TECHNICAL TRANSACTIONS | CZASOPISMO TECHNICZNE
AUTOMATIC CONTROL | AUTOMATYKA
3-AC/2013

ZBIGNIEW KOKOSIŃSKI*, SŁAWOMIR WÓJCIK**

A COMPARISON OF SW/HW IMPLEMENTATIONS OF STREAM CIPHER ENCODERS

PORÓWNANIE IMPLEMENTACJI PROGRAMOWYCH I SPRZĘTOWYCH SZYFRATORÓW STRUMIENIOWYCH

Abstract

In this paper, a new method of stream encoding and decoding is presented. It is developed on the basis of a derangement generator. Stream cipher D has been compared with other stream ciphers – E0, W7 and Phelix. Encoding and decoding algorithms have been implemented in C++ and VHDL programming languages. FPGA synthesis data has been reported for Spartan 3E and Virtex 4 devices from Xilinx. The hardware solution has been tested on the Digilent Nexys 2 500K board. Subsequently, comparative studies have been conducted for software and hardware coders, taking into account average coding time and average throughput for 16 input data files of different sizes. Conclusions resulting from the research are derived.

Keywords: *stream cipher, coder, decoder, coder throughput, FPGA*

Streszczenie

W artykule przedstawiono nową metodę strumieniowego szyfrowania i deszyfrowania danych w oparciu o generator nieporządków. Szyfr strumieniowy D został porównany ze znanymi szyframi strumieniowymi E0, W7 i Phelix. Algorytmy kodowania i dekodowania zaimplementowano w językach programowania C++ oraz VHDL. Podano dane dotyczące syntezy urządzeń sprzętowych w układach programowalnych FPGA typu Spartan 3E oraz Virtex 4 firmy Xilinx. Rozwiązania sprzętowe zostały przetestowane na płycie Digilent Nexys 2 500K. W badaniach porównawczych zbudowanych szyfratorów programowych i sprzętowych uwzględniono średni czas szyfrowania oraz średnią przepustowość dla 16 plików danych o różnych rozmiarach. Sformułowano wnioski z przeprowadzonych badań.

Słowa kluczowe: *szyfr strumieniowy, szyfrator, deszyfrator, przepustowość szyfratora, FPGA*

* Ph.D. Zbigniew Kokosiński, email: zk@pk.edu.pl, Department of Automatic Control and Information Technology, Faculty of Electrical and Computer Engineering, Cracow University of Technology.

** M.Sc. Sławomir Wójcik, Faculty of Electrical and Computer Engineering, Cracow University of Technology (currently with Ericpol, Cracow).

1. Introduction

Data encryption and security is one of the key issues in modern computer and telecommunication systems [4]. A large number of cryptographic systems have been developed having different characteristics. The most popular systems like DES, AES are supported either by government agencies or telecommunication companies while many others are developed and supported by independent private enterprises.

Encryption algorithms usually belongs to one of the two groups: they use block ciphers or stream ciphers. While software encoders are dominating the market, there are also many hardware implementations. Depending on the encryption method, usually one of the two implementations mentioned above is more efficient, e. g. software implementation of LFSR-based encoders is slower than dedicated hardware solution.

Several research results relating to FPGA implementations of stream ciphers were described in [14]. The Stream cipher VMPC (Variably Modified Permutation Composition), was proposed and developed in 2004 by Bartosz Żółtak [18, 19] on the basis of one-way function. It is easy to implement both in software and in hardware. Recent research papers relating to VMPC encryption technology as well as the present software version of VMPCrypt 4 are available at [19]. In 2009, VMPC stream cipher was successfully implemented and tested in FPGA [6].

In this paper, four different stream ciphers, implemented both in hardware and software, are compared. The selected ciphers are:

- E0 – used in wireless data transmission via Bluetooth interface [7];
- W7 – a one-time candidate for a successor of A5 in mobile GSM technology [15];
- Phelix – dedicated for 32-bit platforms, combines encryption with MAC (Message Authentication Code) [16];
- D – a new method, developed recently on the basis of the derangement generation [12].

The software and hardware encoders are characterized by data processing time and throughput computed experimentally.

In the next section, the concept and basic properties of set derangements are explained. In section 3, the D stream cipher is introduced. Section 4 contains a short description of software implementations. FPGA implementations in Xilinx Spartan and Virtex devices are described in section 5. Section 6 brings a comparison of software and hardware encoders. In the last section, some conclusions and remarks are added.

