Computer Science Journal of Moldova, vol.**11**, no.3(33), 2003

Fast DDP-Based Ciphers: Design and Differential Analysis of Cobra-H64

N.A. Moldovyan

Abstract

Use of the controlled operations is considered as a new approach to the design of fast hardware-oriented ciphers. Data-dependent (DD) permutations and DD two-place operations are used in a new ten-round cipher named Cobra-H64 which is fast and cheap when implemented in hardware. The peculiarity of the iterated 64-bit cipher Cobra-H64 is the very fast encryption in the case of frequent change of keys, since no preprocessing is used to perform key scheduling. The whole secret key is directly used in each round. The Cryptosystem Cobra-H64 is characterized by high parallelism of the data transformation. Time delay of one round is $15t_{\oplus}$, where $t_{\oplus}$ is the time delay of the XOR operation. The analysis performed has shown that Cobra-H64 is secure against differential attacks.

Key words: data-dependent operations, controlled permutations, data-dependent permutations, differential characteristics, fast hardware encryption, block cipher

1 Introduction

Data-dependent (DD) permutations (DDP) performed with so called controlled permutation (CP) boxes [1, 2] appear to be very efficient cryptographic primitive for fast hardware encryption.

Controlled two-place operations (CTPO) represent another type of interesting cryptographic primitives [3]. A variant of CTPO was used for the first time in the 64-bit block cipher SPECTR-H64 together with DDP [4]. It is remarkable that in one round of this cipher the

two mutually inverse DDP operations are performed sequentially on the right data subblock. The CTPO and DDP are associated well in this cipher, however the bits of the controlling data subblock are not used uniformly while forming the controlling vectors corresponding to the CP boxes $\mathbf{P}_{32/80}$ and $\mathbf{P}_{32/80}^{-1}$ and some improvement of the round structure of SPECTR-H64 is desirable.

In present paper we design a new block cipher Cobra-H64 oriented to fast hardware encryption using CP-box permutations and CTPO. Regarding general structure and the use of data-dependent operations cryptosystem Cobra-H64 is analogous to SPECTR-H64, but the first cipher includes several features. They consist in that Cobra-H64 uses: 1) two equal CTPO, which are different from that used in SPECTR-H64, 2) fixed permutational involution on the left data subblock, 3) CP boxes $\mathbf{P}_{32/96}$ and $\mathbf{P}_{32/96}^{-1}$ instead of $\mathbf{P}_{32/80}$ and $\mathbf{P}_{32/80}^{-1}$, 4) special switchable fixed permutation between two mutually inverse CP-box operations, and 5) no combining subkeys with the left data subblock while forming controlling vectors.

The paper is organized in the following way: In second section we consider construction of the controlled operational boxes performing DDP. In section 3 we describe general structure of the Cobra-H64. In section 4 we give some justification of the basic primitives used in Cobra-H64 and consider characteristics of DDP and CTPO corresponding to the differences with several active bits. Section 5 discusses design of Cobra-H64 and presents security estimation against differential analysis and discussion of other attacks.

2 Design of the controlled permutations

Controlled permutations can be easy performed with well known interconnection networks (IN) [5, 6] which were proposed to construct key-dependent permutations [7, 8]. Regarding cryptographic applications it is more attractive to use IN to perform variable permutations using some data subblock to specify the current permutation performed on another data-subblock [1, 2]. An operational box $\mathbf{P}_{n/m}$ performing permutations on n-bit binary vectors depending on some controlling

m-bit vector V we shall call controlled permutation box (CP box or simply CPB). Different types of the CP boxes can be constructed using elementary switching elements $\mathbf{P}_{2/1}$ (Fig. 1a) as elementary building blocks performing controlled transposition of two input bits x_1 and x_2.

The layered CPB are constructed as superposition of $S = \frac{2m}{n}$ active layers separated with $S-1$ fixed permutations π_i,... , π_{S-1} that are implemented in hardware as simple connections. Each active layer (Fig. 1a) in a CPB with $2n$-bit input is represented by the set of n parallel elementary boxes $\mathbf{P}_{2/1}$. General structure of the layered CPB is shown in Fig. 1b. In all figures in this paper the solid lines indicate data movement, while dotted lines corresponding to CP boxes indicate the controlling bits. A CP box inverse of the box $\mathbf{P}_{n/m}$ is denoted as $\mathbf{P}_{n/m}^{-1}$.

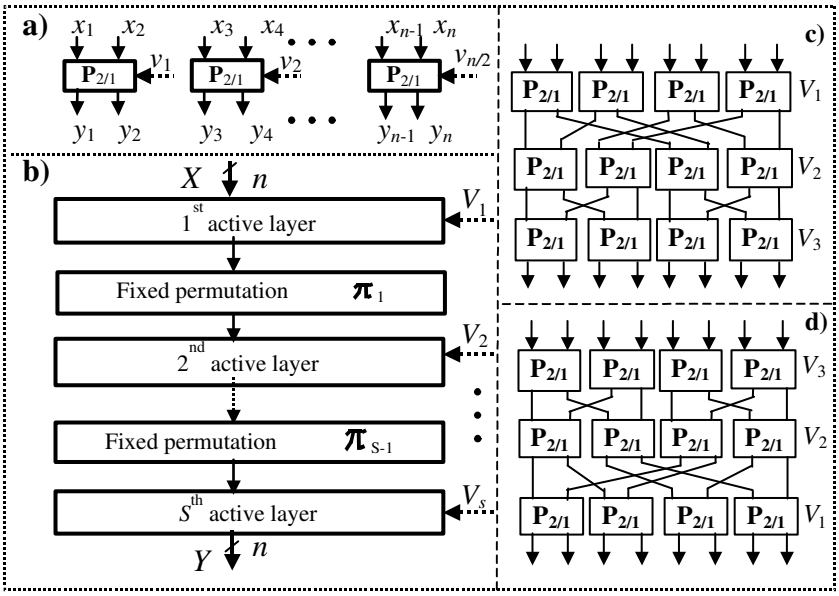

Figure 1. One active layer (a), general structure of the layered CP boxes (b), and topology of the boxes $\mathbf{P}_{8/12}$ (c) and $\mathbf{P}_{8/12}^{-1}$ (d)

We assume that in a layered CP box all elementary switching ele-

ments are consecutively numbered from left to right and from top to bottom and the jth bit of vector V controls the jth switching element $\mathbf{P}_{2/1}$. In accordance with the number of layers the vector V can be represented as concatenation of $2m/n$ vectors $V_1, V_2, ..., V_{2m/n} \in \{0,1\}^{n/2}$, i.e. $V = (V_1, V_2, ..., V_{2m/n})$.

Controlled permutations performed with the box $\mathbf{P}_{n/m}$ can be characterized using an ordered set of the modifications $\{\Pi_0, \Pi_1, ..., \Pi_{2^m-1}\}$, where each modification Π_i, $i = 0, 1, ..., 2^m - 1$, is a fixed permutation of some set of n bits. Permutations Π_i we shall call CP modifications. The execution of the CP operation $\mathbf{P}_{n/m(V)}(X)$ consists in performing the permutation Π_V on $X : Y = \mathbf{P}_{n/m(V)}(X) = \Pi_V(X)$. The following two definitions we use according to [1].

