VERSITA

Tatra Mountains Mathematical Publications
DOI: 10.2478/tmmp-2013-0038
Tatra Mt. Math. Publ. **57** (2013), 101–118

A NEW STREAM CIPHER BASED ON FIALKA M-125

Eugen Antal—Viliam Hromada

ABSTRACT. In 2010, a new cipher Hummingbird by [Engels, D.—Fan, X.–Gong, G.—Hu, H.—Smith, E. M. *Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices*, in: 1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices. Tenerife, Canary Islands, Spain, January 2010] was proposed. It is a combination of both block and stream cipher and its design was inspired and motivated by the Enigma machine. The encryption process of the cipher can be considered as a continuous running of a rotor-cipher. Four block ciphers play the role of the rotors that apply the permutation to the 16-bit words. This cipher motivated us to investigate a new cipher design based on a Fialka cipher machine.

Fialka M-125 is an Enigma based rotor-cipher machine used during the Cold War. It is considered one of the most secure cipher machines. Advantages of this cipher are based on the elimination of the Enigma's known weaknesses. There are no known attacks on this cipher. In this paper we introduce a new cipher based on the Fialka machine. We transform the Fialka encryption algorithm to a modern stream cipher. The rotors are represented as S-boxes and shift registers are used to provide the rotor clocking. We propose three different versions of the cipher and investigate the statistical properties of their outputs. In the article we also provide basic implementation details and basic performance analysis.

1. Introduction

Couple of years ago, cryptography could be divided into two disjoint parts - classic cryptography (usually ciphers and ciphering machines used before the WWII) and modern cryptography (ciphers and machines used after the WWII). But in recent years a new trend has emerged - designs of modern ciphers are inspired by those older "classic" ciphers which are even today considered secure. One of these new modern ciphers is the Hummingbird cipher [3].

2010 Mathematics Subject Classification: 94A60, 68P25.
Keywords: lightweight cryptography, stream ciphers, Fialka M-125.
This work was partially supported by the grant VEGA 1/0173/13 and by the grant APVV--0586-11.

Its design is based on the famous old German ciphering machine Enigma and is considered a combination of block and stream ciphers. This interesting trend led us to an idea of designing a stream cipher that would be based on a different old, but still considered secure, cipher Fialka [7].

Stream ciphers are ciphers where the encryption/decryption transformation changes with each processed symbol. They are said to have a "memory", because usually the next state of the cipher depends either on previous states of the cipher or on previous output of the cipher. Usually, they are constructed as a binary additive cipher where the output of a pseudo-random generator is xored onto a plaintext. Stream ciphers are considered faster and less-hardware-demanding than block ciphers, which makes them suitable for low-resource environments and also for lightweight cryptography. In 2008, the eSTREAM [1, 8] project orientated on new stream ciphers ended, and as a result, several stream ciphers were proposed for use in cryptography, either as software ciphers or hardware ciphers.

Stream ciphers (or their principles) have been presented in the world of securing information for many years. Russian encryption machine called Fialka [7] is an example of a mechanical ciphering machine used in Cold War. Its design resembles a stream cipher, because for each letter of plaintext a different encryption, dependent on previous ones, is used (in other words, two same letters at a different position in a plaintext are likely to encrypt to two different letters in ciphertext). The construction uses a collection of rotors and fixed permutations, on the other hand, modern stream ciphers use linear and non-linear feedback shift registers, non-linear filtering functions and non-linear clocking of registers.

This difference between Fialka construction and the construction of modern stream ciphers inspired us to design a stream cipher, which would follow Fialka construction's principles.

We designed several variants of this stream cipher with different sizes of internal states and different number of rounds. We carried out a number of statistical and performance tests to test our constructions.

Chapter 2 contains the description of Fialka's algorithm and the ciphering machine. In chapter 3 we present our constructions, which were tested and the results are summarized in chapter 4. Chapter 5 provides a summary of our work.

2. Fialka M-125 cipher design

Fialka M-125 is an electro-mechanical rotor-cipher machine (Figure 1) [7]. The cipher machine was created by the Soviet Army. It was first introduced around 1965, and was officially used until the collapse of the Soviet Union in 1991 [7].

