Computer Science Journal of Moldova, vol.**8**, no.3(24), 2000

Fast block ciphers based on controlled permutations

A.A. Moldovyan

Abstract

This paper describes two fast hardware-oriented block ciphers based on combination of the data-dependent and key-dependent permutations. It also introduces a data-dependent transformation of subkeys as an internal key scheduling.

Key words: fast encryption, data-dependent permutation, flexible hardware cipher, internal key scheduling

1 Introduction

Design of ciphers based on data-dependent operations appears to be a new and perspective direction in applied cryptography. Using data-dependent rotations R.Rivest designed extremely simple cryptosystem RC5 [1]. Since RC5 was proposed no practical attacks on this cipher have been found. Different studies [2-3] have provided a good understanding of how RC5's structure and data-dependent rotations contribute to its security. It have been shown that the mixed use of data-dependent rotations and some other simple operations is a very effective way of thwarting differential and linear cryptanalysis. Since several studies provide some theoretical attacks based on the fact that few bits in a register define selection of concrete modification of the current rotation operation, in new 128-bit ciphers MARS [4] and RC6 [5] data-dependent rotations are combined with integer multiplication.

In the case of n-bit registers containing encrypted data subblocks there are available n different modifications of the rotation operation and consequently only $\log_2 n$ bits can be used directly as controlling ones. Rotations are particular modifications of the permutation operation. It is known that a fixed permutation is a linear operation, however permutations depending on encrypted data are non-linear ones and can be used as basic cryptographical primitive in the design of fast hardware-oriented ciphers. The number of possible permutations significantly exceeds the number of different rotations, therefore it is very attractive to use data-dependent permutations (DDP) for increasing the number of the controlling bits [6]. Key-dependent permutations (KDP) are the another case of the controlled permutations (CP).

In present paper it is considered a design approach based on the use of the CP with a large number ($\geq 2^n$) of different modifications. Some variants of the CP operations and two fast iterated cryptoschemes are proposed.

2 Implementation of the controlled permutations

Variable permutations are difficult to be executed on modern microprocessors and we assume the CP operations to be hardware implemented. A box $P_{n/m}$ executing the CP operation (CP-box or simply $P_{n/m}$-box) is shown in Fig. 1(a), where X is an input n-bit string, V is a controlling m-bit string which defines the current variant of the permutation operation. It is supposed the value V to be generated in dependence on encrypted data and/or key. Concrete type of the CP-box permutation $P_{n/m}$ is characterized by an ordered set $\{\Pi_0, \Pi_1, ..., \Pi_{2^m-1}\}$, where all Π_V, $V = 0, 1, ..., 2^m - 1$, are fixed permutations which are to be performed above binary vectors X. Such fixed permutations we shall call the CP-modifications. To execute operation $P_{n/m(V)}(X)$ one performs permutation Π_V obtaining value $\Pi_V(X)$ which is taken as the output $Y = P_{n/m(V)}(X) = \Pi_V(X)$.

Since arbitrary permutation can be represented as a superposition of the transpositions of bits, the CP-boxes can be constructed on the basis of the utilization of the standard elementary CP-boxes $P_{2/1}$ (Fig. 1(b)) which are used, for example, in the ICE encryption

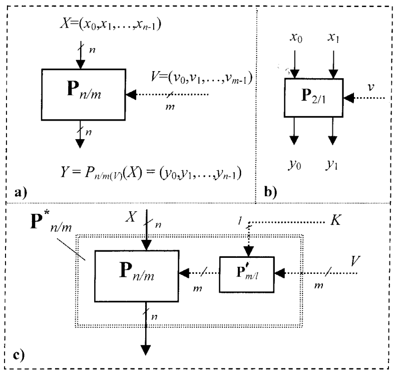
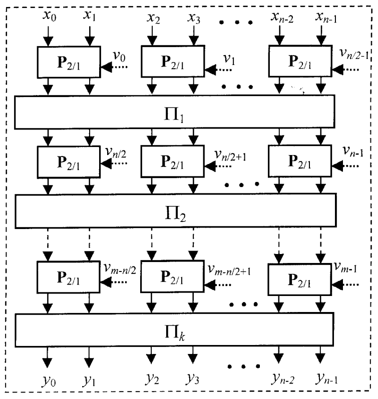


