A New Approach To Block Cipher Design

Joan Daemen, René Govaerts and Joos Vandewalle

Katholieke Universiteit Leuven, Laboratorium ESAT
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
email: joan.daemen@esat.kuleuven.ac.be

Abstract. In this paper we apply the cryptographic finite state machine approach as introduced in [1] to the design of symmetric key block ciphers. Key words in the design approach are simplicity, uniformity, parallelism, distributed nonlinearity and high diffusion. 3-WAY is a block cipher with a block and key length of 96 bits. Key components in the construction of 3-WAY are a 3-bit nonlinear S-box and a linear mapping that can be described by modular polynomial multiplication in $\mathbb{Z}_2^{12}$. The arrangement of the components allows software implementations in the range of 10 Mbit/s on a modern PC and dedicated hardware implementations above 1 Gbit/s using standard technology (1.2μ CMOS). The cipher structure of 3-WAY is shown to be surprisingly strong with respect to both linear and differential cryptanalysis.

1 Introduction

Essentially a block cipher is a keyed permutive mapping (encryption) together with its inverse (decryption). For a practical block cipher it is important that these two mappings can be efficiently implemented in software on a wide variety of processors. For some applications the throughput of software implementations may not be sufficient and dedicated hardware implementations may be necessary. In this case it is an economic advantage if the same circuitry can be used for both encryption and decryption.

In DES [2] the desired properties are realized by iterating a round function that has the so-called Feistel structure. The application of this structure guarantees that encryption and decryption are similar processes, independent of the exact specification of the so-called F-function. With the adoption of the Feistel structure, the design of the F-function can concentrate completely on the desired propagation properties without restrictions imposed by invertibility.

Unfortunately the Feistel structure has important drawbacks. Because only half of the bits of the intermediate result enter the F-function the round function exhibits a large amount of linearity. This linearity is heavily exploited both in the differential cryptanalysis [3] and the linear cryptanalysis [5] of DES. Therefore we propose to use a different, more uniform round structure.

In the cryptographic finite state machine approach [1] the round function is composed of a number of simple invertible steps that treat every bit of the intermediate result in qualitatively the same way. The difference with the Feistel approach is that these steps have to satisfy some additional requirements such as invertibility. For 3-WAY the feature that the same hardware can be used for both encryption and decryption is realized by imposing certain algebraic conditions on the steps. Within these algebraic constraints and the desired propagation properties, the steps are chosen as simple as possible. This has the benefit of a short and elegant cipher description, with no room for possible trapdoors.

After introducing the basic building blocks, the structure of the cipher is presented. Next, we discuss the cryptographic claims and the behaviour of the cipher structure under cryptanalysis. This includes differential cryptanalysis, linear cryptanalysis and attacks based on symmetry. The paper concludes with our most important results. The Appendix contains a reference specification of 3-WAY in the form of a C program.

2 The Basic Building Blocks of the Cipher

2.1 Preliminaries

In this paper all operations will be on binary vectors whose components are indexed starting from 0, e.g. $X = (x_0, x_1, \dots, x_{n-1})^{\mathrm{T}}$. The dimension of a vector is by default denoted by n. If a mapping of vectors is specified in terms of its components, the use of the index i implies the range $0 \leq i < n$. Indices consisting of expressions containing i must be reduced modulo n.

Let μ be a bit permutation that inverts the order of the components of a vector. For $B = \mu(A)$ we have

$$b_i = a_{n-1-i} \ . \tag{1}$$

Clearly $\mu^{-1} = \mu$. This bit permutation plays an important role in the structure of the cipher. The basic building blocks of the cipher γ and θ have been chosen such that $\gamma^{-1} = \mu \circ \gamma \circ \mu$ and $\theta^{-1} = \mu \circ \theta \circ \mu$.

