Fast Encryption Algorithm Spectr-H64

Nick D. Goots, Alexander A. Moldovyan, and Nick A. Moldovyan

Specialized Center of Program Systems "S PECTR",
Kantemirovskaya str., 10, St. Petersburg 197342, Russia
ph./fax.7-812-2453743, spectr@vicom.ru

Abstract. This paper describes a fast hardware-oriented 64-bit block cipher SPECTR-H64 based on combination of the data-dependent permutations and data-dependent transformation of subkeys.

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]. 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 AES candidates 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. It is very attractive to use data-dependent permutations (DDP) for increasing the number of the controlling bits [6]. DDP are easy to be performed with permutation networks (PN) developed previously [7,8]. A PN can be used in the cryptosystems design as an operational block performing so called controlled permutations (CP). CP in the form of key-dependent permutations are used in cipher ICE [9], such use of CP has been shown [10] to be not very effective against differential cryptanalysis though. Different variants of the use of CP are proposed in patents [6,11]. In [6] it is proposed to encrypt data with DDP combining them with modulo 2^{32} addition and bitwise exclusive-OR (XOR) operations ( ). In [11] it is proposed to construct a block cipher based on data-dependent transformation of the round subkeys.

In present paper we consider a fast hardware-oriented 64-bit block cipher based on DDP performed on both the data and the round subkeys. Concrete type of the CP-box permutation $\mathbf{P}_{n/m}$ is characterized by an ordered set $\{ _0, _1,\ldots, _{2^m-1}\}$, where all $_i$, $i = 0,1,\ldots,2^m-1$, are fixed permutations and each number i can be represented as a m-bit control vector $V(v_1,.,. v_m)$ such that $i = v_1+2v_2+\ldots+2^{m-1}v_m$ is hold. Therefore instead of $_i$ we can use notation $\mathbf{P}_V$. Such fixed permutations we shall call the CP-

V.I. Gorodetski et al. (Eds.): MMM-ACNS 2001, LNCS 2052, pp. 275–286, 2001.

modifications. Thus, the notation $Y = P_{n/m}(X,V)$ means $Y = \mathbf{P}_V(X) = {}_i(X)$. To construct CP boxes we use the approach proposed in [8] which is based on standard elementary CP-boxes $P_{2/1}$ (Fig. 6a). Each $P_{2/1}$-box is controlled by one bit v: $y_1 = x_{1+v}$ and $y_2 = x_{2\ v}$.

CP-boxes can be constructed on the basis of the simple layered structure shown in Fig. 5-7. Two CP-boxes $P_{n/m}$ and $P^{-1}_{n/m}$ we shall call mutually inverse, if for all fixed values of the vector V the respective CP-modifications P_V and P^{-1}_V are mutually inverted.

1 Design of the Block Cipher SPECTR-H64

Our design strategy was oriented to the following objectives:

Cryptoscheme should be iterative in structure.

Cryptoscheme should be based on fast operations (CP-box permutations, XOR, fixed permutations, special fast nonlinear functions).

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

1.1 Notations

Let $\{0,1\}^n$ denotes the set of the binary vectors of length n, then $X \ \{0,1\}^n$ means $X = (x_1,.,. x_n)$, where $x_i \ \{0,1\}$ and $i \ \{1,\ldots,n\}$. Let x_1 is the least significant bit and x_n is the most significant bit. Let denote $X_{lo} = (x_1,.,. x_{n/2})$ and $X_{hi} = (x_{n/2+1},.,. x_n)$, then $X_{lo},X_{hi} \ \{0,1\}^{n/2}$ and $X = (X_{lo},X_{hi})$ (or $X = X_{lo}|X_{hi}$, where "|" d enotes concatenation operation). Let $X,Y,A \ \{0,1\}^n$, $c \ \{0,1\}$. Let $X \ A$ denotes bitwise-AND operation (" "), and let denote cyclic rotation ">>>" a s follows:

$$Y = X^{>>> k} \qquad \begin{array}{ll} y_i = x_{i+k} & i = 1,\ldots, n \quad k \\ y_i = x_{i+k\ n} & i = n \quad k+1,\ldots, n \end{array}, \text{ where } 0 \quad k < n.$$

1.2 General Encryption Scheme

General encryption scheme is defined by the following formulas: $C = \mathbf{E}(M,K)$ and $M = \mathbf{D}(C,K)$, where M is the plaintext, C is the ciphertext ($M,C \ \{0,1\}^{64}$), K is the secrete key ($K \ \{0,1\}^{256}$), **E** is encryption function, and **D** is decryption function.

In the block cipher SPECTR-H64 encryption and decryption functions coincide (**E= D = F**). Thus, transformation algorithm is defined by formula:

$$Y = \mathbf{F}(X, Q^{(e)}),$$

where $Q^{(e)} = \mathbf{H}(K,e)$ is the extended key (EK), the last being a function of the secrete key $K = (K_1,.,. K_8)$ and of the transformation mode parameter e ($e = 0$ defines encryption, $e = 1$ defines decryption), the input data block being $X = \begin{array}{ll} M & e=0 \\ C & e=1 \end{array}$.