Figure 1. CP-boxes $P_{n/m}$(a) and $P_{2/1}$(b) and key-dependent CP-box(c)

algorithm [7]. Each $P_{2/1}$-box is controlled by one bit v. If $v = 0$ two input bits are swapped: $y_0 = x_1$ and $y_1 = x_0$. For $v = 1$ input bits are not swapped: $y_0 = x_0$ and $y_1 = x_1$. In general case, if the set $\{\Pi_0, \Pi_1, ..., \Pi_{2^m-1}\}$ is permuted, then the respective CP-box will be modified. This can be used for construction of the controlled DDP-boxes as it is shown in Fig. 1(c), where data-dependent controlling bit string V is permuted by an additional CP-box $P'_{m/l}$. Using a subkey as the controlling bit string of the $P'_{m/l}$-box one can get a key-dependent DDP-box $P^*_{n/l}$.

CP-boxes can be constructed on the basis of the simple cascade structure shown in Fig. 2, where Π_i, $i = 1, 2, ...k$, are fixed permutations. The number of cascades k define the time delay corresponding to the execution of the CP-box permutation. Selecting different variants of fixed permutations one can define different types of the CP-box permutations for given k. For CP-box with cascade structure we have relation $m = nk/2$. For $k = 1$ the number of different modifications is $2^{n/2}$ and the time delay of the CP-box permutation is approximately equal to that of the EXCLUSIVE-OR (XOR) operation ($\oplus$).



x_0 x_1 x_2 x_3 x_{n-2} x_{n-1}
$\mathbf{P}_{2/1}$
v_0 v_1 $v_{n/2-1}$
Π_1
$v_{n/2}$ $v_{n/2+1}$ v_{n-1}
Π_2
$v_{m-n/2}$ $v_{m-n/2+1}$ v_{m-1}
Π_k
y_0 y_1 y_2 y_3 y_{n-2} y_{n-1}

Figure 2. CP-box operation $P_{n/m}$ with cascade structure

Different kinds of controlled permutations are possible. One can use the following construction criteria:

1) simplicity of the hardware implementation;

2) realization of large number of the CP-modifications;

3) realization of unique (pairwise inequivalent) CP-modifications;

4) realization of special CP-modifications.

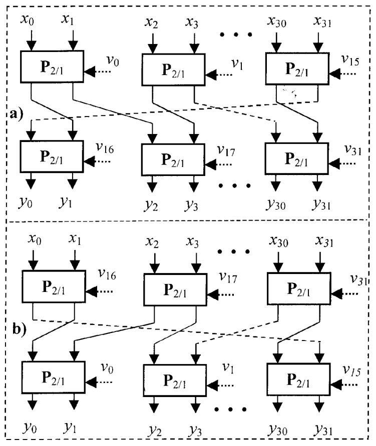

a)
b)
x_0 x_1 x_2 x_3 x_{30} x_{31}
$\mathbf{P}_{2/1}$
v_0 v_1 v_{15} v_{16} v_{17} v_{31}
y_0 y_1 y_2 y_3 y_{30} y_{31}

Figure 3. Structure of the CP-boxes $P_{32/32}$ (a) and $P^{-1}_{32/32}$ (b)

Concrete types of cascade CP-boxes corresponding to criteria 1-3 are presented in Fig. 3. Two CP-boxes $P_{n/m}$ and $P^{-1}_{n/m}$ we shall call mutually inverse, if for all possible values of the vector V the respective CP-modifications Π_V and Π_V^{-1} are mutually inverse. For example, the CP-boxes $P_{32/32}$ (Fig. 3(a)) and $P^{-1}_{32/32}$ (Fig. 3(b)) are mutually inverse.

