Acta Cybernetica **22** (2015) 359–371.

A Novel Cryptosystem Based on Gluškov Product of Automata[*][†]

Pál Dömösi[‡] and Géza Horváth[§]

Abstract

The concept of Gluškov product was introduced by V. M. Gluškov in 1961. It was intensively studied by several scientists (first of all, by Ferenc Gécseg and the automata-theory school centred around him in Szeged, Hungary) since the middle of 60's. In spite of the large number of excellent publications, no application of Gluškov-type products of automata in cryptography has arisen so far. This paper is the first attempt in this direction.

Keywords: cryptosystem, Gluškov product of automata

1 Dedication

This paper is dedicated to the memory of our late colleague, teacher and friend, Professor Ferenc Gécseg who has been a central figure in modern automata theory. He established the world famous research school of Szeged University in automata theory. His death is an irreplaceable loss for the whole research community of theoretical computer science.

2 Introduction

The connection of certain automata through various communication links leads to the notion of composition of automata [9]. A substantial body of literature in this important scientific field has been published by researchers belonging to the automata-theory school centred around Ferenc Gécseg in Szeged, Hungary [8, 9]. The specific concept of automaton also applied in cryptography, the cellular

[*]In memory of Professor Ferenc Gécseg

[†]The second author was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

[‡]Institute of Mathematics and Informatics, College of Nyíregyháza, H-4400 Nyíregyháza, Sóstói út 36, Hungary, E-mail: `domosi@nyf.hu`

[§]Faculty of Informatics, University of Debrecen, H-4028 Debrecen, Kassai út 26, Hungary, E-mail: `horvath.geza@inf.unideb.hu`

DOI: 10.14232/actacyb.22.2.2015.8

automaton, can also be regarded a special composition of automata, where the cells functioning as the members of the composition are composed of one and the same type of elementary automata, and the pattern of the communication links and connections between these elementary automata is a simple network. Despite the large number of publications on compositions of automata (authored predominantly by Hungarian researchers), no cryptographic applications of the results have been disclosed so far.

Several cryptosystems have been designed on the basis of abstract automata. Some of them are based on Mealy automata or their generalization (see, for exaple [1, 14, 20, 21]), some of them are based on cellular automata (see, for example [12, 13, 16, 23]), while [6] is based on automata without outputs . The best-known abstract automata based cryptosystems all share the common problem of serious realization difficulties: some systems are easy to defeat [2, 3, 4, 17, 19, 22], the technical realization of others result in slow performance [6, 7, 12, 21], and still others exhibit difficulties in the choice of the key-automaton [5, 16]. These drawbacks justify the need of novel cryptosystems overcoming these problems. By some experimental results we will show the security of the proposed system. (Serious security analysis should be necessary in the future work.) By an example we show that the technical realization of the novel system is not difficult. Moreover, we give a method to generate key automata easily.

A Gluškov product of automata [11] is loosely defined as a collection of automata that each of which changes its state at discrete time steps by a local transition function of the states and a global input. Moreover, the synchronous action of the local state transitions defines a global transition on the entire product. Thus a Gluškov product of automata is also an automaton. Usually it is assumed that the component automata are connected together according to a directed graph $\mathcal{D}$. The vertices of $\mathcal{D}$ are considered as automata and the edges indicate the existence of communication links. Thus $\mathcal{D}$ has no parallel edges.

An important observation of this paper is that, using the concept of Gluškov product, we can store certain properties of very large automata such that their transitions can be computed easily. By this observation, we can built new secure symmetric block ciphers based on Gluškov product of automata.

3 Preliminaries

We start with some standard concepts and notation. For all notions and notation not defined here we refer to the monographs [8, 9, 10, 15, 18]. A *word* (over Σ) is a finite sequence of elements of some nonempty and finite set Σ. We call the set Σ an *alphabet*, the elements of Σ *letters*. By the *free monoid* Σ^* *generated by* Σ we mean the set of all words (including the *empty word* λ) having catenation as multiplication. We set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$, where the subsemigroup Σ^+ of Σ^* is said to be the *free semigropu generated by* Σ. By an *automaton* we mean a deterministic finite automaton without outputs. In more details, an automaton is an algebraic structure $\mathcal{A} = (A, \Sigma, \delta)$ consisting of the nonempty and finite *state set* A, the nonempty and finite *input set* Σ, and a *transition function* $\delta : A \times \Sigma \to A$. The elements of the state set are the *states*, and the elements of the input set are the *input signals*. An element of A^+ is called a *state word* [1] and an element of Σ^* is called an input word. State and input words are also called *state strings* and *input strings,* respectively. If a state string $a_1 a_2 \cdots a_s$ $(a_1, \ldots, a_s \in A)$ has at least three elements, the states $a_2, a_3, \ldots, a_{s-1}$ are also called intermediate states. It is understood that δ is extended to $\delta^* : A \times \Sigma^* \to A^+$ with $\delta^*(a, \lambda) = a$, $\delta^*(a, xq) = \delta(a, x)\delta^*(\delta(a, x), q), a \in A, x \in \Sigma, q \in \Sigma^*$. In other words, $\delta^*(a, \lambda) = a$ and for every nonempty input word $x_1 x_2 \cdots x_s \in \Sigma^+$ (where $x_1, x_2, \ldots, x_s \in \Sigma$) there are $a_1, \ldots, a_s \in A$ with $\delta(a, x_1) = a_1, \delta(a_1, x_2) = a_2, \ldots, \delta(a_{s-1}, x_s) = a_s$ such that $\delta^*(a, x_1 \cdots x_s) = a_1 \cdots a_s$.