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

Definition 2. *Suppose that for arbitrary $h \leq n$ input bits $x_{\alpha_1}, x_{\alpha_2}, ..., x_{\alpha_h}$ and arbitrary h output bits $y_{\beta_1}, y_{\beta_2}, ..., y_{\beta_h}$ there is at least one value V which specifies a permutation Π_V moving x_{α_i} to y_{β_1} for all $i = 1, 2, ..., h$. Such a $\mathbf{P}_{n/m}$-box is called a CP box of the order h.*

One active layer can be considered as the single-layer CPB $\mathbf{S}_n$. It is evidently that $\mathbf{P}_{2/1} = \mathbf{P}^{-1}_{2/1}$, therefore $\mathbf{S}_n = \mathbf{S}_n^{-1}$. A layered CPB $\mathbf{P}_{n/m}$ can be represented as superposition $\mathbf{P}_{n/m} = \mathbf{S}_{V_1} \circ \pi_1 \circ \mathbf{S}_{V_2} \circ \pi_2 \circ ... \circ \pi_{S-1} \circ \mathbf{S}_{V_{2m/n}}$. The respective box $\mathbf{P}^{-1}_{n/m}$ has the following structure $\mathbf{P}^{-1}_{n/m} = \mathbf{S}_{V_{2m/n}} \circ \pi^{-1}_{2m/n-1} \circ \mathbf{S}_{V_{2m/n-1}} \circ \pi^{-1}_{2m/n-2} \circ ... \circ \pi_1^{-1} \circ \mathbf{S}_{V_1}$. Thus, to construct inverse of the CP box $\mathbf{P}_{n/m}$ it is sufficient to number the boxes $\mathbf{P}_{2/1}$ from left to right and *from bottom to top* and to replace π_i by $\pi^{-1}_{2m/n-i}$. We shall assume that in the boxes $\mathbf{P}^{-1}_{n/m}$ switching elements $\mathbf{P}_{2/1}$ are consecutively numbered from left to right from bottom to top. Note that the vector V_j corresponding to the jth active layer in the box $\mathbf{P}_{n/m}$ controls the $(2m/n-j+1)$th active layer in $\mathbf{P}^{-1}_{n/m}$.

In one round of the cipher SPECTR-H64 there are two DDP on the right data subblock performed in serial using two mutually inverse CPB of the first order: $\mathbf{P}^{-1}_{32/80}$ and $\mathbf{P}_{32/80}$. The use of the CP boxes with

80-bit controlling input stipulates that in SPECTR-H64 the left data subblock L has unbalanced influence on the formation of the controlling vectors corresponding to boxes $\mathbf{P}_{32/80}$ and $\mathbf{P}^{-1}_{32/80}$. To balance the left data subblock's influence on CP-box permutations one can use the boxes $\mathbf{P}_{32/96}$ and $\mathbf{P}^{-1}_{32/96}$ each of which is constructed using four parallel boxes $\mathbf{P}_{8/12}$ and four parallel boxes $\mathbf{P}^{-1}_{8/12}$ shown in Fig. 1c,d. The first and the second group of boxes are separated with permutational involution $\mathbf{I}$ described as follows:

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

The structure of the boxes $\mathbf{P}_{32/96}$ and $\mathbf{P}^{-1}_{32/96}$ is presented in Fig. 2. It is easy to show that both of these CP boxes have the second order and the superposition $\mathbf{P}^{-1}_{32/96(V)} \circ \mathbf{P}_{32/96(V')}$ as well as $\mathbf{P}_{32/96(V)} \circ \mathbf{P}^{-1}_{32/96(V')}$ represents the twelve-layer CPB of maximal order. When using the boxes $\mathbf{P}_{32/96}$ and $\mathbf{P}^{-1}_{32/96}$ one can easy design some mechanism of the formation of the controlling vector for which each bit of L influences exactly three bits of V.

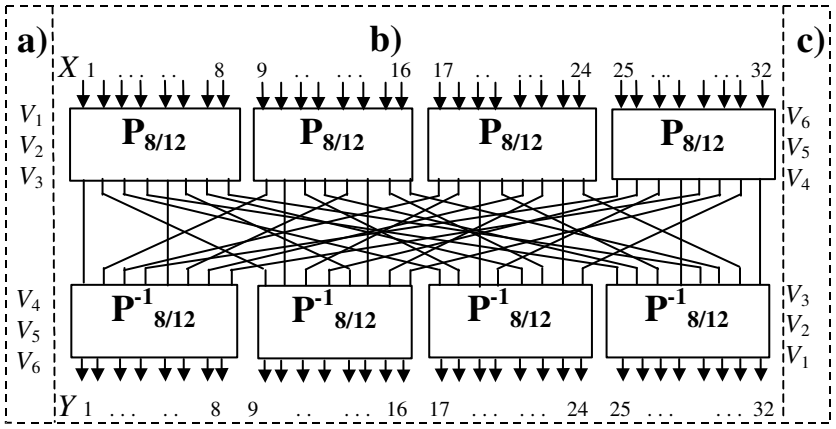

Figure 2. Structure of the CP boxes $\mathbf{P}_{32/96}$ (a) and $\mathbf{P}^{-1}_{32/96}$ (b)

3 The block cipher Cobra-H64

While designing the single key cryptosystem Cobra-H64 our strategy was oriented to the extensive use of the controlled operations in the form of the CP-box operations and CTPO corresponding to the sequential model proposed in [3]. Both of these basic cryptographic primitives are fast and inexpensive while implementing in hardware. Our design criteria were the following:

1. The cryptosystem should be an iterated 64-bit cipher.

2. The cryptalgorithm should be able to perform encryption and decryption with simple and fast change of the sequence of the used subkeys.

3. The cipher should be fast in the case of frequent change of keys. For this reason we use very simple key scheduling.

4. Round transformation of data subblocks should be characterized by high parallelism.

Let us denote $X_l = (x_1, ..., x_{n/2})$ and $X_h = (x_{n/2+1}, ..., x_n)$. Then $X_{lo}, X_{hi} \in \{0,1\}^{n/2}$ and $X = (X_l, X_h)$. Let $X, Y, A \in \{0,1\}^n$, $e \in \{0,1\}$ ($e = 0$ is encryption and $e = 1$ is decryption). Let $Y = X^{\ggg k}$ denote rotation of the word X by k bits ($0 \leq k < n$), where $Y = y_1, ..., y_n$ is the output, $y_j = x_{j+k}$ for $1 \leq j \leq n-k$, and $y_j = x_{j+k-n}$ for $n - k + 1 \leq j \leq n$.