Fig. 4(a) represents the CP-box $P_{32/64}$ which consists of two CP-boxes $P_{32/32}$ and one fixed permutation Π between them. Fixed permutation Π is specified in Table 1 that shows what output bits (o.b.) correspond to which input bits (i.b).

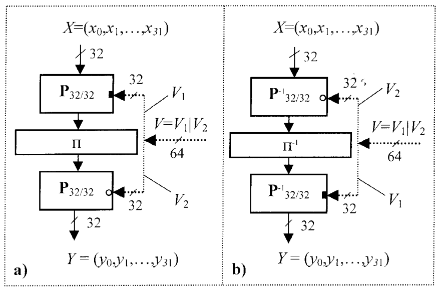

$X=(x_0,x_1,\ldots,x_{31})$
32
$\mathbf{P}_{32/32}$
32
V_1
$V=V_1|V_2$
Π
64
$\mathbf{P}_{32/32}$
32
V_2
32
$Y=(y_0,y_1,\ldots,y_{31})$
a)
$X=(x_0,x_1,\ldots,x_{31})$
32
$\mathbf{P}^{-1}_{32/32}$
32
V_2
$V=V_1|V_2$
Π^{-1}
64
$\mathbf{P}^{-1}_{32/32}$
32
V_1
32
$Y=(y_0,y_1,\ldots,y_{31})$
b)

Figure 4. Structure of the CP-boxes $P_{32/64}$ (a) and $P^{-1}_{32/64}$ (b)

Table 1.

i.b.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
o.b.	0	8	16	24	1	9	17	24	2	10	18	26	3	11	19	27
i.b.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
o.b.	4	12	20	28	5	13	21	29	6	14	22	30	7	15	23	31

Fig. 4(b) represents the CP-box operation $P^{-1}_{32/64}$ which is inverse of the CP-box operation $P_{32/64}$. The CP-box $P^{-1}_{32/64}$ consists of two $P^{-1}_{32/32}$-boxes and one fixed permutation Π^{-1} between them, the last being inverse of Π. Swapping input and output of arbitrary given $P_{n/m}$-box permutation one can obtain respective inverse $P^{-1}_{n/m}$-box permutation. It is not difficult to see that all CP-boxes in Fig. 3 and 4 realize unique CP-modifications for all possible values of the controlling vector V.

3 Ciphers based on controlled permutations

Our design strategy was oriented to the following objectives:

1. Cryptoscheme should be a symmetric block cipher suitable for hardware implementation. This means the used operations must be economically implemented in electronic devices.

2. Cryptoscheme should be iterative in structure.

3. Cryptoscheme should be based on a combination of the DDP and KDP in order to reduce greatly the effectiveness of the differential and linear cryptanalysis.

4. Cryptoscheme should be based on fast operations (CP-box permutations, modulo 2^n addition $(+)$, modulo 2^n subtraction $(-)$, XOR, fixed permutations).

5. Key scheduling should be very simple in order to provide high encryption speed in the case of frequent change of keys.

These design criteria are realized in 64-bit block ciphers shown in Fig. 5 (Cipher 1) and Fig. 6 (Cipher 2), where input data block $L|R$ is represented as concatenation of two 32-bit subblocks L and R. In these cryptoschemes the rotation operations denoted as "$R <<< Z$" (a left-rotation of bit string R by Z bits) or "$>>> Z$" (right-rotation by Z bits) are used. Utilization of the fixed rotations (fixed permutations) does not raise the hardware implementation cost and does not

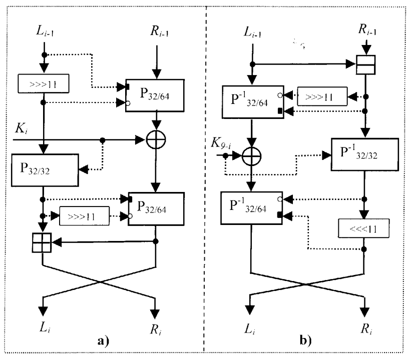


Figure 5. Encryption (a) and decryption (b) round function based on the CP-box operations

give rise to any additional time delay while ciphering. For both cryptoschemes it is supposed to use $r = 8$ encryption iterations (rounds) and 256-bit secrete key, which is represented as a set of 32-bit subkeys $K_1, K_2, ..., K_8$. Table 2 presents the i-th encryption (decryption) round of Cipher 1 and an approximate estimation of the time delay T corresponding to the respective ciphering steps ($t_{\oplus}$ is the time delay of the XOR operation; $L \leftrightarrow R$ denotes swapping subblocks L and R).