In the sequel, we will consider the transition of an automaton in this extended form and thus we will denote it by the same Greek letter δ. If b is the last letter of $\delta(a, w)$ for some $a, b \in A, w \in \Sigma^*$ then we say that w *takes* the automaton from its state a into state b, and we also say that the automaton *goes* from state a into state b under the effect of w. The automaton $\mathcal{B} = (B, Y, \delta_{\mathcal{B}})$ with $B \subseteq A$, $Y \subseteq \Sigma$ and $\delta_{\mathcal{B}}(a, x) = \delta(a, x), a \in B, x \in Y$ is a *subautomaton* of $\mathcal{A}$. In particular, if $B \subseteq A$ and $Y = \Sigma$ then $\mathcal{B}$ is a state-subautomaton of $\mathcal{A}$. Moreover, if $B = A$ and $Y \subseteq \Sigma$ then $\mathcal{B}$ is an input-subautomaton of $\mathcal{A}$. The automaton $\mathcal{C} = (C, \Sigma_{\mathcal{C}}, \delta_C)$ is isomorphic to $\mathcal{A}$ if there are bijective mappings $\tau_1 : C \to A, \tau_2 : \Sigma_{\mathcal{C}} \to \Sigma$ with $\tau_1(\delta_C(c, x)) = \delta(\tau_1(c), \tau_2(x)), c \in C, x \in \Sigma_{\mathcal{C}}$. If $\Sigma_{\mathcal{C}} = \Sigma$ and $\tau_2(x) = x, x \in \Sigma$ then we say that $\mathcal{C}$ is state isomorphic to $\mathcal{A}$. In this case, we also say that $\mathcal{A}$ is a state-isomorphic copy of $\mathcal{C}$ and vice versa.[2]

The transition matrix of an automaton is a matrix with rows corresponding to each input and columns corresponding to each state; the state $\delta(a, x)$ is put at the entry of any row indicated by an input $x \in \Sigma$ and any column indicated by a state $a \in A$. If all rows of the transition matrix are permutations of the state set then we speak about a *permutation automaton.*

Next we prove the following statement.

Proposition 1. *Given a permutation automaton $\mathcal{A} = (A, \Sigma, \delta)$, for every pair $b \in A, x \in \Sigma$, there exists exactly one $a \in A$ with $\delta(a, x) = b$.*

Proof. Assume that there exists no $a \in A$ with $\delta(a, x) = b$. Then the row of of the transition matrix labeled by x does not contain b. But then $\mathcal{A}$ is not a permutation automaton, a contradiction.

Next we assume that there are $a_1, a_2 \in A$ with $a_1 \neq a_2$ $\delta(a_1, x) = b$ and $\delta(a_2, x) = b$. Then the row of of the transition matrix labeled by x contains b two times, a contradiction again. $\square$

Let $\mathcal{A}_i = (A_i, \Sigma_i, \delta_i)$ be automata where $i \in \{1, \ldots, n\}$, $n \geq 1$. Take a finite nonvoid set Σ and a *feedback function* $\varphi_i : A_1 \times \cdots \times A_n \times \Sigma \to \Sigma_i$ for every $i \in \{1, \ldots, n\}$. A *Gluškov-type product* of the automata $\mathcal{A}_i$ with respect

[1]The empty word is not considered as a state word.

[2]Obviously, then the bijective mapping $\tau_1 : C \to A$ unambigously determines the state isomorphism of $\mathcal{C}$ onto $\mathcal{A}$.

to the feedback functions φ_i $(i \in \{1,\dots,n\})$ is defined to be the automaton $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(\Sigma, (\varphi_1,\dots,\varphi_n))$ with state set $A = A_1 \times \cdots \times A_n$, input set Σ, transition function δ given by $\delta((a_1,\dots,a_n),x) = (\delta_1(a_1,\varphi_1(a_1,\dots,a_n,x)),\dots,\delta_n(a_n,\varphi_n(a_1,\dots,a_n,x)))$ for all $(a_1,\dots,a_n) \in A$ and $x \in \Sigma$.

We shall use the feedback functions $\varphi_i, i \in \{1,\dots,n\}$ in an extended sense as mappings $\varphi_i^* : A_1 \times \cdots \times A_n \times \Sigma^* \to \Sigma_i^*$, where $\varphi_i^*(a_1,\dots,a_n,\lambda) = \lambda$, and $\varphi_i^*(a_1,\dots,a_n,px) = \varphi_i^*(a_1,\dots,a_n,p)\varphi_i(\delta_1(a_1,\varphi_1^*(a_1,\dots,a_n,p)),\dots,\delta_n(a_n,\varphi_n^*(a_1,\dots,a_n,p)),x)$, $a_i \in A_i$, $i \in \{1,\dots,n\}, p \in \Sigma^*, x \in \Sigma$. In the sequel, $\varphi_i^*, i \in \{1,\dots,n\}$ will also be denoted by φ_i.