General structure of the encryption round of SPECTR-H64 suites very well to satisfy our design criteria, therefore we have used this cipher as a prototype when developing 64-bit cryptosystem Cobra-H64. Instead of one CP-box operation and one CTPO used to perform internal key scheduling (IKS) in SPECTR-H64 we use two CTPO of the same type. To gain better spreading of the avalanche effect caused by these CTPO operations two fixed permutations ($\mathbf{I}$ and $\pi^{(e)}$) are used in Cobra-H64. Permutation $\mathbf{I}$ is a permutational involution performed on the left data subblock L. Permutation $\pi^{(e)}$ is some special switchable permutation performed on the right data subblock R. Fixed permutation $\pi^{(0)}$ is executed when encrypting and fixed permutation $\pi^{(1)} = \left(\pi^{(0)}\right)^{-1}$ is executed when decrypting. To change $\pi^{(0)}$ for $\pi^{(1)}$ we use a single-layer CPB. Time delay introduced by such switchable

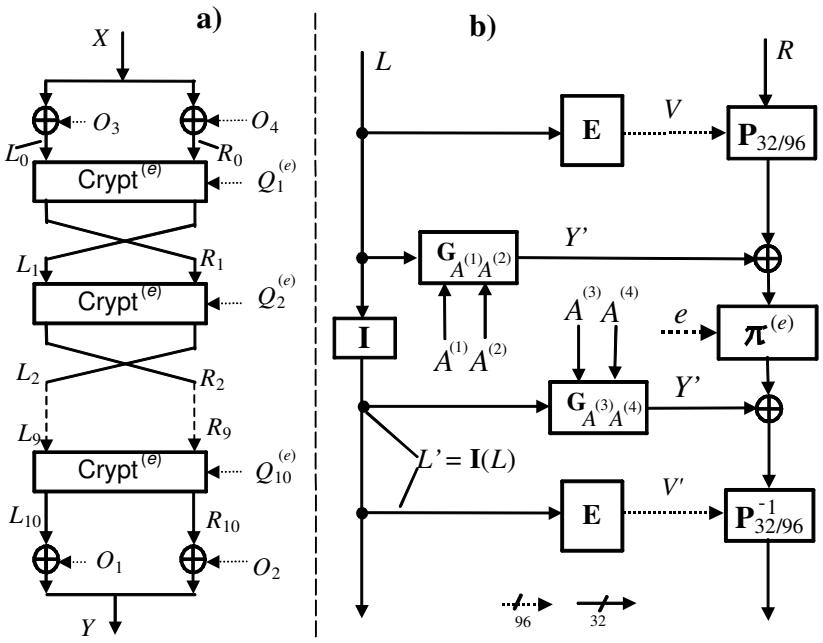

Figure 3. General structure of Cobra-H64 (a) and procedure $\mathbf{Crypt}^{(e)}$ (b)

fixed permutation equals about $t_{\oplus}$ (time delay of the XOR operation).

Cobra-H64 is a new ten-round iterated block cipher with 64-bit input and 128-bit secret key. The general encryption scheme is described by the following formulas: $C = \mathbf{T}^{(e=0)}(M, K)$ and $M = \mathbf{T}^{(e=1)}(C, K)$, where M is the plaintext, C is the ciphertext ($M, C \in \{0,1\}^{64}$), K is the secret key ($K \in \{0,1\}^{128}$), $\mathbf{T}$ is the transformation function, and $e \in \{0,1\}$ is a parameter defining encryption ($e = 0$) or decryption ($e = 1$) mode. The secret key is considered as concatenation of four 32-bit subkeys K_i, $i = 1, 2, 3, 4$: $K = (K_1, K_2, K_3, K_4)$. Cobra-H64 uses no preprocessing to transform subkeys. The extended key $Q^{(e)}$ is formed as simple sequence of subkeys K_i taken in respective order. Iterative structure of Cobra-H64 is shown in Fig. 3a and can

be described as follows. Encryption begins with initial transformation. Then 10 rounds with procedure $\mathbf{Crypt}^{(e)}$ (see Fig. 3b) followed by final transformation are performed.

First data block M is divided into two 32-bit subblocks L and R. Then initial transformation is executed which consists in XORing the data subblocks with 32-bit subkeys: $L_0 = L \oplus O_3$ and $R_0 = R \oplus O_4$. Ten consecutive encryption rounds are performed in accordance with the following algorithm:

1. For $j = 1$ to 9 do:
   1.1. Execute transformation

$$(L_j, R_j) = \mathbf{Crypt}^{(e)}(L_{j-1}, R_{j-1}, Q_j^{(e)}),$$

   1.2. Swap the data subblocks: $T := R_j$, $R_j := L_j$, $L_j := T$.
2. Execute transformation $(L_{10}, R_{10}) = \mathbf{Crypt}^{(e)}(L_9, R_9, Q_{10}^{(e)})$.

Procedure $\mathbf{Crypt}(e)$ represents round encryption function, where $Q_i^{(e)}$ is the 128-bit round key used in the ith encryption round. Encryption finishes with procedure of final transformation: $L' = L_{10} \oplus O_1$ and $R' = R_{10} \oplus O_2$. The ciphertext block is $C = (L', R')$.

3.1 Formation of the round keys

Each of the round keys $Q_i^{(e)}$ consists of four e-dependent round subkeys $A^{(1)}, A^{(2)}, A^{(3)}, A^{(4)} \in \{0,1\}^{32}$, i.e. $Q_i^{(e)} = (A^{(1)}, A^{(2)}, A^{(3)}, A^{(4)})_i^{(e)}$. Figure 4a and Table 1 specify round subkeys and their correspondence to the secret key. Subkeys K_i $(i = 1, ..., 4)$ are used directly in each round avoiding preprocessing of the secret key.

In each of ten rounds each of these subkeys is used while performing two operations $\mathbf{G}$. Execution of the operation $\mathbf{G}$ can be represented as performing IKS before subkeys are combined with the right data subblock. Taking into account that three inputs of the operation $\mathbf{G}$ are different one can see that the role of each subkey changes from one round to another one.

Change of the encryption mode is performed as simple swapping subkeys K_i with single-layer box $\mathbf{P}_{128/1(e)}$ which is represented by two

Table 1. Specification of the round subkeys

$i=$	1	2	3	4	5	6	7	8	9	10
$A^{(1)}=$	O_1	O_4	O_3	O_2	O_1	O_1	O_2	O_3	O_4	O_1
$A^{(2)}=$	O_2	O_1	O_4	O_3	O_4	O_4	O_3	O_4	O_1	O_2
$A^{(3)}=$	O_3	O_2	O_1	O_4	O_3	O_3	O_4	O_1	O_2	O_3
$A^{(4)}=$	O_4	O_3	O_2	O_1	O_2	O_2	O_1	O_2	O_3	O_4

parallel boxes $\mathbf{P}_{64/1(e)}$. The box $\mathbf{P}_{64/1(e)}$ is some single-layer CPB in which all elementary switching elements are controlled with the same bit e. The pairs (K_1, K_3) and (K_2, K_4) are inputs of the corresponding boxes $\mathbf{P}_{64/1(e)}$ (see Fig. 4a). Four 32-bit outputs of two boxes $\mathbf{P}_{64/1(e)}$ are the e-dependent subkeys O_j $(j = 1, 2, 3, 4)$.

Correct change of the encryption mode for the decryption one is also defined by the respective change of the fixed permutation $\pi^{(e)}$ described below.

Thus, we have $O_j = K_j$, if $e = 0$, and $O_1 = K_3$, $O_2 = K_4$, $O_3 = K_1$, $O_4 = K_2$, if $e = 1$.