Table 2.

Step	Encryption	$T \approx$
# 1	$V := L; \quad L := L >>> 11;$ ‘ $R := P_{32/64(V\|L)}(R)$	$4t_{\oplus}$
# 2	$L := P_{32/32(K_i)}(L); \quad R := R \oplus K_i$	$2t_{\oplus}$
# 3	$V := L >>> 11; \quad R := P_{32/64(L\|V)}(R)$	$4t_{\oplus}$
# 4	$L := L + R; \quad L \leftrightarrow R$	$10t_{\oplus}$
Step	(Decryption)	$T \approx$
# 1	$R := R - L;$	$10t_{\oplus}$
# 2	$V := R >>> 11; \quad L := P^{-1}_{32/64(R\|V)}(L)$	$4t_{\oplus}$
# 3	$R := P^{-1}_{32/32(K_{9-i})}(R); \quad L := L \oplus K_{9-i}$	$2t_{\oplus}$
# 4	$V := R; R := R <<< 11; L := P^{-1}_{32/64(R\|V)}(L); L \leftrightarrow R$	$4t_{\oplus}$

One can note the following peculiarities contributing essentially to the security of the cryptoschemes:

1. The used CP-boxes realize relatively large number of different CP-modifications (2^{32} for $P_{32/32}$ and $P^{-1}_{32/32}$; 2^{64} for $P_{32/64}$ and $P^{-1}_{32/64}$).

2. Each round subkey is used in two different ways. Firstly, it serves as an operand while performing XOR operation over one data subblock. Secondly, it is used as controlling vector while performing $P_{32/32}$-box or $P^{-1}_{32/32}$-box permutation above other data subblock.

3. Cryptoscheme in Fig. 6 differs by the use of the **data-dependent transformation of the round subkeys** [8].

4. The dependency of the output bits on the input bits spreads mainly due to use of the data bits as controlling ones while performing the CP-box permutations. The arithmetic operations contribute additionally to the avalanche effect.

4 Discussion

Hardware-oriented ciphers based on CP-box operations can be attributed to so called flexible ciphers. An interesting class of flexible ciphers is related to the use of key-dependent operations. Controlled permutations are very interesting cryptographic primitive to design ciphers with key-dependent operations which are free of precomputations usually used in software-oriented flexible ciphers [9]. Cryptoschemes in Fig. 5 and Fig. 6 can be considered as representatives of the fast hardware-oriented "indeterminate" ciphers which can be used under conditions of the frequent key change, since they use no complex key scheduling. Actually in Cipher 2 a very fast internal key scheduling is used which corresponds to the execution of the CP-box permutation above the round subkeys before they are combined with data. The internal key scheduling is not complex, but it changes from one data block to another one strengthening significantly the cryptoscheme against differential and linear crpytanalysis. It introduces no time delay, since it is executed in parallel with some data ciphering operations.

In the proposed cryptoschemes several steps in each round keep one data subblock constant, however all its bits influence transformation of another one. This distinguish Ciphers 1 and 2 from Feistel-like cryptoschemes in which a half of data bits do not take part in the round transformation F. Because the Feistel structure converts arbitrary n-bit function F into a $2n$-bit cipher it is easy to compose different Feistel-like cryptosystems based on the KDP and DDP. This trend in the block cipher design is a topic of independent consideration. Prospects of the use of the CP-box permutations for construction of the F-functions are connected with comparatively high rapidity of their execution and simplicity of the hardware implementation.

