On Software Implementation of Fast DDP-based Ciphers

Nikolay A. Moldovyan[1], Peter A. Moldovyanu[1], and Douglas H. Summerville[2]
(Corresponding author: Nikolay A. Moldovyan)

Specialized Center of Program Systems "SPECTR",[1]
Kantemirovskaya, 10, St.Petersburg 197342, Russia. (Email: info@cobra.ru)
Binghamton University, [2]
PO Box 6000, Binghamton NY, 607-777-2942. (Email: dsummer@binghamton.edu)

(Received Sept. 17, 2005; revised and accepted Oct. 16 & Nov. 23, 2005)

Abstract

Data-dependent (DD) permutations (DDP) are discussed as a cryptographic primitive for the design of fast hardware, firmware, and software encryption systems. DDP can be performed with so called controlled permutation boxes (CPB) which are fast while implemented in cheap hardware. The latter defines the efficiency of the embedding of CPB in microcontrollers and microprocessors when adding a new instruction that allows one to perform DDP. Different types of fast software and firmware encryption algorithms combining DDP with fast arithmetic operations are described. The proposed ciphers are free of the key preprocessing that provides high performance in the case of frequent change of keys. Presented results show the ciphers are secure against differential analysis. Other attacks are also considered.

Keywords: Block cipher, cryptanalysis, data-dependent permutations, differential characteristics, fast encryption, software implementation

1 Introduction

Data-dependent (DD) permutations (DDP) suites well to the design of fast and secure block ciphers [12]. The DDP can be performed with so called controlled permutation (CP) boxes (CPB) of different orders [12, 17], which are fast while implemented in cheap hardware. In certain sense DDP represent generalization of the DD rotations (DDR) used earlier in cryptosystems RC5 [14], RC6 [15],and MARS [4]. DDP have significantly more different modifications as compared with DDR (from 2^n to $n!$ versus n, where n is the size of the transformed data subblock). CPB can be easily embedded in microcontrollers and general purpose CPUs and used while designing fast firmware and software encryption systems. The hardware-oriented cryptosystem SPECTR-H64 [7] is an example of 64-bit ciphers extensively using DDP. In one round of SPECTR-H64 two mutually inverse CPB operations of the first order are performed on the right data subblock providing efficient hardware implementation. That twelve-round cipher uses also several other sound structural features which can be used in the design of new cryptosystems.

In present paper we propose a variant of the fast CPB instruction and consider the design and security of the block DDP-based ciphers oriented to firmware and software implementation.

The paper is organized in the following way: In the second section we characterize DDP performed with CPB presenting detailed structure of the symmetric CPB used in the designed cryptalgorithms. Use of the symmetric CPB simplifies realization of the design criteria corresponding to selection of the current modifications of DDP. A variant of the switchable CPB $\mathbf{P}^{(e)}_{32;32}$ suitable for embedding in microcontrollers and CPUs is described. Addition of the CP instruction in the standard set of the CPU commands require only about 1800 additional transistors to be formed when manufacturing CPU. In Section 3 we present two 64-bit block ciphers appropriate for firmware implementation and a software-oriented 128-bit cipher. These fast firmware and software-oriented ciphers are based on DDP performed with $\mathbf{P}^{(e)}_{32;32}$-instruction and simple arithmetic operations. In Section 4 we consider differential properties of the used DDP as well as security estimation of the proposed ciphers.

Notation: Let $\{0,1\}^s$ denote the set of all s-bit binary vectors $X = (x_1, ..., x_s)$. Let $/X/$ denote numerical value (or simply value) of the vector $(x_1, ..., x_s)$ and $/X/ = \sum_{i=1}^{s} x_i 2^{i-1}$.

Let $X \oplus Y$ denote the bit-wise XOR operation performed on X and Y : $X, Y \in \{0,1\}^s$.

Let $Y = X^{\lll k}$ denote rotation operation to the least significant bits of the word X by k bits.

Let $X = (X_l, X_h)$ denote concatenation of binary vec-

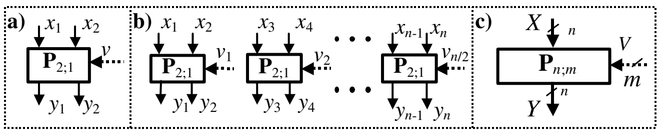

Figure 1: Notation of the CP boxes $\mathbf{P}_{2;1}$ (a) and $\mathbf{P}_{n;m}$ (c) and structure of the active layer (b)

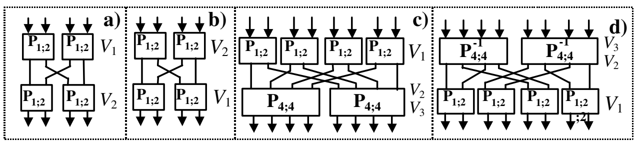

Figure 2: Structure of the first-order boxes $\mathbf{P}_{4;4}$ (a), $\mathbf{P}^{-1}_{4;4}$ (b), $\mathbf{P}_{8;12}$ (c), and $\mathbf{P}^{-1}_{8;12}$ (d)

tors $X_l, X_h \in \{0,1\}^{n/2}$. Then for $X \in \{0,1\}^n$ we can write $X_l = (x_1, ..., x_{n/2})$ and $X_h = (x_{n/2+1}, ..., x_n)$.

Let "$+_n$" ("$-_n$") denote addition (subtraction) modulo 2^n. Let $Y = X +_n A$ ($Y = X -_n A$) denote operation on X and A output of which is vector Y that $/Y/ = /X/ +_n /A/$ ($/Y/ = /X/ -_n /A/$).