In 12-round block cipher EK is represented as concatenation of 14 subkeys:

$$Q^{(e)} = (Q^{(e)}_{IT}, Q^{(e)}_{1},\ldots Q^{(e)}_{12}, Q^{(e)}_{FT}),$$

where $Q^{(e)}_{IT}, Q^{(e)}_{FT} \in \{0,1\}^{32}$ and $j = 1,\ldots,12$ $Q^{(e)}_{j} = (Q^{(1,e)}_{j},\ldots,Q^{(6,e)}_{j})$, where $h = 1,\ldots 6$ $Q^{(h,e)}_{j} \in \{0,1\}^{32}$.

Thus, EK is a sequence of 74 32-bit binary vectors. Output value Y is the ciphertext C in the encryption mode or is a plaintext M in the decryption mode.

Description of the encryption algorithm (function **F**) is given in section 1.3 and procedure of the formation of EK $Q^{(e)} = \mathbf{H}(K,e)$ is given in section 1.4.

1.3 Encryption Algorithm

The algorithm (function **F**) is designed as sequence of the following procedures: *initial transformation* ***IT***, 12 rounds with procedure ***Crypt***, and *final transformation* ***FT***.

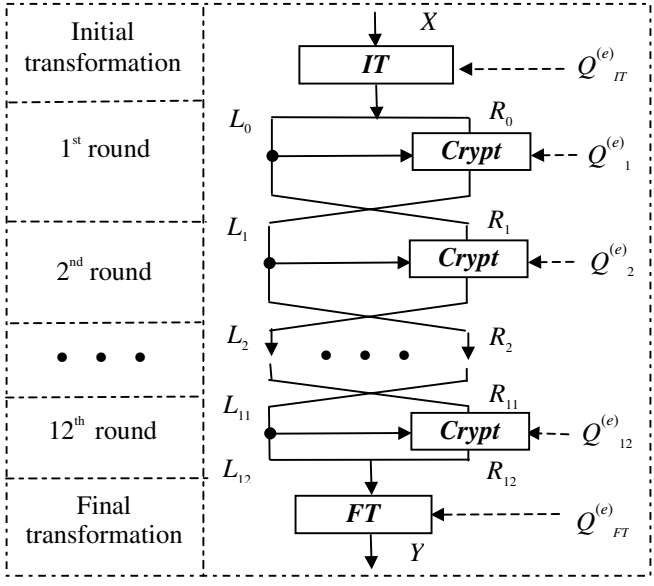

Fig. 1. General encryption scheme

Ciphering (Fig. 1) begins with the procedure ***IT.***

$$Y = \mathbf{IT}(X, Q^{(e)}_{IT}).$$

Then data block Y is divided into two blocks L_0 and R_0, i.e. $(L_0,R_0) = Y$, where $L_0,R_0 \in \{0,1\}^{32}$. After that 12 sequential rounds are performed with procedures ***Crypt*** in accordance with the formulas:

$$L_j = \mathbf{Crypt}\,(R_{j-1},L_{j-1},Q^{(e)}_{j}); \qquad R_j = L_{j-1}, \quad \text{where } j = 1,\ldots,12.$$

The final transformation ***FT*** is executed after the 12-th round:

$$Y = \boldsymbol{FT}(X, Q^{(e)}_{\mathrm{FT}}),$$

where $X = (R_{12}, L_{12})$ is the output of 12-th round.

Initial transformation *IT*. It has the following form: $Y = \boldsymbol{IT}(X,A)$, where X,Y $\{0,1\}^{64}$, A $\{0,1\}^{32}$. Fig.3a shows the scheme of this transformation which defines transposition of each pair of bits $x_{2j-1}x_{2j}$ ($j = 1,...,32$) of X if $a_j = 1$, otherwise ($a_j = 0$) the bits are not transposed. After that each even bit is inverted.

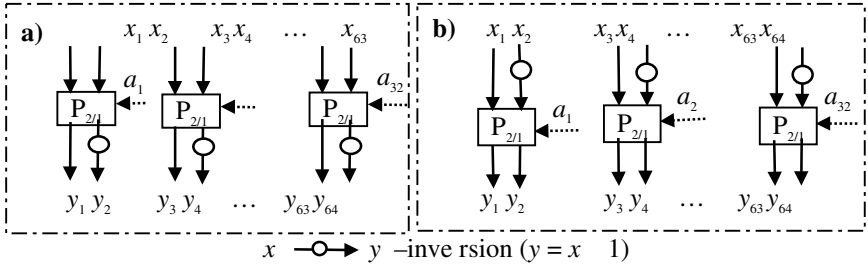

Fig. 2. Initial transformation IT (a) and final transformation FT (b)