We can imagine this structure as a working model in the following way. The product is a collection of automata so that every member of this collection is supplied with a transformer which is a special type of finite state transducer. The transformers, realizing the feedback functions mentioned above, are able to get an input vector containing a common external input sign and the state of all component automata. They can each transform this input vector into an appropriate input sign for their component automaton. The product is at work along a discrete time scale in the following way: all transformers of the product get a common external input sign x, and simultaneously, all transformers get the value of the instantaneous states $a_1,\dots,a_n$ of all component-automata as input information. Induced by this this input vector $(a_1,\dots,a_n,x)$, the transformers produce an input sign $x_i = \varphi_i(a_1,\dots,a_n,x), i \in \{1,\dots,n\}$ for their component-automata. Then, these (transformed) input signs take every component-automaton into a new (not necessarily different) state $\delta_i(a_i,x_i) = \delta_i(a_i,\varphi_i(a_1,\dots,a_n,x))$, and then, in the next time period, the whole process takes place again. We will use several generalizations and several restrictions of this concept. If the transformers are able to produce not only single input signs but entire input words (strings of input signs), then induced by the inner input sign x and the value of the instantaneous states $a_1,\dots,a_n$ they produce a (possibly empty) input word $\varphi_i(a_1,\dots,a_n,x)$ working as microprocessors, for their component automata then we get the model of the *generalized product.*

If we assume that transformers do not necessarily have access to all the instantaneous states of component automata, but only some restricted subset, then we will get the models of several special types of the products [8, 9].

It is clear that, by definition, a Gluškov product is a parallel working system. Since parallel working Gluškov product is not appropriate for block cipher, we define its sequentially working version called *sequentially working Gluškov product.*

Consider the above defined Gluškov product modifying its transition function in the following way. Let δ be given by
$\delta((a_1,\dots,a_n),x) = (\delta_1(a_1,\varphi_1(a_1,\dots,a_n,x)),\delta_2(a_2,\varphi_2(a_1',a_2,\dots,a_n,x)),\dots,\delta_{n-1}(a_{n-1},\varphi_{n-1}(a_1',\dots,a_{n-2}',a_{n-1},a_n,x)),\delta_n(a_n,\varphi_n(a_1',\dots,a_{n-1}',a_n,x)))$ for all $(a_1,\dots,a_n) \in A$ and $x \in \Sigma$, where, in order, $a_1' = \delta_1(a_1,\varphi_1(a_1,\dots,a_n,x)), a_2' = \delta_2(a_2,\varphi_2(a_1',\dots,a_n,x)),\dots,a_{n-1}' = \delta_{n-1}(a_{n-1},\varphi_{n-1}(a_1',\dots,a_{n-2}',a_{n-1},a_n,x))$.

Given a function $f : X_1 \times \cdots \times X_n \to Y$, we say that f is *really independent of its i-th variable* if for every pair $(x_1,\dots,x_n),(x_1,\dots,x_{i-1},x_i',x_{i+1},\dots,x_n) \in X_1 \times \cdots \times X_n$, $f(x_1,\dots,x_n) = f(x_1,\dots,x_{i-1},x_i',x_{i+1},\dots,x_n)$. Otherwise we say that f *really depends on its i-th variable.*

A (finite) *directed graph* (or, in short, a *digraph*) $\mathcal{D} = (V, E)$ (of order $n > 0$) is a pair consisting of sets of *vertices* $V = \{v_1, \ldots, v_n\}$ and *edges* $E \subseteq V \times V$. Elements of V are sometimes called *nodes*. If $|V| = n$ then we also say that $\mathcal{D}$ is a digraph of order n.

Given a digraph $\mathcal{D} = (V, E)$, we say that the above defined Gluškov product (sequentially working Gluškov product) is a $\mathcal{D}$-product (sequentially working $\mathcal{D}$-product) if for every pair $i, j \in \{1, \ldots, n\}$, $(i, j) \notin E$ implies that the feedback function φ_i is really independent of its j-th variable.

By a *key automaton* we mean a sequentially working Gluškov product having the following properties:

- it consists of automata components that are state isomorphic to each other so that their state sets also coincide with each other,
- it has the same state and input sets which are sets of all strings with a given length over a fixed alphabet,
- it is a permutation automaton.

4 Encryption and Decryption

Both of the encryption and decryption apparatus use the same key automaton and they use the same pseudorandom generator. We have to use the same pseudorandom blocks during the encryption and decryption processes, because otherwise decryption is impossible, and these pseudorandom blocks have to be secret, otherwise the system is vulnerable. Modern block ciphers create different ciphertext each time when they encrypt the same plaintext. To reach this goal, we have to change the seed of the pseudorandom generator each time when we use encryption. It is not too difficult to satisfy all these properties: we need two blocks, one is constant, secret and part of the key, let us call it ,,core vector", and the other block is changed each time when we use encryption, this one is public, – it is the first block of the ciphertext, – and let us call it ,,initialization vector". The recent seed can be calculated as a function of these two blocks. The most simple solution is to use the exclusive or (bitwise addition modulo 2) operator. In this way the seed will be secret, both of the encryption and decryption process calculate the same seed, they can calculate the same secret pseudorandom blocks, and the seed and the pseudorandom blocks are changed each time, when we use encryption.

