6)(v-7)...(v-6-(n-2)) =$$
$$= (v-6)(v-7)...(v-4-n),$$

where $n = m-1$ and $2 \le n \le v-4$, provided that at least one such a path can be found, or

$$N(x^S, x^T, m) = 0$$

otherwise.

Outline of proof:

Proof is based on the assumption that path can be constructed. Then, it is shown that certain shorter paths can be constructed and the number of them is given by Theorem 4. Finally, Lemmas 1, 2 together with some previous results (especially, corollaries to Theorem 2) show that the shorter path can be linked with x^T in a fixed number of steps. Such composed paths will fulfill conditions of Theorem 5 and their minimal number can be provided.

Remark:

If some assumptions are lifted (say b. and/or d.) similar results can be found. In such cases it may be required to introduce additional assumptions about the path (specially its length).

3. GRAPH-BASED BLOCK CIPHER

The proposed algorithm uses paths between two vertices of a graph. To construct the series of paths we apply the table of preferences of visiting the vertices of the graph. For every vertex $x \in X$, the table of preferences is a list of numbers of the vertices $x_i \in X$ such that $u(x, x_i) \in U$ giving the sequence of visiting the points x_i. The cipher graph *G* must be defined in such a way, that it is connective and it has exactly 256 edges; this means that it has at least 24 vertices (We bear in mind the conclusion of Theorem 1). Next, we assign to all the edges $u_j, j = 1,2,...,256$, in a random way, the binary numbers of the range $0...255$, called the labels $k^j, j = 1,2,...,256$ (assuming that all the labels are different). To obtain the working path for the encryption the actual character, one performs an algorithm of finding paths between two vertices, x^S and x^T; the numbers of the vertices belonging to the generated path are written, in sequence, in the vector *WAY;* this vector is returned always after application of the procedure of searching the next path between x^S and x^T. Finally, $u_{i_r}, u_{i_{r-1}}, ..., u_{i_1}$ are the edges belonging to the path linking x^S and x^T, written in the inverse order.

To complete the algorithm of encryption the characters, we define a cryptographically strong bijective map (permutation) Π,

$$\Pi : \{0,1,...,255\} \to \{0,1,...,255\},$$

and the cyclic shift to the left with b bits (denoted by the symbol <<< b).

The input of the cipher is the sequence of 8-bit characters $z_1, z_2, ..., z_n$. The other parts of the cryptosystem immediate: the table of preferences, the set of labels k^j, $j = 1,2,...,256$, the bijective function Π, and the structure of the graph G can be considered as elements of the secret key. The paths needed for encryption of each character can be generated according to algorithms given in [8], [9], [10]. The action of the algorithm can be written symbolically in the following way.

Algorithm of encryption:

Input:

$z_1, z_2, ..., z_n$, the characters that should be encrypted;

x^S, the initial vertex of the path;

x^T, the final vertex of the path;

r, the length of the path beeing analyzed;

the table of preferences of the visited vertices.

For i := 1 **to** n **do**

begin

temp := z_i

Generate next path linking $\boldsymbol{x^S}$ and $\boldsymbol{x^T}$**;**

For j:=r **downto** 1 **do begin**

temp := temp XOR k^j

temp := temp <<< 1

temp := Π(temp)

End { For j }

w_i := temp;

Return encrypted character w_i;

End { For i}

Output:

$w_1, w_2, ..., w_n$, the encrypted characters.

The above algorithm of encryption is very simple. It has been implemented in PASCAL and C++. Both versions are very effective in time and the obtained ciphertexts satisfy the required statistical tests (see [11]).

4. GENERAL FORMULATION OF A SYMMETRIC CIPHER

The block cipher (alternatively: the pseudo-random function, the cryptographic primitive) can be written in the form of the following abstract two-argument map

$F(.,.): K \times P \to C$,

where

K is the secret key space,

$P = \{0,1\}^l$ is the plaintext (domain) space,

$C = \{0,1\}^L$ is the ciphertext space.

If we fix the secret key k, substitute it as the first argument of the function $F(.,.)$ and make the function transforming the sequence of plaintexts $p_1, p_2, ...$ (second arguments) to ciphertexts $c_1, c_2, ...$ (images of $F(.,.)$), then we have the block cipher constructed.

To apply the function $F(.,.)$ for random bits generation we must do three steps (all of them written symbolically) of the following algorithm.

Algorithm 1.

