DE GRUYTER
OPEN

TATRA
MOUNTAINS
Mathematical Publications

DOI: 10.2478/tmmp-2014-0027
Tatra Mt. Math. Publ. **60** (2014), 101–116

A MICRO-CONTROLLER IMPLEMENTATION OF A FIALKA M-125 BASED STREAM CIPHER

Eugen Antal — Viliam Hromada

ABSTRACT. In 2013, a new stream cipher was proposed in Antal, E.–Hromada, V.: *A new stream cipher based on Fialka M-125*, Tatra Mt. Math. Publ. **57** (2013), 101–118. Its design was inspired and motivated by a Soviet encryption machine Fialka M-125. The authors proposed three versions of the cipher with different inner state bit-lengths. They provided the design, software implementation on a personal computer and a preliminary statistical and performance analysis of the cipher.

In this article we extend their work by implementing all three versions of the cipher on two different micro-controllers: EBV SoCrates evaluation board [*Official SoCrates webpage (EBV SoCrates evaluation board),* www.rockerboards.org] and STM32F407VG [*Official STM webpage (STM32F407VG),* `www.st.com`]. We evaluate the performance of all implementations on both platforms. We also investigate the possibilities of performing a simple power analysis of the implementation of the 8-bit version of the cipher implemented on STM32F407VG micro-controller. It stems from our experiments that we are able to determine a part of the secret key of the cipher by observing the power trace (power consumption) of the encryption/decryption process.

1. Introduction

Cryptography protects critical data in various applications, ranging from personal computers to resource constrained devices such as RFID tags and embedded micro-controllers. Nowadays, there are many modern and secure cryptographic algorithms designed for practical use, unfortunately, not all of these algorithms can be implemented on low-resource devices. This leads to a new trend

2010 Mathematics Subject Classification: 94A60, 68P25.
Keywords: micro-controller, side-channel analysis, stream ciphers, Fialka M-125.
This work was partially supported by grants VEGA 1/0173/13, APVV-0586-11, NATO SPS 984520 and project "International center of excellence for research on intelligent and secure information and communications technologies and systems" ITMS 26240120039.

in cryptography, called lightweight cryptography, oriented on designing fast and secure cryptosystems implementable on a low-resource environment. Usually, these ciphers have a smaller key-space (e.g., 80 bits) and use fast and simple operations, e.g., XOR, bit rotation, bit shifting, small look-up tables, etc.

Apart from completely new concepts, some ciphers are based on older encryption algorithms. Designs of these ciphers are inspired by those older "classic" ciphers which are even today considered secure. One of these new modern ciphers is the Hummingbird cipher [7]. Its design is based on the famous old German ciphering machine Enigma and is considered a combination of block and stream ciphers. Another modern stream cipher based on an older ciphering machine Fialka M-125 was proposed in [5]. The design of Fialka M-125 itself was inspired by Enigma, but with a number of improvements that removed Enigma's known weaknesses and it is still considered to be secure, even with today's computational power.

Nowadays, it is not sufficient to investigate only the algorithmic security of the cipher (direct attacks), but also the security of the implementation itself (indirect attacks). In practice, the attacker can obtain valuable information by observing physical effects resulting from the device's behaviour, e.g., power consumption, electromagnetic emission, etc. In combination with the knowledge of the cryptographic algorithm, the attacker can determine the secret parameters from the measurement of these physical effects. These types of attacks are called Side-Channel Attacks (SCA) [8].

The popularity of low-resource devices has been increasing few years in the past. They are used in many different areas, e.g. telemedicine, access controls, communication, etc. They may contain sensitive data, which must be protected from unauthorized access. Therefore, it is important to secure these data by means of cryptography.

In this article, we analyse the implementation of the stream cipher based on Fialka M-125 on two different micro-controllers. We implemented the cipher on two devices: EBV SoCrates evaluation board [2] and STM32F407VG [3]. We analysed the performance of the cipher on both platforms. We also carried out a side-channel analysis of the cipher's implementation on the micro-controller STM32F407VG.

This paper is structured as follows. Chapter 2 contains a short description of the design of used stream cipher. In chapter 3 we present two different micro-controllers used for our purposes and the measurement methodology. The results of performance analysis and SCA are also summarized in Chapter 3. Chapter 4 provides a summary of our work.

2. Fialka M-125 based stream cipher

This section contains the information about the principle of the Fialka M-125 cipher and the stream cipher design. We will focus on parts which we used in our experiments. For detailed information about the Fialka M-125 including the stream cipher design please refer to [5], [6].