There is a fixed positive integer k which is the number of the rounds (see later). Before the encryption procedure, the pseudorandom generator gets its initialization vector as a true random sign $r_1 \ldots r_n \in \Sigma^n$, where the pseudorandom alphabet Σ is also the plaintext and the ciphertext alphabet simultaneously. This initialization vector will be also the first block of the ciphertext.

The encryption procedure is the following. The apparatus reads the plaintext block-by-block and, after reading the next plaintext block $a_1 \cdots a_n \in \Sigma^n$ (first the first block), it generates the second, third, etc. blocks of the ciphertext in the following way.

First the random number generator generates a word $w_1 \cdots w_k$ of pseudorandom sequences, where $w_1, \ldots, w_k \in \Sigma^n$. The key automaton $\mathcal{A} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{A}})$ goes from state $(a_1, \ldots, a_n)$ into state $(c_1, \cdots, c_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_k)$, where $a_1 \cdots a_n$ is the referred next plaintext block. The state $(c_1, \ldots, c_n)$ will be performed sequentially such that, in order, we specify the state $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1)$ by $(a_1, \ldots, a_n)$ and w_1, the state $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 w_2)$, by $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1)$ and w_2, $\ldots$, the state $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ by $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{k-2})$ and w_{k-1}, the state $(c_1, \ldots, c_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_k)$ by $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ and w_k.

Let $w_i = (x_1, \ldots, x_n)$ where $x_1, \ldots, x_n \in \Sigma$ for some $i \in \{1, \ldots, k\}$ and let us define $(d_1, \ldots, d_n)$ and $(e_1, \ldots, e_n)$ by $(e_1, \ldots, e_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_i)$ and $(d_1, \ldots, d_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{i-1})$ if $i > 1$, moreover, $(e_1, \ldots, e_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1)$ and $(d_1, \ldots, d_n) = (a_1, \ldots, a_n)$ if $i = 1$.

Clearly, then $(e_1, \ldots, e_n) = \delta_{\mathcal{A}}((d_1, \ldots, d_n), (x_1, \ldots, x_n))$.

This transition will be performed sequentially in the following way.

$e_1 = \delta_1(d_1, \varphi_1(d_1, d_2, \ldots, d_n, (x_1, \ldots, x_n)),$
$e_2 = \delta_2(d_2, \varphi_2(e_1, d_2, d_3, \ldots, d_n, (x_1, \ldots, x_n)),$
$\ldots$
$e_{n-1} = \delta_{n-1}(d_{n-1}, \varphi_{n-1}(e_1, \ldots, e_{n-2}, d_{n-1}, d_n, (x_1, \ldots, x_n)),$
$e_n = \delta_n(d_n, \varphi_n(e_1, \ldots, e_{n-1}, d_n, (x_1, \ldots, x_n)).$

Applying the above procedure in k round, we finally receive the state $(c_1, \ldots, c_n)$. Then, concatenating the calculated blocks, we will get the ciphertext $c_1 \cdots c_n$.

The decryption procedure is the following. Before the decryption procedure, the pseudorandom generator gets the first ciphertext block as its initialization vector $r_1 \ldots r_n \in \Sigma^n$.

Then the apparatus reads the ciphertext block-by-block and, after reading the next ciphertext block $c_1 \cdots c_n \in \Sigma^n$ (first the second block), it generates the first, second, third, etc. blocks of the plaintext back in the following way.

First the random number generator generates the same word $w_1 \cdots w_k$ of pseudorandom sequences as at the encryption. Recall that the key automaton is a permutation automaton. Therefore, by Proposition 1, it has exactly one state $(a_1, \ldots, a_n)$ from which the key automaton goes into the state $(c_1, \ldots, c_n)$ under the effect of $w_1 \cdots w_k$. Then, applying the transition $(c_1, \cdots, c_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_k)$ the plaintext block $a_1 \cdots a_k$ can be unambiguously recovered.

We specify the state $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ by $(c_1, \ldots, c_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_k)$ and w_k, the state $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{k-2})$ by $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ and w_{k-1}, $\ldots$, the state $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1)$ by $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 w_2)$ and w_2, the state $(a_1, \ldots, a_n)$ by $\delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1)$ and w_1.

The vectors $w_i, (d_1, \ldots, d_n)$, and $(e_1, \ldots, e_n)$ are defined in the same way as it is done at the encryption procedure. In more details, similarly as previously, let $w_i = (x_1, \ldots, x_n)$ where $x_1, \ldots, x_n \in \Sigma$ for some $i \in \{1, \ldots, k\}$ and let us define

$(d_1,\dots,d_n)$ and $(e_1,\dots,e_n)$ by $(e_1,\dots,e_n) = \delta_{\mathcal{A}}((a_1,\dots,a_n),w_1\cdots w_i)$ and $(d_1,\dots,d_n) = \delta_{\mathcal{A}}((a_1,\dots,a_n),w_1\cdots w_{i-1})$ if $i>1$, moreover, $(e_1,\dots,e_n) = \delta_{\mathcal{A}}((a_1,\dots,a_n),w_1)$ and $(d_1,\dots,d_n) = (a_1,\dots,a_n)$ if $i=1$.