1) $k \xleftarrow{R} K$, the random choice of the secret key k,

2) $s \xleftarrow{R} P$, the random choice of an initial condition (seed) s,

3) $F_k(s) := F(k,s) \in C$, calculation of the sequence of bits as the value of the function F for chosen values k and s.

To obtain more bits (a long stream of bits) we should go through the following algorithm.

Algorithm 2.

1) generate the sequence of secret keys $k_1, k_2, k_3, ...$, each according to Step 1 of Algorithm 1,

2) generate the sequence of initial conditions $s_1, s_2, s_3 ...$, each according to Step 2 of Algorithm 1,

3) for every pair of arguments, k_i, s_i, calculate the bits values according to Step 3 of Algorithm 1,

4) construct the stream of bits G as:

$G = F_{k_1}(s_1) \| F_{k_2}(s_2) \| F_{k_3}(s_3) \| ....$,

where $\|$ denotes the concatenation of strings.

In a lot of widely used cryptosystems, the secret key space is the set of bit sequences, e.g., $K = \{0,1\}^k$. In this paper we propose to use, as the secret key space K, the set of graphs being subgraphs (with certain assumed properties) of our basic graph G. Moreover, for

simplicity of calculations in this presentation, we assume $l = L = 8$, so the other spaces used in the cryptosystem are: $P = \{0,1\}^8$, $C = \{0,1\}^8$.

The basic graph G of the cryptosystem has v vertices and e edges; each vertex and edge has its individual number. Every edge of the graph is associated with the label being some 8-bit number. Every vertex has its individual table of preferences which describes sequence of the neighbor vertices when the paths are constructed. The secret keys used in the cryptosystem are the subgraphs $k_i, i = 1,2,...,$ generated from the basic graph G (e.g., the paths of certain length starting from a fixed vertex x^S) according to some rule based on the tables of preferences associated to the vertices of the graph.

A certain subgraph k, being the secret key in our cryptosystem, can be interpreted as a sequence of words (bytes, elements of the space $\{0,1\}^8$, the labels associated to the edges of the subgraph) of random length r (the length is dependent on the way the path is generated) of the form
$k = \left(k^1, k^2, ..., k^r\right).$

Now, the cipher function $F(.,.)$ is the composition of r round operation $f(k^j,.)$, where in each round subkey (the label assigned to the graph edge) k^j of the key k is applied and the initial plaintext $s = s_1$ is iterated according to
$s_{j+1} = f(k^j, s_j), j = 1,2,...,r,$
and the output of $F(.,.)$ is $c = s_{r+1}$. Every round should be built according to usual conditions to guarantee good cryptographic properties of $F(.,.)$ (see, e.g., [12]). In the particular case of our cipher we applied the XOR operation of the subkey k^j and the plaintext s_j and permutation of bits in the resulting byte.

The general scheme presented in this section is now implemented for certain graph scheme.

5. STREAM CIPHER

To apply the algorithm of the graph cipher applied for bits generation one must do the following.

1) Construct the graph G with v vertices and e edges, $\frac{(v-1)(v-2)}{2} + 1 \le e \le \frac{v(v-1)}{2}$;
2) Assign the labels $k^p \in [0,255]$, $p = 1,2,...,n$, to the edges;
3) Assign the tables of preferences to the vertices of the graph;
4) Choose the seed $s \in [0,255]$;
5) Choose the way to generate the keys $k_i = (k_i^1, k_i^2, ..., k_i^r)$, e.g.,
   a) choose the initial vertex x_1^S and generate a long path starting from it, using the tables of preferences (e.g., the length of the path is 18),
   b) choose all the 8 element subsets of the edges from this path and place their numbers to rows of the table in a natural order (leaving the order of edges governed by the path unchanged),
   c) change the order the rows of the matrix in a random way using the pseudorandom numbers generator for the indexes of rows, obtaining the keys $k_i = \left(k_i^{j_1}, k_i^{j_2}, ..., k_i^{j_8}\right)$, $i = 1,2,...,i_{\max}$,
6) Generate the sequence of bits according to:
   $G(s) = F_{k_1}(s) \,||\, F_{k_2}(s+1) \,||\, F_{k_3}(s+2) \,||\, ...$
   $.... \,||\, F_{k_{i_{\max}}}(s + i_{\max} - 1),$
   where addition of the function's argument is modulo 2^8.
7) Select next path and repeat the procedure of points 5b, 5c, 6. After exploiting all the paths starting from the vertex x_1^S, choose other initial vertices x_i^S, $i = 2,3,...,v$ and continue the procedure of points 5, 6.

The final step of the procedure is statistical verification of the generated stream of bits.