2. Derangements

The D stream cipher introduced in this paper is developed on the basis of a generation of a specific class of n-permutations with no constant points (no 1–cycles) called derangements. Combinatorial properties of derangements are described in depth in [5, 9]. Several methods for the generation of all set derangements sequentially or in a parallel linear array model are published in the literature [1–3, 8, 13].

The representation of partial derangements is derived from a representation of permutations by iterative decomposition of symmetric permutation group S_n into cosets [12]. Some particular properties of derangements are also established.

Now we introduce representations of the considered combinatorial objects by means of integer sequences (codewords) defined as choice functions of indexed families of sets.

Let $\langle A_i \rangle_{i \in I}$ denote an indexed family of sets $A_i = A$, where: $A = \{1, ..., n\}$, $I = \{1, ..., n\}$, $1 \le n$. Any mapping f which 'chooses' one element from each set $A_1, ..., A_n$ is called a choice function of the family $\langle A_i \rangle_{i \in I}$. If, for every $i \ne j$, a suplementary condition: $a_i \ne a_j$, for $a_i \in A_i$ and $a_j \in A_j$, is satisfied then any choice function $\alpha = \langle a_i \rangle_{i \in I}$ that belongs to the indexed family $\langle A_i \rangle_{i \in I}$ is called n-permutation of the set A.

Let us now define permutations with forbidden positions and derangements [12]. A permutation π of n-element set $A = \{1, ..., n\}$ with a forbidden position i is the sequence $\langle \pi(1), \pi(2), ..., \pi(n) \rangle$, where $\pi(i) \ne i$, for some $1 \le i \le n$.

Let $\langle P_i \rangle_{i \in I}$ be an indexed family of sets $P_i \subseteq A$, where $P_i = \{1, ..., i\}$, $1 \le i \le n - 1$, and $P_n = P_{n-1}$. Any choice function $\alpha = \langle p_i \rangle_{i \in I}$, that belongs to Carthesian product $\times_{i \in I} P_i$ represents a permutation of A with a forbidden position i if and only if:

$$(p(i) \ne i) \vee [(p(i) = i) \Rightarrow \exists j : (i < j \le n) \wedge p(j) = i] \tag{1}$$

Any n-permutation with n forbidden positions $i \ne \pi(i)$, $1 \le i \le n$, is called a derangement $\delta(n)$ with the forbidden set A.

Let $\langle D_i \rangle_{i \in I}$ be an indexed family of sets $D_i \subseteq A$, where $D_i = P_i$, $1 \le i < n$, and $F = \{f_1, f_2, ..., f_k\}$, $F \subseteq I = \{1, ..., n\}$, $0 \le k \le n$, be the forbidden set. Any choice function $\delta(n, k) = \langle d_i \rangle_{i \in I}$, that belongs to Carthesian product $\times_{i \in I} D_i$ represents a derangement of A if and only if:

$$\forall d(i) \in F : (d(i) \ne i) \vee [(d(i) = i) \Rightarrow \exists j : (i < j \le n) \wedge d(j) = i] \tag{2}$$

The new cipher D belongs to a group of derangement ciphers, working on bits or strings. There exist $\frac{n!}{e}$ different derangements of n-element set. The generation algorithms for derangements can be found in [1–3, 8, 12, 13]. For $n = 32$, $D(32) \approx \frac{32!}{e} \approx 10^{35}$.

3. *D* stream cipher

The new cipher D, proposed by the first author of the article, belongs to a group of derangements ciphers, working on bits or strings. However, encoding scheme on the basis of derangement operation can not provide nontrivial encodings of specific strings like 0 or 1 sequences. Therefore, the generated derangements are processed further with the help of a key stream S_i generated by a linear feedback shift register (LSFR) – see Fig. 1.

The hardware-oriented algorithm for generating set derangements is developed in the parallel counter model augmented by a triangular permutation network and is a modification of the permutation generation algorithm [11].

The triangular permutation network is built of two-state cells (2-permuters) [10, 11]. Each cell requires a separate control signal. The permutation network can perform $n - 1$ transpositions ($P(i)$, $P(k)$), i.e. can produce any n-permutation of its inputs on outputs.