2 Design of the Controlled Permutations

DDP can be easy performed with well-known interconnection networks (IN) [1, 5] which were proposed to construct key-dependent permutations [9, 13]. However such use of IN do not effectively thwarts differential cryptanalysis [2, 16]. Different types of IN can be constructed using elementary switching elements $\mathbf{P}_{2;1}$ (Figure 1a) as elementary building blocks performing controlled transposition of two input bits x_1 and x_2. In the general case each $\mathbf{P}_{2;1}$-box is controlled with one bit v and forms two-bit output (y_1, y_2), where $(y_1, y_2) = (x_1, x_2)$, if $v = 0$, and $(y_1, y_2) = (x_2, x_1)$, if $v = 1$. So called layered IN contain several unified active layers (Figure 1b) separated with fixed permutations. A layered IN represents a superposition $\mathbf{L}^{(V_1)} \circ \pi_1 \circ \mathbf{L}^{(V_2)} \circ \pi_2 \circ ... \circ \pi_{S-1} \circ \mathbf{L}^{(V_s)}$, where π_j, $j = 1, 2, ..., s-1$, are fixed permutations defining particular type of IN, $V = V_1, V_2, \cdots, V_s$ is the controlling vector, and $s = 2m/n$ is the number of active layers $\mathbf{L}^{(V_i)}$, $i = 1, 2, ..., s$, each of which contains $n/2$ boxes $\mathbf{P}_{2;1}$. Layered IN are fast and cheap in hardware, therefore they are attractive to be used as CPB performing DDP. In this paper a layered CPB with n-bit input and m-bit control input is denoted as $\mathbf{P}_{n;m}$ (Figure 1c). In present paper dotted lines corresponding to CP boxes indicate the controlling bits.

One can easy construct the layered box $\mathbf{P}^{-1}_{n;m}$ which is inverse of $\mathbf{P}_{n;m}$-box: $\mathbf{P}^{-1}_{n;m} = \mathbf{L}^{(V_s)} \circ \pi^{-1}_{s-1} \circ \mathbf{L}^{(V_{s-1})} \circ \pi^{-1}_{s-2} \circ ... \circ \pi^{-1}_1 \circ \mathbf{L}^{(V_1)}$. In accordance with the structure of the CPB $\mathbf{P}_{n;m}$ and $\mathbf{P}^{-1}_{n;m}$ we shall assume that in CPB denoted as $\mathbf{P}_{n;m}$ the boxes $\mathbf{P}_{2;1}$ are consecutively numbered from left to right *from top to bottom* and in CPB denoted as $\mathbf{P}^{-1}_{n;m}$ the boxes $\mathbf{P}_{2;1}$ are numbered from left to right *from bottom to top.* Thus, the vector V_j controls the j-th active layer in the box $\mathbf{P}_{n;m}$ and the $(s-j+1)$-th layer in $\mathbf{P}^{-1}_{n;m}$. Figure 2 presents examples of the layered CPB. We shall indicate controlling vector as upper index: $\mathbf{P}^{(V)}_{n;m}$. All possible values $/V/$ specify some set of fixed permutations $\{\Pi_0, \Pi_1, ..., \Pi_{2^m-1}\}$. We shall call them modifications of the CPB operation $\mathbf{P}^{(V)}_{n;m}$. The following two definitions we use according to [12].

Definition 1. *The CPB $\mathbf{P}_{n;m}$ and $\mathbf{P}^{-1}_{n;m}$ are mutual inverses, if for all possible values of the vector V the corresponding CP modifications $\Pi_{/V/}$ and $\Pi^{-1}_{/V/}$ are mutual inverses.*

Definition 2. *The CPB $\mathbf{P}_{n;m}$-box is called a CP box of the order h ($1 \le h \le n$), if for arbitrary index set $i_1, i_2, ..., i_h$ and arbitrary index set $j_1, j_2, ..., j_h$ ($i_\alpha \ne i_\beta$ and $j_\alpha \ne j_\beta$ for $\alpha \ne \beta$) there is at least one vector V which specifies a permutation $\Pi_{/V/}$ moving x_{i_α} to y_{j_α} for all $\alpha = 1, 2, ..., h$.*

Using a Benes-like recursive structure of IN one can construct CPB of the first, second, ..., $(n/4)$th, and nth orders containing $\log_2 n$, $\log_2 n + 1$,...,$2\log_2 n - 2$, and $2\log_2 n - 1$ layers [17]. In algorithms below we shall use the second-order boxes $\mathbf{P}_{32;96}$ (Figure 3a) and $\mathbf{P}^{-1}_{32;96}$ (Figure 3b). These CPB are constructed using boxes $\mathbf{P}_{8;12}$ (Figure 2c) and $\mathbf{P}^{-1}_{8;12}$ (Figure 2d) as main building blocks. The connection between outputs of $\mathbf{P}_{8;12}$ and inputs of $\mathbf{P}^{-1}_{8;12}$ is described as the following fixed permutational involution $\mathbf{I}$:

$$\begin{aligned}\mathbf{I} \;=\; & (1)(2,9)(3,17)(4,25)(5)(6,13)(7,21)(8,29)\\ & (10)(11,18)(12,26)(14)(15,22)(16,30)(19)\\ & (20,27)(23)(24,31)(28)(32).\end{aligned}$$

Due to symmetric structure mutually inverse CPB $\mathbf{P}_{32;96}$ and $\mathbf{P}^{-1}_{32;96}$ differ only with the distribution of controlling bits over boxes $\mathbf{P}_{2;1}$.

When performing DDP operations with the boxes $\mathbf{P}_{32;96}$ we form 96-bit controlling vector depending on 32-bit data subblock. Let L be a controlling data subblock. Thus, bits of $L = (l_1, ..., l_{32})$ are used on the average three times while defining the controlling vector. When designing respective extension box it is reasonable to use the following criteria:

Criterion 1 [7]. *Let $X = (x_1, ..., x_{32})$ is the input vector of the $\mathbf{P}^{(V)}_{32;96}$-box. Then for all L and i the bit x_i should be permuted depending on six different bits of L.*

Criterion 2. *For all L each bit l_i, where $i \in \{1, 2, ..., 16\}$, should not influence the permutation of the bit x_i and each bit l_j, where $j \in \{17, 18, ..., 32\}$, should not influence*

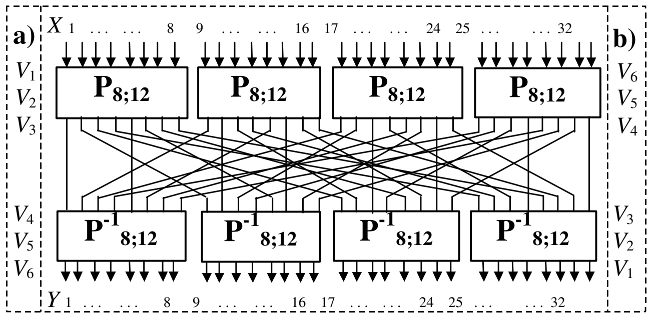
a)
b)
X 1 ... ... 8 9 ... ... 16 17 ... ... 24 25 32
V_1
V_2
V_3
V_4
V_5
V_6
$\mathbf{P}_{8;12}$
$\mathbf{P}^{-1}_{8;12}$
Y 1 ... ... 8 9 ... ... 16 17 ... ... 24 25 32

Figure 3: Structure of the CP boxes $\mathbf{P}_{32;96}$ (a) and $\mathbf{P}^{-1}_{32;96}$ (b)

the permutation of the input bits moved to y_j.