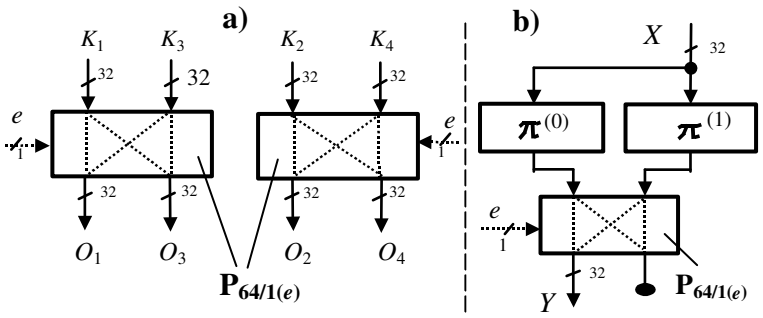

Figure 4. Swapping of subkeys (a) and the $\pi^{(e)}$ operation (b)

3.2 Data-dependent permutations

To perform DDP Cobra-H64 uses CP boxes of the second order $\mathbf{P}_{32/96(V)}$ and $\mathbf{P}^{-1}_{32/96(V')}$ described in section 2 (Fig. 2). To perform DDP corresponding to one round only 32 bits of the controlling data subblock are available, and for fixed key the DDP operations performed with CP boxes can have only $z \leq 2^{32} \ll 32!$ different modifications. Concrete set of 2^{32} CP-box modifications ordered correspondingly to the values of the controlling data subblock defines concrete kind of the DDP operation. Thus, different variants of DDP can be implemented using different mechanisms of the formation of the controlling vectors V and V'.

Let us consider some effective DDP performed on the right data subblock passing through the CP box $\mathbf{P}^{-1}_{32/96(V')}$ of the ith round, the involution $\mathbf{I}$ in the left branch of the cryptoscheme, and the CP box $\mathbf{P}_{32/96(V)}$ of the $(i+2)$th round, where $i = 1, ..., 8$. The values V and V' depending on all bits of the input data block define about 2^{64} different modifications of such "distributed" DDP represented by the superposition $\mathbf{P}^{-1}_{32/96(V')} \circ \mathbf{I} \circ \mathbf{P}_{32/96(V)}$.

3.3 Fixed permutations, $\pi^{(e)}$ and I

Switchable fixed permutation $\pi^{(e)}$ performs permutation $\pi^{(0)}$ when enciphering and $\pi^{(1)}$ when deciphering. Permutations $\pi^{(1)}$ and $\pi^{(0)}$ contain two cycles. The first cycle corresponds to identical permutation of the most significant input bit x_{32}. The second cycle is represented by the rotation of the bit string $(x_1, x_2, ..., x_{31})$ by 5 bits for $\pi^{(1)}$ and by 26 bits for $\pi^{(0)}$. The permutation $\pi^{(e)}$ improves the resultant DDP corresponding to performing sequential operations $\mathbf{P}_{32/96(V)}$ and $\mathbf{P}^{-1}_{32/96(V')}$. Indeed, even in the case $V = V'$ the superposition $\mathbf{P}_{32/96(V)} \circ \pi^{(e)} \circ \mathbf{P}^{-1}_{32/96(V')}$ forms an effective CP-box permutation all modifications of which are permutations having the same cycle structure (all modifications contain one cycle with length 1 and one cycle with length 31).

The structure of the switchable permutation is presented in Fig. 4b.

Possibility to change direct permutation for its inverse is provided by the use of the single-layer CP box $\mathbf{P}_{64/1(e)}$. Right (left) 32-bit input of the box $\mathbf{P}_{64/1(e)}$ is connected with output of the direct (inverse) permutation $\pi^{(0)}$ ($\pi^{(1)}$). The left output of $\mathbf{P}_{64/1(e)}$ is the output of the switchable permutation $\pi^{(e)}$.

Permutational involution $\mathbf{I}$ performed on the left data subblock is used to strengthen the avalanche effect. Let $y_{i'}$ and $y_{j'}$ be the output bits corresponding to the input bits x_i and x_j. To design involution $\mathbf{I}$ we have used the following two criteria: 1) $\forall i,j : |j-i| \leq 3$ should be $|j'-i'| \geq 4$; 2) $\forall i$ should be $|i-i'| \geq 6$.

These criteria are satisfied by the involution $\mathbf{I}$:

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

Due to this involution changing one bit of the left data subblock causes inversion from 2 to 8 bits of the right data subblock after the outputs of both operations $\mathbf{G}$ are XOR-ed with R. Note that permutation $\pi^{(e)}$ introduces no violation in this rule, since $\pi^{(e)}$ shifts the bits of R by 5 digit except the most significant bit (the right most one) which is not shifted. After combining outputs of the operations $\mathbf{G}$ with the right data subblock two bits of R change deterministically and 6 bits change statistically with probability 1/2 (see Table 4).

3.4 Formation of the controlling vector, V

Controlling vectors for the both CP boxes $\mathbf{P}_{32/96(V)}$ and $\mathbf{P}^{-1}_{32/96(V')}$ are formed using the same extension box $\mathbf{E}$ implemented with simple connections. The input of the $\mathbf{E}$-box is L. Let the vector $V = (V_1, V_2, V_3, V_4, V_5, V_6)$ be the output of the $\mathbf{E}$-box. The extension box provides the following relations: $V_1 = L_l$, $V_2 = L_l^{>>>6}$, $V_3 = L_l^{>>>12}$, $V_4 = L_h$, $V_5 = L_h^{>>>6}$, and $V_6 = L_h^{>>>12}$. Inverting a bit in L corresponds to the inversion of three bits of V. Thus, each bit of L influences three boxes $\mathbf{P}_{2/1}$ in the box $\mathbf{P}_{32/96(V')}$ and three $\mathbf{P}_{2/1}$-boxes in $\mathbf{P}_{32/96(V')}$. While designing the box $\mathbf{E}$ we used the following criterion: Permutation of each input bit of CPB must be defined by six different bits of L.

Table 2. Distribution of bits of the vector V

V_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
V_2	7	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6
V_3	13	14	15	16	1	2	3	4	5	6	7	8	9	10	11	12
V_4	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
V_5	23	24	25	26	27	28	29	30	31	32	17	18	19	20	21	22
V_6	29	30	31	32	17	18	19	20	21	22	23	24	25	26	27	28

Due to realization of this criterion each bit of L influences exactly six bits of R. It is easy to see that such distribution of the controlling bits provides that arbitrary input bit of the boxes $\mathbf{P}_{32/96}$ and $\mathbf{P}_{32/96}^{-1}$ moves to each output position with the same probability, if L is a uniformly distributed random variable. Correspondence between bits of the controlling data subblock L and elementary switching elements is given in the Table 2.

3.5 Non-linear operation G

To develop nonlinear operation $\mathbf{G}$ performing transformations $Y = \mathbf{G}(L, A^{(1)}, A^{(2)})$ (or $Y = \mathbf{G}(L, A^{(3)}, A^{(4)})$) we have used the sequential model of the CTPO [3] in the following form:

$$(y_1, ..., y_{32}) = \left(f_1^{(L)}(A^{(1)}, A^{(2)}), ..., f_{32}^{(L)}(A^{(1)}, A^{(2)})\right),$$

where for all i the generating functions f_i have the form $f_i = a_i^{(1)} \oplus f_i'(a_1^{(1)}, ..., a_{i-1}^{(1)}, A^{(2)}, L)$. This provides that mapping $A^{(1)} \to Y$ is bijective. To provide also the mapping $L \to Y$ to be bijective we use the functions f' with analogous structure:

$$f_i' = l_i \oplus \phi_i(a_1^{(1)}, ..., a_{i-1}^{(1)}, l_1, ..., l_{i-1}, A^{(2)}, L).$$

In the general case each function f_i can be represented in a unique form for each value i including different numbers of variables. In operation $\mathbf{G}$ we use a unified function with fixed number of variables:

$$f = f(z_1, ..., z_8) = z_8 \oplus z_7 \oplus \phi(z_1, ..., z_6),$$

where $\phi = z_1z_2 \oplus z_2z_3 \oplus z_3z_1 \oplus z_4z_2 \oplus z_5z_1 \oplus z_6z_2z_3$.