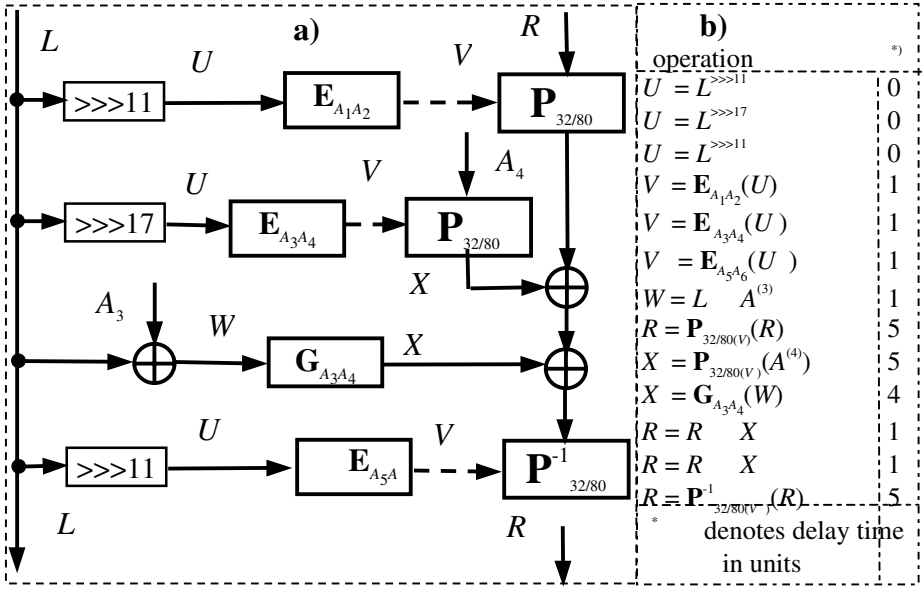

Fig. 3. Procedure Crypt (a) and time delays of individual operations (b)

Final transformation *FT*. Procedure ***FT*** is inverse of the procedure ***IT***. Transformation ***FT*** has the following form: $Y = \boldsymbol{FT}(X,A)$, where X,Y $\{0,1\}^{64}$, A $\{0,1\}^{32}$. Initially (Fig. 3b) each even bit of the input block is inverted and then each

pair of bits with indices $2j$-1 and $2j$ (j = 1,...,32) is transposed, if a_j = 1, otherwise (a_j = 0) the bits are not transposed.

Procedure *Crypt*. This procedure has the form: R = ***Crypt***(R,L,A_1,A_2,..., A_6), where L,R,A_1,A_2,..., A_6 $\{0,1\}^{32}$. Thus, ***Crypt*** transforms data subblock R under control of the data subblock L and 192-bit subkey. It uses the following operations: cyclic rotation "$>>>$" b y fixed amount, XOR operation " ", non -linear function **G**, data-dependent permutations $\mathbf{P}_{32/80}$ and $\mathbf{P}^{-1}_{32/80}$, and extension operation **E** over control vector**.** The sequence of the performed transformations is represented in Fig. 4.

Non-linear function G. Justification of the function **G** is given in section 2. Formally this transformation can be represented as follows: X = **G**(W,A,B), where X ,W,A, B $\{0,1\}^{32}$. Realization of the function **G** is defined by the following expression:

$$X = M_0 \quad M_1 \quad (M_2 \quad A) \quad (M_2 \quad M_5 \quad B) \quad (M_3 \quad M_5) \quad (M_4 \quad B),$$

where binary vectors M_0, M_1,... M_5 are expressed recursively through W as follows:

$$M_0 = (m_1^{(0)}, m_2^{(0)}, \ldots m_{32}^{(0)}) = (w_1, w_2, \ldots w_{32}) \quad \text{and} \quad j = 1,\ldots,5$$

$$M_j = (m_1^{(j)}, m_2^{(j)}, m_3^{(j)}, \ldots m_{32}^{(j)}) = (1, m_1^{(j-1)}, m_2^{(j-1)}, \ldots m_{31}^{(j-1)}).$$

Taking into account that some operations in function **G** are performed in parallel it is easy to see that delay time of this function is about 4 .

Controlled permutation boxes $\mathbf{P}_{32/80}$ and $\mathbf{P}^{-1}_{32/80}$. CP-box transformation is represented in the following form: $Y = \mathbf{P}_{32/80}(X,V)$, where X, Y $\{0,1\}^{32}$, V $\{0,1\}^{80}$, X is the input, V is the control vector. In necessary cases we use also the notation $\mathbf{P}_{32/80(V)}(X)$. Detailed mathematical representation depends on concrete realization of the CP box $\mathbf{P}_{32/80}$. In general, each output bit y_j (j = 1,…,32) is a Boolean function of the variables $\{x_i\}$ and $\{v_i\}$, i.e. y_j = $_j(x_1,\ldots x_{32}; v_1,\ldots v_{80})$.