Criterion 3. *For all i the bit l_i should define exactly three bits of V.*

We suppose that Criterion 1 (2) should improve differential (nonlinear) characteristics of the $\mathbf{P}^{(V)}_{32;96}$-box. Below we use the extension box $\mathbf{E}$ forming the controlling vector V for CPB $\mathbf{P}^{(V)}_{32;96}$. This extension box defines distribution of the controlling bits l_i over boxes $\mathbf{P}_{2;1}$ described in Table 1, where lines indicate active layers and bits of the controlling data subblock L are written at the positions of the respective elementary switching boxes. Let some 32-bit data subblock L is the input of the $\mathbf{E}$-box. Let the vector $V = (V_1, V_2, V_3, V_4, V_5, V_6)$, where $\forall i\ V_i \in \{0,1\}^{16}$, be the output of the $\mathbf{E}$-box. The extension box $\mathbf{E}$ provides the correspondence between L and V specified in Table 1.

Due to symmetric structure of $\mathbf{P}_{32;96}$ its modifications $\Pi_{/V/}$, where $V = (V_1, V_2, ...V_6)$, and $\Pi_{/V'/}$, where $V' = (V_6, V_5..., V_1)$ are mutually inverse. This property of the symmetric CPB can be used in order to construct switchable CP boxes. This idea can be realized using very simple transposition box $\mathbf{P}^{(e)}_{96;1}$ implemented as some single layer CPB consisting of three parallel single-layer boxes $\mathbf{P}^{(e)}_{2\times16;1}$ (Figure 4a). Input of each $\mathbf{P}^{(e)}_{2\times16;1}$-box is divided into 16-bit left and 16-bit right inputs. The box $\mathbf{P}^{(e)}_{2\times16;1}$ contains 16 parallel $\mathbf{P}^{(e)}_{2;1}$-boxes controlled with the same bit e. For example, $\mathbf{P}^{(0)}_{2\times16;1}(U) = U$ and $\mathbf{P}^{(1)}_{2\times16;1}(U) = (U_h, U_l)$, where $U = (U_l, U_h) \in \{0,1\}^{32}$. The left (right) inputs of the $\mathbf{P}^{(e)}_{2;1}$-boxes correspond to the left (right) 16-bit input of the box $\mathbf{P}^{(e)}_{2\times16;1}$. If the input vector of the box $\mathbf{P}^{(e)}_{96;1}$ is $(V_1, V_2, ...V_6)$, then at the output of $\mathbf{P}^{(e)}_{96;1}$ we have $V' = (V_1, V_2, ..., V_6)$ (if $e = 0$) or $V' = (V_6, V_5, ..., V_1)$ (if $e = 1$). Structure of the switchable CPB $\mathbf{P}^{(L,e)}_{32;32}$ is shown in Figure 4b.

In hardware the box $\mathbf{P}_{2;1}$ can be implemented using 12 transistors. The operational box $\mathbf{P}^{(L,e)}_{32;32}$ can be implemented with 1728 transistors. This figure is less than number of transistors required to implement an anticipated carry adder performing the addition modulo 2^{32}. The time delay of some CP box is defined by the number of active layers. Time delay of one layer is approximately equal to that of the XOR operation ($t_\oplus$). Time delay of the $P^{(L,e)}_{32;32}$-box operation ($6t_\oplus$) is less than that of the addition modulo 2^{32} with high-speed carry. Straightforward estimates show that the $\mathbf{P}^{(L,e)}_{32;32}$-instruction can be added in microprocessor within less than 0.3 mm^2. Thus, the CPB $\mathbf{P}^{(L,e)}_{32;32}$ can be easily implemented as a new fast instruction of some 32-bit processors and microcontrollers.

Another interesting variant is embedding the nine-layer CP box $\mathbf{P}^{(V)}_{32;144}$ of the maximal order $h = 32$ [6]. The operation $\mathbf{P}^{(V)}_{32;144}(X)$ can perform arbitrary given bit permutation on 32-bit words. The hardware implementation cost of this instruction is about the same as that of the switchable CPB $\mathbf{P}^{(L,e)}_{32;32}$. The time delay of the operation $\mathbf{P}^{(V)}_{32;144}$ is about $9t_\oplus$. Performing the operation $\mathbf{P}^{(V)}_{32;144}$ takes 1-2 cycles (depending on the architecture of the hypothetical microcontroller or CPU). Operation $\mathbf{P}^{(V)}_{32;144}$ can be used for cryptographic purposes (construction of fast ciphers and hash functions) and for some other special purposes. For example, the instruction $\mathbf{P}^{(V)}_{32;144}$ allows to perform on a 32-bit word $X = (X_1, X_2, X_3, X_4)$ different variants of rotation operation: $Y = X^{<<<g}$, $Y = (X_1^{<<<g_1}, X_2^{<<<g_2}, X_3^{<<<g_3}, X_4^{<<<g_4})$, $Y = ((X_1, X_2)^{<<<g_2}, (X_3, X_4)^{<<<g_5})$, $Y = \left(X_1^{<<<g_1}, (X_2, X_3, X_4)^{<<<g_6}\right)$, where $g \in \{0,1\}^5$, $g_1, g_2, g_3, g_4 \in \{0,1\}^3$, $g_5 \in \{0,1\}^4$, and $g_6 \leq 10111$. In addition to being well-suited towards cryptographic purposes, $\mathbf{P}^{(V)}_{32;144}$ can be used for fast and efficient implementations of a number of common software functions. A prominent example is the bit-reversal permutation, which is used in a number of Fast Fourier Transform (FFT) algorithms. A large number of multimedia applications apply the Discrete Cosine Transform (DCT) or Discrete Fourier Transform (DFT) as steps in the processing of multimedia data. Many implementations rely on the FFT to perform these transforms. On a general-purpose uniprocessor, a bit-reversal operation can require 50 or more cycles to execute. The instruction $\mathbf{P}^{(V)}_{32;144}$ could perform a single bit-reversal in as little as one cycle. Other permutations could be used to dramatically increase the performance of higher-radix FFTs.