2.2 The Nonlinear Substitution γ

The mapping γ is defined for vectors whose dimension is a multiple of 3. If $B = \gamma(A)$ and the dimension $n = 3k$ we have

$$b_i = \bar{a}_i \oplus \bar{a}_{i+k} a_{i+2k} \ . \tag{2}$$

In fact γ is the parallel execution of k substitutions, acting upon 3-bit blocks (called ***triplets***) consisting of bits a_j, a_{j+k} and a_{j+2k}. The effect of γ on a single triplet can be seen in Table 1.

Actually 3 is the minimum size for an invertible nonlinear substitution box. The box used in γ is the one with the best nonlinear properties (see Sect. 5) that has rotational symmetry and has the desired interaction with μ.

x	000	001	010	100	110	101	011	111
$\gamma(x)$	111	010	100	001	011	110	101	000

Table 1. The effect of γ on a single triplet.

2.3 The Linear Substitution θ

A vector A can be interpreted as a binary polynomial $a(x) = \sum a_i x^i$. The mapping θ is defined for vectors whose dimension is a multiple of 12. If $B = \theta(A)$ and the dimension $n = 12h$ we have

$$b(x) = e(x^h)a(x) \bmod (1 + x^{12h}) \tag{3}$$

with

$$e(x) = 1 + x + x^2 + x^3 + x^5 + x^6 + x^{10} \ . \tag{4}$$

In fact θ is the parallel execution of h substitutions acting upon 12-bit blocks consisting of bits $a_j, a_{j+h}, a_{j+2h}, \dots a_{j+11h}$.

The linear substitution θ was chosen such that every output bit depends on 7 input bits. This is realized in a multiplication by a polynomial with 7 terms modulo $(1+x^m)$. Since m has to be a multiple of 3, the smallest value of m equal to $3 \cdot 2^k$ for some k is 12. The desired interaction with μ is realized by imposing the condition $e(x)e(x^{-1}) \bmod (1+x^{12}) = 1$. From all the candidate polynomials we chose the one with the best propagation properties (see Table 4).

3 The Structure of the Block Cipher

Let π_1 and π_2 be two bit permutations such that $\pi_1 \circ \mu \circ \pi_2 = \mu$, hence the choice of π_1 fixes π_2. For 3-WAY these are blockwise rotations of vector subblocks of length 32 to facilitate software implementations. The encryption process consists of the iterative application of a number of *rounds* r. One 3-WAY round consists of the subsequent application of θ, π_1, γ and π_2 and is denoted by ρ:

$$\rho = \pi_2 \circ \gamma \circ \pi_1 \circ \theta \ .$$

Before every round the intermediate result is XORed with a vector that depends on the secret key and the round number. XORing with K_i is denoted by $\delta(K_i)$. The last round is followed by an extra application of δ and θ. We have

$$\mathrm{E}_K = \theta \circ \delta(K_r) \circ \rho \circ \delta(K_{r-1}) \circ \cdots \circ \rho \circ \delta(K_1) \circ \rho \circ \delta(K_0)$$

with E_K denoting the encryption operation under secret key K. The order of the components and their interaction with μ causes decryption to be a very similar operation to encryption. We can prove

$$\mathrm{D}_K = \mu \circ \left(\theta \circ \delta(K_0') \circ \rho \circ \delta(K_1') \circ \cdots \circ \rho \circ \delta(K_{r-1}') \circ \rho \circ \delta(K_r')\right) \circ \mu$$

with the round keys given by $K'_j = \mu(\theta(K_{r-j}))$.

For efficiency reasons the key schedule is kept as simple as possible. The key length is the block length and every encryption round key is equal to the global key K XORed with a round constant C_j with small Hamming weight. The decryption round keys can be computed by XORing round constants with the so-called decryption key $K' = \mu(\theta(K))$.