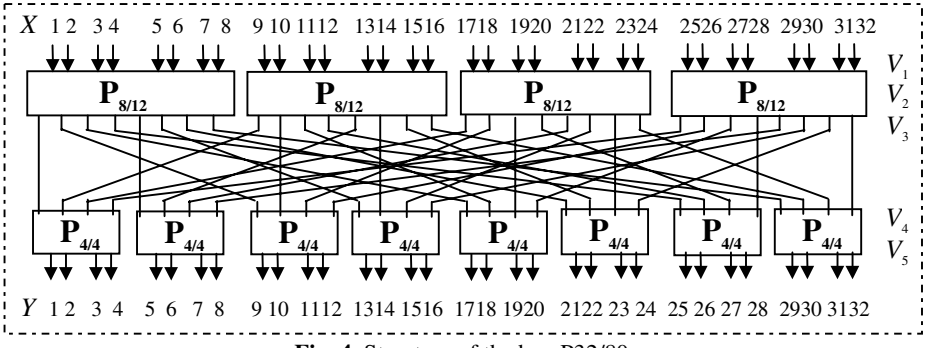

Fig. 4. Structure of the box P32/80

Construction scheme of the box $\mathbf{P}_{32/80}$ is shown in Fig. 5. Box $\mathbf{P}_{32/80}$ has 32-bit input, 32-bit output, and 80-bit control input. This CP box consists of five layers of 16 parallel elementary CP boxes $\mathbf{P}_{2/1}$ with some fixed connection between layers. The first three layers are structurally combined in four boxes $\mathbf{P}_{8/12}$ and the last two are combined in eight boxes $\mathbf{P}_{4/4}$. Design schemes of the boxes $\mathbf{P}_{2/1}$, $\mathbf{P}_{4/4}$, and $\mathbf{P}_{8/12}$ are shown in Fig. 6.

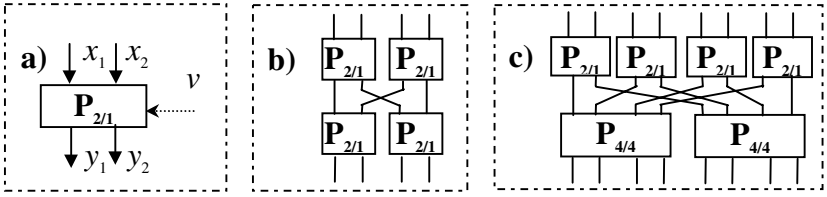

Fig. 5. Structure of the boxes P2/1 (a), P4/4 (b), and P8/12 (c)

In the box $\mathbf{P}_{32/80}$ all elementary CP boxes are numbered consequently from left to right and from top to bottom. In accordance with such enumeration the j-th bit of vector V controls the elementary box $\mathbf{P}_{2/1}^{(j)}$ and vector V can be represented as concatenation of the five vectors $V_1,V_2,V_3,V_4,V_5 \quad \{0,1\}^{16}$, i.e. $V = (V_1|V_2|V_3|V_4|V_5)$. Two lower bits x_1 and x_2 of the transformed data subblock are the input of the first box $\mathbf{P}_{2/1}^{(1)}$. Respectively, two higher bits x_{31} and x_{32} of the transformed data subblock are the input of the 16-th elementary box $\mathbf{P}_{2/1}^{(16)}$. Respectively, the bits y_1 and y_2 of the value Y correspond to the box $\mathbf{P}_{2/1}^{(65)}$. Bits y_{31} and y_{32} correspond to the box $\mathbf{P}_{2/1}^{(80)}$.

Interconnection between the third and the fourth layers of the box $\mathbf{P}_{32/80}$ has the form of the following involution:

$$\mathbf{I} = (1)(2,9)(3,17)(4,25)(5)(6,13)(7,21)(8,29)(10)(11,18)(12,26)$$

$$(14)(15,22)(16,30)(19)(20,27)(23)(24,31)(28)(32)$$

One can remark that interconnection between boxes $\mathbf{P}_{8/12}$ and $\mathbf{P}_{4/4}$ is designed in accordance with the "each-to-each" principal, i.e. the i-th output digit of the j-th box $\mathbf{P}_{8/12}$ is connected with the j-th input digit of the i-th box $\mathbf{P}_{4/4}$. It is easy to see that boxes $\mathbf{P}_{2/1}$, $\mathbf{P}_{4/4}$, $\mathbf{P}_{8/12}$ are the CP boxes of the *first order*. The CP box $\mathbf{P}_{n/m}$ of the order h ($0 \quad h \quad n$) is a CP box which has at least one CP-modification $_V$ moving arbitrary given h bits in the given h output positions. Taking into account the design structure it is easy to see that the box $\mathbf{P}_{32/80}$ is also of the first order.

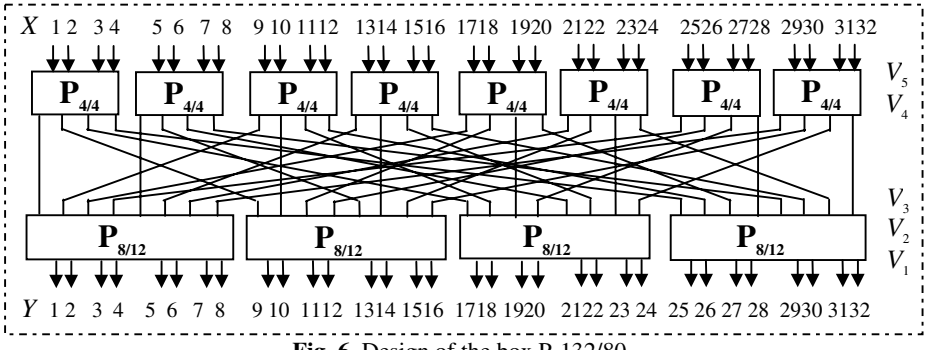

Fig. 6. Design of the box P-132/80

Formal representation of the *inverse transformation* $\mathbf{P}^{-1}_{32/80}$ has the following form:

$$Y = \mathbf{P}^{-1}_{32/80}(X,V), \qquad \text{where } X, Y \in \{0,1\}^{32}, V \in \{0,1\}^{80}.$$

Box $\mathbf{P}^{-1}_{32/80}$ (Fig. 7) can be constructed analogously to the design of the box $\mathbf{P}_{32/80}$ with exception that input and output are swapped.