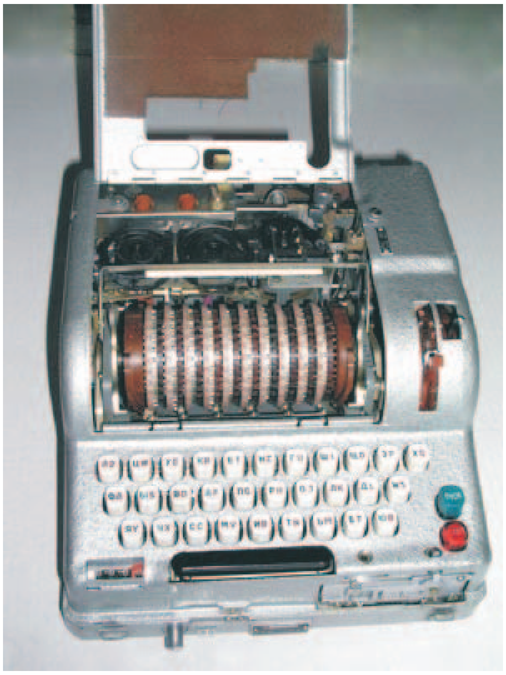

Figure 1. The Fialka cipher machine [6].

The design of the Fialka cipher machine is largely based on the well-known cipher machine Enigma. There are some major differences in the machines construction that can imply the effort to correct the known Enigma's weaknesses. The main improvements are [7]:

- The number of rotors is increased up to 10.
- The Enigma's simple rotor stepping is replaced by advanced - blocking pin based two directional rotor stepping.
- Any input letter can be self-encrypted.

In this article we do not provide a full encryption mechanism of the Fialka cipher, and we do not investigate all the cipher's components in depth. Instead, we will focus only on some parts of the cipher, necessary to keep the cipher's main operation principle.

For complete technical details and electronic description please refer to [7] and web page http://www.cryptomuseum.com/crypto/fialka/index.htm that also includes Fialka simulators. For description and for mathematical model of the Fialka cipher please refer to [2].

2.1. Construction of the cipher

In this section we describe the Fialka machine's main mechanical components. We focus on components that represent the main principle of the cipher's design.

The Fialka M-125 device itself can be divided into several parts, based on their functionality [7]:

- **Keyboard** containing 30 keys , that serves as an input of the cipher machine. Pressing a key sends an electrical signal from the key into the output of the cipher machine through several electro-mechanical components of the device listed below.
- **Card reader** that allows to use punched cards as a fixed permutation of the input alphabet. Fialka also contains a metal triangle that can be inserted into the card reader replacing the punched cards with identity.
- The core of the encryption mechanism of the Fialka cipher machine is realized by a set of 10 **rotors**. After encrypting a letter, rotors rotate to new positions, so in the next step a different permutation of the input alphabet is realized.
- The set of 10 rotors is linked to the static parts of the machine by a connection to two static components, the **entry disc** (from the right) and the **reflector** (from the left). These components have similar structure like the rotors. Each of them contains 30 contacts. The functionality of the reflector is important, it is used to reflect the signal back to the set of rotors. The reflector's wiring is special, it enables the **self-encryption** of the input letter. Because of the special reflector, a different operation mode has to be used for encryption and decryption.

The encryption process is based on electro-mechanical principles. The complex electrical circuit serves to perform the encryption of the input letter. The corresponding mechanical components change the configuration of the cipher (rotors are moved into new positions, so in the next step a different permutation of the input alphabet is realized). A configuration of the electrical circuit depends on the initial settings of the machine, and is changed after each encrypted letter. The output letter can be printed to a tape, or punched using a 5-bit code [7]. The full encryption procedure is summarized in Algorithm 1.

Based on article [2] we can shortly describe the encryption mathematically. We can map the plaintext and ciphertext alphabet to the ring

$$\mathbb{Z}_N = \{0, 1, \ldots, N-1\},$$

where N is the number of symbols (in original Fialka $N = 30$).

Algorithm 1 Fialka encryption procedure.

```
Require: Secret key $K$
configure the cipher (initial settings) using the secret key $K$
for all pressed keyboard letters do
    keyboard to card reader substitution;
    card reader to entry disc substitution;
    for i:=1 to 10 do
        rotor substitution;
    end for
    reflector substitution;
    for i:=10 to 1 do
        inverse rotor substitution;
    end for
    card reader to entry disc inverse substitution;
    keyboard to card reader inverse substitution;
    print the output;
    perform rotor clocking;
end for
```

The encryption process performs the following operation in each step [2]:

1. a single letter $x \in \mathbb{Z}_N$ is encrypted
$$y = Enc(x) = \left(IP^{-1} \circ ROT^{-1}[t] \circ REF \circ ROT[t] \circ IP\right)(x),$$
where IP is an initial permutation given by keyboard/card reader/entry disc substitutions (key-dependent), REF is a reflector, and ROT is a time (and key) dependent permutation given by the rotors;
2. internal state of the machine is updated, the rotors are rotated into new positions.