3.1 Software and Hardware Implications

The choice of the cipher structure is hardware oriented. The round function (preceded by the round key XOR) can be implemented as the state transition function of a finite state machine. Encryption is performed by loading the plaintext into the state register and iterating the finite state machine r times. The state register now contains the intermediate value that is one application of δ and θ short to be the legitimate ciphertext. The ciphertext is obtained at the output of the step θ in the round function logic. If decryption is performed, the bits are loaded into the state register and to the output register in reverse order. The decryption key can be computed on-chip in a single clock cycle from the encryption key thanks to the accessibility of the first stage of the round function logic. The total gate delay of the finite state machine can be made as small as 4 XORs, 1 NAND and 1 multiplexer, allowing clock speeds of over 150 MHz. If the number of iterations r is equal to 11, encryption (or decryption) of an n-bit block will take 12 clock cycles. The data can be loaded on and off the chip $h = n/12$ bits per clock cycle. For 3-WAY $h = 8$, hence a clock speed of 125 MHz implies an encryption speed of 1 Gbit/s.

In software the (time-consuming) execution of μ before and after decryption can be avoided by writing separate routines for encryption and decryption. The steps γ and θ can be efficiently programmed using bitwise XOR, OR, complementation and shifting. A straightforward C implementation allows an encryption speed of over 2 Mbit/s on a 66 MHz 80486 processor. We expect that optimization and the use of coding in assembly language allows a speedup by at least a factor of 5.

4 Cryptographic Claims

The usefulness of a cryptographic function is based on assumptions about its security. In our opinion these assumptions have to be made explicit in the form of a clear and practical cryptographic claim that accompanies the publication of the cipher. This cryptographic claim serves initially as a challenge for the cryptologic community. As time passes and no weaknesses have been found that refute the cryptographic claim, the cipher can gain credibility. For a system engineer or a user who believes in the validity of the cryptographic claim, it serves as a specification of the security of the cipher.

The additional protection obtained by applying encryption is limited by the external parameters of the cipher system. For instance, an adversary who knows that the plaintext has some type of redundancy can find the key by exhaustive key search. Expressed in number of encryptions, the work factor of this attack depends only on the key length m. An adversary who has temporarily access to a block encryptor(decryptor), can encrypt(decrypt) some chosen plaintext(ciphertext). This can be considered as a partial table reconstruction for the used key. This partial table can be used to gain information about the plaintext in future encryptions. The probability of success for this attack depends only on the block length n. Both examples are instances from the class of attacks that do not exploit the internal structure of the cipher, denoted by ***black box cryptanalysis***. We consider a cryptographic function *BB-secure* if under all circumstances there are no better attacks than black box cryptanalysis.

A block cipher has two external parameters: the block length n and the key length m. 3-WAY is claimed to be BB-secure with respect to its external parameter n (= 96), that is both the block and key length.

5 Cryptanalysis

In this section we want to give a motivation for the choice of the structure and components of the cipher. First we discuss the behaviour of the cipher structure under differential [3] and linear [5] cryptanalysis. Then we treat the measures that are taken against attacks that exploit symmetry in the cipher.

5.1 Differential Cryptanalysis

Differential cryptanalysis exploits the high-probability propagation of certain differences of pairs of plaintext blocks into differences in the corresponding pairs of intermediate results to obtain information about the key.

The cipher consists of the alternation of linear steps $(\pi_1 \circ \theta \circ \pi_2)$ and nonlinear steps γ. A difference X' before the linear step gives a difference $Y' = $ MX' after the step. Here M is the matrix representation of $(\pi_1 \circ \theta \circ \pi_2)$.

The analysis of the difference propagation through the nonlinear step can be made using the pairs XOR distribution table of γ confined to a triplet, given in Table 2. The entry in this table in row x' and column y' represents the number of input pairs of triplets with XOR x' whose corresponding output pairs have XOR y'. If nothing is known about the absolute values of the input triplets, it can be seen that every nonzero input XOR triplet can propagate to four different output XOR triplets, each with probability 1/4. An input XOR and an output XOR are called *compatible* if the input XOR can propagate to the output XOR through γ. From Table 2 it can be seen that an input XOR triplet and an output XOR triplet are compatible if they have an odd number of 1-bits in common, i.e., if their bitwise AND has odd Hamming weight.