If for CP-box $\mathbf{P}_{32/80}$ enumeration of the elementary boxes $\mathbf{P}_{2/1}$ is given from left to right and from top to bottom for $\mathbf{P}^{-1}_{32/80}$ enumeration is given from left to right from bottom to top, then the same control vector $V=V_1|V_2|V_3|V_4|V_5$ defines a pair of respective inverse modifications $\mathbf{P}_{32/80(V)}$ and $\mathbf{P}^{-1}_{32/80(V)}$. Such design of the box $\mathbf{P}^{-1}_{32/80}$ allows one to use the same extension box $\mathbf{E}$ to form control vector for both the box $\mathbf{P}_{32/80}$ and the box $\mathbf{P}^{-1}_{32/80}$. Structure of the CP boxes $\mathbf{P}_{32/80}$ and $\mathbf{P}^{-1}_{32/80}$ provides that superposition $\mathbf{P}_{32/160} = \mathbf{P}_{32/80} * \mathbf{P}^{-1}_{32/80}$ is a CP box of the maximal order [12].

Extension box E. The extension box $\mathbf{E}$ is used to form an 80-bit control vector the given 32-bit value. Formal representation of the extension transformation is:

$$V = (V_1|V_2|V_3|V_4|V_5) = \mathbf{E}(U,A,B) = \mathbf{E}_{A|B}(U),$$

where $V \in \{0,1\}^{80}$, $V_1,V_2,V_3,V_4,V_5 \in \{0,1\}^{16}$, $U,A,B \in \{0,1\}^{32}$.

Actually the vectors V_1,V_2,V_3,V_4,V_5 are determined accordingly to the formulas:

$$V_1 = U_{hi}; \; V_2 = ((U \; A)_{hi}); \; V_5 = ((U \; A)_{lo}), \; V_3 = ((U \; B)_{hi}); V_4 = ((U \; B)_{lo}),$$

where fixed permutations and are the following:

$$(Z) = Z_{hi}^{\gg 1} | Z_{lo}^{\gg 1} \qquad \text{and} \qquad (Z) = Z_{hi}^{\gg 5} | Z_{lo}^{\gg 5}.$$

Correspondence between control bits of both the vector V and the vector U for box $\mathbf{P}_{32/80}$ and for $\mathrm{P}^{-1}_{32/80}$ are given in Tables 2.

Table 1.

E																	
V_1	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	U_{hi}
V_2	26	27	28	29	30	31	32	25	18	19	20	21	22	23	24	17	$(U \; A)_{hi}$
V_3	30	31	32	25	26	27	28	29	22	23	24	17	18	19	20	21	$(U \; B)_{hi}$
V_4	14	15	16	9	10	11	12	13	6	7	8	1	2	3	4	5	$(U \; B)_{lo}$
V_5	10	11	12	13	14	15	16	9	2	3	4	5	6	7	8	1	$(U \; A)_{lo}$

For example, in Table 2 the number 19 means that bit u_{19} controls the box $\mathrm{P}^{(3)}_{2/1}$ (i.e. u_{19} is used as control bit v_3), 22 corresponds to value u_{22} a_{22} used as control bit v_{29}, and 7 corresponds to value u_7 b_7 used as control bit v_{58}.

Extension box is designed in such a way that lower 16 bits of the vector U take part in controlling the boxes $\mathrm{P}_{4/4}$ only and the higher 16 bits control the boxes $\mathrm{P}_{8/12}$ only. This provides the uniformity of the use of each bit of vector U in controlling the box $\mathrm{P}_{32/80}$. This also provides avoiding repetition of arbitrary bit of U to be used twice in some set of five control bits defining the path of each input bit to each output position.

1.4 Procedure of the Formation of the Extended Key

Extended encryption key represents $Q^{(e)} = \mathbf{H}(K,e)$ some sequence of 32-bit subkeys K_i (i = 1,…,8), in which parameter e defines encryption (e = 0) or decryption (e = 1) mode. Sequences $Q^{(0)} = \mathbf{H}(K,0)$ and $Q^{(1)} = \mathbf{H}(K,1)$ are different in general, but partially they coincide. Respective modification of the extended key is performed with single-layer box $\mathbf{P}_{256/1(e)}$ which is represented by four parallel boxes $\mathbf{P}_{2\ 32/1(e)}$. Four couples of subkeys K_i (Fig. 2), i.e. (K_1,K_2), (K_3,K_4), (K_5,K_6), and (K_7,K_8) are inputs of the corresponding boxes $\mathbf{P}_{2\ 32/1(e)}$. Each box $\mathbf{P}_{2\ 32/1(e)}$ represent 32 parallel elementary boxes $P_{2/1(e)}$.