To recover $d_1\cdots d_n$, the following equalities are used.

By $e_n = \delta_n(d_n,\varphi_n(e_1,\dots,e_{n-1},d_n,(x_1,\dots,x_n))$, we can determine d_n,

by $e_{n-1} = \delta_{n-1}(d_{n-1},\varphi_{n-1}(e_1,\dots,e_{n-2},d_{n-1},d_n,(x_1,\dots,x_n))$, we can determine d_{n-1},

$\dots$,

by $e_2 = \delta_2(d_2,\varphi_2(e_1,d_2,\dots,d_n,(x_1,\dots,x_n))$,we can determine d_2,

by $e_1 = \delta_1(d_1,\varphi_1(d_1,d_2,\dots,d_n,(x_1,\dots,x_n))$,we can determine d_1.

Thus we can get the plaintext block in k rounds back.

Therefore, if all of $\varphi_1,\dots,\varphi_n$ can be computed easily, then the proposed system could be effective.

To sum up, the discussed cryptosystem is a block cipher. Since the key automaton is a permutation automaton, for every ciphertext there exists exactly one plaintext making the encryption and decryption unambiguous. Moreover, there is a huge number of corresponding encoded messages to each plaintext so that several encryptions of the same plaintext yield several distinct ciphertexts.

5 Example

Next we consider a special key automaton for which the proposed cryptosystem is effective and secure. We are going to use a sequentially working $\mathcal{D}$-product of automata for *key automaton* in this Section.

Let Σ be the set of all binary strings with a given length $\ell \geq 1$ and let n be a positive integer.

Let $\mathcal{A}_1 = (\Sigma,\Sigma\times\Sigma,\delta_{\mathcal{A}_1})$ be a permutation automaton and let $\mathcal{A}_i = (\Sigma,\Sigma\times\Sigma,\delta_{\mathcal{A}_i}), i=2,\dots,n$ be state-isomorphic copies of $\mathcal{A}_1$ such that $\mathcal{A}_1,\dots,\mathcal{A}_n$ are pairwise distinct.[3] Given a digraph $\mathcal{D}=(V,E)$ with $V=\{1,\dots,n\}, E=\{(n,1),(1,2),\dots,(n-1,n)\}$ define the Gluškov-type product, called $\mathcal{D}$-product, $\mathcal{A}_{\mathcal{D}} = \mathcal{A}_1\times\cdots\times\mathcal{A}_n(\Sigma^n,(\varphi_1,\dots,\varphi_n))$ of $\mathcal{A}_1,\dots,\mathcal{A}_n$ so that for every $(a_1,\dots,a_n)$, $(x_1,\dots,x_n)\in\Sigma^n, i\in\{1,\dots,n\}$,

$\varphi_1(a_1,\dots,a_n,(x_1,\dots,x_n)) = (a_n\oplus x_n, x_1)$, where $a_n\oplus x_n$ is the bitwise addition modulo 2 of a_n and x_n,

$\varphi_i(a_1,\dots,a_n,(x_1,\dots,x_n)) = (a_{i-1}\oplus x_{i-1}, x_i), i=2,\dots,n$ where $a_{i-1}\oplus x_{i-1}$ is the bitwise addition modulo 2 of a_{i-1} and x_{i-1}.

Then the sequentially working version of $\mathcal{A}_{\mathcal{D}}$ is the automaton $\mathcal{B} = (\Sigma^n,\Sigma^n,\delta_{\mathcal{B}})$, where for every $(a_1,\dots,a_n),(x_1,\dots,x_n)\in\Sigma^n$, $\delta_{\mathcal{B}}((a_1,\dots,a_n),(x_1,\dots,x_n)) = (b_1,\dots,b_n)$ such that

$b_1 = \delta_{\mathcal{A}_1}(a_1,\varphi_1(a_1,\dots,a_n,(x_1,\dots,x_n))$

and $\varphi_1(a_1,\dots,a_n,(x_1,\dots,x_n)) = (a_n\oplus x_n, x_1)$,

[3]In other words, for every $i,j\in\{1,\dots,n\}$, $i\neq j$ implies $\mathcal{A}_i\neq\mathcal{A}_j$.

$b_2 = \delta_{\mathcal{A}_2}(a_2, \varphi_2(b_1, a_2, \ldots, a_n, (x_1, \ldots, x_n))$,
and $\varphi_2(b_1, a_2, \ldots, a_n, (x_1, \ldots, x_n)) = (b_1 \oplus x_1, x_2)$,
...
$b_{n-1} = \delta_{\mathcal{A}_{n-1}}(a_{n-1}, \varphi_{n-1}(b_1, \ldots, b_{n-2}, a_{n-1}, a_n, (x_1, \ldots, x_n))$,
and $\varphi_{n-1}(b_1, \ldots, b_{n-2}, a_{n-1}, a_n, (x_1, \ldots, x_n)) = (b_{n-2} \oplus x_{n-2}, x_{n-1})$,
$b_n = \delta_{\mathcal{A}_n}(a_n, \varphi_n(b_1, \ldots, b_{n-1}, a_n, (x_1, \ldots, x_n))$,
and $\varphi_n(b_1, \ldots, b_{n-1}, a_n,(x_1, \ldots, x_n)) = (b_{n-1} \oplus x_{n-1}, x_n)$.