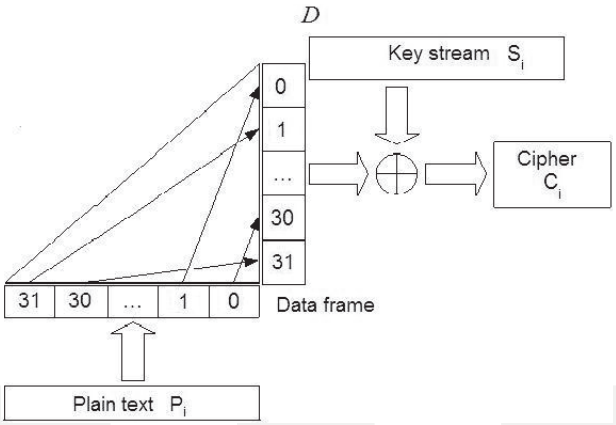

Fig. 1. The idea of D encoding scheme for $n = 32$ with the permutation network and a key stream

The control sequences are produced in $O(1)$ average time per generated object. The output sequences are then obtained from the control sequences in $O(n)$ time.

The control circuit is organized in the following way [17]. With every i-th column of the triangular network ($1 \le i \le n$) the i-th ring counter is associated with the initial state from the '1–out–of–i' code. All column counters form the parallel counter with $n!$ different states. Clock enable signal for the i-th ring counter is a product of carry signals (overflows) from all ring counters preceding it. The state of the permutation network is controlled by $n - 1$ synchronous up-down counters (UDC), where UDC(i) counts mod $(i + 1)$ depending on the cipher bit C(i). For C(i) = 1, UDC(i) counts up, otherwise UDC(i) counts down. We assume, that k = UDC(i).

The asynchronous setup of each ring counter and global reset for all ring counters is provided. If the j-th bit of the i-th ring counter $b_i^j = 1$, for $1 \le j \le (i - 1)$, then in the i-th column of the network only one cell denoted by C[i, j] is activated to perform the corresponding transposition τ_i^j. If $b_i^i = 1$, $1 \le i \le n$, then all cells in the i-th column are in the 'identity' state.

After setting the initial state of the network, the control circuit generates consecutive states of network in a constant time (one clock period) and the permutation network generates subsequent configurations representing permutations. In order to recognize a derangement permutation, an additional logic based on formula (2) is needed [12], and, on average, $(e - 1)$ extra clock periods are required to find such a permutation.

Valid n-derangements are detected by a logic function V checking if the condition given in (2) is satisfied:

$$V = \prod_{i=1}^{n-1} FV(i)v_i = \prod_{i=1}^{n-1} FV(i)\left(\overline{b_i^i} + \sum_{j=i+1}^{n} b_j^i\right) \tag{3}$$

where:

FV – a binary forbidden set vector: $FV(i) = 1$ iff position i is forbidden, otherwise $FV(i) = 0$;

v_i – the function detecting if the condition (5) for the forbidden position i is satisfied; in fact, for technical reasons, v_i should be rewritten in the form:

$$v_i = \left(\overline{b_i^i} + \left(\left(...(b_{i+1}^i + b_{i+2}^i) + ...\right) + b_{n-1}^i\right) + b_n^i\right) \tag{4}$$

The above logic functions can be computed in $O(n)$ time which matches the network propagation delay. Because the size of the network is limited and the constant factor hidden in the function $O(n)$ is very low, for most applications we may assume that consecutive network configurations are generated in constant time.

The hardware complexity of the generator is $O(n^2)$, and the network propagation delay is $O(n)$. For practical applications, the networks size is limited and the propagation delay can be considered constant.

4. Software implementations

An application in C++ has been developed in MS Visual Studio 2005 for MS Windows platforms with MS .NET Framework 2.0 installed. The compiler has been set for maximum speed.

The user interface provides selection of paths to access input files and write output files into a given destination. A secret key section is common for all algorithms. It is possible to generate a key, read/write from/to a file, input the key via a keyboard. A key length varies from 8 to 256. Selection of the cipher tab provides setting additional parameters for that cipher. It is possible to generate public parameter (nonce) and provide the MAC-tag for the message authentication (MAC – Message Authentication Code).

After setting all necessary parameters the type of operation (encryption or decryption) is selected. The log window allows the user to trace the consecutive steps of program setting and execution. The progress of data processing is visualized. The main window of the application is shown in Fig. 2.