Permutation ROT in a $2l$-rotor machine (two independent parts, each has l rotors - see Section 2.3) is given as [2]

$$ROT[t] = \sigma^{(l)}[t] \circ \rho^{(1)}[t] \cdots \sigma^{(2)}[t] \circ \rho^{(l-1)}[t] \circ \sigma^{(1)}[t] \circ \rho^{(l)}[t], \tag{1}$$

where $\rho^{(i)}$'s are clockwise rotors, $\sigma^{(i)}$'s are counter-clockwise rotors (so $\sigma^{(l)}$ is rotor number 1, $\rho^{(1)}$ is rotor number 2, etc.) and t represents the state of the cipher varying with time (each processed symbol) [2].

2.2. Rotors

The core of the encryption mechanism of the Fialka cipher machine is realized by rotors. A rotor is a cylinder [5] from a non-conductive material with two bases containing a set of electrical contacts. These contacts represent the inputs and the outputs of the rotor. The contacts are wired (connected) inside the discs that can be imagined as a fixed permutation of 30 input/output positions. The actual permutation of the input alphabet depends also on the relative position of the rotor (it can be placed in 30 different positions) [7].

Two different types of rotors are known to exist: a rotor with fixed wiring called PROTON-1, and adjustable rotor called PROTON-2, respectively, [7]. The core of adjustable rotors containing the contacts and the wiring could be exchanged between different rotors, or could be rotated or turned over inside a single rotor. The PROTON-2 rotors bring only additional rotor settings (increasing the key space of the cipher), but the rest of the encryption mechanism still remains the same. We will focus on the standard PROTON-1 rotors.

The rotors are divided into two independent parts, where one part rotates clockwise and the other part rotates in opposite direction (details explained in Section 2.3). Let S be a permutation realized by some rotor in a default position. If the clockwise rotor in time t is moved by $c[t]$ steps from a default position, then the permutation $\rho[t]$ can be expressed as [2]

$$\rho[t] = S(x + c[t]) - c[t], \tag{2}$$

where operations $+, -$ are in $\mathbb{Z}_N$ (modulo N). Similarly, for counter-clockwise rotor we get [2]

$$\sigma[t] = S(x - c[t]) + c[t]. \tag{3}$$

2.3. Rotor clocking

The clocking (rotation) of rotors is more complicated than in the case of standard rotor-cipher machines like Enigma. The rotors of the Fialka machine are divided into two independent parts. If we mark the rotors with numbers in increasing order with numbers from 1 to 10, the rotors marked with odd numbers represent one independent part and the rest another independent part (see Figure 3) [2].

The rotors in these two independent parts are rotated in opposite direction (Figure 3. Each rotor has a number of blocking pins on its perimeter (see Figure 2). Presence of the blocking pin in a specific position (that was different in case of even-numbered rotors from the case of odd-numbered ones) prevents all the following connected rotors from rotation. In case of even-numbered rotors the blocking pin blocks all the even-numbered rotors to the right of that rotor. And in case of odd-numbered rotors the blocking pin blocks all the odd-numbered rotors to the left of that rotor [2, 6]. How the rotors are connected illustrates the dashed line in Figure 3.

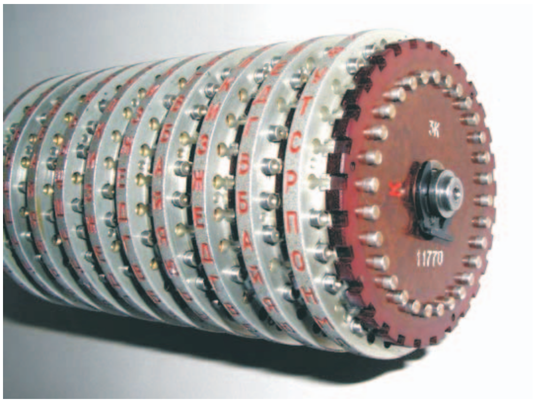

Figure 2. Set of 10 rotors [6].

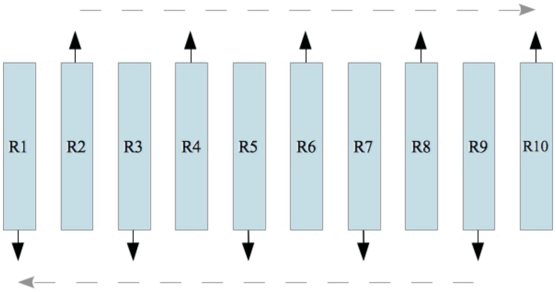

Figure 3. Rotation direction of two independent parts of rotors [2].