Fialka M-125 is an electro-mechanical rotor-cipher machine created by the Soviet Army. It was officially used until the collapse of the Soviet Union in 1991 [12]. The design of this cipher machine is based on Enigma with some major differences in the machine construction.

2.1. Construction of the Fialka M-125 cipher-machine

The Fialka M-125 functionality can be divided into the following parts [12]:

- **Keyboard** that serves as an input. 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.
- 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 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 [12].

The full encryption procedure is described in [6].

2.2. Mathematical preliminaries

We 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 $N = 30$).

The encryption process performs the following operation in each step [6]:

(1) a single letter $x \in \mathbb{Z}_N$ is encrypted

$$y = Enc(x) = \big(IP^{-1} \circ ROT^{-1}[t] \circ REF \circ ROT[t] \circ IP\big)(x);$$

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 as is given in [6])

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

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) [6].

The core of the encryption mechanism of the Fialka cipher machine is realized by a set of rotors. These rotors are divided into two independent parts, where one part rotates clockwise and the other part rotates in the opposite direction.

Let S be a permutation realized by a 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 [6]:

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

where operations $+, -$ are in $\mathbb{Z}_N$ (modulo N). Similarly, for counter-clockwise rotor we get [6]:

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

The clocking (rotation) of the rotors plays an important part in the cipher's security.

The rotors in these two independent parts are rotated in opposite directions. Each rotor has a number of blocking pins on its perimeter (described in detail in [6]). Presence of the blocking pin in a specific position prevents all the following connected rotors from rotating.

In the case of even-numbered rotors the blocking pin blocks all even-numbered rotors to the right of that rotor. And in the case of odd-numbered rotors the blocking pin blocks all odd-numbered rotors to the left of that rotor [6], [10].

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

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 ith 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 a polynomial multiplication, e.g., if the rotor is moved in clock t, we get [6]:

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

2.3. Stream cipher design

To keep the Fialka's operation principle we need to represent the rotor permutation (considered as a simple S-box—in our case we have chosen the S-boxes which are known to be resistant against linear and differential cryptanalysis [13]) and the corresponding blocking pin based rotor clocking.

Based on the equation (2), we can describe the rotor permutation S using the Fig. 1, where λ is a rotor offset and operations $+, -$ are in $\mathbb{Z}_N$ (modulo N). In this construction (see Fig. 2) a counter λ is used ($c[t]$ in the equation (2)) to perform the rotation of the rotors together with modular addition and subtraction which represents the change of the rotor permutation.

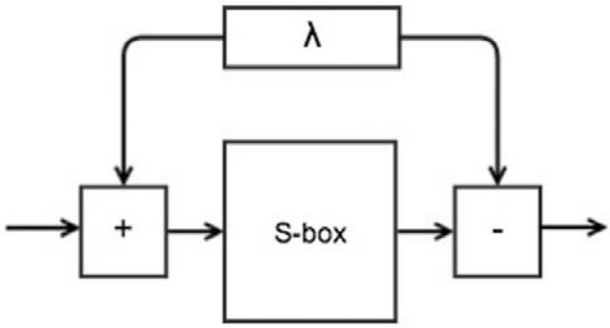

FIGURE 1. Design of a single rotor.

The rotor clocking based 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 Fig. 2. 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 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.