Thus, the CP-box instruction $\mathbf{P}^{(V)}_{32;144}$ can replace the already embedded rotation operation, economizing hardware resources and reducing to a minimum the hardware cost of the implementation of the CP-box instruction. If the CPU makers support encryption method based on DDP, then cryptographers will have the possibility to develop different variants of the software-oriented ciphers and hash functions based on DDP providing performance 400 - 1000 Mbit/s and more. In present paper we consider the instruction $\mathbf{P}^{(L,e)}_{32;32}$ which is oriented to cryptographic use.

Table 1: Distribution of bits of vector L in $\mathbf{P}_{32;96}$-box

V_1	l_7	l_8	l_1	l_2	l_{16}	l_{15}	l_{10}	l_9	l_5	l_6	l_3	l_4	l_{11}	l_{12}	l_{13}	l_{14}
V_2	l_9	l_{10}	l_{11}	l_{12}	l_1	l_2	l_7	l_8	l_{13}	l_{14}	l_{15}	l_{16}	l_5	l_6	l_3	l_4
V_3	l_{13}	l_{14}	l_{15}	l_{16}	l_5	l_6	l_3	l_4	l_1	l_2	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}
V_4	l_{21}	l_{22}	l_{29}	l_{30}	l_{25}	l_{26}	l_{23}	l_{24}	l_{31}	l_{32}	l_{27}	l_{28}	l_{17}	l_{18}	l_{19}	l_{20}
V_5	l_{31}	l_{32}	l_{27}	l_{28}	l_{17}	l_{18}	l_{19}	l_{20}	l_{29}	l_{30}	l_{25}	l_{26}	l_{21}	l_{22}	l_{23}	l_{24}
V_6	l_{19}	l_{20}	l_{23}	l_{24}	l_{27}	l_{28}	l_{29}	l_{30}	l_{21}	l_{22}	l_{17}	l_{18}	l_{32}	l_{31}	l_{25}	l_{26}

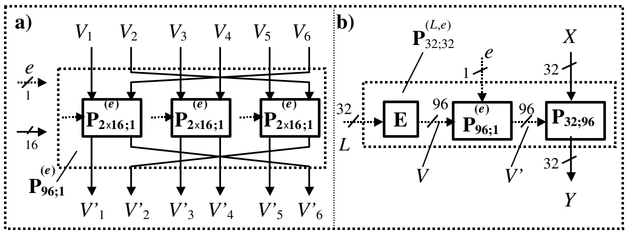

Figure 4: Structure of the switchable CP boxes $\mathbf{P}^{(e)}_{96;1}$ (a) and $\mathbf{P}^{(L,e)}_{32;32}$ (b)

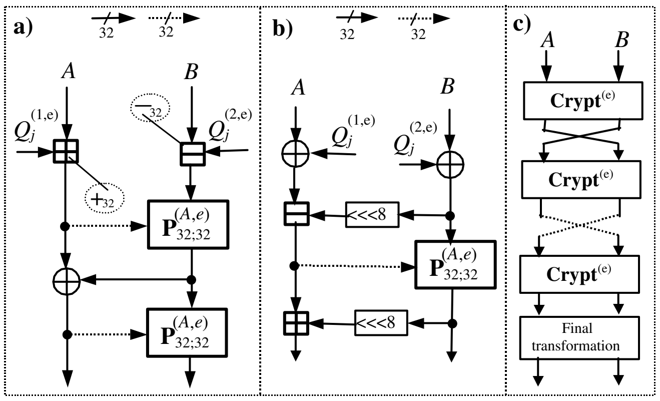

Figure 5: Procedure $\mathbf{Crypt}^{(e)}$ of Cobra-F64a (a) and Cobra-F64b (b) and general transformation scheme (c)

3 Firmware- and Software Oriented Ciphers

3.1 Firmware-suitable 64-bit Block Ciphers

We propose two variants of the firmware-oriented ciphers Cobra-F64a and Cobra-F64b with 64-bit input and 128-bit key $K = (K_1, K_2, K_3, K_4)$, where $\forall i\ K_i \in \{0,1\}^{32}$. No secret key preprocessing is used. While performing j round transformation subkeys are used directly as round 32-bit subkeys $Q^{(1,e)}_j, Q^{(2,e)}_j$, where $j = 1, ..., R+1$ and $e = 0, 1$. The number of rounds is $R = 16$ for Cobra-F64a and $R = 20$ for Cobra-F64b. Correspondence between secret key and round keys is defined by Table 2 and the following formulas:

$$\begin{aligned}(Q^{(1,1)}_1, Q^{(1,1)}_{R+1}) &= (Q^{(1,0)}_{R+1}, Q^{(1,0)}_1),\\ (Q^{(2,1)}_1, Q^{(2,1)}_{R+1}) &= (Q^{(2,0)}_{R+1}, Q^{(2,0)}_1),\text{ and}\\ (Q^{(1,1)}_j, Q^{(2,1)}_j) &= (Q^{(2,0)}_{R-j+2}, Q^{(1,0)}_{R-j+2})\ \text{ for all } j = 2, \ldots, R.\end{aligned}$$

Input 64-bit data block X is divided into two 32-bit subblocks A and B. Encryption and decryption described by the general formula $Y = \mathbf{F}^{(e)}(X, K)$ are performed in two stages (Figure 5): (1) R rounds with e-dependent procedure $\mathbf{Crypt}^{(e)}$ and (2) final transformation. The value $e = 0$ corresponds to encryption and $e = 1$ corresponds to decryption. Due to peculiarities of the structure of the round transformation of Cobra-F64a and Cobra-F64b initial transformation is not used. For both ciphers the data ciphering algorithm can be represented as follows:

1) For $j = 1$ to $R-1$ do:
$\{(A, B) := \mathbf{Crypt}^{(e)}(A, B, Q^{(1,e)}_j, Q^{(2,e)}_j)$; $(A, B) := (B, A)\}$.

2) For $j = R$ do:
$\{(A, B) := \mathbf{Crypt}^{(e)}(A, B, Q^{(1,e)}_j, Q^{(2,e)}_j)\}$.

3) Perform final transformation: $Y = (Y_l, Y_h) := (A \oplus Q^{(1,e)}_{R+1}, B \oplus Q^{(2,e)}_{R+1})$ for Cobra-F64b or $Y = (Y_l, Y_h) := (A -_{32} Q^{(1,e)}_{R+1}, B +_{32} Q^{(2,e)}_{R+1})$ for Cobra-F64a, where Y is the 64-bit output data block.