Rotor positions influence the individual rotor permutations according to equations (2), and (3), respectively. However, they also influence the (absolute) position of blocking pins in a given time, which in turn influence the positions of rotors in the next step.

Rotors marked with 2 and 9 are rotated by one position in each step because these are not controlled by any blocking pin [7].

Let us simplify the situation to a single set of l rotors (one independent part), whose stepping is connected using blocking pins. The first rotor rotates freely, i.e., in each clock $c[t+1] = c[t] + 1 \mod N$. We can describe the positions of blocking pins on a rotor by a polynomial $a(x) \in \mathbb{Z}_2[x]$. The coefficient $a_i = 0$, if the blocking pin is present at i-th position, and $a_i = 1$ if the blocking pin is not present. The default position has $i = 0$, the next position in the direction of rotation has $i = 1$. Then a single step of the rotor can be simply written as

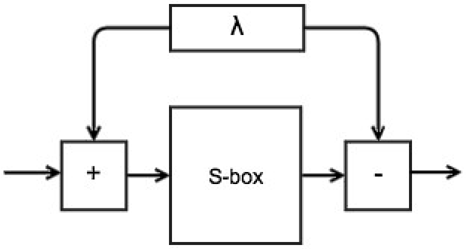

Figure 4. Design of a single rotor.

a polynomial multiplication, e.g., if the rotor is moved in clock t, we get, see [2]

$$a(x)[t+1] = x \cdot a(x)[t] \mod (x^N + 1). \tag{4}$$

Further information about rotor clocking with an example can be found in [2].

3. New stream cipher design

In Section 2 we introduced the Fialka like rotor-machine's main operation principle. The core encryption mechanism of the cipher are the rotor permutation and advanced blocking pin based rotor clocking. In this section we will introduce a modern stream cipher based on these principles. We will show, how to convert this historical cipher machine to the family of modern ciphers.

To keep the Fialka's operation principle we need to represent the rotor permutation and the corresponding blocking pin based rotor clocking.

In a modern stream cipher construction the rotor permutations can be considered as a simple S-box. Based on equation 2, we can describe the rotor permutation S using the Figure 4, where λ is a rotor offset and operations $+, -$ are in $\mathbb{Z}_N$ (modulo N). In this construction (see Figure 5) a counter λ is used ($c[t]$ in equation (2)) to perform the rotation of the rotors and use modular addition and modular subtraction to apply the rotor's offset (change the rotor permutation).

The rotor clocking based on Section 2.3 is performed and controlled by blocking pins available on the rotors. In our case the pins are designed as circular binary shift registers V_i in Figure 5. The binary value in a specific position (e.g., the rightmost bit) controls the movement of the next rotor. If the blocking pin is available (the register's value at a specific position is 0) or the value of the incoming signal from the previous rotors is 0, all the following rotors (of the corresponding independent set of rotors) are not rotated, thus none of the following shift registers V are shifted and values of λ remain unchanged.

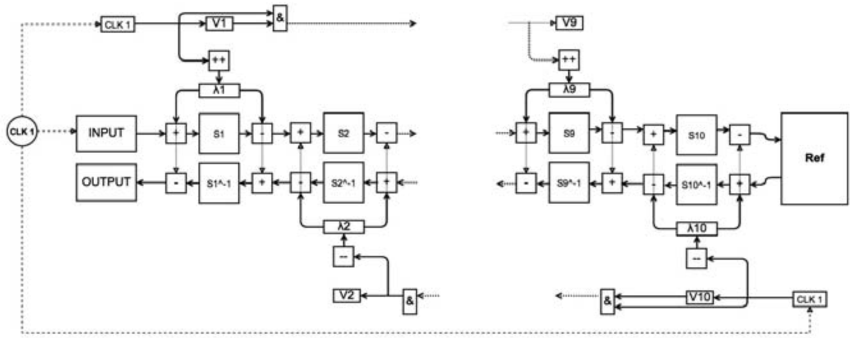

Figure 5. New cipher design (short).

The Fialka cipher (based on Figure 5) can be constructed using the following operations (brackets contain the notation in Figure 5):

- S-box substitution (S_i, Ref),
- Modular addition and subtraction ($+.-$),
- Counter (λ, $++, --$),
- Circular shift registers (V),
- Binary AND (&).