In second section several simple variants of the CP-box permutations have been proposed. One can design other CP-boxes with different structures. CP-boxes with matrix structure realize $n!$ different permutation modifications [10], but they are slower. A permutation Π over n ordered elements can include $k \geq 1$ independent cycles with the length $n' \leq n$. Permutations containing only one cycle of the length

n (one-cycle permutations) are of special interest to be used as CP-modifications. It is possible to implement a matrix CP-box realizing only one-cycle permutations, the number of different CP-modifications being $(n-1)!$.

The hardware-realization cost of such two variants of CP-boxes with matrix structure equals about $4n^2$ nand gates ($\approx 4 \cdot 10^3$ for $n = 32$). The gate count for CP-boxes with cascade structure is $2kn$ (128 for $P32/32$ and 256 for $P_{32/64}$). Taking this into account we have decided in favour of the use of several cascade CP-boxes instead of one CP-box with matrix structure, the controlled one-cycle permutations (COCP) being very attractive though. In our future research we shall try to design fast COCP-boxes with lowered gate count.

While ciphering many different plaintext blocks, two input bits of the CP-boxes differ with probability 1/2, therefore in the case of cascade CP-boxes one bit of the controlling vector influences on the average one output bit. Let us consider one encryption round of Cipher 1. The single input bit of the left (right) subblock influences on the average $G_{LR}^{(1)} \approx 5$ ($G_{RR}^{(1)} \approx 1$) bits of the right half of the output and $G_{LL}^{(1)} \approx 4$ ($G_{RL}^{(1)} = 1$) bits of the left half of the output. For bits of the key one can consider analogous coefficients $G_{KL}^{(1)} \approx 3$ and $G_{KR}^{(1)} \approx 4$. A comparison of the values $G_{LR}^{(1)}$, $G_{LL}^{(1)}$, $G_{RL}^{(1)}$, $G_{RR}^{(1)}$, G_{KL}, and G_{KR} for one ciphering round of Ciphers 1 and 2 is given in Table 3.

Table 3.

Cryptoscheme	$G_{LR}^{(1)}$	$G_{LL}^{(1)}$	$G_{RR}^{(1)}$	$G_{RL}^{(1)}$	G_{KR}	G_{KL}
Cipher 1 (encryption)	5	4	1	1	3	4
Cipher 1 (decryption)	5	1	4	1	3	1
Cipher 2 (encryption)	7	6	1	1	2	2
Cipher 2 (decryption)	7	1	6	1	2	0

To consider $i \geq 2$ ciphering rounds one can calculate respective coefficients using the following approximate formulas:

$$G_{LL}^{(i)} \approx G_{LL}^{(i-1)} G_{LL}^{(1)} + G_{LR}^{(i-1)} G_{RL}^{(1)}; \qquad G_{LR}^{(i)} \approx G_{LL}^{(i-1)} G_{LR}^{(1)} + G_{LR}^{(i-1)} G_{RR}^{(1)};$$
$$G_{RL}^{(i)} \approx G_{RL}^{(i-1)} G_{LL}^{(1)} + G_{RR}^{(i-1)} G_{RL}^{(1)}; \qquad G_{RR}^{(i)} \approx G_{RL}^{(i-1)} G_{LR}^{(1)} + G_{RR}^{(i-1)} G_{RR}^{(1)}.$$

Calculating values $G_{LL}^{(3)}$, $G_{LR}^{(3)}$, $G_{RL}^{(3)}$, and $G_{RR}^{(3)}$ one can ascertain that after three rounds every input bit influences statistically almost all output bits. For example, in the case of Cipher 1 (Cipher 2) one can obtain for encryption $G_{LL}^{(3)} \approx 109$ (307), $G_{LR}^{(3)} \approx 130$ (350), $G_{RL}^{(3)} \approx 26$ (50), $G_{RR}^{(3)} \approx 31$ (57). The case $G > 32$ can be interpreted as repeated influence. It should be noted that these coefficients do not take into account the additional contribution of the modulo 2^n addition (subtraction) to the avalanche effect.