The procedure $\mathbf{Crypt}^{(e)}$ is described in Figure 5a (for Cobra-F64a) and in Figure 5b (for Cobra-F64a). Both variants of the procedure $\mathbf{Crypt}^{(e)}$ are based on the instruction $\mathbf{P}^{(L,e)}_{32;32}$ in which the controlling vector is specified with the left data subblock. Such DDP contributes mostly to the security of Cobra-F64a and Cobra-F64b. The last cipher uses only one DDP operation in each round versus two DDP operations in one round of Cobra-F64b. For this reason we define to perform 20 encryption rounds with Cobra-F64b and 16 rounds with Cobra-F64a.

In a cheap implementation these ciphers provide performance about 20 Mbit/s for some microcontroller working at 30 MHz.

Table 2: Key scheduling in Cobra-F64a (F64b and S128) for encryption

$j =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$Q_j^{(1,0)} =$	K_1	K_2	K_3	K_4	K_2	K_1	K_4	K_3	K_1	K_2	K_4	K_3	K_1	K_4	K_2	K_3	K_1	K_4	K_3	K_1	K_2
$Q_j^{(2,0)} =$	K_4	K_3	K_1	K_2	K_3	K_2	K_1	K_4	K_2	K_3	K_1	K_2	K_3	K_1	K_3	K_4	K_2	K_1	K_4	K_2	K_3

3.2 Software Encryption System Cobra-S128

While having possibility to use a fast CP-box instruction $\mathbf{P}_{32;32}^{(L,e)}$ described in Section 2 one can use the following design criteria:

1) Operation $\mathbf{P}_{32;32}^{(L,e)}$ should be used to perform DDP, i.e. controlling vector should be dependent on one of data subblocks.

2) Encryption round should combine CP-box operation with fast operations (XOR, addition and subtraction modulo 2^{32}, transposition of 32-bit words).

3) The cipher should be free of the key preprocessing. This provides higher rapidity in the case of frequent change of keys.

4) Encryption and decryption should be performed with the same algorithm. Reason for this design criterion is to provide cheaper hardware implementation if necessary.

Using these criteria we have developed 128-bit block cipher Cobra-S128. It is a twelve-round iterated cryptosystem with 128-bit secret key $K = (K_1, K_2, K_3, K_4)$ represented as concatenation of four 32-bit subkeys. While ciphering subkeys $Q_j^{(e)}$ are used directly as round 32-bit subkeys $Q_j^{(1,e)}$ and $Q_j^{(2,e)}$ which for $j = 1, ..., 12$ and $e = 0, 1$ are specified with Table 2 and the following correspondence formulas: $Q_j^{(1,1)} = Q_{R-j+1}^{(2,0)}$ and $Q_j^{(2,1)} = Q_{R-j+1}^{(1,0)}$.

Input 128-bit data block X is divided into four 32-bit subblocks A, B, C, and D (Figure 6a). Data ciphering described by the general formula $Y = \mathbf{F}^{(e)}(X, K)$ is performed in three stages (Figure 6b): (1) initial transformation, (2) execution of twelve encryption rounds, and (3) final transformation. Transformation of the input data block can be represented as follows:

1) Execute initial transformation: $(A, B, C, D) := A \oplus Q_1^{(1,e)}, B \oplus Q_2^{(1,e)}, C \oplus Q_3^{(1,e)}, D \oplus Q_4^{(1,e)})$.

2) For $j = 1$ to 11 do: $\{(A, B, C, D) := \mathbf{Crypt}^{(e)}(A, B, C, D, Q_j^{(1,e)}, Q_j^{(2,e)})$; $(A, B, C, D) := (B, A, D, C)\}$

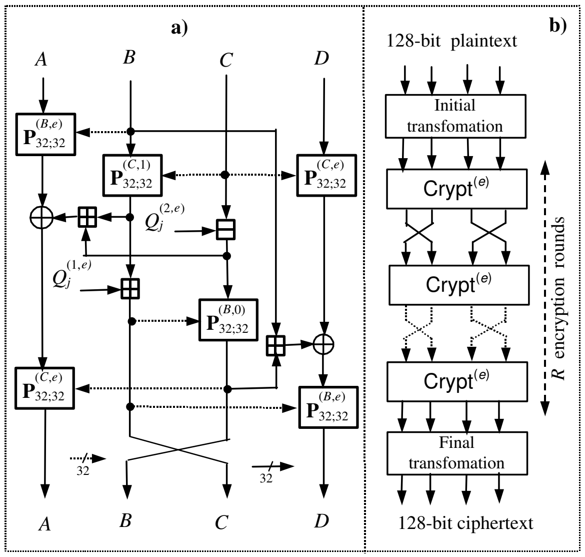

Figure 6: Encryption round of Cobra-S128 (a) and general structure of the cipher (b)

3) For $j = 12$ do: $\{(A, B, C, D) := \mathbf{Crypt}^{(e)}(A, B, C, D, Q_j^{(1,e)}, Q_j^{(2,e)})$;

4) Perform final transformation: $Y := (A \oplus Q_{12}^{(2,e)}, B \oplus Q_{11}^{(2,e)}, C \oplus Q_{10}^{(2,e)}, D \oplus Q_9^{(2,e)})$, where Y is 128-bit output data block.

The second stage is performed using subkeys $Q_j^{(1,e)}$ and $Q_j^{(2,e)}$ and procedure $\mathbf{Crypt}^{(e)}$ described in Figure 6a. The value $e = 0$ corresponds to encryption and $e = 1$ corresponds to decryption. We have estimated that Cobra-S128 can provide performance about 400 Mbit/s for some hypothetical Pentium-like processor having CP-box instruction $\mathbf{P}_{32;32}^{(e)}$. To measure "real" encryption speed we have composed a program implementing a model of cipher Cobra-S128 in which each CP-box operation have been replaced by addition modulo 2^{32} operation. Such model-based experiment showed performance 430 Mbit/s for twelve rounds and 650 Mbit/s for eight rounds. Using transformations analogous to that of Cobra-S128 we have also composed models implementing several DDP-based hash functions which have shown perfomance 1 - 2 Gbit/s.

Hash functions are faster as compared with ciphers with analogous structure of transformation, since they do not require the intermediate transformed data blocks to be saved. Detailed description and the analysis of the DDP-based hash functions represent a topic of the individual research.