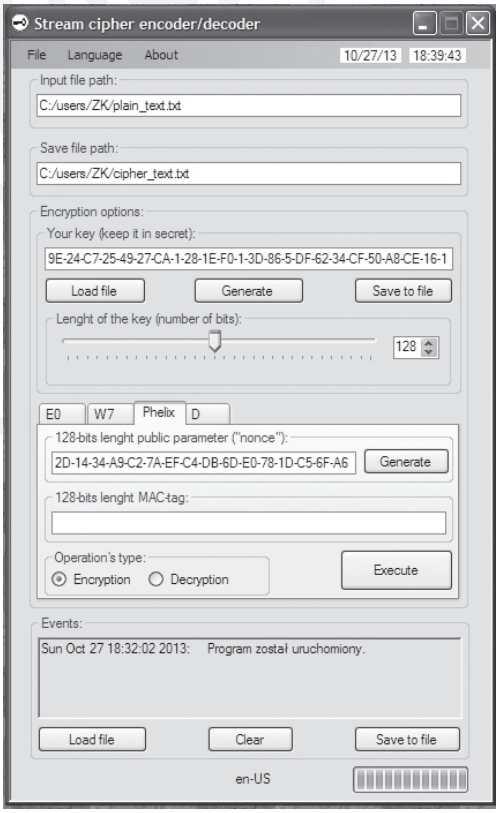

Fig. 2. GUI of the stream cipher encoder/decoder

5. Hardware implementations

The stream cipher encoder/decoder has been implemented in VHDL in Xilinx WebPack ISE v.8.2.03i. The destination device Spartan 3E-500 FG320 on Digilent Nexys2 development board has been used. The FPGA circuit has got 500 000 equivalent gates. A USB port is provided for the power supply and communication between the PC and FPGA. Embedded SDRAM has a capacity of 16MB. A quartz oscillator runs with 50 MHz frequency.

Data transmission between PC and FPGA memory is always 8-bit, but data processing within FPGA is either 8-bit (E0 and W7 ciphers) or 32-bit in buffered mode (Phelix and D ciphers).

During the synthesis phase, we have used options Optimization goal – Speed and Optimization Effort – Normal. In the implementation phase, the option Optimization Strategy – Area has been used. The collected data from the synthesis reports are presented in Table 1.

E0 encoder employs the simplest architecture, while D encoder employs the most complex architecture. The complexity of the structure also has influence on the minimal clock period. Normalization of the clock period for all encoders at 20 [ns] has become possible by means of the flag system that was introduced for synchronization of internal transitions in all finite state machines (FSM) that control the work of encoders.

As we will see in the next section, the clock frequency does not necessarily influence the device's throughput: f. i. W7 and E0 encoders, with the highest and the lowest throughput respectively, have very similar minimal clock periods.

Table 1

Synthesis data for Spartan 3E-500 device

Encoder	Input IOB	Output IOB	Bi-direct. IOB	Number of LUTs	Number of slices	Number of Gates	Minimal clock period [ns]
E0	19	56	24	1555	963	24 711	11.681
W7	19	56	24	3186	1685	38 769	11.758
Phelix	19	56	24	5721	3088	65 528	27.729
D	19	56	24	7900	4151	63 188	116.653
Available resources	**232** (all types)			**9312**	**4656**	**500 000**	–

Table 2

Synthesis data for Virtex4 family

Encoder	Number of LUTs	Number of slices	Number of gates	Minimal clock period [ns]
E0	1802	1069	19 855	4.875
W7	3282	1808	33 488	4.684
Phelix	5683	3054	58 579	13.224
D	7826	4088	55 177	64.719
Available resources	**12 288**	**6144**	–	–

In Table 2, synthesis data for Virtex 4 family of Xilinx FPGAs are shown for comparison with the Spartan 3 device. There are several differences between implementations resulting from architectural differences. The most interesting result is a significantly higher possible speed of the Virtex 4 implementations.

6. Comparison of software and hardware encoders

In the conducted experiments, the application presented in section 3 was tested on a 2.0 GHz computer with 3.0 GB RAM, running under 32-bit Windows Vista OS. Speed of encoding was measured for 16 data files within the range of 1-16 MB which corresponds with the maximum memory size 16 MB in the hardware encoder implementations on Spartan 3E FPGA. The linear growth of encoding time with file size is observed in Fig. 3. The average throughput of 115,72 Mbit/s for Phelix, 27,78 Mbit/s for D, 11,57 Mbit/s for E0 and 7,70 Mbit/s for W7 was obtained.