Ciphers described above have been implemented in software and tested. Experimental statistic examination has shown that four rounds sufficed to get uniform correlation between input and output bits. Preliminary analysis of these cryptoschemes has shown that they are secure against differential and linear attacks, although much more work on cryptanalysis of these ciphers is to be done. An objective of the present paper is to focus on the CP-box permutations as a source of cryptographic strength.

The hardware implementation of the proposed cryptoschemes appear to be very fast and inexpensive. One can note that chip makers can support encryption technique based on CP by adding a CP-box permutation instruction to the CPU. In this case it will be possible to compose very fast ($>$ 300 Mbit/s for Pentium-like processors) software encryption algorithms.

References

[1] Rivest R.L. *The RC5 Encryption Algorithm.* Lect. Notes Comput. Sci. Vol. 1008. pp. 86-96. Berlin: Springer-Verlag, 1995.

[2] Kaliski B.S., Yin Y.L. *On differential and linear cryptanalysis of the RC5 encryption algorithm.* Lect. Notes Comput. Sci. Vol. 963. pp. 171-184. Berlin: Springer-Verlag, 1995.

[3] Biryukov A., Kushilevitz E. *Improved cryptanalysis of RC5.* Lect. Notes Comput. Sci. Vol. 1403. pp. 85-99. Berlin: Springer-Verlag, 1998.

[4] Rivest R.L., Robshaw M.J.B., Sidney R., Yin Y.L. *The RC6 Block Cipher.* Proc. of 1st Advanced Encryption Standard Candidate Conference, Venture, California, Aug. 20-22, 1998 (see also at http://www.nist.gov/aes).

[5] Burwick C., Coppersmith D., D'Avingnon E., Gennaro R., Halevi Sh., Jutla Ch., Matyas Jr.S.M., O'Connor L., Peyravian M., Safford D., Zunic N. 1998. *MARS - a Candidate Cipher for AES.* Proc. of 1st Advanced Encryption Standard Candidate Conference, Venture, California, Aug. 20-22, 1998 (see also at http://www.nist.gov/aes).

[6] Moldovyan A.A., Moldovyan N.A. *A method of the cryptographical transformation of binary data blocks.* Russian patent N 2141729. Bull. N 32. Nov. 20, 1999.

[7] Kawn M. *The design of the ICE encryption algorithm.* Lect. Notes Comput. Sci. Vol. 1267. pp. 69-82. Berlin: Springer-Verlag, 1997.

[8] Maslovsky V.M., Moldovyan A.A., Moldovyan N.A. *A method of the block encryption of discrete data.* Russian patent N 2140710. Bull. N 30. Oct. 27, 1999.

[9] Moldovyan A.A., Moldovyan N.A. *Software Encryption Algorithms for Transparent Protection Technology.* Cryptologia, 1998. V. 22. No 1. pp. 56-68.

[10] Zima V.M., Moldovyan A.A., Moldovyan N.A., Savlukov H.B. *Encryption block.* Russian patent N 2140715. Bull. N 30. Oct. 27, 1999.

A.A.Moldovyan,
Specialized Center of Program Systems "SPECTR",
Kantemirovskaya, 10,
St-Petersburg 197342, Russia;
e-mail: *spectr@vicom.ru*

Received November 2, 2000

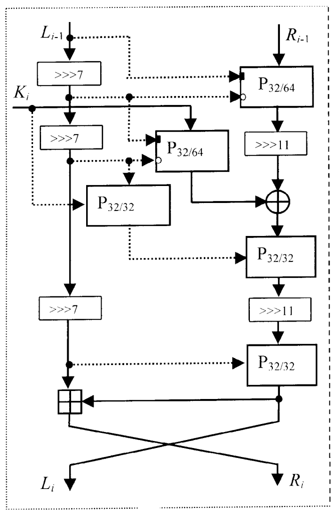


Figure 6. Cryptosystem with internal key scheduling