6. PUBLIC-KEY CRYPTOSYSTEM

The scheme for public-key cryptosystem called "Polly Cracker" can be found in [7]. The general idea behind this scheme is to:

a. construct polynomials over finite field F;
b. take an arbitrary vector $z \in F^n$ as a private

key and the subset $B=\{q_i\}$ of the polynomials over finite field *F*, such that, for every *i*, $q_i(z)=0$, as a public-key;

c. encrypt a message *m* obtaining cipher polynomial *C* using the public-key (a randomly chosen element generated by *B*);
d. message *m* can be decrypted by finding the value of polynomial *C* at *z*.

Having described public-key cryptosystem "Polly Cracker", one can move to its special case based on graph 3-coloring. The problem of graph 3-coloring is NP class. To formulate the cryptosystem in terms of graph theory, we introduce:
The public-key is the graph $G(X,U)$, that is the graph with the set of vertices *X* and the set of edges *U*;
The private key is the proper 3-coloring of the graph using colors $c\in\{1,2,3\}$ and the map assigning $x\mapsto c$ for $x\in X$, according to graph 3-coloring rule.

Once graph 3-coloring is known, the base $B=B(G)$ is constructed. *B* is constructed from a polynomial derived from the variables $\{t_{x,c}\}$, and $B=B_1\cup B_2\cup B_3$ for

$$B_1=\{t_{x,1}+t_{x,2}+t_{x,3}-1 : x\in X\}$$
$$B_2=\{t_{x,c}t_{x,d} : x\in X, c\le d\in\{1,2,3\}\}$$
$$B_3=\{t_{x,c}t_{y,c} : u(x,y)\in U\}$$

Then, the zero point of all polynomials from *B* can be computed by taking $t_{x,c}=1$, if the vertex *x* has color *c,* and 0 otherwise.

Further references to this public key cryptosystem will be given in the secret sharing section.

In a similar way other graph based "Polly Cracker" schemes can be constructed. One of the examples can be "perfect code" graph described in [7].

All these implementations, like described above graph 3-coloring system, have the following features:

a. knowing $G(X,U)$ is equivalent to knowing subset $B=\{q_i\}$ of polynomials over finite field *F*;
b. knowing NP class problem (resulting from graph structure) is equivalent to knowing vector $z\in F^n$;
c. encryption process takes place like in general "Polly Cracker" description;
d. to decrypt message *m*, value of received polynomial (derived from the graph $G(X,U)$ structure) at *z* should be calculated.

7. SECRET SHARING

Graphs have applications to secret sharing on two levels:

a. theoretical research into secret sharing problems;
b. as structures used for secret sharing schemes.

Concerning the first application, theoretical results will be presented, the most of them coming from [13]. Let us define the scheme of the method. Thus, let:
P denotes the set of *w* participants of the secret, where *w* is an integer;
K denotes the shared key;
Γ is the access structure, that is, the set of subsets of *P*. The elements of Γ are those subsets of participants that should be able to compute the key. Such subsets in Γ are called authorized subsets;
S is the set of the shares created by the key distribution rule (function) $f: P\to S$.

A perfect secret sharing scheme realizing the access structure Γ is a method of sharing the secret key *K* among the participants from *P* in such a way that the following two properties are satisfied :

1. If an authorized subset of participants $B\subseteq P$ pool their shares, then they can determine the value of *K*.
2. If an unauthorized subset of participants $B\subseteq P$ pool their shares, then they can determine nothing about the value of *K*.

In a perfect secret sharing scheme realizing an access structure Γ, the information rate for P_i is the ratio

$$q_i=\frac{\log_2|K|}{\log_2|S(P_i)|},$$

where $S(P_i)$ denotes the set of possible shares that P_i might receive and $S(P_i)\subseteq S$.

The information rate of the scheme is

denoted by q and is defined as $q = \min\{q_i : 1 \le i \le w\}$, where w is the total number of participants of the secret as defined above. Optimal perfect sharing scheme has $q = 1$. Such a scheme is called the ideal one.

Once basic terms are established some theoretical results and methods can be presented :

Graphs can be used to visualize and ease design of access structures. For example, monotone circuit construction method is used first to build a monotone circuit that "recognizes" the access structure, and then to build the secret sharing scheme from the description of the circuit.

Yet, more general theoretical result can be stated. Suppose, $G(X,U)$ is a complete multipartite graph. Then, there is an ideal scheme realizing the access structure $\mathrm{Cl}(U)$ on the participant set *X*. ($\mathrm{Cl}(U)$, the closure of *U*, means that all edges of the graph are used in the access structure). This short and elegant result, although at first looks like another "existence" theorem, is indeed powerful tool in construction access structures for sharing schemes.