Using some substitution of variables in the general form of the function f one can obtain concrete unified generating functions. In the operation $\mathbf{G}$ for all $i \in \{1, ..., 32\}$ we use the following substitution:

$$\begin{pmatrix} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 \\ l_{i-3} & l_{i-2} & l_{i-1} & a_{i-1}^{(2)} & a_{i-1}^{(1)} & a_i^{(2)} & a_i^{(1)} & l_i \end{pmatrix}.$$

Thus, we have the following generating functions:

$$y_i = f_i = l_i \oplus a_i^{(1)} \oplus l_{i-3}l_{i-2} \oplus l_{i-2}l_{i-1} \oplus l_{i-3}l_{i-1} \oplus a_{i-1}^{(2)}l_{i-2} \oplus a_{i-1}^{(1)}l_{i-3} \oplus$$

$$\oplus a_i^{(2)}l_{i-2}l_{i-1},$$

where l_j, $a_j^{(1)}$, $a_j^{(2)}$ are components of the vectors $L, A^{(1)}, A^{(2)} \in \{0,1\}^{32}$, respectively, and $(l_{-2}, l_{-1}, l_0) = (1, 1, , 1)$, $a_0^{(1)} = a_0^{(2)} = 1$.

Operation $\mathbf{G}$ can be represented in the vector form by the following expression:

$$Y = L_0 \oplus A_0^{(1)} \oplus (L_3 \otimes L_2) \oplus (L_2 \otimes L_1) \oplus (L_3 \otimes L_1) \oplus (A_1^{(2)} \otimes L_2) \oplus (A_1^{(1)} \otimes L_3) \oplus$$

$$\oplus (A_0^{(2)} \otimes L_2 \otimes L_1),$$

where $\otimes$ denotes bit-wise AND operation and binary vectors L_j, $A_j^{(1)}$, and $A_j^{(2)}$ are expressed as follows:

$$L_0 = L = (l_1, l_2, ..., l_{32}),\ L_1 = (1, l_1, l_2, ..., l_{31}),\ L_2 = (1, 1, l_1, l_2, ..., l_{30}),$$
$$L_3 = (1, 1, 1, l_1, l_2, ..., l_{29}),\ A_0^{(1)} = A^{(1)} = (a_1^{(1)}, a_2^{(1)}, ..., a_{32}^{(1)}),$$
$$A_1^{(1)} = (1, a_1^{(1)}, a_2^{(1)}, ..., a_{31}^{(1)}),\ A_0^{(2)} = A^{(2)} = (a_1^{(2)}, a_2^{(2)}, ..., a_{32}^{(2)}),$$
$$A_1^{(2)} = (1, a_1^{(2)}, a_2^{(2)}, ..., a_{31}^{(2)}),\ A_2^{(2)} = (1, 1, a_1^{(2)}, a_2^{(2)}, ..., a_{30}^{(2)}).$$

In the operation $\mathbf{G}_{A^{(3)},A^{(4)}}$ the operands $A^{(3)}$ and $A^{(4)}$ play correspondingly the role of the operands $A^{(1)}$ and $A^{(2)}$ in the operation $\mathbf{G}_{A^{(1)},A^{(2)}}$ described above.

4 Discussion

4.1 Peculiarities of Cobra-H64

Cipher Cobra-H64 presents an example of the extensive use of the CP-box operations. They are used in three different ways: 1) as DDP that are one of two basic cryptographic primitives, 2) to swap subkeys when changing encryption mode for decryption one, and 3) to switch permutation $\pi^{(e)}$ when changing ciphering mode.

The second basic cryptographic primitive is represented by operation $\mathbf{G}$. Due to high parallelism of the general structure of Cobra-H64, to perform one round of Cobra-H64 takes only about $15t_{\oplus}$. Time delay of ten rounds is about $150t_{\oplus}$. Encryption speed can be estimated as ≈ 0.43 bit$/t_{\oplus}$. This figure is close to that of the cipher SPECTR-H64 (≈ 0.44 bit$/t_{\oplus}$).

Analogously to SPECTR-H64 the cryptosystem Cobra-H64 is fast in the case of frequent change of keys, since it is free of the key preprocessing. The Cobra-H64 has the following features:

1. In each round it uses the secret key entirely.

2. Internal key scheduling in Cobra-H64 is implemented with two CTPO executed in parallel with the CP-box operation $\mathbf{P}_{32/96}$.

3. Round transformation includes special permutational involution performed on the left data subblock.

4. Round transformation includes special switchable permutation $\pi^{(e)}$ performed on the right data subblock. Permutation $\pi^{(e)}$ prevents appearance of the weak keys with the structure $K = (X, X, X, X)$, where $X \in \{0,1\}^{32}$.

5. After the ith round each bit of R_{i-1} influences statistically all bits of L_i.

6. For operations $\mathbf{G}$, $\mathbf{P}_{32/96}$, and $\mathbf{P}_{32/96}^{-1}$ it is quite easy to calculate characteristics corresponding to differences with few number of active bits.

Drawing an analogy with other DDP-based cryptosystems: SPECTR-H64 [9] and CIKS-1 [10], we estimate the cipher Cobra-H64 suits well to cheap and fast hardware implementation. While implemented in hardware the DDP-based ciphers are competitive with

AES [10, 11] that is superior to majority of known ciphers.

4.2 Some properties of the controlled operations

Let $\delta_{h|i_1,...,i_n}$ (or simply δ_h) be the difference with h active (non-zero) bits and $i_1, ..., i_h$ be the numbers of digits corresponding to active bits. Let $p(\delta \stackrel{\mathbf{G}}{\rightarrow} \delta')$ and $p(\delta \stackrel{r}{\rightarrow} \delta')$ be the probabilities that input difference δ passing the operation $\mathbf{G}$ and rth encryption round transforms into output difference δ'.

Avalanche effect corresponding to DDP is caused by the use of data to define the values V and V'. Each bit of the left data subblock influences three bits of each controlling vector. Each controlling bit influences two bits of the right data subblock. Thus, due to DDP one bit of L influences statistically about 12 bits of R.

In the case when some difference with one active bit $\delta^L_{1|i}$ passes through the left branch of the cryptoscheme it influences three elementary switching elements permuting six different bits of the right data subblock. The elementary box $\mathbf{P}_{2/1}$ controlled with active bit we shall call active. If the input difference δ^R of the CP box $\mathbf{P}_{32/96}$ has no active bits (i.e. we have δ^R_0), then the difference $\delta^L_{1|i}$ can cause the following events depending on the right data subblock:

1) generate no difference (or generate at the output of $\mathbf{P}_{32/96}$ the difference ${\delta'}^R_0$) with probability 2^{-3};

2) generate difference ${\delta'}^R_2$ with probability $3 \cdot 2^{-3}$;

3) generate difference ${\delta'}^R_4$ with probability $3 \cdot 2^{-3}$;

4) generate difference ${\delta'}^R_6$ with probability 2^{-3}.