An input XOR and an output XOR are compatible if all their component triplets are compatible. Hence if an input XOR has ℓ nonzero triplets it is compatible with $2^{2\ell}$ different output XORs. The input XOR will propagate to any of these compatible output XORs with probability $2^{-2\ell}$. The number of nonzero

	000	001	010	100	011	101	110	111
000	8	-	-	-	-	-	-	-
001	-	2	-	-	2	2	-	2
010	-	-	2	-	2	-	2	2
100	-	-	-	2	-	2	2	2
011	-	2	2	-	-	2	2	-
101	-	2	-	2	2	-	2	-
110	-	-	2	2	2	2	-	-
111	-	2	2	2	-	-	-	2

Table 2. Pairs XOR distribution table for γ confined to a single triplet.

triplets in an XOR vector will be called the *propagation weight* or simply *weight* of an XOR and denoted by $\mathrm{w_p}()$.

A one-round *characteristic* consists of an input XOR X' and an output XOR Y' that is compatible to $\mathrm{M}X'$ (for ease of notation the round boundaries are taken after π_1 and before γ here). The probability of this characteristic is the probability that X' will propagate to Y' through ρ if the absolute values of the inputs are unknown and independent. This probability is equal to $2^{\mathrm{w_p}(Y')}$. By iterating one-round characteristics multiple-round characteristics can be constructed. An f-round characteristic Ω consists of a string of XORs $X'_0, X'_1, \ldots X'_f$ such that every X'_j is compatible with $\mathrm{M}X'_{j-1}$. If the absolute values of the input and the intermediate results are unknown and independent the probability of this characteristic is the product of the probabilities that X'_j will propagate to X'_{j+1} through ρ for $0 \leq j < f$. This probability is equal to $2^{-2\mathrm{w_p}(\Omega)}$ where $\mathrm{w_p}(\Omega)$ is the propagation weight of the characteristic Ω. We have

$$\mathrm{w_p}(\Omega) = \sum_{0<j\leq f} \mathrm{w_p}(X'_j) \ . \tag{5}$$

We are interested in the probability that X'_0 will propagate to X'_f irrespective of the intermediate XOR values. This probability will be the sum of the probabilities of all possible f-round characteristics starting with X'_0 and ending with X'_f. Only the characteristic(s) with the lowest weight will essentially contribute to this sum.

In practice we cannot guarantee the independence of the absolute values of input and intermediate results for complex characteristics. Correlations between absolute bits can occur. In the following subsection we will discuss how to find correlations between different variables consisting of the parity of certain subsets of bits.

5.2 Linear Cryptanalysis

In linear cryptanalysis high correlations between sums modulo 2 (parity) of a subset of input bits and the parity of subsets of output bits are exploited to obtain information about key bits.

The parity of a number of bits of a vector X can be denoted by $V^{\mathrm{T}}X$ where V specifies which bits are included in the binary sum. V is called a *selection vector.*

The correlation between two binary variables is expressed by a *correlation coefficient* between -1 and 1. If the correlation coefficient is e, the probability that the variables are equal is $(1+e)/2$. Two variables are correlated if the correlation coefficient differs from 0.

For $Y = \mathrm{M}X$ the parity $V_{\mathrm{y}}^{\mathrm{T}}Y$ is equal to $V_{\mathrm{x}}^{\mathrm{T}}X$ with $V_{\mathrm{x}} = \mathrm{M}^{\mathrm{T}}V_{\mathrm{y}}$. The two parities have a correlation of $+1$. If we have $Y = \mathrm{M}X + \mathrm{K}$ with K a constant vector, the correlation can also be -1 depending on the value of the constant K. From the rotation invariant properties of θ and its interaction with μ it can easily be proven that $\mathrm{M}^{\mathrm{T}} = \mathrm{M}^{-1}$ hence $V_{\mathrm{y}} = \mathrm{M}V_{\mathrm{x}}$.