As visible on Figure 5 - we need only 8 shift registers, $V2$ and $V9$ are unused, because the last rotors in each independent part do not control any further rotor and the rotor rotation is unnecessary operation in this case.

The complete identical Fialka's operation can be constructed using 8 shift registers, 8 AND gates, 10 counters, 10 S-boxes for the rotor with 10 corresponding inverse S-boxes, 1 S-box for the reflector with the corresponding inverse S-box.

For an $n - bit$ version of the cipher the size of the λ counter is $n - bits$, the size of a single V shift register is $2^n - bits$ and size of a single S-box is $2^n - bits$. We adapted the encryption process from Section 2.1, where an $n - bit$ vector $x \in \mathbb{Z}_2^n$ is encrypted

$$y = Enc(x) = \left(ROT^{-1}[t] \circ REF \circ ROT[t]\right)(x),$$

where $y \in \mathbb{Z}_2^n$ represents ciphertext, REF is a reflector realized by a bijective $n{-}bit$ to $n{-}bit$ S-box, and ROT is a permutation given by the rotors. We omitted the IP and its inverse, because it does not contribute to cipher's security.

Permutation ROT is given as

$$ROT[t] = \sigma^{(l)}[t] \circ \rho^{(1)}[t] \cdots \sigma^{(2)}[t] \circ \rho^{(l-1)}[t] \circ \sigma^{(1)}[t] \circ \rho^{(l)}[t], \tag{5}$$

where l is a number of rotors in one independent part, $\rho^{(i)}$ and $\sigma^{(i)}$ represent rotor permutation realized by bijective $n - bit$ to $n - bit$ S-boxes and t represents the state of the cipher - the state of the shift registers Vi. The decryption process

$$x = Dec(y) = \left(ROT^{-1}[t] \circ REF^{-1} \circ ROT[t]\right)(y),$$

where $x \in \mathbb{Z}_2^n$ represents plaintext, $y \in \mathbb{Z}_2^n$ represents ciphertext, REF^{-1} is realized by the inverse S-box of REF, and ROT is a permutation given by the rotors.

The potentially highest period of the Fialka cipher is N^l [2], where l is the number of rotors. The clocking period of the system of l rotors, with $(w_1, w_2, \dots, w_{l-1}, w_l)$ blocking pins on individual rotors is (regardless of the position of blocking pins)

$$T_l = \frac{N^l}{\prod_{i=1}^{l-1} \gcd(N - w_i, N)}. \tag{6}$$

In the following sections we will introduce three different versions of the cipher, 4-bit, 5-bit and 8-bit construction.

All these versions are based on a construction presented in Figure 6 which could be found in Appendix B of this paper.

3.1. 4-bit construction

Based on the new design described in section 3, we implemented a 4-bit version of this stream cipher, $n = 4$, i.e., the input/output of the cipher are 4 bits long. We used S-boxes that have good properties against linear and differential cryptanalysis from [9].

We consider two types of key, one for a longer period of time and one that is changed for each encryption.

The output period of this 4 bit version is 2^{20} [2].

As the secret key of this cipher we used initial values of the shift registers V_i's and initial λ_i's.

The 4-bit construction can be summarized:

- 11 S-boxes $\mathbb{Z}_2^4 \to \mathbb{Z}_2^4$ with their respective inverses,
- Key length: 200 bits,
- Period: 2^{20}.

The key length can be reduced to 192 bits if we ignore the V_2 and V_9 shift registers.

3.2. 5-bit construction

The 5-bit cipher design, $n = 5$, is similar to the version in section 3.1.

- 11 S-boxes $\mathbb{Z}_2^5 \to \mathbb{Z}_2^5$ with their respective inverses,
- Key length: 370 bits,
- Period: 2^{25}.

In this case the key length can be also reduced by 10 bits. The only problem of the software implementation is, that the input has to be split by 5 bits, so the optimal required length of the input on 8-bit processors is 40 bits.

3.3. 8-bit construction

Increasing the bit-length of the cipher causes larger internal state of the cipher. In case of 8-bit construction based on Figure 6 we need ten 8-bit to 8-bit S-boxes. S-boxes on rotors take 20480 bits, with the same amount required for inverse S-boxes and 4096 bits for the reflector, together with its inverse. The shift registers take 2560 bits for all rotors. The counters take 80 bits.