Average probabilities $p(\delta_q \stackrel{\mathbf{P}_{32/96}}{\rightarrow} \delta'_g)$ corresponding to input and output differences of the $\mathbf{P}_{32/96}$-box with several active bits ($q = 0, 1, 2$ and $g = q + a$, where $a = -2, 0, 2, 4$) are presented in Table 3. These probabilities depend on the number of active bits in the difference passing through the left branch of the encryption round. The values $p(\delta_{q|i_1,...,i_q} \rightarrow \delta_g)$ have been calculated assuming that all sets of numbers $i_1, ..., i_q$ corresponding to positions of the input active bits are equiprobable. (Note that positions of the active bits of the output dif-

Table 3. Average probabilities $p(\delta_q \overset{\mathbf{P}_{32/96}}{\rightarrow} \delta'_g)$

δ^L	$\delta_0 \rightarrow \delta'_0$	$\delta_0 \rightarrow \delta'_2$	$\delta_0 \rightarrow \delta'_4$	$\delta_1 \rightarrow \delta'_1$	$\delta_1 \rightarrow \delta'_3$
δ_0	1	0	0	1	0
δ_1	2^{-3}	$1.5 \cdot 2^{-2}$	$1.5 \cdot 2^{-2}$	$1.17 \cdot 2^{-3}$	$1.59 \cdot 2^{-2}$
δ_2	2^{-6}	$1.5 \cdot 2^{-4}$	$1.88 \cdot 2^{-3}$	$1.38 \cdot 2^{-6}$	$1.88 \cdot 2^{-4}$

δ^L	$\delta_1 \rightarrow \delta'_5$	$\delta_2 \rightarrow \delta'_0$	$\delta_2 \rightarrow \delta'_2$	$\delta_2 \rightarrow \delta'_4$	$\delta_2 \rightarrow \delta'_6$
δ_0	0	0	1	0	0
δ_1	$1.41 \cdot 2^{-2}$	$1.55 \cdot 2^{-11}$	$1.08 \cdot 2^{-3}$	$1.36 \cdot 2^{-2}$	$1.15 \cdot 2^{-2}$
δ_2	$1.06 \cdot 2^{-2}$	$1.55 \cdot 2^{-13}$	$1.38 \cdot 2^{-6}$	$1.69 \cdot 2^{-4}$	$1.72 \cdot 2^{-3}$

Table 4. Formulas describing avalanche in the operation **G** corresponding to changing the single input bit l_i ($\Delta l_i = 1$)

#	Expression	Probability
1	$\Delta y_i = \Delta l_i$	$p(\Delta y_i = 1) = 1$
2	$\Delta y_{i+1} = \Delta l_i (l_{i-1} \oplus l_{i-2} \oplus a^{(2)}_{i+1} l_{i-1})$	$p(\Delta y_{i+1} = 1) = 1/2$
3	$\Delta y_{i+2} = \Delta l_i (l_{i-1} \oplus l_{i+1} \oplus a^{(2)}_{i+1} \oplus a^{(2)}_{i+2} l_{i+1})$	$p(\Delta y_{i+2} = 1) = 1/2$
4	$\Delta y_{i+3} = \Delta l_i (l_{i+1} \oplus l_{i+2} \oplus a^{(1)}_{i+2})$	$p(\Delta y_{i+3} = 1) = 1/2$

ference are not fixed.) One can show that for arbitrary input difference δ_q of the CP box and corresponding output difference δ'_g the relation $q \oplus g = 0$ always holds.

Avalanche effect corresponding to the operations **G** is defined by its structure which provides each bit of $l_1, l_2, ..., l_{29}$ influences four output bits. Bits from l_{30} to l_{32} influence different number u of output bits (see Tables 4 and 5). When performing operation **G**, change of the ith input bit ($\Delta l_i = 1$) causes deterministic change of the ith output bit ($\Delta y_i = \Delta l_i$), and probabilistic change of the $(i+1)$th, $(i+2)$th, and $(i+3)$th output bits (see Table 4).

From the formulas presented in Table 4 one can see that alteration

Table 5. Values of the probability $p(\delta_{1/i} \stackrel{\mathbf{G}}{\rightarrow} \delta'_g)$

i	u	$\delta_{1\|i} \rightarrow \delta'_{1\|i}$	$\ldots\delta'_{2\|i,i+1}$	$\ldots\delta'_{2\|i,i+2}$	$\ldots\delta'_{2\|i,i+3}$	$\ldots\delta'_2$	$\ldots\delta'_3$	$\ldots\delta'_4$
1-29	4	2^{-3}	2^{-3}	2^{-3}	2^{-3}	$1.5 \cdot 2^{-2}$	$1.5 \cdot 2^{-3}$	2^{-3}
30	3	2^{-2}	2^{-2}	2^{-2}	-	2^{-1}	2^{-2}	-
31	2	2^{-1}	2^{-1}	-	-	2^{-1}	-	-
32	1	1	-	-	-	-	-	-

of the input bit l_i causes deterministic alteration of the output bit y_i and probabilistic alteration of the output bits y_{i+1}, y_{i+2}, and y_{i+3}. For uniformly distributed random variable L and $i \leq 29$ the output bits y_{i+1}, y_{i+2}, and y_{i+3} change with probability $p = 0.5$. Values of the probability $p(\delta_{1|i} \stackrel{\mathbf{G}}{\rightarrow} \delta'_{1|i})$ corresponding to different i are given in Table 5, where u is the number of the output bits depending on the ith input bit.

5 Differential cryptanalysis

We have considered different variants of the differential cryptanalysis (DCA). We have obtained that the fewer active bits in the difference the higher the probability of the differential characteristic. The best our attack corresponds to two-round difference with one active bit. Let $\delta = (\delta^L_0, \delta^R_{1|k})$, where $1 \leq k \leq 32$, be the input difference. For all k the difference $\delta = (\delta^L_0, \delta^R_{1|k})$ passing two rounds transforms with probability $P(2)$ into the output difference $\delta = (0, \delta_{1|j})$, where $1 \leq j \leq 32$. In the first round the active bit passes the right branch with probability 1. In the second round the active bit passes the left branch and do not generate any active bits in the right branch with probability $P(2)$. The following two cases contribute mostly to the probability of this two-round characteristic.

Case 1. Each of two operations $\mathbf{G}$ produces at its output the difference with one active bit. The $\mathbf{P}_{32/96}$-box produces transformation $\delta^R_0 \rightarrow \delta^R_{2|i,j'}$, where $j' = \pi^{(e \oplus 1)}(j)$ and $j = \mathbf{I}(i)$. Then difference $\delta^R_{2|i,j'}$ after XORing with two output differences of the both operations $\mathbf{G}$ gives

zero difference. Zero difference passes the $\mathbf{P}_{32/96}^{-1}$-box with probability 2^{-3}.

Case 2. Each of two operations $\mathbf{G}$ produces at its output difference with one active bit. The $\mathbf{P}_{32/96}$-box produces at the output zero difference. Then XORing two outputs of the operation $\mathbf{G}$ with the right data subblock produces two-bit difference $\delta_{2|i',j}^{(R)}$, where $i' = \pi^{(e)}(i)$ and $j = \mathbf{I}(i)$. Then passing the $\mathbf{P}_{32/96}^{-1}$-box the difference $\delta_{2|i',j}^{(R)}$ transforms into zero difference.