Of course, the values of the feedback functions can be computed easily. By the encryption procedure, using the transition matrices of the component automata, we can specify easily the state b_1 from a_1, a_n, x_n, x_1, the state b_2 from a_2, b_1, x_1, x_2, $\ldots$, the state b_{n-1} from $a_{n-1}, b_{n-2}, x_{n-2}, x_{n-1}$, the state b_n from $a_n, b_{n-1}, x_{n-1}, x_n$. On the other hand, all component automata of the key automaton are permutation automata. Therefore, by the decryption procedure, using again the transition matrices of the component-automata, we can specify unambiguously the state a_n from $b_{n-1}, b_n, x_{n-1}, x_n$, the state a_{n-1} from $b_{n-2}, b_{n-1}, x_{n-2}, x_{n-1}$, $\ldots$, the state a_2 from b_1, b_2, x_1, x_2, the state a_1 from a_n, b_1, x_n, x_1.

6 Avalanche Effect

The avalanche effect is a very important property of block ciphers. We say the block cipher has avalanche effect when a small change in the plaintext block (or in the key) results a significant change in the corresponding ciphertext block, and also small change in the ciphertext block (or in the key) results a significant change in the corresponding plaintext block after decoding. In section 4 we introduced a very simple key automaton, which works well, but it has just limited avalanche effect. Suppose we have a plaintext block $a = (a_1, \ldots, a_n) \in \Sigma^n$, a pseudorandom block $w_1 = (x_1, \ldots, x_n) \in \Sigma^n$ and the key automaton $\mathcal{B} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{B}})$ goes to the ciphertext block $b = (b_1, \ldots, b_n) \in \Sigma^n$ from a by the effect of w_1. (In short, $\delta_{\mathcal{B}}(a, w_1) = b$.) Let us define $c = (a_1, \ldots, a_{i-1}, c_i, a_{i+1}, \ldots, a_n) \in \Sigma^n$, where $a_i \neq c_i$, $1 < i < n$, and calculate the $d = \delta_{\mathcal{B}}(c, w_1)$ value. We will see that d starts with $b_1, \ldots, b_{i-1}$ so changing a_i to c_i has no effect for the first $i-1$ part of the ciphertext block. However, from the i-th part, we have appropriate avalanche effect. This is the same with the pseudorandom block, changing x_i to c_i ($x_i \neq c_i$, $1 < i < n$) has no effect for the first $i-1$ part of the ciphertext block, but it has appropriate avalanche effect from the i-th part of the ciphertext. The solution is simple. We should repeat the encoding procedure twice. First calculate the $a' = \delta_{\mathcal{B}}(a, w_1)$ block, then calculate the $b = \delta_{\mathcal{B}}(a', w_1)$ ciphertext block.

Unfortunately, the situation during the decoding is worst. Suppose we have the $b = (b_1, \ldots, b_n) \in \Sigma^n$ ciphertext block, the $w_1 = (x_1, \ldots, x_n) \in \Sigma^n$ pseudorandom block and the key automaton $\mathcal{B} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{B}})$ goes to the ciphertext block b from the paintext block $a = (a_1, \ldots, a_n) \in \Sigma^n$ by the effect of w_1. (In short, $\delta_{\mathcal{B}}(a, w_1) = b$.) Let us define $\gamma_{\mathcal{B}}$ such that $\gamma_{\mathcal{B}}(\delta_{\mathcal{B}}(a, w), w) = a$ for each $a, w \in \Sigma^n$. In this case $\gamma_{\mathcal{B}}(b, w_1) = a$. Now let us define the $d = (b_1, \ldots, b_{i-1}, d_i, b_{i+1}, \ldots, b_n) \in \Sigma^n$, where

$b_i \neq d_i$, $1 < i \leq n$. Comparing a and $\gamma_{\mathcal{B}}(d, w_1)$ we can recognize that changing the i-th part of the ciphertext block has effect only on the i-th and $i-1$-th part of the plaintext block. This means we can not have appropriate avalanche effect during decoding using only the above defined $\gamma_{\mathcal{B}}$ function. To solve this problem, we have to use the $\delta_{\mathcal{B}}$ function twice during the decoding process.

Finally, we created the following function, which has 3 parameters, can do the encoding and the decoding, and – based on experimental results, – it has appropriate avalanche effect during the encoding and the decoding process:

$$f(a, w_1, w_2) = \gamma_{\mathcal{B}}(\gamma_{\mathcal{B}}(\delta_{\mathcal{B}}(\delta_{\mathcal{B}}(a, w_1), w_1), w_2), w_2).$$

This function first receives the plaintext block a and two pseudorandom blocks w_1 and w_2.