The analysis of correlations between inputs and outputs of the nonlinear step can be made using the linear approximation table of γ confined to a triplet, given in Table 3. In this table the entry e_{ij} in row v_i and column v_j represents the deviation of the number of occurrences $v_i^{\mathrm{T}}x = v_j^{\mathrm{T}}y$ from 4. The corresponding correlation coefficients can easily be found by dividing these entries by 4. It can be seen that every nonzero input parity $v_i^{\mathrm{T}}x$ is correlated with four output combinations $v_j^{\mathrm{T}}y$, each with correlation $\pm 1/2$. An input selection and an output selection are called *compatible* if their corresponding parities are correlated. By comparing Tables 2 and 3 it can be seen that the compatibility conditions for selections are the same as those for XORs.

	000	001	010	100	011	101	110	111
000	4	-	-	-	-	-	-	-
001	-	−2	-	-	2	−2	-	−2
010	-	-	−2	-	−2	-	2	−2
100	-	-	-	−2	-	2	−2	−2
011	-	−2	2	-	-	2	2	-
101	-	2	-	−2	2	-	2	-
110	-	-	−2	2	2	2	-	-
111	-	−2	−2	−2	-	-	-	2

Table 3. Linear approximation distribution table for γ confined to a single triplet.

An input selection and an output selection to γ are compatible if all their component triplet selections are compatible. Moreover, if an input selection has weight ℓ it is compatible with $2^{2\ell}$ different output selection vectors with correlation coefficient $\pm 2^{-\ell}$.

A one-round *linear approximation* consists of an input selection V_{x} and an output selection V_{y} that is compatible with $\mathrm{M}V_{\mathrm{x}}$. The correlation coefficient of this linear approximation is equal to $\pm 2^{\mathrm{w_P}(V_{\mathrm{y}})}$. By combining one-round linear approximations with equal intermediate selection vectors multiple-round linear approximations can be constructed. An f-round linear approximation Λ consists of a string of selections $V_0, V_1, \ldots V_f$ such that every V_j is compatible with $\mathrm{M}V_{j-1}$. The correlation coefficient of this linear approximation is the product of the correlation coefficients of the one-round linear approximations defined by V_j and V_{j+1} for $0 \leq j < f$. This correlation coefficient is equal to $2^{-\mathrm{w_p}(\Lambda)}$ where $\mathrm{w_p}(\Lambda)$ is the weight of the linear approximation Λ. We have

$$\mathrm{w_p}(\Lambda) = \sum_{0<j\leq f} \mathrm{w_p}(V_j) \ . \tag{6}$$

5.3 Analogy between Differential and Linear Cryptanalysis

From this discussion it can be seen that in the case of 3-WAY there is an strong analogy between differential cryptanalysis and linear cryptanalysis. The resistance against linear and differential cryptanalysis can be investigated by interpreting the same propagation structures, called *propagation chains* in two different ways. Both in differential and linear cryptanalysis the effort of a successful attack is in the order of magnitude of $2^{2\mathrm{w_p}(\Omega)}$ encryptions where Ω is an f-round propagation chain and f is the number of rounds minus 1 [3, 5].

Once the length has been fixed, the single criterion for the choice of the blockwise bit rotations π_1 and π_2 is the elimination of propagation chains with low weight.

5.4 Attacks based on Symmetry

An important class of attacks is based on the exploitation of symmetry in the cryptographic function. A well known example is the method to reduce exhaustive keysearch of DES by making use of the complementation property. More recent examples can be found in [4] where the regularity in the key schedule is used to construct chosen key attacks and speed up exhaustive key search.

The round keys are equal to the global key XORed with the round constants. The idea is to choose round constants as simple as possible that eliminate all exploitable symmetric properties. This choice is not affected by propagation chain considerations.