Let us denote probability of the event $(\delta_0^L, \delta_1^R) \xrightarrow{2} (\delta_0^L, \delta'^R_1)$ corresponding to cases 1 and 2 as P' and P'', respectively, i.e. $P(2) \approx P' + P''$. (We neglect small contribution of other events corresponding to the formation of two and more pairs of active bits at outputs of some active $\mathbf{P}_{2/1}$-boxes in one of the CP boxes $\mathbf{P}_{32/96}^{-1}$ or $\mathbf{P}_{32/96}$.)

Let us consider Case 1 that is shown in Fig. 5. The difference passes two decryption rounds in the following way. In the first round the difference $\delta_{1|k}^R$ goes through the right branch of the cryptoscheme and transforms into the difference $\delta_{1|i}^R$ ($i \in \{1, ..., 32\}$) with probability 1. After swapping subblocks we have $\delta_{1|i}^L$ and δ_0^R. In the second round the active bit goes through the left branch producing difference $\delta_{1|i}$ at the output of the upper operation $\mathbf{G}$ (event A_1) with probability $p_1 = 2^{-3}$ and difference $\delta_{1|j}$ at the output of the lower operation $\mathbf{G}$ (event A_2) with probability p_2. At the same time the difference $\delta_{1|i}^L$ causes formation of the two-bit difference $\delta_{2|i,j'}$ at the output of the operation $\mathbf{P}_{32/96}$ (event A_3) with probability p_3.

After being XORed with difference $\delta_{1|i}$ the difference $\delta_{2|i,j'}^R$ transforms in $\delta_{1|j'}^R$. Passing fixed permutation $\pi^{(1)}$ difference $\delta_{1|j'}^R$ transforms deterministically in $\delta_{1|j}^R$ producing zero difference $\delta_0^R = 0$ after XORing with output of the lower operation $\mathbf{G}$. Zero difference passes the operation $\mathbf{P}_{32/96}^{-1}$ (event A_4) with probability $p_4 = 2^{-3}$. After swapping we have the difference $(\delta_0^L, \delta_{1|j}^R)$ at the output of the second round.

Values p_1, p_2, p_3, p_4 are presented in Table 6, where each row corresponds to different subsets of numbers i and p_i is probability that for given $\delta_{1|i}^L$ zero difference passes the right branch of the second round.

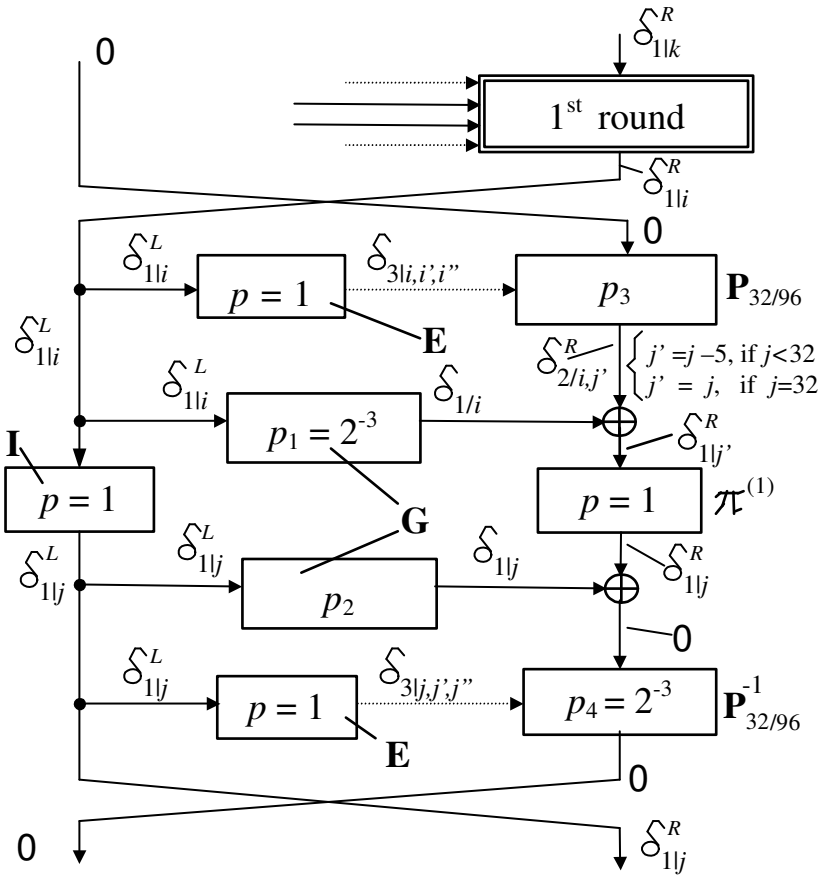

Figure 5. Difference $(0, \delta_{1|k})$ passing through two decryption rounds (Case 1)

Each subset is characterized by equal values p_1, p_2, p_3 and p_4. Probability $p_i = p\left((\delta^L_{1|i}, 0) \xrightarrow{2} (0, \delta^R_{1|j})\right) = p_1 p_2 p_3 p_4$ corresponds to the event that for given i after the second round and swapping left and right data subblocks input difference $(\delta^L_{1|i}, 0)$ transforms into output difference $(0, \delta^R_{1|j})$, where i is fixed and $1 \le j \le 32$. Considering that the key and the left data subblock are independent random values it is easy to see that events A_1, A_2, A_3 and A_4 are pairwise independent. In the estimation given below we neglect some weak interdependence between events A_3 and A_4. The probability P' can be calculated with the following formula

$$P' = p(k \to i) \sum_{i=1}^{32} p_i \approx 1.33 \cdot 2^{-20}.$$

For $i = 13$ Table 6 gives "approximated" values p_1, p_2, and p_3. (Calculation of the contribution of the value $i = 13$ has some peculiarities, since output bits y'_{15} and y'_{16} of the upper operation **G** after permutation $\pi^{(1)}$ are XORed with output bits y''_{20} and y''_{21} of the lower operation **G**. Thus, for $i = 13$ the following differences $\delta^R_{2|13,14}$, $\delta^R_{2|13,15}$, $\delta^R_{2|13,16}$, $\delta^R_{2|13,17}$, and $\delta^R_{2|13,18}$ contribute to P'. The resultant contribution is $\approx 1,7 \cdot 2^{-23}$.)

Calculation of the probability p_3 is not difficult, but require to consider each elementary switching element of the box $\mathbf{P}_{32/96}$ and all connections between active layers of the last. For example, let us consider the difference $\delta^L_{1|5}$. The 5th bit controls one elementary box $\mathbf{P}_{2/1}$ in each of three upper active layers of the CP box $\mathbf{P}_{32/96}$ (see Table 2), namely the 5th, 31st, and 41st boxes $\mathbf{P}_{2/1}$. For $i = 5$ we have $j = 18$ in correspondence with the operations **I** and $\pi^{(1)}$. The following three different variants corresponding to the generation of the output difference $\delta^R_{2|5,13}$ are to be taken into account:

1. Depending on the value R input bits of the 5th elementary box are different with probability $p = 0.5$, input bits corresponding to the 31st elementary box are equal ($p = 0.5$), input bits corresponding to the 41st elementary box are equal ($p = 0.5$).