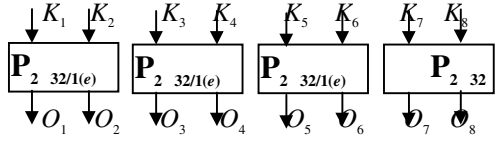

Fig. 7. Switching of subkeys

In Fig. 2 eight 32-bit outputs of four boxes $\mathbf{P}_{2\ 32/1(e)}$ are denoted as O_j (j = 1,…,8): $O_{2j-1}=K_{2j-1+e}$ and $O_{2j}=K_{2j-e}$ (j = 1,...,4). Extended key is specified in Table 1.

Table 2.

$Q^{(e)}_{IT}$	j =	1	2	3	4	5	6	7	8	9	10	11	12	$Q^{(e)}_{FT}$
O_1	$Q^{(1,e)}_j$=	K_1	K_8	K_5	K_4	K_1	K_6	K_7	K_4	K_2	K_6	K_5	K_3	O_2
	$Q^{(2,e)}_j$=	K_2	K_6	K_7	K_3	K_2	K_8	K_5	K_3	K_1	K_8	K_7	K_4	
	$Q^{(3,e)}_j$=	O_6	O_1	O_2	O_5	O_7	O_3	O_4	O_8	O_6	O_1	O_2	O_5	
	$Q^{(4,e)}_j$=	O_7	O_4	O_3	O_8	O_6	O_1	O_2	O_5	O_7	O_4	O_3	O_8	
	$Q^{(5,e)}_j$=	K_3	K_5	K_6	K_2	K_4	K_7	K_6	K_1	K_4	K_5	K_8	K_1	
	$Q^{(6,e)}_j$=	K_4	K_7	K_8	K_1	K_3	K_5	K_8	K_2	K_3	K_7	K_6	K_2	

Let r denotes the number of encryption rounds. Then $Q^{(0)} = (Q^{(0)}_{IT},Q^{(0)}_1,\ldots,Q^{(0)}_r,Q^{(0)}_{FT})$ is the encryption key, where $Q^{(0)}_j = (Q^{(1,0)}_j,\ldots,Q^{(6,0)}_j)$ is j-th extended round subkey. Let $Q^{(1)} = (Q^{(1)}_{IT},Q^{(1)}_1,\ldots,Q^{(1)}_r,Q^{(1)}_{FT})$ is the decryption key. To provide possibility to perform encryption and decryption with the same algorithm the following conditions must hold:

$$Q^{(1)}_{IT} = Q^{(0)}_{FT}; \qquad Q^{(1)}_{FT} = Q^{(0)}_{IT}; \qquad \text{and} \qquad j = 1,\ldots r$$

$$Q^{(1,1)}_j = Q^{(5,0)}_{r+1-j}; \quad Q^{(5,1)}_j = Q^{(1,0)}_{r+1-j};$$

$$Q^{(2,1)}_j = Q^{(6,0)}_{r+1-j}; \quad Q^{(6,1)}_j = Q^{(2,0)}_{r+1-j};$$

$$Q^{(3,1)}_j = Q^{(3,0)}_{r+1-j}; \quad Q^{(4,1)}_j = Q^{(4,0)}_{r+1-j}.$$

2 Justification of the Function G

Let define mapping **G**: $\{0,1\}^{32}$ $\{0,1\}^{32}$ $\{0,1\}^{32}$ $\{0,1\}^{32}$ as a vector Boolean function $Y = \mathbf{G}(X,A,B)$ in the following form: $(y_1,\ldots y_{32}) = (f_1(X,A,B),\ldots f_{32}(X,A,B))$, where f_i are some generating Boolean functions. If i equation $f_i = f_i(x_1,\ldots x_i,A,B)$ is hold, then such mapping is a sequential model of the controlled substitutional operation. If the functions $f_i = x_i$ $f_i(x_1,\ldots,x_{i-1},A,B)$ are used as generating functions, then the vector function **G** performs bijective mapping over X [13]. To simplified general scheme one can consider a single unificated function of fixed number of variables as a prototype of generating functions. In SPECTR-H64 the following unificated function is used:

$$f_i = f(z_1,\ldots z_8) = f'(z_1,\ldots z_7) \quad z_8 = (z_1,\ldots z_6) \quad z_7 \quad z_8,$$

where $(z_1,\ldots z_6) = z_1z_2z_3 \quad z_1z_4 \quad z_2z_5 \quad z_3z_6$ is a bent function [14].

Discrete Fourier transformation of the function f [14] consists of 64 elements taking on the value $|U| = 32$ for all vectors $= (z_1,\ldots z_6,0,0,)$ and of 192 elements having value $U = 0$. Taking into account these facts as well as other special properties of the bent functions we have the following characteristic of the function f:

non-linearity of the function f is $N(f) = 2^{n-1} - 2^{(n+2)/2-1} = 112$; this value is sufficiently close to maximally possible non-linearity for Boolean functions of 8 variables ($N_{max} = 2^{n-1} - 2^{n/2-1} = 120$);

f is correlationally immune function relatively all vectors $(z_1,\ldots z_8)$, where $(z_7 \quad z_8)$ 0. This means that arbitrary particular function obtained by fixing arbitrary seven or less variables (except simultaneous fixing z_7 and z_8) is a balanced one;

f has good self-correlation properties (avalanche effect), since for all non-zero avalanche vectors $(z_1,\ldots z_8)$ $\{0,1\}^8$, except $(0,\ldots 0,0,1)$, $(0,\ldots 0,1,0)$, and $(0,\ldots 0, 1,1)$, propagation criterion of the degree 8 is fulfilled, i.e. the function $f = f(z_1,\ldots z_8)$ $f(z_1 \quad z_1,\ldots z_8 \quad z_8)$ is balanced one;

the degree of the algebraic normal form of function f equals 3 ($deg(f) = 3$).