Then, it calculates the $a' = \delta_{\mathcal{B}}(a, w_1)$ value.

In the next round, it calculates the $a'' = \delta_{\mathcal{B}}(a', w_1)$ value.

In the next round, it calculates the $a''' = \gamma_{\mathcal{B}}(a'', w_2)$ value.

In the next round, it calculates the $b = \gamma_{\mathcal{B}}(a''', w_2)$ value, which is the ciphertext block.

Decoding done with the same function, but it has different parameters: $f(b, w_2, w_1)$. In this case the same f function first receives the ciphertext block b and the two pseudorandom blocks w_2 and w_1 in the opposite order.

Then, it calculates the $a''' = \delta_{\mathcal{B}}(b, w_2)$ value.

In the next round, it calculates the $a'' = \delta_{\mathcal{B}}(a''', w_2)$ value.

In the next round, it calculates the $a' = \gamma_{\mathcal{B}}(a'', w_1)$ value.

In the next round, it calculates the $a = \gamma_{\mathcal{B}}(a', w_1)$ value, which is the plaintext block.

For protection against chosen ciphertext attack, we recommend to repeat this procedure at least twice during the encoding and decoding process, with different pseudorandom numbers. For example, the ciphertext block b can be calculated from the plaintext block a by the function $f(f(a, w_1, w_2), w_3, w_4)$, with four pseudorandom number blocks w_1, w_2, w_3, w_4, and then, we can decipher the plaintext block a from the ciphertext block b using the function $f(f(b, w_4, w_3), w_2, w_1)$.

7 Experimental Results

We have been developed some practical tests using 16 bytes (128 bits) long input blocks, output blocks and pseudorandom blocks. It has been done for the cases when both of the encryption and decryption algorithms in Chapter 4 have been modified as it is formulated in Chapter 6.

7.1 Keyspace Size

Using the above mentioned parameters with 256 possible states, (1 byte long states,) we need 16 automata, having a transition matrix $2^{16} = 65536$ lines and $2^8 = 256$

columns. Each cell of the automaton contains 1 byte long data. (One state.) The size of the matrix is 16 megabytes, and the number of possible matrices is $256!^{65536}$, where the exclamation mark means the factorial operation. This is much more than good enough protection against brute-force attack. When we use isomorphic automata, this huge number should be further increase to have $256!^{65536} * 256!^{15} = 256!^{65551}$ possible keys.

7.2 Speed Test Results

The practical tests of the encoding and decoding algorithm were done on an average table PC, (3,1 GHz Intel Core I3-2100 processor, 4 Gigabyte RAM). The program we used was a well written C# implementation. The results of the speed tests of the 8 bit version can be seen in the table 1.

Table 1: Results of the speed tests

size (bytes)	encoding time	decoding time	encoded bytes per second
131104	00.0169140	00.0164919	7751212
524336	00.0572925	00.0573531	9151913
1048656	00.1111786	00.1098338	9432175
33556496	03.8841316	04.0200288	8639382
134225936	16.0446227	16.1320934	8365789

The results of the speed tests show that using an average PC, the encoding time is more than 7 megabytes per second, and decoding time is about the same.

7.3 Effectiveness of the Avalanche Effect

We used to test the avalanche effect in the following way. We chose 1000000 random plaintext blocks, encoded them, and then we changed 1 bit in each plaintext block, encoded again, then we calculated the number of the different bytes in the ciphertext blocks pair-wise. The opposite case has been also tested, namely there were chosen 1000000 random ciphertext blocks, we decoded them, and then we changed 1 bit in each ciphertext block, decoded again, and calculated the number of the different bytes in each plaintext blocks pair-wise. The results can be seen in the table 2.

Table 2: Results of the avalanche effect of encoding and decoding

different characters in one block	encoding	decoding
0-12	0	0
13	24	32
14	1771	1743
15	58851	59028
16	939354	939197

When we change only one bit in the plaintext block, the difference between the corresponding ciphertext blocks will be really huge in the majority of the cases. The same effect can be seen in the opposite case, changing one bit in the ciphertext block results huge difference in the plaintext block as well.

We created another table as well. In this table we calculated the optimal avalanche effect. We had choosen 2×1000000 completely random blocks, and then calculated the difference between them pair-wise. The results can be seen in the table 3.

Table 3: Results of the avalanche effect of complete random blocks

different characters in one block	
0-12	0
13	32
14	1693
15	58681
16	939594

By our experimental results, we can conclude that the algorithm has the optimal avalanche effect, and an appropriate speed (more than 7 megbyte/s). Of course the speed of the algorithm depends on the hardware and the programming language / program code as well.

8 Conclusion and Future Work

This paper is devoted to propose a novel cryptosystem based on Gluškov product of automata. By a simple example, its utility is shown. The avalanche effect tests show good results. Moreover, some experimental results show the effectiveness. However, serious security analysis and rigorous machine-independent investigation should be necessary in the future work.

References

[1] Atanasiu, A. A class of coders based on gsm. Acta Informatica, 29 (1992), 779-791.

[2] Bao, F. Cryptoanalysis of partially known cellular automata. IEEE Trans. on Computers, 53 (2004), 1493-1497.