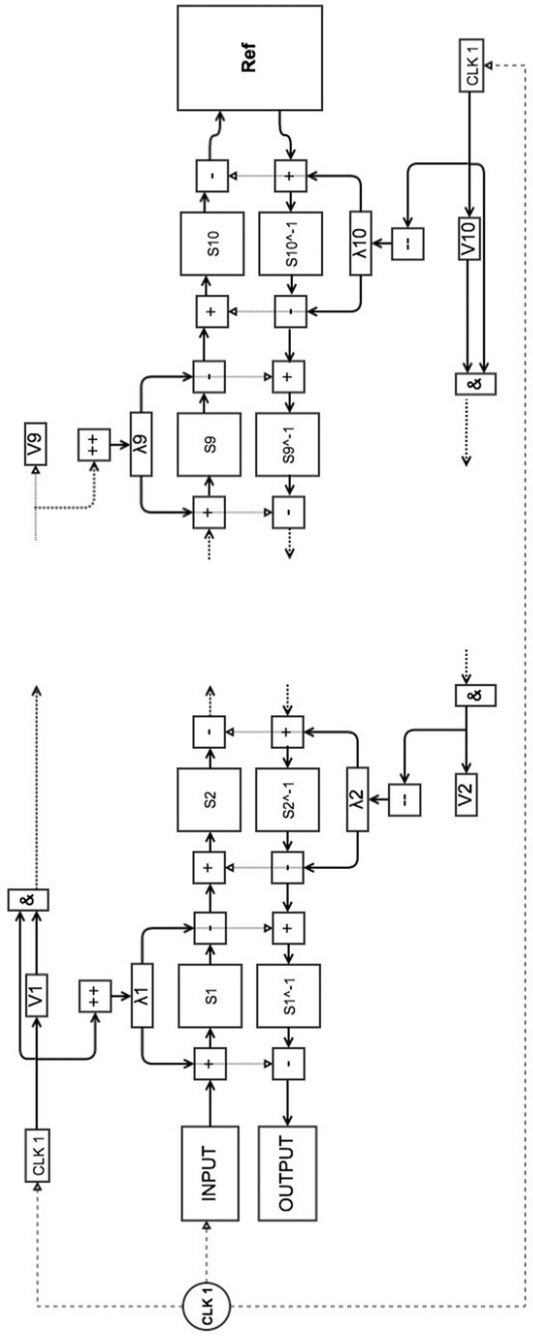

FIGURE 2. New cipher design (short).

The Fialka cipher (based on Fig. 2) can be constructed using the following operations (brackets contain the notation in Fig. 2):

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

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.

For more information about the stream cipher design please refer to [5].

3. Micro-controller implementation

This section describes the two different micro-controllers used for our purposes, the measurement methodology, the performance analysis and the SPA.

3.1. Implementation platforms

We implemented the stream cipher described on two different platforms: EBV SoCrates evaluation board and STM32F407VG in Section 2. Following descriptions were taken directly from manufacturers' websites ([3], [2] respectively).

The device STM32F407VG is a programmable micro-controller based on ARM Cortex-M4 32-bit RISC processor with a frequency up to 168 MHz. It contains 1MB of flash memory and 196kB of SRAM. It offers a wide-range of peripherals: three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers four motor control, two general-purpose 32-bit timers and a TRNG. It also supports up to 140 I/O ports with interrupted capability and up to 15 different communication interfaces. The board is powered by USB. The controller can be directly programmed with dedicated IDE. The program can be written in C/C++ programming language. The compiled applications are directly flashed into the device [3].

The device EBV SoCrates evaluation board is a controller with an operating system support. It is based on the Dual-Core ARM Cortex-A9 MPCore clocked at 925 MHz. It contains 128MB of DDR3 memory and it also supports microSD cards. Its wide variety of interfaces includes 1 Gbit Ethernet, embedded USB--bluster II, temperature sensor, real tie clock, UART/USB Converter, etc. [2].

We used a standard C++ implementation of the stream cipher from [5] on both platforms.

We performed the power consumption measurement (see Section 3.3) using the Picoscope 6403A [1], the oscilloscope with 5GS/s and 256 MS memory for samples.

3.2. Performance measurement

To analyse the performance of the stream cipher on micro-controllers we performed the same performance measurement as in the case of the measurement on personal computers in [5]. We have to note that our implementation is not optimized for 32-bit micro-controllers. We used the same implementation as in [5]. Some operations can be optimized to 32-bit platform (e.g., rewrite some operations into assembler), or we can use the FPGA on SoCrates to increase the performance.

On micro-controller EBV SoCrates we used the same measurement methodology using a standard C system library for time measurement.

The STM32F407VG micro-controller required slight modifications, because it does not support the used system library for time measurement and we also modified the maximal amount of data that can be encrypted/decrypted due to lower amount of memory. In this case we reduced the maximal data length and increased the number of tests. The time measurement on the STM micro-controller was performed using the *Systick timer*—a special system interrupt [4]. The *Systick timer* clock source is 168MHz so it performs 168000000 ticks per second. We set the interrupt time to 1 *ms* by dividing the system clock using the following instruction:

```
SysTick_Config(SystemCoreClock/1000);
```

So the interrupt will be called once per each 1 *ms*. To count the number of elapsed time we used a special counter to count the number of interrupts. We set the value of the counter to *zero* when we start our measurement and increment the counter value in each interruption. At the end of the measurement we save the counter's current value. The following code snippet contains the counting:

```
extern __IO uint32_t counter;

// in stm32f4xx_it.c is the interruption handler
void SysTick_Handler(void)
{
        counter++;
}
```

The results of our performance measurements are summarized and compared with the original results from [5] below.