4 Security Estimation of the DDP-based Ciphers

Security of the DDP-based ciphers described above is based on the properties of the $\mathbf{P}^{(L,e)}_{32;326}$-box operations. It is easy to see that a single input bit of the box $\mathbf{P}^{(L,e)}_{32;32}$ moves to each output position with the same probability provided L is a uniformly distributed random variable. Avalanche effect is caused by the use of data to define the value of the controlling vector V' (see Figure 4). One bit of the vector L influences three bits of V'. Each controlling bit influences two bits of the input data subblock. Thus, when performing one CP-box operation one bit of the controlling data subblock influences 6 bits of the permuted binary vector X.

Let Δ_q be the difference with arbitrary q active (nonzero) bits. Let $\Delta_{q|i_1,...,i_q}$ be the difference with q active bits and $i_1, ..., i_q$ be the numbers of digits corresponding to active bits. Note that Δ_1 corresponds to one of the differences $\Delta_{1|1}, \Delta_{1|2}, ..., \Delta_{1|32}$. Let $P(\Delta_q \xrightarrow{\mathbf{F}} \Delta_g)$ be the probability that input difference Δ_q transforms into output difference Δ_g while passing some operation $\mathbf{F}$. Let $P(R) = P(\Delta_q \xrightarrow{R} \Delta_g)$ be the probability that Δ_q transforms into Δ_g while passing through R encryption rounds.

Let us consider the box $\mathbf{P}^{(L,e)}_{32;32}$ in the case when some difference with one active bit Δ^L_1 corresponds to some controlling data subblock L. Passing through the extension box (see Figure 4) this difference forms difference Δ^V_3 at the controlling input of the internal CP box $\mathbf{P}_{32;96}$. Thus, the difference Δ^L_1 influences three switching elements $\mathbf{P}_{2;1}$ of the box $\mathbf{P}_{32;96}$ permuting six different bits of the permuted vector X. Depending on the input value and input difference Δ^X_q of the $\mathbf{P}_{32;96}$-box the output differences Δ^Y_g with different number of active bits can be formed. Table 3 presents probabilities of different output differences corresponding to differences $\Delta^L_{q'}$ and Δ^X_q with few active bits ($q', q \in \{0, 1, 2\}$).

Differential cryptanalysis of Cobra-S128.

Our best variant of the differential cryptanalysis (DC) corresponds to two-round characteristic with difference $\Delta_2 = (\Delta^A_1, \Delta^B_0, \Delta^C_0, \Delta^D_1)$. The fact that differential characteristics with few active bits are the most efficient ones seems to be a general property of the DDP-based ciphers (see for example DC of RC5 [3, 8]). The difference Δ_2 passes two rounds in the following way (see Figure 7). It is easy to see that this difference passes the first round with probability 1 and after swapping subblocks it transforms in $\Delta_2 = (\Delta^A_0, \Delta^B_1, \Delta^C_1, \Delta^D_0)$.

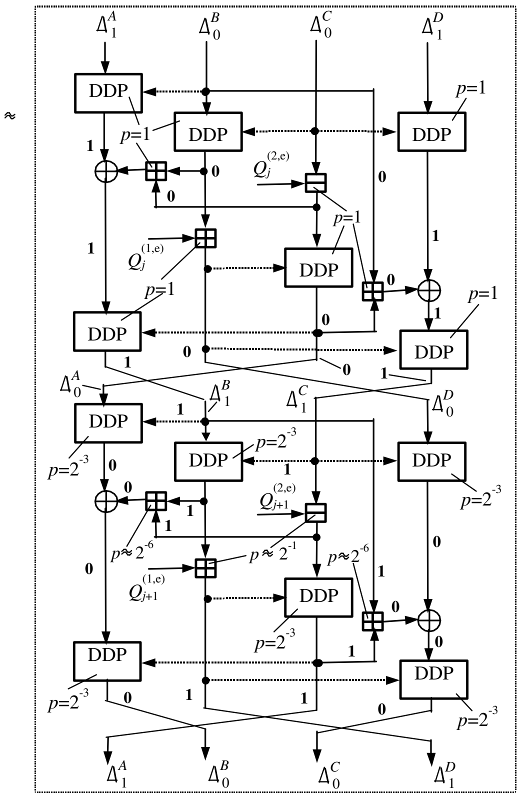

Figure 7: Formation of the two-round differential characteristic in Cobra-S128

In the second round at different steps of the round transformation the subblocks B and C with one active bit are combined two times using modulo 2^{32} addition operation. Each of these two modulo 2^{32} additions produces at its output zero difference with probability $p_1 \approx 2^{-6}$ ($p'_1 = 2^{-5}$ corresponds to the probability that for the current differences $\Delta^B_{1|i}$ and $\Delta^C_{1|i'}$ we have $i = i'$ and $p''_1 \approx 2^{-1}$ corresponds to the probability that no carry bits are formed). In addition to this each active bit influences three DDP operations. Each DDP operation produces no active bits in the respective transformed data subblock with probability $p_2 = 2^{-3}$. While combining differences Δ^B_1 and Δ^C_1 with subkeys $Q^{(1,e)}_{j+1}$ and $Q^{(2,e)}_{j+1}$, respectively, the modulo 2^{32} addition and subtraction operations produce on the average no active bits with probability $p_+ \approx 2^{-1}$ and $p_- \approx 2^{-1}$, respectively. Taking into account these elementary events forming the two-round characteristic we obtain its probability:

$$P(2) = p_1^2 p_2^6 p_+ p_- \approx 2^{-32}.$$

Using these value one can calculate the minimal number $R_{\min} = 8$ of the encryption rounds for which Cobra-S128 is undistinguishable from a random cipher while using differential cryptanalysis. Indeed, for eight rounds of Cobra-S128 we have $P(8) = P^4(2) \approx 2^{-128} < P_{\text{rand}} = 2^{-118}$, where P_{rand} is the probability to have output difference $(\Delta^A_0, \Delta^B_1, \Delta^C_1, \Delta^D_0)$ in the case of the random cipher.