Table 6. Probabilities corresponding to different values $i \in \{1, ..., 32\}$

i	p_1	p_2	p_3	p_4	p_i	$p(k \to i)$
1,5	2^{-3}	2^{-3}	$1.06 \cdot 2^{-9}$	2^{-3}	$1.06 \cdot 2^{-18}$	2^{-5}
2,9	2^{-3}	2^{-3}	2^{-13}	2^{-3}	2^{-22}	2^{-5}
8	2^{-3}	2^{-2}	$1.25 \cdot 2^{-11}$	2^{-3}	$1.25 \cdot 2^{-19}$	2^{-5}
12	2^{-3}	2^{-1}	$1.25 \cdot 2^{-11}$	2^{-3}	$1.25 \cdot 2^{-18}$	2^{-5}
13*	2^{-2}	2^{-2}	$1.7 \cdot 2^{-11}$	2^{-3}	$1.7 \cdot 2^{-18}$	2^{-5}
16	2^{-3}	1	$1.25 \cdot 2^{-11}$	2^{-3}	$1.25 \cdot 2^{-17}$	2^{-5}
$17 \le i \le 32$	$\neq 0$	2^{-3}	0	2^{-3}	0	2^{-5}
other i	2^{-3}	2^{-3}	$1.25 \cdot 2^{-11}$	2^{-3}	$1.25 \cdot 2^{-20}$	2^{-5}

2. Input bits corresponding to the 31st elementary box are different ($p = 0.5$), input bits corresponding to the 5th elementary box are equal ($p = 0.5$), input bits corresponding to the 41st elementary box are equal ($p = 0.5$).

3. Input bits corresponding to the 41st elementary box are different ($p = 0.5$), input bits corresponding to the 5th elementary box are equal ($p = 0.5$), input bits corresponding to the 31st elementary box are equal ($p = 0.5$).

Since each bit of the subblock L controls six different bits of R the probability of each of these events is exactly 2^{-3}. One of output bits of the 5th elementary box moves to the 5th digit at the output of the CP box $\mathbf{P}_{32/96}$ with probability 2^{-5} (it must pass five active layers having the single route). The same probability corresponds to the second output bit of the 5th elementary box to be moved to the 13th output of $\mathbf{P}_{32/96}$. Thus, we have probability $p' = (2^{-5})^2 2^{-3} = 2^{-13}$ that the first event generates difference $\delta^R_{2|5,13}$. Probability that the second event generates difference $\delta^R_{2|5,13}$ is $p'' = 0$ (input bits of the 31st elementary box never move simultaneously to 5th and 13th output digits of $\mathbf{P}_{32/96}$). Probability that the third event generates difference $\delta^R_{2|5,13}$ is $p''' = 2^{-9}$. The resultant probability that difference

$\delta^L_{1|5}$ produces difference $\delta^R_{2|5,13}$ is $p \approx p' + p'' + p''' \approx 1.06 \cdot 2^{-9}$. In analogous way one can consider differences $\delta^L_{1|i}$ for $i = 1, ..., 32$. Note that differences $\delta^L_{1|i}$, where $17 \leq i \leq 32$ do not contribute to P'.

Case 2 is analogous to the Case 1, except upper CP box produces zero output difference and two active bits are annihilated with probability 0.5 when they pass through the same active elementary switching element of the lower CP box. Contribution of this case can be calculated analogously to Case 1. For Case 2 one can obtain $P'' \approx 2^{-20}$ and then $P(2) \approx P' + P'' \approx 1.33 \cdot 2^{-20} + 2^{-20} \approx 1.16 \cdot 2^{-19}$. For comparison we have also applied this technique of the DCA to SPECTR-H64 and have obtained $P(2) = 1.15 \cdot 2^{-13}$ [12]. Using values $P(2)$ we can find that Cobra-H64 for $r \geq 8$ and SPECTR-H64 with $r \geq 10$ rounds are indistinguishable from a random cipher with differential attack.

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 keys $K' = (X, Y, X, Y)$ or $K'' = (X, X, X, X)$, where $X, Y \in \{0,1\}^{32}$, are not weak, since encryption and decryption require change of the parameter e (indeed, from Fig. 3 it is easy to see that $\mathbf{T}^{(e=0)}(C, K'') \neq M$, where $C = \mathbf{T}^{(e=0)}(M, K'')$). It seems to be difficult to calculate a semi-weak key-pair for presented ciphers, if it is possible at all. This shows the other role of the switchable permutation $\pi^{(e)}$. For example, for SPECTR-H64 which uses no switchable operations for all X the 256-bit key $K = (X, X, X, X, X, X, X, X)$ is weak.

Conclusion

1. Controlled two-place operations and DDP appear to be efficient cryptographic primitives.

2. Cobra-H64 is a fast cipher when implementing in cheap hardware.

3. It is fast in the case of frequent change of keys.

4. The use of switchable operations prevents weak keys in the case when no key scheduling precomputation is used.

5. While designing DDP-based ciphers much attention should be payed to the structure of the **E**-box.

References

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

[2] A.A. Moldovyan. *Fast block ciphers based on controlled permutations.* Computer Science Journal of Moldova. 2000, vol. 8, no. 3, pp. 270–283.

[3] B.V.Izotov, A.A.Moldovyan, N.A.Moldovyan. *Controlled operations as a cryptographic primitive,* MMM-2001 Proceedings. LNCS, Springer-Verlag, 2001, vol. 2052, pp. 230–241.

[4] Goots N.D., Moldovyan A.A., Moldovyan N.A. *Fast encryption algorithm SPECTR-H64,* MMM-2001 Proceedings. LNCS, Springer-Verlag, 2001, vol. 2052, pp. 275–286.

[5] V.E. Benes. *Mathematical theory of connecting networks and telephone trafic,* Academic Press, New York, 1965.

[6] C. Clos. *A study of nonblocking switching networks,* Bell System Technical J. 1953, vol.32, pp. 406–424.

[7] M. Portz. *A generallized description of DES-based and Benes-based Permutationgenerators.* LNCS, Springer-Verlag, vol.718. (1992) pp. 397–409.

[8] M. Kwan. *The design of the ICE encryption algorithm,* FSE'97 Proceedings. LNCS, Springer-Verlag, vol. 1267 (1997), pp. 69–82.

[9] O. Koufopavlou, N. Sklavos and P. Kitsos. *Cryptography in Wireless Protocols: Hardware and Software Implementations,* tutorials

proceedings of IEEE International Symposium on Circuits & Systems (ISCAS'03), Volume II, pp. 383–424, Thailand, May 25–28, 2003.

[10] N. Sklavos, A. A. Moldovyan and O. Koufopavlou. *Encryption and Data Dependent Permutations: Implementation Cost and Performance Evaluation.* MMM-2003 Proceedings. LNCS, Springer-Verlag, vol. 2776 (2003), pp. 343–354.

[11] N. Sklavos and O. Koufopavlou. *Architectures and VLSI Implementations of the AES-Proposal Rijndael,* IEEE Transactions on Computers, 2002, Vol. 51, Issue 12, pp. 1454–1459.

[12] B.V. Izotov, N.D. Goots, A.A. Moldovyan and N.A. Moldovyan *Fast Ciphers for Cheap Hardware: Differential Analysis of SPECTR-H64,* MMM-2003 Proceedings. LNCS, Springer-Verlag, vol. 2776 (2003), pp. 454–457.

Received March 3, 2003

N.A. Moldovyan,

Specialized Center of Program Systems "SPECTR",
Kantemirovskaya str., 10, St. Petersburg 197342, Russia;
ph./fax.7–812–2453743,
E–mail: *nmold@cobra.ru*