Many undesirable symmetric properties are special cases of one of the two following properties:

- There are affine mappings $\tau_\mathrm{k}, \tau_\mathrm{p}$ and τ_c, such that for some keys $\tau_\mathrm{c} \circ \mathrm{E}_{\tau_\mathrm{k}(K)} \circ \tau_\mathrm{p}$ is equal to E_K or D_K.
- There are keys such that the last $(r-q)$ rounds of the cipher (encryption or decryption) under one key are the same mapping as the first $(r-q)$ rounds of (encryption or decryption) under another key with q small.

The round constants are derived from the state c_j of a linear feedback shift register with length 8. In polynomial representation we have

$$\mathrm{c}_j(x) = (1 + x + x^3)x^j \bmod (1 + x^4 + x^8) \ . \tag{7}$$

The order of the feedback polynomial is 12, hence $x^{12} = 1 \bmod (1 + x^4 + x^8)$. The calculation of the round constants C_j in polynomial representation is

$$\mathrm{C}_j(x) = (x^{2\mathrm{h}} + x^{3\mathrm{h}} + x^{8\mathrm{h}} + x^{9\mathrm{h}})\mathrm{c}_j(x) \tag{8}$$

with $12h = n$. For the inverse round constants we have $\mathrm{C}'_j = \mu(\theta(\mathrm{C}_{r-j}))$. It can be seen that

$$\mathrm{C}'_j(x) = (x^{2\mathrm{h}} + x^{3\mathrm{h}} + x^{8\mathrm{h}} + x^{9\mathrm{h}})\mathrm{c}'_j(x) \tag{9}$$

with $\mathrm{c}'_j(x)$ given by

$$\mathrm{c}'_j(x) = (1 + x^4 + x^5 + x^7)x^j \bmod (1 + x^4 + x^8) \ . \tag{10}$$

The encryption and decryption round constants can be generated and applied with the same circuitry. The only difference is the initial value of the 8-bit linear feedback shift register. Observe that the difference between the round constants of two subsequent encryption or decryption rounds is different for all cases.

6 Results

A C program has been written that determines whether there are propagation chains with a weight per round smaller than a given lower bound ℓ for the 3-Way structure. Basically this program executes a pruned tree search for low weight propagation chains for all initial propagation vectors with weight smaller than ℓ.

This program was used to select a permutation π_1 for the 96-bit version of 3-Way. Excellent results were obtained with a permutation where the 96-bit vector is divided into three 32-bit words. Two of the three words are cyclically shifted, one by 1 bit position and one by 10 bit positions. If $B = \pi_1(A)$ we have

$$b_i = a_{(i+10) \bmod 32}, b_{i+32} = a_{i+32}, b_{i+64} = a_{(i-1) \bmod 32 + 64} \text{ for } 0 \leq i < 32 \ . \tag{11}$$

For this choice of π_1, all propagation chains of 5 rounds or more have a weight not smaller than 6 per round. This implies that m-round characteristics of 5 rounds or more have maximum probability 2^{-12m} and that m-round linear approximations have maximum correlation 2^{-6m}. To give an idea of the impact of these figures we compare them with the analogous figures for the Data Encryption Standard. DES has an iterative characteristic with a probability of $2^{-3.6}$ per round [3] and a 15-round linear approximation with a correlation coefficient of $2^{-21.2}$ or $2^{-1.4}$ per round [5].

These strong results with respect to differential and linear cryptanalysis are the consequence of the propagation properties of the linear step θ. In Table 4 the interaction between the Hamming weight (**not** the propagation weight) of $a(x)$ and $b(x) = e(x)a(x) \bmod (1 + x^{12})$ can be observed. For nonzero vectors, the sum of the Hamming weight of $a(x)$ and $b(x)$ is at least 8. By a good choice of π_1 the linear mapping M inherits these good properties with respect to propagation chains. Table 5 gives the number of pairs $(X, \mathrm{M}X)$ for given propagation