To reduce the size of internal state, in this construction we will consider only one S-box with its inverse for all the rotors. The other components remain without a change. This causes that the size of the internal state is reduced by **36864 bits**. The key space of this cipher is also too high. We need 2560 bits for shift registers and 80 bits for the counters. So in this construction we left out the shift registers from the cipher's key. The registers will be fixed. We can consider the key as 80 bits required to initialize the counters. If we want to increase the size of key space we can additionally use 80 bits to shift the 10 shift registers from initial position.

The 8-bit cipher design for $n = 8$ can be summarized:

- 2 S-boxes $\mathbb{Z}_2^8 \to \mathbb{Z}_2^8$ with its respective inverse.
- Key length: 80 bits (or 160 bits).
- Period: 2^{40}.
- Required internal state: 10832 bits.

Reference implementation of the 8-bit version can be found in the Appendix of this article.

4. New stream cipher analysis

We analysed the performance of the software implementation on 2.8 GHz Intel Core i7, 4GB 1333 MHz DDR3, Mac OS X 10.7.5.

- 4-bit version - 61 $Mbit/sec$–367 $cycles/B$,
- 5-bit version - 82 $Mbit/sec$–273 $cycles/B$,
- 8-bit version - 124 $Mbit/sec$–180 $cycles/B$.

We also performed statistical testing of our construction. We generated one hundred 1MB sequences with each version of the cipher and applied the NIST Statistical Test Suite [4] to each set of sequences at a significance level $\alpha = 0.01$. The version of the cipher passed the tests if at least 96 sequences from the appropriate set passed the tests (their p-values were larger than 0.01). The results (PASS / FAIL) are presented in tables below.

Our next goal was to made a trade-off between performance and security, we tried to increase the encryption speed by reducing the number of rotors and by replacing the modular addition and subtraction by XOR. Lowering the number of rotors causes smaller key length, $4-bit$ and $5-bit$ versions of the cipher with $2l$ rotors have a key length of $2l*n+2l*2^n$. In case of $8-bit$ version, rotors have a key length of $2l*n$ (or $2l*n*2$). We also replaced the non-linear operation of modular addition/subtraction with a faster linear operation XOR.

In the following tables, each column represents one type of the cipher - first row represents the number of rotors and used operation (*mod* means modular addition/subtraction, *xor* means XOR), second row represents the average speed of encryption measured in megabits per second and the third row indicates whether the cipher version passed or failed the statistical testing.

Results for the 4-bit version:

Table 1. 4-bit reduced rotors and XOR version results.

Rotors / operation	10/mod	10/xor	8/mod	8/xor
Speed in $Mbit/sec$	61	80	72	92
NIST rand. tests	PASS	FAIL	FAIL	FAIL

We applied the same experiments to 5-bit version of our implementation. Only the version with 8 rotors and XOR failed the NIST tests. It is vital to note, that in case of the 5 bit version, if we reduce it to 8 rotors we have the same period like in case of 4-bit version with 10 rotors.

Results for the 5-bit version:

Table 2. 5-bit reduced rotors and XOR version results.

Rotors / operation	10/mod	10/xor	8/mod	8/xor
Speed in $Mbit/sec$	78	83	113	136
NIST rand. tests	PASS	PASS	PASS	FAIL

Results for the 8-bit version:

The fastest construction that has passed the statistical evaluation is the 8-bit version with 6 rotors and operation XOR. Its output speed is 208 Mbps which corresponds to 107 cycles per byte. We compared these performance results with eSTREAM candidates' results presented at:

- http://www.ecrypt.eu.org/stream/perf/pentium-4-a.

Table 3. 8-bit reduced rotors and XOR version results.

Rotors / operation	10/mod	10/xor	8/mod	8/xor	6/mod	6/xor
Speed in $Mbit/sec$	124	134	155	159	200	208
NIST rand. tests	PASS	PASS	PASS	PASS	PASS	PASS

Unfortunately, our design was slower than most of eSTREAM ciphers intended for software use. For comparision, their performance results are:

- HC-128 - 4 cycles per byte,
- Rabbit - 10 cycles per byte,
- Salsa20/12 - 11 cycles per byte,
- SOSEMANUK - 6 cycles per byte.

We also compared the performance of our design with the performance of lightweight cipher Hummingbird [3] mentioned in the introduction. The resulting output speed of Hummingbird was 165 Mbps. This suggests that our design is in general faster, since our fastest construction has an output speed of 208 Mbps. However, this comparision is only theoretical, because Hummingbird is a hardware orientated cipher, with implementations done mostly in FPGA, while we tested its performance as a software cipher written in C language.