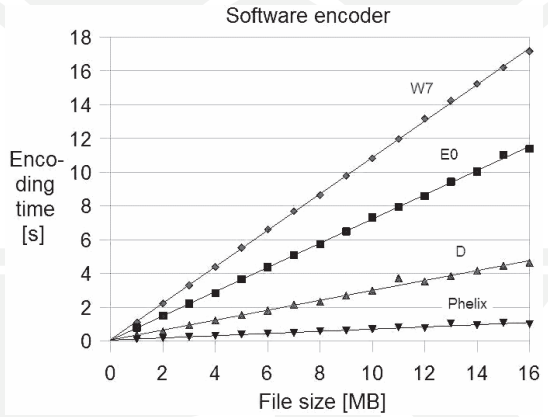

Fig. 3. Comparison of encoding times of the software encoders (W7, E0, D and Phelix)

In the conducted experiments, the hardware encoders presented in section 3 were tested on Xilinx Spartan 3E FPGA. Speed of encoding was measured for 16 data files within the range of 1-16 MB. The linear growth of encoding time with file size is observed in Fig. 4. The average throughput of 16,26 Mbit/s for W7, 12,69 Mbit/s for D, 12,67 Mbit/s for Phelix and 10,20 Mbit/s for E0 was obtained.

Simultaneously, W7 has got the lowest throughput among software encoders. It justifies a conclusion that encoders composed on LFSR are devoted mostly to hardware implementations. E0 encoder, which is also built on LFSR, delivers another data. Its hardware version takes the last place and its software version the 3rd place. Its low hardware throughput is due to an inefficient key stream generator, which produces only one key bit in one clock cycle. Implementations of encoders processing 32-bit characters (Phelix and D) are definitely the fastest among the tested software versions. The Phelix encoder outperforms all other software encoders. Also its hardware version has the high throughput.

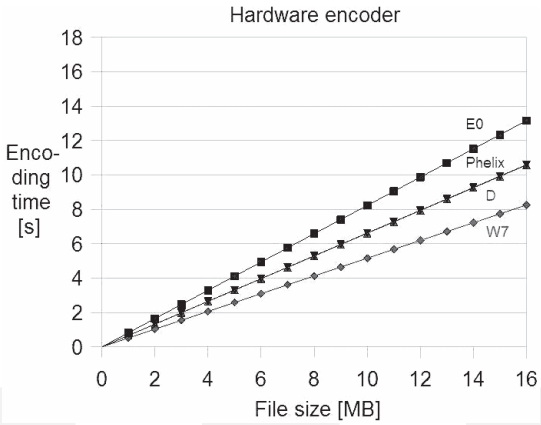

Fig. 4. Comparison of encoding times of the hardware encoders (E0, Phelix, D and W7)

The comparison of hardware and software encoder throughputs is depicted in Fig. 5. The most efficient hardware implementation of the encoder algorithm is that for W7. Its FPGA implementation reveals higher throughput than the software version, while the clock frequency of Nexys 2 board is 40 times lower then that of the processor.

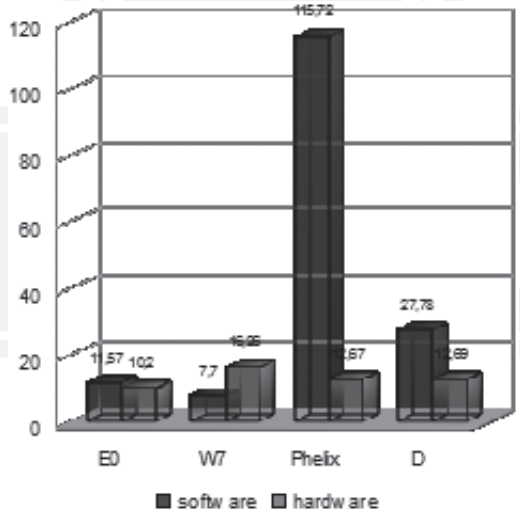

Fig. 5. Comparison of SW/HW encoder throughputs (E0, W7, Phelix, D)

7. Concluding remarks

All four encoding schemes were successfully implemented and tested in software and hardware. The software implementation of the new D stream cipher implemented by the authors is much better than E0 and W7 in terms of the average throughput, but the winning

algorithm in this category is Phelix. The differences between hardware versions of encoders are less visible. The fastest hardware encoding provides W7, while D and Phelix encoders occupy the second place. The slowest one in this category is the E0 encoder.