As one would expect, the performance results of both micro-controller implementations are worse than the results of a PC implementation. What is more, the 8-bit version of the cipher performs almost equally as the 5-bit version. This is caused by the fact that the clocking of the 8-bit version requires larger data

TABLE 1. 4-bit, 5-bit and 8-bit version performance results on personal computer from [5].

bits / operation	8/mod	8/xor	5/mod	5/xor	4/mod	4/xor
Speed in *Mbit/sec*	124	134	78	83	61	80
Speed in *cycles/byte*	180	167	287	269	367	280

TABLE 2. 4-bit, 5-bit and 8-bit version performance results on EBV SoCrates.

bits / operation	8/mod	8/xor	5/mod	5/xor	4/mod	4/xor
Speed in *Mbit/sec*	5.6	5.7	4.9	6.0	3.8	4.6
Speed in *cycles/byte*	1321	1298	1510	1233	1947	1608

TABLE 3. 4-bit, 5-bit and 8-bit version performance results on STM32F407VG.

bits / operation	8/mod	8/xor	5/mod	5/xor	4/mod	4/xor
Speed in *Mbit/sec*	1.01	1.1	0.87	0.98	0.67	0.78
Speed in *cycles/byte*	1330	1221	1544	1371	2005	1723

types (256 bits for shift registers — see Section 2.3) so the number of operations is higher than in other versions. Also, in the PC implementation, one register was represented by four 64-bit variables. This does not perform well on 32-bit platforms and even if we replaced this with eight 32-bit variables, the resulting rotor clocking time would be even larger due to a higher number of operations.

3.3. Simple power analysis

The power consumption of a cryptographic device may provide much information about the operations that take place and the involved parameters. To measure a power consumption, a small resistor is inserted in series with the power of ground input [9], [11].

Simple power analysis (SPA) is a technique where the attacker collects information about the device behaviour by measuring the power consumption and directly interprets the measured power consumption. If the execution path depends on the processed data, then SPA can be directly used to reveal the sequence of executed instructions [11].

According to the results of [5], the 8-bit version can be considered to be the most secure version of the cipher. Our results from Section 3.2 show that

the implementation of this version on micro-controller has a drawback—a longer rotor-clocking time. It also stems from the cipher design that the clocking of rotors for one encrypted input is done at least once and at most four times. This applies for both sets of rotors, so together the clocking is done at least twice and at most eight times. We were therefore interested whether it would be possible to use SPA to measure the number of clockings and determine the corresponding part of the key.

We carried out the power consumption measurement of the STM32F407VG micro-controller with oscilloscope. The results are presented on Fig. 4, 5 and 6, where the horizontal axis represents the elapsed time in microseconds and the vertical axis represents the measured power consumption in millivolts. The micro-contoller board was prepared for the measurement by inserting a small resistor (10 ohm) in series with the power (see Fig. 3, 3V power source VDD near jumper JP1). Unfortunately, the resistance was too high and the microprocessor would not start. Therefore, we switched the resistor with a different one with a resistance of 2.2 ohm. This proved to be useful for our purposes, because not only the micro-processor would start, but we were able to correctly measure the power consumption. To get better results, we disabled the STM interruptions used in communication with the serial port, to eliminate the possible noise caused by these interruptions. Of course, to further improve the measuring conditions, one could carry out the measurements in a dedicated shielded environment, etc.

At first, we measured the power consumption of a modified rotor clocking––we forced the rotors to clock once, twice, three and four times and measured the power trace in each case by using a suitably placed trigger signal. In each case we clocked the rotors twice with the same number of clockings (1, 2, 3 and 4) for two independent rotor sets.

The vertical lines on Fig. 4 represent the trigger signal measured by the oscilloscope. The measured power trace can be seen between the trigger signals. It can be seen directly from the measurement that the power trace differs for each number of clockings. For example, first two parts of the trace on Fig. 4 represent the consumption of the clocking of four rotors (each part represents one rotor set). Next two parts of the trace represent the consumption of the clocking of three rotors, etc. We can therefore distinguish different number of clockings by simply checking the power trace shape visually and determine the time of each individual clocking.

To further demonstrate the SPA of the implementation, on Fig. 5 we present the power trace of a longer period of time. To remove the possibility that the power trace is influenced by the trigger signal, we measured the consumption again, this time without the trigger.