From general form of the function f one can obtain concrete generating functions of the sequential model of the substitutional operation, using the following substitution of variables:

z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
x_i	$x_{i\ 1}$	$x_{i\ 2}$	$x_{i\ 5}$	b_i	a_i	$x_{i\ 3}$	$x_{i\ 4}$

i $\{1,\ldots,n\}$, where $n = 32$. Thus, we have the following generating functions:

$$y_i = f_i = x_i \quad x_{i-1} \quad x_{i-2}a_i \quad x_{i-2}x_{i-5}b_i \quad x_{i-3}x_{i-5} \quad x_{i-4}b_i,$$

where x_j, a_j, b_j are components of the vectors X,A,B $\{0,1\}^{32}$ ($i = 1,\ldots 32$), respectively, and $(x_{-4},x_{-3},x_{-2},x_{-1},x_0) = (1,1,1,1,1)$ is the initial condition. The function $\mathbf{G} = \mathbf{G}(L \quad A)$ is bijective over L for arbitrary fixed values of subkeys A,B.

3 Statistic Examination and Discussion

Investigation of statistic properties of SPECTR-H64 has been carried out with standard tests, which have been used in [15] for five AES finalists. The following dependence criteria have been used: 1) the average number of output bits changed when changing 1input bit; 2) the degree of completeness; 3) the degree of the avalanche effect; 4) the degree of strict avalanche criterion.

In the criteria we have used the dependence matrix a_{ij} and the distance matrix b_{ij} are defined as follows:

$$a_{ij} = \#\{X \in \mathbf{X}, K \in \mathbf{K} \mid (\mathbf{F}(X^{(i)}, K))_j \neq (\mathbf{F}(X, K))_j\}$$

for $i = 1,\ldots, n$ and $j = 1,\ldots, m$ and

$$b_{ij} = \#\{X \in \mathbf{X}, K \in \mathbf{K} \mid w(\mathbf{F}(X^{(i)}, K) \oplus \mathbf{F}(X, K)) = j\}$$

for $i = 1,\ldots, n$ and $j = 0,\ldots, m$, where $w(W)$ is the Hamming weight of the components of the vector W, and the vector $X^{(i)}$ denotes the vector obtained by complementing the i-th bit of X. The *degree of completeness* is defined as:

$$d_c = 1 - \frac{\#\{(i, j) \mid a_{ij} = 0\}}{nm}.$$

The *degree of the avalanche effect* has been calculated in accordance with the expression:

$$d_a = 1 - \frac{1}{nm} \sum_{i=1}^{n} \left| \frac{1}{\#\mathbf{X} \cdot \#\mathbf{K}} \sum_{j=1}^{m} 2^j b_{ij} - m \right|.$$

The *degree of strict avalanche criterion* has been calculated as:

$$d_{sa} = 1 - \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \left| \frac{2 a_{ij}}{\#\mathbf{X} \cdot \#\mathbf{K}} - 1 \right|.$$

Experiments have been performed in cases “one key and 10000 texts” and “100 keys and 100 texts” (Table 3). In general, results on statistic testing of SPECTR-H64 are analogous to that of AES finalists [15].

Table 3.

Number of round	#K=1, #X=10000;				#K=100, #X=100.			
	(1)	(2)=d_c	(3)=d_a	(4)=d_{sa}	(1)	(2)=d_c	(3)=d_a	(4)=d_{sa}
12	32.044695	1.000000	0.998514	0.991975	32.039656	1.000000	0.998591	0.992178
10	32.040497	1.000000	0.998508	0.992011	32.045756	1.000000	0.998489	0.991799
8	32.049603	1.000000	0.998273	0.991963	32.042373	1.000000	0.998440	0.991953
6	32.047088	1.000000	0.998495	0.991937	32.045458	1.000000	0.998494	0.992088
4	32.013722	1.000000	0.998547	0.991473	32.003152	1.000000	0.998279	0.991663
3	29.213473	1.000000	0.911581	0.872055	29.213597	1.000000	0.912476	0.873493
2	27.931805	1.000000	0.872869	0.513196	27.927716	1.000000	0.872741	0.524154
1	25.220356	0.515459	0.788136	0.140762	25.187814	0.515137	0.787119	0.160758

Because of very simple key scheduling used in SPECTR-H64 it appears to be important to study how a single bit of key influences statistically ciphertext. For this purpose we have used the criteria 1-4 mentioned above with exception that the matrix of dependencies a_{ij} and matrix of distances b_{ij} have been defined relatively key as follows:

$$a_{ij} = \#\{X \quad \boldsymbol{X},K \quad \boldsymbol{K} \mid (\mathbf{F}(X,K^{(i)}))_j \quad (\mathbf{F}(X,K))_j\}$$

for $i = 1,.,. n$ and $j = 1,.,. m$,

$$b_{ij} = \#\{X \quad \boldsymbol{X},K \quad \boldsymbol{K} \mid w(\mathbf{F}(X,K^{(i)}) \quad \mathbf{F}(X,K)) = j\}$$

for $i = 1,.,. n$ and $j = 0,.,. m$, where $K^{(i)}$ denotes the vector obtained by complementing the i-th bit of K.