Properties of the D stream cipher were not verified via cryptanalysis so far. It is expected that selection of key scheduling scheme shall play an important role. In order to increase robustness of the proposed method on cryptanalytic attacks, application of derangements D(n) for n different then the power of two might be considered.

It is possible to develop encoding schemes similar to D encoder on the basis of other derangement generation algorithms [1, 3, 9, 13]. Alternative hardware encoder may be constructed with the parallel derangement generator in the linear array model [2].

The idea of using derangements instead of classical permutations may lead to the modification of VMPC one-way function [18, 19] into the VMDC (Variable Modified Derangement Composition) one-way function. In this way, the VMPC encryption algorithm would become the VMDC encryption algorithm. However, many details of the former scheme, like the key scheduling algorithm, should be adapted to the new cipher. It would be interesting to compare cryptographic properties of both variants.

References

[1] Akl S.G., *A new algorithm for generating derangements*, BIT 20, 1980, 2-7

[2] Akl S.G., Calvert J.M., Stojmenovič I., *Systolic generation of derangements*, Proc. Int. Workshop on Algorithms and Parallel VLSI Architectures II, Elsevier, 1992, 59-70.

[3] Baril J.-L., Vajnovszki V., *Gray code for derangements*, Discrete Applied Mathematics, 140, 2004, 207-221.

[4] Denning D.E.R., *Cryptography and data security*, Addison-Wesley, 1982.

[5] Erickson M.J., *Introduction to Combinatorics*, Wiley Interscience, 83, 1996, 119-120.

[6] Gajos T., *A hardware implementation of VMPC stream cipher encoder and decoder in programmable logic*, M.S. Thesis, Cracow University of Technology, Cracow 2009 (in Polish).

[7] Gehrman C., Persson J., Smeets B., *Bluetooth Security*, Artech House, 2004.

[8] Gupta P., Bhattacharjee G.P., *A parallel derangement generation algorithm*, BIT 29, 1989, 14-22.

[9] Graham R.L., Knuth D.E., Patashnik O., *Concrete Mathematics*, 2nd ed., Addison-Wesley Publishing Company, 1994, 194-195.

[10] Kokosiński Z., *On generation of permutations through decomposition of symmetric groups into cosets*, BIT 30, 1990, 583-591 (available at www.pk.edu.pl/~zk/pubs.html).

[11] Kokosiński Z., *Circuits generating combinatorial configurations for sequential and parallel computer systems*, Cracow University of Technology, Cracow, Poland, Monograph, 160, 1993.

[12] Kokosiński Z., *On parallel generation of partial derangements, derangements and permutations*, Proc. Int. Conf. Parallel Processing and Applied Mathematics PPAM'2007, Gdansk, Poland, Lecture Notes in Computer Science, Vol. 4967, 2008, 219-228 (available at www.pk.edu.pl/~zk/pubs.html).

[13] Korsh J.F., LaFolette P., *Constant time generation of derangements*, Information Processing Letters, Vol. 90, 2004, 181-186.
[14] Rogawski M., *Stream ciphers in FPGA structures*, IX Konferencja Zastosowań Kryptografii, Enigma'2005, Warszawa 2005 (in Polish).
[15] Thomas S., Deven A., Berson T., Gong G., *The W7 stream cipher Algorithm*, Internet draft, April 2002.
[16] Whiting D., Schneier B., Lucks S., Muller F., *Phelix fast encryption and authentication in a single cryptographic primitive*, ECRYPT Stream Cipher Project Report 2005/027, 2005.
[17] Wójcik S., *A hardware implementation of stream encoder and decoder of the data in programmable Spartan 3 FPGA devices*, M.Sc. Thesis, Cracow University of Technology, Cracow 2010 (in Polish).
[18] Żółtak B., *VMPC one-way function and stream cipher*, Proc. Int. Conf. Fast Software Encryption FSE'2004, Delhi, February 2004.
[19] VMPCrypt: http://www.vmpcrypt.pl/