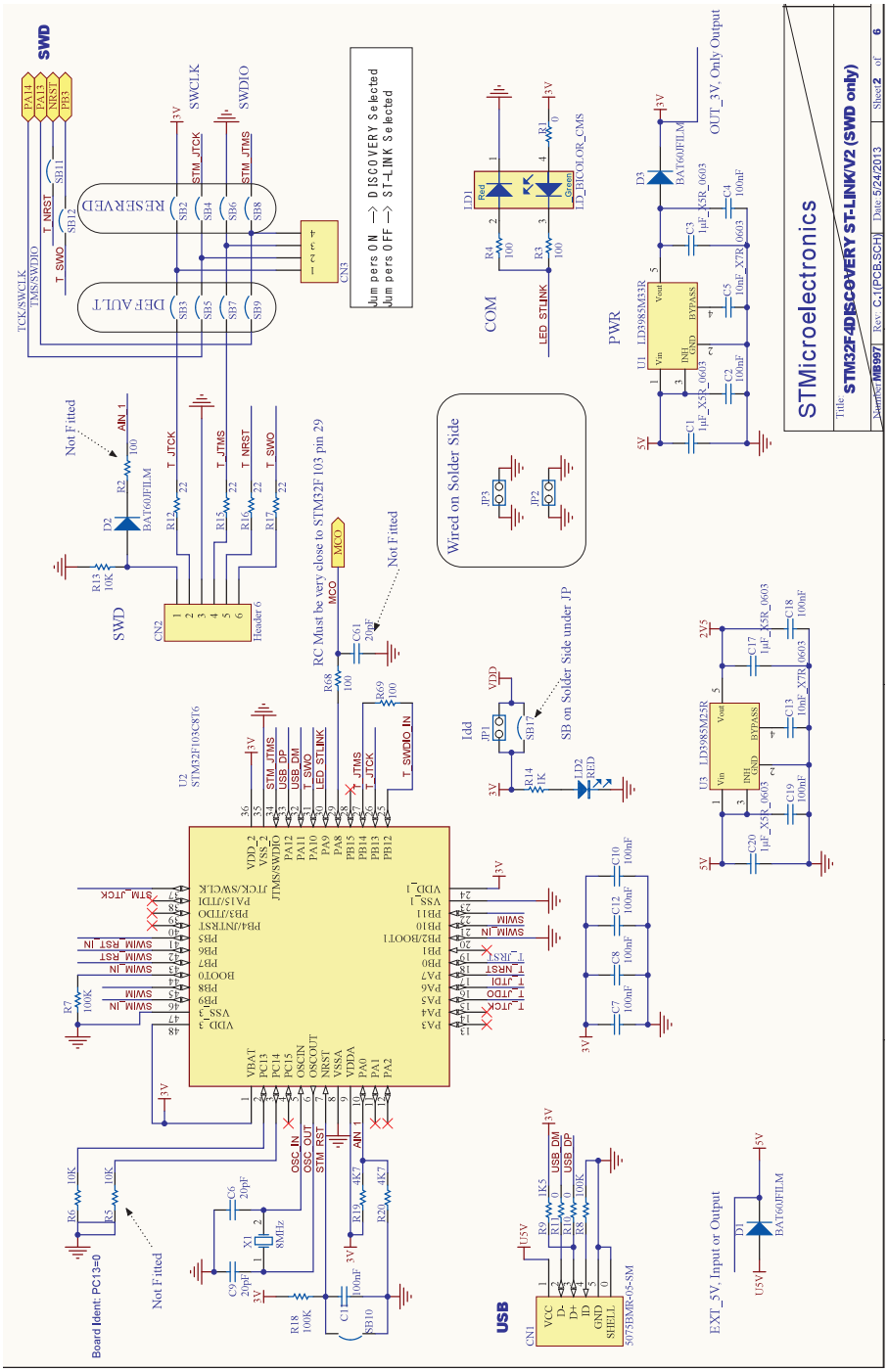

FIGURE 3. STM32F407VG schematics.