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	12	-	-	-	-	-
2	-	-	-	-	-	-	60	-	-	-	6	-	-
3	-	-	-	-	-	180	-	-	-	40	-	-	-
4	-	-	-	-	255	-	-	-	240	-	-	-	-
5	-	-	-	180	-	-	-	600	-	-	-	12	-
6	-	-	60	-	-	-	804	-	-	-	60	-	-
7	-	12	-	-	-	600	-	-	-	180	-	-	-
8	-	-	-	-	240	-	-	-	255	-	-	-	-
9	-	-	-	40	-	-	-	180	-	-	-	-	-
10	-	-	6	-	-	-	60	-	-	-	-	-	-
11	-	-	-	-	-	12	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	-	1

Table 4. Number of pairs $(a(x), b(x))$ with $b(x) = e(x)a(x) \bmod (1 + x^{12})$ with the **Hamming** weight of $a(x)$ given at the left and the Hamming weight of $b(x)$ at the top

weight of X and MX for the mentioned choice of π_1. It can be seen that in a propagation chain every vector with a propagation weight w not larger than 4 must be followed by a vector with weight not smaller than $8 - w$. Hence, it can be deduced from Table 5 that there are no propagation chains of even length with weight smaller than 4 per round.

7 Conclusions

3-WAY is a block cipher that is the product of a new design approach. The cipher is suitable for both software and hardware implementations. It is shown that for 3-WAY the resistance against both differential and linear cryptanalysis can be studied using the *same* propagation structures. The high resistance against these types of cryptanalysis is realized by the combination of high diffusion (θ, π_1, π_2) and distributed nonlinearity (γ).

References

[1] J. Daemen, R. Govaerts and J. Vandewalle, A Hardware Design Model for Cryptographic Algorithms, *Computer Security-Esorics '92*, pp. 419–434. Lecture Notes in Computer Science, vol. 648, Springer-Verlag, Berlin 1992.

[2] *Data Encryption Standard*, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[3] E. Biham and A. Shamir, Differential Cryptanalysis of DES-like Cryptosystems, *Journal of Cryptology*, Springer-Verlag, Vol. 4, No. 1, pp. 3–72, 1991.

	1	2	3	4	5	6	7	8
1	-	-	-	-	-	-	96	-
2	-	-	-	-	-	480	-	-
3	-	-	-	-	1440	-	-	55
4	-	-	-	2040	-	7	168	5335
5	-	-	1440	-	25	313	12480	71138
6	-	480	-	7	313	?	?	?
7	96	-	-	168	12480	?	?	?
8	-	-	55	5335	71138	?	?	?
9	-	19	1122	28012	265865	?	?	?
10	-	195	6381	90042	431964	?	?	?
11	39	836	18775	119868	457174	?	?	?
12	25	1883	20751	113010	776241	?	?	?
13	32	2017	17408	159098	2682584	?	?	?
14	13	1677	21418	469917	6262878	?	?	?

Table 5. Number of pairs (X, MX) with a given input propagation weight (left) and output propagation weight (top).

[4] E. Biham, New Types of Cryptanalytic Attacks Using Related Keys, *Abstracts Eurocrypt '93.*

[5] M. Matsui, Linear Cryptanalysis Method for DES Cipher, *Abstracts Eurocrypt '93.*