Table 3: Values of probability $P\left(\left(\Delta_q^X \xrightarrow{\mathbf{P}^{(L,e)}_{32;32}} \Delta_g^{(Y)}\right) \Big/ \Delta_{q'}^L\right)$

	$\Delta_0^X \to \Delta_0^Y$	$\Delta_0^X \to \Delta_2^Y$	$\Delta_0^X \to \Delta_4^Y$	$\Delta_1^X \to \Delta_1^Y$	$\Delta_1^X \to \Delta_3^Y$
Δ_0^L	1	0	0	1	0
Δ_1^L	2^{-3}	$1.5 \cdot 2^{-2}$	$1.5 \cdot 2^{-2}$	$1.17 \cdot 2^{-3}$	$1.59 \cdot 2^{-2}$
Δ_2^L	2^{-6}	$1.5 \cdot 2^{-4}$	$1.88 \cdot 2^{-3}$	$1.38 \cdot 2^{-6}$	$1.88 \cdot 2^{-4}$

	$\Delta_1^X \to \Delta_5^Y$	$\Delta_2^X \to \Delta_0^Y$	$\Delta_2^X \to \Delta_2^Y$	$\Delta_2^X \to \Delta_4^Y$	$\Delta_2^X \to \Delta_6^Y$
Δ_0^L	0	0	1	0	0
Δ_1^L	$1.41 \cdot 2^{-2}$	$1.55 \cdot 2^{-9}$	$1.08 \cdot 2^{-3}$	$1.36 \cdot 2^{-2}$	$1.15 \cdot 2^{-2}$
Δ_2^L	$1.06 \cdot 2^{-2}$	$1.55 \cdot 2^{-8}$	$1.38 \cdot 2^{-6}$	$1.69 \cdot 2^{-4}$	$1.72 \cdot 2^{-3}$

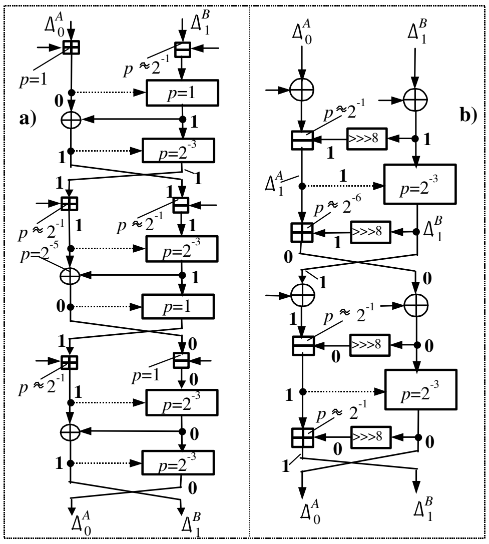

Figure 8: Formation of tree-round characteristic in Cobra-F64a (a) and two-round characteristic in Cobra-F64b (b)

Security of the firmware encryption systems. As in the case of Cobra-S128 the best characteristics of Cobra-F64a and Cobra-F64b correspond to differences with few active bits. The best characteristics of these ciphers are three-round iterative characteristic for Cobra-F64a (Figure 8a) with probability $P(3) \approx 2^{-21}$ and two-round one for Cobra-F64b (Figure 8b) with probability $P(2) \approx 2^{-12}$. Results of our DC-security estimation of the ciphers Cobra-F64a and Cobra-F64b are presented in Table 4, where for comparison we present also our results on DC of SPECTR-H64. Obtained results show that all considered DDP-based ciphers are secure against DC. Our analysis has also shown that SPECTR-H64's security against DC is more sensitive to the structure of the extension box as compared with Cobra-F64a and Cobra-F64b. In hardware SPECTR-H64 seems to be faster as compared with Cobra-F64a and Cobra-F64b. However except CPB permutations the hardware-oriented cipher Spectr-H64 uses an additional specially designed cryptographic primitive (operation **G** [7]) that makes the firmware implementation significantly more expensive. Besides, the ciphers Cobra-F64a and Cobra-F64b are significantly faster in software provided the CPB instruction $\mathbf{P}^{(L,e)}_{32;32}$ is implemented.

Rough linear cryptanalysis (LC). Detailed analysis of the linear characteristics of the CP-box operations we present in [6]. Our preliminary LC of the ciphers Cobra-S128, Cobra-F64a, and Cobra-F64ba has shown that they are secure against linear attacks. Our preliminary study of their security against LC has shown that structures of these ciphers are also suitable for calculation of the biases of the linear characteristics in the case of few active bits. Our rough analysis has shown that the best one-round linear characteristics have bias $b \leq 2^{-6}$ In accordance with these results the ciphers are undistinguishable with LC from a random cipher for $R \geq 7$.

Comments on other attacks. High degree of the algebraic normal form and the complexity of the Boolean function describing round transformation of the described ciphers prevent the interpolation and high order differential attacks. In spite of the use of very simple key scheduling the described ciphers are secure against slide attack due to non-periodic use of the round subkeys and data-dependent subkey transformations.

Comments on key scheduling. The used key scheduling is secure against basic related-key attacks. In spite of the simplicity of the key schedule the "symmetric" keys $K' = (X, Y, X, Y)$ or $K'' = (X, X, X, X)$ are not weak, since encryption and decryption require change of the parameter e. Indeed, from Figure 5 and 6 it is easy to see that for all considered ciphers we have $\mathbf{F}^{(e=0)}(C, K'') \neq M$, where $C = \mathbf{F}^{(e=0)}(M, K'')$. For comparison we can note that for all X the key $K'' = (X, X, X, X)$ is weak for SPECTR-H64 that does not use switchable CPB operations. It seems to be difficult to calculate a semi-weak key-pair for the ciphers Cobra-S128, Cobra-F64a, and Cobra-F64b, if it is possible at all. In the case when keys are not changed often one can use one of the known key scheduling procedures providing pseudorandom generation of the round keys.

Table 4: DC-security estimation of the DDP-based ciphers

Cipher	R	Difference	Probablity	P_{rand}	$R_{\min}$
Cobra-S128	12	$(\Delta_1^A, \Delta_0^B, \Delta_0^C, \Delta_1^D)$	$P(2) = 2^{-32}$	2^{-115}	8
Cobra-F64a	16	(Δ_1^A, Δ_0^B)	$P(3) = 2^{-21}$	2^{-59}	9
Cobra-F64b	20	(Δ_0^A, Δ_1^B)	$P(2) = 2^{-12}$	2^{-59}	10
SPECTR-H64	12	(Δ_0^A, Δ_1^B)	$P(2) = 1.1 \cdot 2^{-13}$	2^{-59}	10