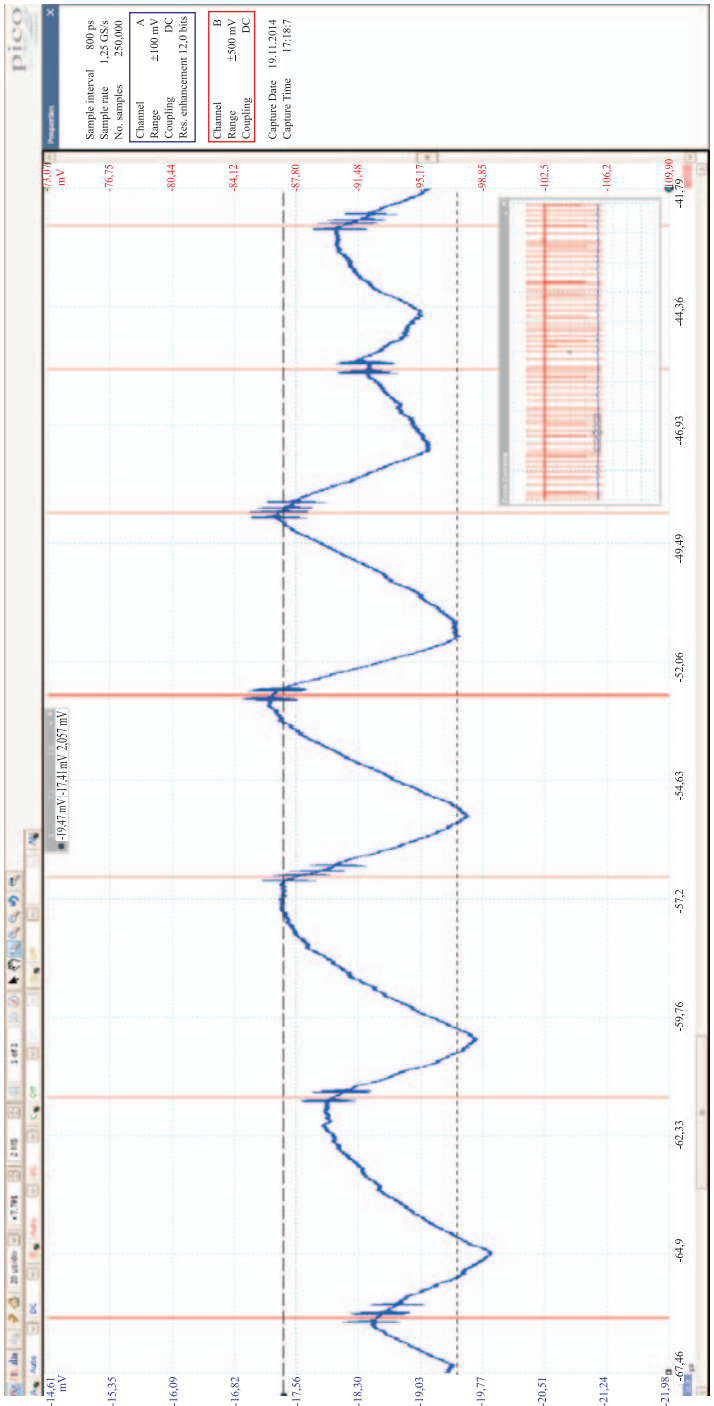

FIGURE 4. Power trace with trigger signal.