We have not yet made any assumptions on the security of this cipher. The security relies heavily on the original Fialka's security and as far as Fialka is considered secure, i.e., there is not a successful attack - and as far as we know there is not - then we consider our construction secure. Moreover, we have chosen the S-boxes which are known to be resistant against linear and differential cryptanalysis [9]. Practically any S-box could be used, e.g., AES S-box in the 8-bit version, since it is an S-box with well-studied properties. Of course, further security investigation is needed to ensure that this construction can be considered secure for everyday use.

5. Summary

In the article, we presented a new construction of a stream cipher based on a Russian encryption machine Fialka. We carried out performance and statistical tests of three versions of our construction - 4-bit version, 5-bit version and 8-bit version. We investigated the number of rotors needed to successfully pass the statistical NIST-testing suite. Our results show that the 4-bit version with 10 rotors and modular addition/subtraction passes the tests. However, any modification of the 4-bit version (XOR operation, less number of rounds) does not pass the statistical testing. Both 5-bit versions with 10 rotors pass the tests and also the 8-rotor version with modular addition and subtraction passes the tests. 8-bit version could be reduced even further - to 6 rotors and it still passed statistical tests with both modular addition/subtraction and XOR operation. Future research includes hardware implementation, further security research of this type of construction. Also, we would like to investigate whether this construction could be modified to a lightweight construction suitable for RFID tags.

Appendix A. Software implementation

In this appendix we present a reference implementation in **C** of the $8-bit$ version of our cipher design. The corresponding S-boxes have to be inserted into the source code.

```c
#ifndef __FIALKACIPHER_H
#define __FIALKACIPHER_H

#include <stdint.h> // uint64_t
#include <math.h>

typedef unsigned char byte;
typedef uint64_t pin_t;

#define ROT     10    // number of rotoros
#define N       8     // N-bit cipher
#define MOD     256   // pow(2,n)

#define XOR 1 // comment to use modular operation

const byte REF_E[MOD]         = {/*Reflector S-box*/};
const byte REF_D[MOD]         = {/*inverse ref. S-box*/};
const byte SBOX[MOD]          = {/*rotor S-box*/};
const byte SBOX_INV[MOD]      = {/*inverse rotor S-box*/};

pin_t pins[ROT][4]; // 1 binary vector - 4 * 64 bits
byte lambda[ROT];    // 1 lambda - 8 bits

void setKey(pin_t k1[][4], byte k2[]){
    for(int i=0; i < ROT; i++){
        for (int j=0; j < 4; j++) {
            pins[i][j] = k1[i][j];
        }
        lambda[i] = k2[i];
    }
}

```

```
/* function & method prototypes */
 void wheelStepping(void);
 byte rotors(byte input);
 byte rotorsInv(byte input);

void encrypt(byte PT[],byte CT[],const long size);
void decrypt(byte PT[],byte CT[],const long size);

/* impl.*/
inline void wheelStepping(void){
    for(register short i=1; i < 10; i=i+2){
        pins[i][0] = ((pins[i][0]>>1)|(pins[i][1]<<63));
        pins[i][1] = ((pins[i][1]>>1)|(pins[i][2]<<63));
        pins[i][2] = ((pins[i][2]>>1)|(pins[i][3]<<63));
        pins[i][3] = ((pins[i][3]>>1)|(pins[i][0]<<63));
        lambda[i]++;
        lambda[i]%=MOD;
        if((pins[i][0]&0x0000000000000001) == 0) break;
    }
    for(register short i=8; i >= 0; i=i-2){
        pins[i][0] = ((pins[i][0]<<1)|(pins[i][3]>>63));
        pins[i][1] = ((pins[i][1]<<1)|(pins[i][0]>>63));
        pins[i][2] = ((pins[i][2]<<1)|(pins[i][1]>>63));
        pins[i][3] = ((pins[i][3]<<1)|(pins[i][2]>>63));
        lambda[i] += MOD -1;
        lambda[i]%=MOD;
        if((pins[i][0]&0x0000000000000001) == 0) break;
    }

}

inline byte rotors(const byte input){
    register byte x = input;
#ifdef XOR /* XOR */
    for(register short i=0; i < 10; i++){
        x = x^lambda[i];
        x = SBOX[x];
        x = x^lambda[i];
    }
#else /* +,- MOD 2^n */
    for(register short i=0; i < 10; i++){
        x+= lambda[i], x %= MOD;
        x = SBOX[x];
        x += MOD - lambda[i], x %= MOD;
    }
#endif
    return x;
}

inline byte rotorsInv(byte input){
    register byte x = input;
#ifdef XOR /* XOR */
    for(register short i=9; i >= 0; i--){
        x = x^lambda[i];
        x = SBOX_INV[x];
        x = x^lambda[i];
    }
#else /* +,- MOD 2^n */
    for(register short i=9; i >= 0; i--){
        x+= lambda[i], x %= MOD;
        x = SBOX_INV[x];
        x += MOD - lambda[i], x %= MOD;
    }
#endif
    return x;
}

inline void encrypt(byte *PT,byte *CT, const long size){
    for(long i=0; i < size; i++){
        byte x = PT[i];
        x = rotors(x);
        x = REF_E[x];
        x = rotorsInv(x);
        wheelStepping();
        CT[i] = x;
    }
}

inline void decrypt(byte *PT, byte *CT, const long size){
  for(long i=0; i < size; i++){
        byte x = CT[i];
        x = rotors(x);
        x = REF_D[x];
        x = rotorsInv(x);
        wheelStepping();
        PT[i] = x;
  }
}
#endif
```