After describing theoretical component some more practical results will be presented. Graph *n*-coloring finds applications in many fields of cryptography. Two examples are :

a. public-key cryptosystem "Polly Cracker" bases on graph 3-coloring;
b. zero-knowledge proof based on graph 3-colorability.

Two approaches to secret sharing for graph *n*-coloring will be presented. Due to the lack of space only rough sketch of idea behind solutions will be shown. The volume of information required for detailed description would easily fill whole paper.

1. The participants do not know underlying graph structure $G(X,U)$.

   In this case *G* is extended by some new edges and possibly some vertices. Then, graph divided into *w* pieces (number of participants) according to wanted access structure. Depending on the boundary conditions (*n*, *w*, access structure, and graph extension algorithm) decent results can be presented. They are usually of combinatorial nature, hence allowing good "bottom" estimates of the structure strength. At this moment we are working on their generalization to one elegant model.
2. The participants know underlying graph structure $G(X,U)$.

   The example of such a problem is the public-key cryptosystem "Polly Cracker" based on graph 3-coloring, since $G(X,U)$ is a public-key in this case. Hence, method used above (adding edges, vertices and sharing) cannot be applied.

   Even in such case guidelines for efficient secret sharing can be found. They heavily depend on the structure of *G*, but what may come as a surprise: secret can be shared in an efficient way among number of participants $w > n$ (the number of colors). Certainly, such secret sharing scheme will not be perfect but, yet, can be made of NP computational complexity for special cases. Yet, again authors are busy working on their generalization to one elegant model.

Other results concerning secret sharing can be found in [14], [15]. There are also some results concerning zero-knowledge proofs, the protocols based on graph isomorphism and graph 3-colorability.

8. AUTHENTICATION OF DIGITAL STREAMS

In the paper [16] authors presented a graph-based scheme of authentication of digitals streams over a lossy networks. In their model the data stream being authenticated is represented as a contiguous subset of packets $\{P_1, P_2, ..., P_n\}$. For description of the authentication procedure some directed graph without loops and with *n* nodes (vertices) is used. Every node of the graph corresponds to the data packet of the same number. One packet in the stream, say P_{sig}, is signed with a public-key signature algorithm such as RSA. The directed edges of the graph, $\vec{u}(x_i, x_j)$, connecting the *i*-th and *j*-th node inform that the hash function of the packet P_i is placed in

the packet P_j if $i > j$, and some Message Authentication Code (MAC) of P_i is placed in P_j if $i < j$. If both the contents and source of P_j can be authenticated then also the receiver is capable of verifying the contents and source of P_i. Any packet P_i can be authenticated if and only if there is a path from P_i to the signature packet P_{sig} that only includes nodes corresponding to received (that is, not lost during transmission) packets. Therefore one should construct the authentication graph that (in some reasonable frames) it maximizes the probability $\Pr(P_i \to P_{sig})$ of linking P_i and P_{sig} by the path. Examples of such schemes are presented in the paper [16].

9. NUMERICAL RESULTS

The encryption algorithm was based on a directed graph with 24 vertices and 256 arcs. Outgoing arcs from each vertex were randomly assigned preference levels used during the creation of paths through the graph. Each arc also had a label selected randomly from the set {0..255} without repetitions. Additionally the encryption algorithm used a random single cycle permutation.

Using depth-first search successive 18-arc paths were selected, each starting from a fixed vertex. For each path all 8 element combinations of the 18 arcs were generated in lexicographic order. Each such combination was used to encrypt a single plaintext character. For the purpose of the statistical tests the plaintext characters took on successive values from the set {0..256} repeated as many times as needed.

The encryption of a single character was performed in 8 stages with each stage consisting of a bitwise sum modulo 2 of the plaintext character and the label of an arc from the current 8-element combination and then the application of the single cycle permutation to the result. The result of each stage became the input to the next stage. At each stage the next arc from the current combination was used. The result of the last stage became the ciphertext character.

The tests were carried out on:
100 sequences of 10 Mbits
10 sequences of 100 Mbits
1 sequence of 1 Gbit.

The tests performed and their results are shown in Table 1. Each entry consists of the number of tests which resulted in the rejection of the null hypothesis in relation to the total number of tests. All tests were performed with a significance level of 0.01.