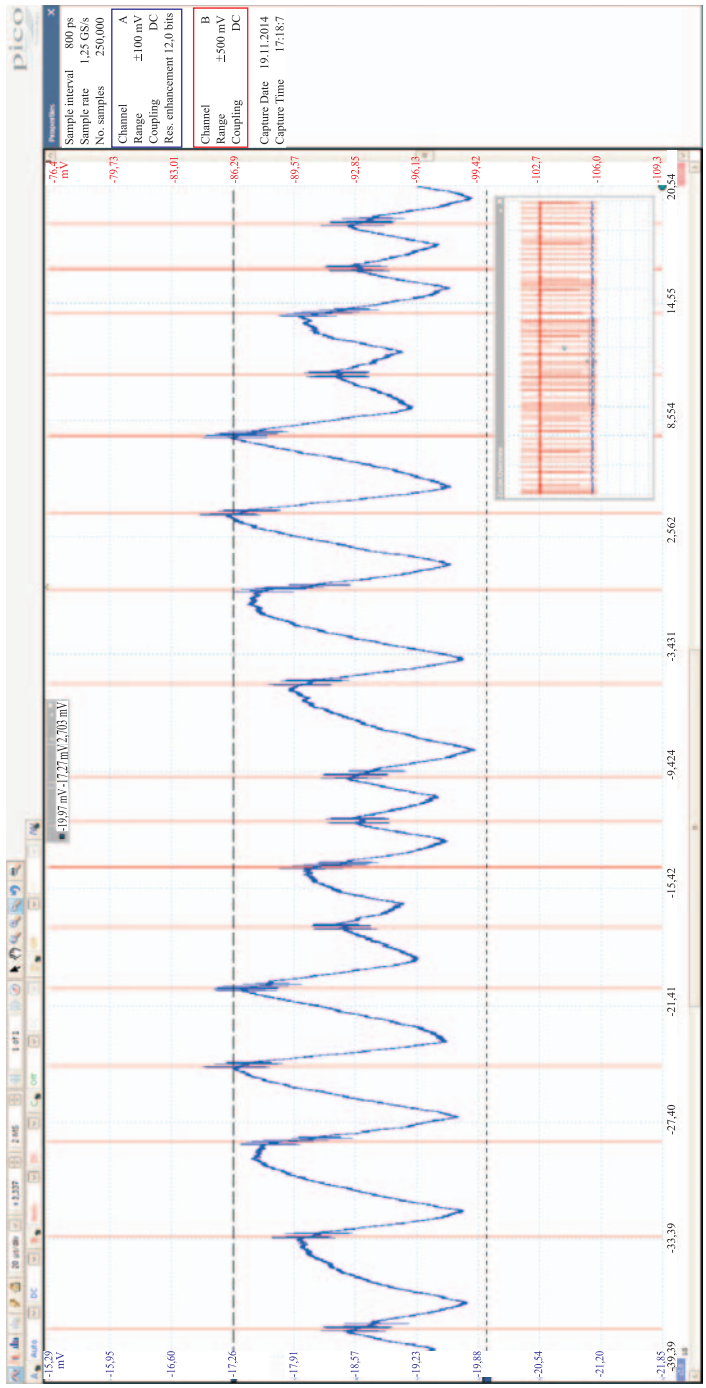

FIGURE 5. Power trace with trigger signal.

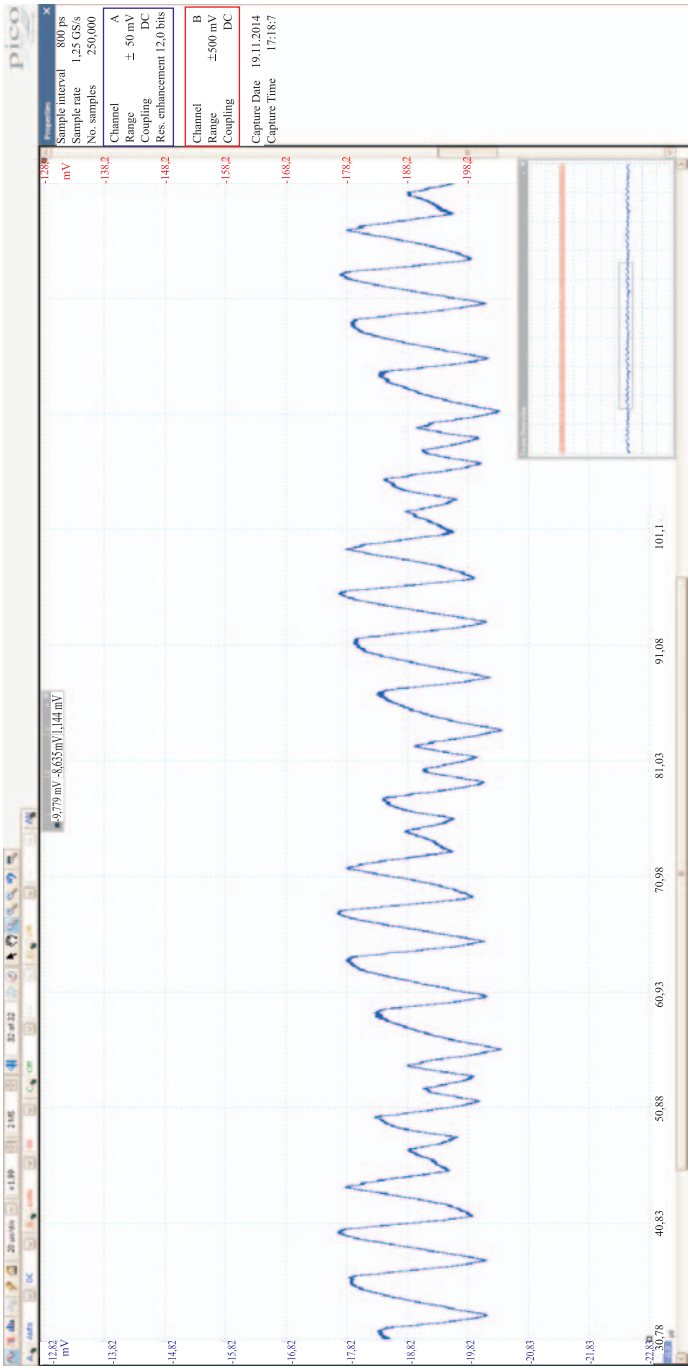

FIGURE 6. Power trace without trigger signal.

Fig. 6 shows that the trigger signal does not influence the power trace significantly and hence our measurements are correct even with the trigger signal.