5 Conclusion

In the present paper we have shown that DDP, earlier used in several hardware-oriented 64-bit ciphers, can be also effectively used when designing fast cryptosystems suitable to software implementation. We have proposed to embed some CP-box instruction in general purpose processors and in different types of microcontrollers and smart cards. A simple variant of the fast switchable $\mathbf{P}_{32;32}^{(L,e)}$-box instruction has been designed and used in one 128-bit software and two firmware-oriented 64-block ciphers illustrating efficiency of the cryptographic use of this instruction. More advanced $\mathbf{P}_{32;144}^{(V)}$-box instruction can perform all possible bit-permutation operations on 32-bit words. Each of such operations can be specified by the controlling vector V and it is not difficult to find value $/V/$ for all possible permutational operations including special ones. This spreads significance of the advanced CP-box instruction far beyond cryptographic applications and can attract serious attention of the CPU manufactures, since a cheap embedding of the $\mathbf{P}_{32;144}^{(V)}$-box instruction imparts attractive properties to the general purpose processors. One of the lasts is the possibility to get more than 500 Mbit/s encryption speed in software. The ability to perform special permutations, such as bit-reversal, can significantly improve the performance of multimedia applications which rely on efficient DCT and DFT algorithms. Comparing with other CPB instruction architectures [10, 11] the proposed one is characterized in its reversibility providing very fast implementation of mutually inverse bit permutations of arbitrary type.

Regarding cryptographic use we have shown that CP-box operations are efficient primitive contributing significantly to the security against differential and linear cryptanalysis. DDP suite well to estimating security against these attacks. We hope that this work will attract more attention of cryptographic community to DDP in respect of the cryptanalysis of the published cryptalgorithms and designing new DDP-based block ciphers, hash functions, and key expansion algorithms.

Acknowledgement

This work was carried out as part of the AFRL funded project #1994p which supported the authors.

References

[1] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Trafic*, Academic Press, New York, 1965.

[2] A. Biryukov and S. Even, *Cryptanalysis of the Ports Interconnection Network Block Cipher*, Technical report CS-0887, Technion (reported at DIMACS'97 "Cryptography and network security").

[3] A. Biryukov and E. Kushilevitz, "Improved cryptanalysis of RC5," in *Advances in cryptology - EUROCRYPT'98 Proceedings*, LNCS 1403, pp. 85-99, Springer-Verlag, 1998.

[4] C. Burwick, D. Coppersmith, and E. D'Avingnon et al., "MARS - a candidate cipher for AES," in *1st AES Candidate Conference Proceedings of Venture*, California, Aug. 20-22, 1998.

[5] C. Clos, "A study of nonblocking switching networks," *Bell System Technical Jornal*, vol. 32, pp. 406-424, 1953.

[6] N. D. Goots, A. A. Moldovyan, and N. A. Moldovyan, "Variable bit permutations: linear characteristics and pure VBP-based cipher," *Computer Science Journal of Moldova*, vol. 13, no. 1, pp. 84-109, 2005.

[7] N. D. Goots, A. A. Moldovyan, and N. A. Moldovyan, "Fast encryption algorithm SPECTR-H64," in *Proceedings of the International workshop "Methods, Models, and Architectures for Network Security"*, LNCS 2052, pp. 275-286, Springer-Verlag, 2001.

[8] B. S. Kaliski and Y. L. Yin, "On differential and linear cryptanalysis of the RC5 encryption algorithm," in *Advances in cryptology - CRYPTO'95 Proceedings*, LNCS 963, pp. 171-184, Springer-Verlag, 1995.

[9] M. Kwan, "The design of the ICE encryption algorithm," in *Proceedings of the 4th International Workshop, Fast Software Encryption - FSE '97*, LNCS 1267, pp. 69-82, Springer-Verlag, 1997.

[10] R. B. Lee, Z. J. Shi, and X. Yang, "Efficient permutation instructions for fast software cryptography," *IEEE Micro*, vol. 21, no 6, pp. 56-69, 2001.

[11] R. B. Lee, Z. J. Shi, R. L. Rivesr, and M. J. B. Robshaw, "On permutation operations in Cipher Design", in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, vol. 2, pp. 569-579, Las Vegas, Nevada, Apr. 5-7, 2004.

[12] A. A. Moldovyan and N. A. Moldovyan, "A cipher based on data-dependent permutations," *Journal of Cryptology*, vol. 15, no. 1, pp.61-72, 2002.

[13] M. Portz, "A generallized description of DES-based and Benes-based permutationgenerators," *Advances in Cryptology - ASIACRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques*, LNCS 718, pp. 397-409, Springer-Verlag, 1992.

[14] R. L. Rivest, "The RC5 encryption algorithm," in *Proceedings of the 2nd International Workshop "Fast Software Encryption - FSE'94"*, LNCS 1008, pp. 86-96, Springer-Verlag, 1995.

[15] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 block cipher," in *1st AES Candidate Conference Proceedings of Venture*, California, Aug. 20-22, 1998.

[16] B. V. Rompay, L.R. Knudsen, and V. Rijmen, "Differential cryptanalysis of the ICE encryption algorithm," in *Proceedings of the 6th International Workshop, "Fast Software Encryption - FSE'98"*, LNCS 1372, pp. 270-283, Springer-Verlag, 1998.

[17] N. Sklavos, N. A. Moldovyan, A. A. Moldovyan, and O. Koufopavlou, "CHESS-64, a block cipher based on data-dependent operations: design variants and hardware implementation efficiency," *Asian Journal of Information Technology*, vol. 4, no. 4, pp. 323-334, 2005.

Nikolay A. Moldovyan is an honored inventor of Russian Federation (2002), a chief researcher with the Specialized Center of Program Systems "SPECTR", and a Professor with the Saint Petersburg Electrical Engineering University. His research interests include computer security and cryptography. He has authored or co-authored more than 50 patents and 200 scientific articles, books, and reports. He received his Ph.D. from the Academy of Sciences of Moldova (1981). Contact him at: nmold@cobra.ru.

Peter A. Moldovyanu is a division head with the Specialized Center of Program Systems "SPECTR". His research interests include cryptography, communication and network security. He has authored or co-authored 6 patents and more than 40 scientific papers. He received his Ph.D. from the St. Petersburg State University of Information Technologies, Mechanics, and Optics (2004). Contact him at: ma@cobra.ru.

Douglas H. Summerville is an Associate Professor in the Department of Electrical & Computer Engineering at the State University of New York at Binghamton. His research interests include computer and network security, intrusion detection, and networking. He has authored or co-authored more than 30 scientific articles. He received his Ph.D. in Electrical Engineering from the State University of New York (1997) and B.E. in Electrical Engineering from The Cooper Union for the Advancement of Science and Art, New York, (1991). Contact him at: dsummer@binghamton.edu.