[3] Bao, F. and Igarashi, Y. Break finite automata public key cryptosystems. In: Fülöp, Z., Gécseg F., eds., Proc. 22nd Int. Coll. On Automata Languages and Programming - ICALP'95, Szeged, Hungary, July 10-14, 1995, LNC 944, Springer-Verlag, Berlin, 1995, 147-158.

[4] Biham, E. Cryptoanalysis of the chaotic map cryptosystem suggested at EUROCRYPT'91. In: Davies, D. W., ed., Proc. Conf. Advances in Cryptology - EUROCRYPT'91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, LNCS 547 Springer-Verlag, Berlin, 1991, 532-534.

[5] Clarridge, A., Salomaa, K. A Cryptosystem Based on the Composition of Reversible Cellular Automata. In: Dediu, A.-H., Ionescu, A.-M., Martn-Vide, C., eds., Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings. Volume 5457 of Lecture Notes in Computer Science, pages 314-325, Springer, 2009.

[6] Dömösi, P. A Novel Cryptosystem Based on Finite Automata without Outputs. In: Ito, M., Kobayashi, Y., Shoji, Kunitaka, S., eds., AFLAS'08, Proc. Int. Conf. On Automata, Formal languages and Algebraic Systems, Kyoto, Japan, 20-22 September 2008, World Scientific, New Jersey, London, Singapore, Beijing, Shanghai, Hong Kong, Taipei, Chennai, 2010, 23-32.

[7] Dömösi, P. A novel stream cipher based on finite automata. Cryptosystem Based on Finite Automata without Outputs. In: Vlad, M. S. and Sgarciu, V., eds., Intellisec - The 1st International Workshop on Intelligent Security Systems, 11-14 November, 2009, Printech, Bucharest, Romania, 2009, 16-25.

[8] Dömösi, P. and Nehaniv, C. L. Algebraic theory of automata networks. An introduction. SIAM Monographs on Discrete Mathematics and Applications, 11. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

[9] Gécseg, F. Products of Automata. EATCS Monogr. Theoret. Comput. Sci. 7, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1986.

[10] Gécseg, F. and Peák, I. Algebraic theory of automata. Akadémiai Kiadó Budapest (Hungary), 1972.

[11] Gluskov, V. M. Abstract theory of automata (in Russian). Uspekhi Mat. Nauk, 16 (101) (1961), 3-62; correction, ibid., 17 (104) (1962), 270.

[12] Guan, P. Cellular automaton public key cryptosystem. Complex Systems, 1 (1987), 51-56.

[13] Gutowitz, H. A. Method and Apparatus for Encryption, Decryption, and Authentication Using Dynamical Systems. US P 5,365,589, 1994.

[14] Gysin, M. One-key cryptosystem based on a finite non-linear automaton. Dawson, E., Golic,J., eds., Proc. Int. Conf. Proceedings of the Cryptography: Policy and Algorithms, CPAC95, Brisbane, Queensland, Australia, July 3-5, 1995. Lecture Notes in Computer Science 1029 Springer-Verlag, Berlin, 1995, 165-163.

[15] Hopcroft, J.E., Motwani, R., and Ullman, J. D. Introduction to Automata Theory (second edition). Addison-Wesley Series in Computer Science, Addison-Wesley Co., Reading, MA, 2001.

[16] Kari, J. Cryptosystems based on reversible cellular automata. University of Turku, Finland, April, 1992, preprint.

[17] Meier, W. and Staffelbach, O. Analysis of pseudo-random sequences generated by cellular automata. In: Davies, D. W., ed., Proc. Conf. Advances in Cryptology EUROCRYPT91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, LNCS 547 Springer-Verlag, Berlin, 1991, 186-199.

[18] Menezes, A. J., Oorschot, P. C., Vanstone, S. A. Handbook of Applied Cryptography. CRC Press Series on Discrete Mathematics and Its Applications, CRC Press LLC, Boca Raton, FL, USA, 1996, 2001, 2008.

[19] Meskaten, T. On finite automaton public key cryptosystems. TUCS Technical Report No. 408, Turku Centre for Computer Science, Turku, 2001, 1-42.

[20] Rayward-Smith, V. J. Mealy machines as coding devices. In: H. J. Beker and F. C. Piper, eds., Cryptography and Coding, Claredon Press, Oxford, 1989.

[21] Tao, R. Finite Automata and Application to Cryptography. Springer-Verlag, Berlin, 2009.

[22] Wichmann, P. Cryptoanalysis of a modified rotor machine. In: Quisquarter, J.-J., Vandewalle, J., eds., Proc. Conf. Advances in Cryptology - EUROCRYPT'89, Workshop on the Theory and Applications of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, LNCS 434, Springer-Verlag, Berlin, 1990, 395-402.

[23] Wolfram, S. Cryptography with Cellular Automata. In: Hugh, C. W., ed., Proc. Conf. Advances in Cryptology - CRYPTO'85, Santa Barbara, California, USA, August 18-22, 1985, LNCS 218, Springer-Verlag, Berlin, 1986, 429-432.

Received 13th January 2015