As we noted in Fig. 2 (Section 2.3), part of the cipher key consists of 8 circular shift registers ($V1 \dots V10$, registers $V9$ and $V2$ are left from the design). These shift registers represent the presence or the absence of the blocking pins on the corresponding rotors (detailed described in [5]). These registers are divided into two parts based on the clocking direction of the rotors (the registers $V1, V3, V5, V7$ and $V4, V6, V8, V10$). Using the SPA, we can measure the power consumption of both parts separately. One shift register from both parts ($V1$ and $V10$) is shifted in each computation cycle, the rest of the registers are shifted according to the previous shift register value on a specific position (described in [5]). By observing the power consumption (power trace) of clocking on both separated clocking parts we can guess the number of performed circular shift register shifts (also the changes of corresponding counter λ). For example by observing the clocking of registers $V1, V3, V5$ and $V7$ we can estimate the corresponding value of 1 bit at least on one - and at most on all four registers. If one shift is performed, we can estimate one bit of the $V1$ register on a specific position (the rightmost bit in our implementation). If two shifts are performed, we can estimate that the pin is present (the rightmost bit is 1) on the $V1$ and a pin is absent on the $V3$ (the rightmost bit is 0). If three shifts are performed, guessed bits on $V1$ is 1, on $V3$ is 1 and on the $V5$ is 0. In this manner we reconstruct the structure of the shift registers and the part of the cipher key by observing the power trace and guessing the number of performed shifts.

4. Summary

In this article, we presented two implementations of the stream cipher based on Fialka M-125 on two different platforms - STM32F407VG and EBV SoCrates Evaluation Board. We evaluated the performance of all versions of the cipher on both devices and compared it with the original results. We also carried out a simple power analysis of the 8-bit version of the cipher implemented on the STM device. Our analysis suggests that it is possible to determine a part of the key - the number of clocked rotors - by simply measuring the device power consumption and then by observing parts of the power trace, since it is possible to directly match a certain shape of a power trace to a certain number of clocked rotors and also read the aggregate time of the rotor clocking directly from the power trace.

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

REFERENCES

[1] *Official Piko Technology webpage,* www.picotech.com
[2] *Official SoCrates webpage (EBV SoCrates evaluation board),* www.rockerboards.org
[3] *Official STM webpage (STM32F407VG),* www.st.com.
[4] *STMP Systick interrupt.* http://electronics-homemade.com/STM32F4-LED-Toggle-Systick.html.
[5] ANTAL, E.—HROMADA, V.: *A new stream cipher based on Fialka M-125*, Tatra Mt. Math. Publ. **57** (2013), 101–118.
[6] ANTAL, E.—ZAJAC, P.: *Key space and period of Fialka M-125 cipher machine*, Cryptologia 2013 (accepted).
[7] ENGELS, D.—FAN, X.—GONG, G.—HU, H.—SMITH, E. M.: *Hummingbird: ultra-lightweight cryptography for resource-constrained devices*, in: 1st Internat. Workshop on Lightweight Cryptography for Resource-Constrained Devices, Tenerife, Canary Islands, Spain, 2010 (R. Sion et al., eds.), Lecture Notes in Comput. Sci., Vol. 6054, Springer, Berlin, 2010, pp. 3–18.
[8] ZAJAC, P.—GALLO, O.: *Simple power analysis demonstration on Arduino platform*, EE, časopis pre elektrotechniku a energetiku, informačné a komunikačné technológie **19** (2013), 15–19.
[9] KOCHER, P.—JAFFE, J.—JUN, B.: *Differential power analysis*, in: Advances in Cryptology—CRYPTO '99, 19th Annual Internat. Cryptology Conf. Santa Barbara, CA, USA, 1999 (M. Wiener, ed.), Lecture Notes in Comput. Sci., Vol. 1666, Springer, Berlin, 1999, pp. 388–397.
[10] PERERA, T.—HAMER, D.: *General introduction: Russian cold war era M-125 and M-125-3MN Fialka cipher machines*, 2005, http://enigmamuseum.com/mfialka.htm.
[11] QUISQUATER, J. J.: *Side channel attacks,* www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf.
[12] REUVERS, P.—SIMONS, M.: *Fialka M-125: Detailed Description of the Russian Fialka Cipher Machines*. PAHJ Reuvers & MJH Simons, 2009.
[13] ZAJAC, P.—JÓKAY, M.: *On S-boxes with low multiplicative complexity*, in: Tatracrypt 2012, 12th Central European Conference on Cryptology. Smolenice, Slovak Republic, Slovak Academy of Sciences, July, 2012, pp. 47–48.

Received November 24, 2014

Institute of Computer Science and Mathematics
Slovak University of Technology
Ilkovicova 3
SK–812-19 Bratislava
SLOVAKIA

E-mail: eugen.antal@stuba.sk
viliam.hromada@stuba.sk