Appendix B. Block diagram of the proposed cipher

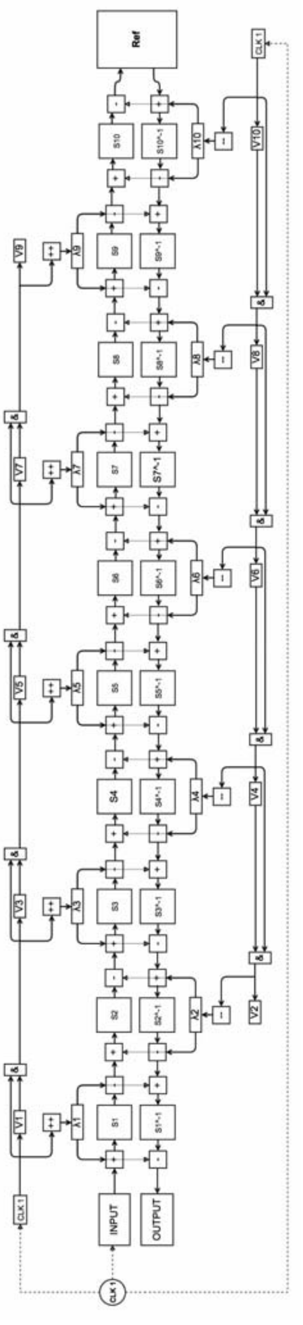

FIGURE 6. New cipher design.

Acknowledgements. The authors are grateful to an anonymous reviewer for helpful comments and remarks.

REFERENCES

[1] ADAMKO L.—VOJVODA, M.—JÓKAY, M.: *Statistical Analysis of ECRYPT eSTREAM Phase3 Ciphers*, EE časopis pre elektrotechniku a energetiku (2008), 193–196,

[2] ANTAL, E.—ZAJAC, P.: *Analysis of the Fialka M-125 cipher-machine*, in: Cryptologia, 2013, (to appear)

[3] ENGELS, D.—FAN, X.—GONG, G.—HU, H.—SMITH, E. M.: *Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices*, in: 1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices. Tenerife, Canary Islands, Spain, January 2010.

[4] RUKHIN, A. ET. AL.: *Sp 800-22 Rev. 1a. a Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Technical Report, Gaithersburg, MD, United States, 2010.

[5] GROŠEK, O.—VOJVODA, M.—ZAJAC, P.: *Classical Ciphers*. STU, 2007. (In Slovak)

[6] PERERA, T.—HAMER, D.: *General introduction: Russian cold war era M-125 and M-125-3MN Fialka cipher machines*. `http://enigmamuseum.com/mfialka.htm`, 2005.

[7] REUVERS, P.—SIMONS, M.: *Fialka M-125: Detailed description of the Russian Fialka cipher machines*, PAHJ Reuvers & MJH Simons, 2009.

[8] *New Stream Cipher Designs - The eSTREAM Finalists*, (M. Robshaw, and O. Billet, eds.) LNCS Vol. 4986, Springer, Heidelberg 2008.

[9] ZAJAC, P.—JÓKAY, M.: *On S-boxes with low multiplicative complexity*. in: Tatracrypt 2012 : 12th Central European Conference on Cryptology. Smolenice, Slovak Republic, July 2012, Slovak Academy of Sciences pp. 47–48.

Received September 15, 2013

Institute of Computer Sci. and Math.
Slovak University of Technology
Ilkovičova 3
SK-812–19 Bratislava
SLOVAKIA
E-mail: eugen.antal@stuba.sk
viliam.hromada@stuba.sk