BIBLIOGRAPHY

[1] http://csrc.nist.gov/encryption/aes/

[2] http://www.cosic.esat.kuleuven.ac.be/nessie/

[3] J.Berstel, D.Perrin, Theory of Codes, Academic Press 1985.

[4] Z.Kotulski, J.Szczepański, K.Górski, A. Paszkiewicz, A. Zugaj, Application of discrete chaotic dynamical systems in cryptography - DCC method; International .Journal of Bifurcation and Chaos. 9(6), 1121-1135, 1999.

[5] K.C.Kakoulis, I.G. Tollis, On the complexity of the Edge Label Placement problem, Computational Geometry Theory and Applications 18(1), 1-17, Feb 2001.

[6] R.Wilson, Introduction to Graph Theory, Addison Wesley, London 1996.

[7] N.Koblitz, Algebraic Aspects of Cryptography, Springer-Verlag, Berlin 1998.

[8] L.Fratta, U.G.Montanari, All simple paths in a graph by solving a system of linear equations, Nota Interna No. B71-11 of Consiglio Nalzonalk dells Riscerche, Institut di Elaborazione Della Informazione, Pisa 1971, Oct;

[9] G.S.Hura, Enumeration of all simple paths in a directed graph using Petri Net - a systematic approach, Microel and Reliab. Vol. 23, No. 1 pp. 157-159, 1983;

[10] R.B.Misra and K.B.Misra, Enumeration of all simple paths in a communication network, Microel. And Reliab. Vol. 20, pp. 419-426, 1980;

[11] A.Paszkiewicz, Przykłady zastosowań teorii grafów do konstrukcji szyfrów, IV Krajowa Konferencja Zastosowań Kryptografii Enigma'2000, K-179-183.

[12] J.Daemen, L.R.Knudsen, V.Rumen, Linear Frameworks for Block Ciphers, Designs, Codes and Cryptography 22, 65-87, 2001.

[13] D.R.Stinson, Cryptography. Theory and Practice. CRC Press, Boca Raton 1995.

[14] G.Simmons, Geometric Shares Secret and/or Shared Control Schemes, Adv. in Cryptology-Proc of CRYPTO'90, Springer Verl. 1991, pp. 216-241;

[15] G.Simmons, How to (really) share a secret, Adv. in Cryptology-Proc of CRYPTO'88, Springer Verl. 1990, pp. 390-448;

[16] S.Miner, J.Staddon, Graph-Based Authentication of Digital Streams, preprint

Table 1. The results of ststistical tests performed for the numerical data

Test	10 Mbit sequences	100 Mbit sequences	1 Gbit sequence
FIPS 140-1 test suite in 20000 bit subsequences	0 / 50000	1 / 50000	0 / 50000
equidistribution of 1 bit blocks in 16384 bit subsequences (χ^2 test)	633 / 61000	604 / 61030	605 / 61035
equidistribution of 1 bit blocks in whole sequence (χ^2 test)	2 / 100	0 / 10	0 / 1
equidistribution of 8 bit blocks in 131072 bit subsequences (χ^2 test)	75 / 7600	77 / 7620	84 / 7629
equidistribution of 8 bit blocks in whole sequence (χ^2 test)	1 / 100	0 / 10	0 / 1
equidistribution of 48 bit blocks in 786432 bit subsequences (Kolmogorov-Smirnov test)	15 / 1200	16 / 1270	12 / 1271
linear complexity in 1024 bit subsequences	9937 / 976500	-	-
randomness of Walsh-Hadamard power spectrum in 1024 bit subsequences of the initial 65536 bit subsequence (Feldman's test)	82 / 6400	-	-
randomness of Walsh-Hadamard power spectrum in the whole initial 65536 bit subsequence (Feldman's test)	0 / 100	-	-
entropy of 8 bit blocks in whole sequence (Maurer's test)	-	0 / 10	0 / 1
entropy of 9 bit blocks in whole sequence (Maurer's test)	-	1 / 10	0 / 1
entropy of 10 bit blocks in whole sequence (Maurer's test)	-	0 / 10	0 / 1
entropy of 11 bit blocks in whole sequence (Maurer's test)	-	0 / 10	0 / 1
entropy of 12 bit blocks in whole sequence (Maurer's test)	-	0 / 10	0 / 1
entropy of 13 bit blocks in whole sequence (Maurer's test)	-	-	0 / 1
entropy of 14 bit blocks in whole sequence (Maurer's test)	-	-	0 / 1
entropy of 15 bit blocks in whole sequence (Maurer's test)	-	-	0 / 1