Obtained results (Table 4) show that 7 rounds are quite sufficient to get necessary key's influencing output text (key's propagation property).

Table 4.

Number of round	#*K*=400, #*X*=25;				#*K*=100, #*X*=100.			
	(1)	(2)=d_c	(3)=d_a	(4)=d_{sa}	(1)	(2)=d_c	(3)=d_a	(4)=d_{sa}
12	32.050219	1.000000	0.998271	0.992005	32.051301	1.000000	0.998197	0.992040
10	32.053772	1.000000	0.998188	0.991940	32.048439	1.000000	0.998298	0.991881
8	32.051826	1.000000	0.998193	0.992168	32.049461	1.000000	0.998261	0.992045
6	31.911817	1.000000	0.995094	0.979446	31.913402	1.000000	0.995040	0.979268
5	31.518421	1.000000	0.983716	0.915390	31.512109	1.000000	0.983649	0.915872
4	30.735910	1.000000	0.960020	0.787367	30.734479	1.000000	0.959858	0.787278
3	28.476102	1.000000	0.889878	0.565557	28.470965	1.000000	0.889718	0.565777
2	26.810414	0.750000	0.837825	0.247683	26.806360	0.750000	0.837699	0.247604
1	21.870804	0.250000	0.683463	0.036559	21.869943	0.250000	0.683436	0.036482

Cryptosystem SPECTR-H64 uses a very fast internal key scheduling 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 cryptanalysis. It introduces no time delay, since it is executed in parallel with some data ciphering operations.

The hardware-realization cost of SPECTR-H64 equals about 100,000 transistors. Experimental cryptochips have been made with 1.2-micron technology, about 1Gbit/s encryption speed being obtained.

The hardware implementation of the proposed cryptoscheme appears to be very fast and inexpensive. One can note that chipmakers 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 (> 400 Mbit/s for Pentium-like processors) software encryption algorithms.

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

References

1. Rivest, R.L.: The RC5 Encryption Algorithm. Fast Software Encryption - FSE'94 Proceedings. Springer-Verlag, LNCS Vol. 1008. (1995) 86-9 6
2. Kaliski, B.S. and Yin, Y.L.: On Differential and Linear Cryptanalysis of the RC5 Encryption Algorithm. Advances in Cryptology – C RYPTO'95 Proceedings. Springer-Verlag, LNCS Vol.963. (1995) 171-1 84
3. Biryukov, A., Kushilevitz, E.: Improved Cryptanalysis of RC5. Advances in Cryptology – Eurocrypt'98 Proceedings. Springer-Verlag, LNCS Vol. 1403. (1998) 85-99
4. Rivest, R.L., Robshaw, M.J.B., Sidney, R., Yin, Y.L.: The RC6 Block Cipher. Proceeding of 1st Advanced Encryption Standard Candidate Conference, Venture, California, (Aug 20-22 1998) (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.: MARS – a Candidate Cipher for AES. Proceeding of 1st Advanced Encryption Standard Candidate Conference, Venture, California, (Aug 20-2 2 1998) (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. Benes, V.E.: Mathematical Theory of Connecting Networks and Telephone Traffic. Academic Press, New York (1965)
8. Waksman, A.A.: Permutation Network, Journal of the ACM. Vol.15, no 1 (1968) 159-16 3
9. Kawn, M.: The Design of the ICE Encryption Algorithm. Fast Software Encryption - FSE'97 Proceedings. Springer-Verlag, LNCS Vol. 1267. (1997) 69-82
10. Van Rompay, L.R. Knudsen, V.: Rijmen Differential Cryptanalysis of the ICE Encryption Algorithm, Fast Software Encryption – FSE 98 Proceeding. Springer-Verlag LNCS Vol. 1372. (1998) 270-2 83
11. 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)
12. Goots, N.D., Izotov, B.V., Moldovyan, A.A., Moldovyan, N.A.: Design of two place operation for fast flexible cryptosystems. Bezopasnosti informatsionnyh tehnologii, Moscow, MIFI, no 4 (2000, in Russian)
13. Goots, N.D., Izotov, B.V., Moldovyan, N.A.: Controlled permutations with symmetric structure in block ciphers, Voprosy zaschity informatsii, Moscow, VIMI, no 4 (2000) 57-65 (in Russian).
14. Nyberg, K.: Constructions of Bent Functions and Difference Sets, Advances in Cryptology –E urocrypt'90 Proceedings. Springer Verlag (1991) 151-16 0
15. Preneel, B., Bosselaers, A., Rijmen, V., Van Rompay, B., Granboulan, L., Stern, J., Murphy, S., Dichtl, M., Serf, P., Biham, E., Dunkelman, O., Furman, V., Koeune, F., Piret, G., Quisquater, J-J., Knudsen, L., Raddum, H.: Comments by the NESSIE Project on the AES Finalists. (24 may 2000) (http://www.nist.gov/aes)