Appendix

File threewayref.c

```
/*********************************************************************\
*                                                                     *
* C specification of the threeway block cipher                        *
*                                                                     *
\*********************************************************************/

#define   STRT_E   0x0b0b /* round constant of first encryption round */
#define   STRT_D   0xb1b1 /* round constant of first decryption round */
#define     NMBR       11 /* number of rounds is 11                   */

typedef   unsigned long int  word32 ;
                 /* the program only works correctly if long = 32bits */



void mu(word32 *a)       /* inverts the order of the bits of a */
{
int i ;
word32 b[3] ;

b[0] = b[1] = b[2] = 0 ;
for( i=0 ; i<32 ; i++ )
   {
   b[0] <<= 1 ; b[1] <<= 1 ; b[2] <<= 1 ;
   if(a[0]&1) b[2] |= 1 ;
   if(a[1]&1) b[1] |= 1 ;
   if(a[2]&1) b[0] |= 1 ;
   a[0] >>= 1 ; a[1] >>= 1 ; a[2] >>= 1 ;
   }

a[0] = b[0] ;      a[1] = b[1] ;      a[2] = b[2] ;
}


void gamma(word32 *a)   /* the nonlinear step */
{
word32 b[3] ;

b[0] = a[0] ^ (a[1]|(~a[2])) ;
b[1] = a[1] ^ (a[2]|(~a[0])) ;
b[2] = a[2] ^ (a[0]|(~a[1])) ;

a[0] = b[0] ;      a[1] = b[1] ;      a[2] = b[2] ;
}
```

```
void theta(word32 *a)    /* the linear step */
{
word32 b[3];

b[0] = a[0] ^  (a[0]>>16) ^ (a[1]<<16) ^     (a[1]>>16) ^ (a[2]<<16) ^
               (a[1]>>24) ^ (a[2]<<8)  ^     (a[2]>>8)  ^ (a[0]<<24) ^
               (a[2]>>16) ^ (a[0]<<16) ^     (a[2]>>24) ^ (a[0]<<8)  ;
b[1] = a[1] ^  (a[1]>>16) ^ (a[2]<<16) ^     (a[2]>>16) ^ (a[0]<<16) ^
               (a[2]>>24) ^ (a[0]<<8)  ^     (a[0]>>8)  ^ (a[1]<<24) ^
               (a[0]>>16) ^ (a[1]<<16) ^     (a[0]>>24) ^ (a[1]<<8)  ;
b[2] = a[2] ^  (a[2]>>16) ^ (a[0]<<16) ^     (a[0]>>16) ^ (a[1]<<16) ^
               (a[0]>>24) ^ (a[1]<<8)  ^     (a[1]>>8)  ^ (a[2]<<24) ^
               (a[1]>>16) ^ (a[2]<<16) ^     (a[1]>>24) ^ (a[2]<<8)  ;

a[0] = b[0] ;      a[1] = b[1] ;      a[2] = b[2] ;
}

void pi_1(word32 *a)
{
a[0] = (a[0]>>10) ^ (a[0]<<22);
a[2] = (a[2]<<1)  ^ (a[2]>>31);
}

void pi_2(word32 *a)
{
a[0] = (a[0]<<1)  ^ (a[0]>>31);
a[2] = (a[2]>>10) ^ (a[2]<<22);
}

void rho(word32 *a)    /* the round function       */
{
theta(a) ;
pi_1(a) ;
gamma(a) ;
pi_2(a) ;
}

void rndcon_gen(word32 strt,word32 *rtab)
{                            /* generates the round constants */
int i ;

for(i=0 ; i<=NMBR ; i++ )
   {
   rtab[i] = strt ;
   strt <<= 1 ;
   if( strt&0x10000 ) strt ^= 0x11011 ;
   }
}
```

```
void encrypt(word32 *a, word32 *k)
{
char i ;
word32 rcon[NMBR+1] ;

rndcon_gen(STRT_E,rcon) ;
for( i=0 ; i<NMBR ; i++ )
   {
   a[0] ^= k[0] ^ (rcon[i]<<16) ;
   a[1] ^= k[1] ;
   a[2] ^= k[2] ^ rcon[i] ;
   rho(a) ;
   }
a[0] ^= k[0] ^ (rcon[NMBR]<<16) ;
a[1] ^= k[1] ;
a[2] ^= k[2] ^ rcon[NMBR] ;
theta(a) ;
}


void decrypt(word32 *a, word32 *k)
{
char i ;
word32 ki[3] ;          /* the ‘inverse’ key             */
word32 rcon[NMBR+1] ;   /* the ‘inverse’ round constants */

ki[0] = k[0] ; ki[1] = k[1] ; ki[2] = k[2] ;
theta(ki) ;
mu(ki) ;

rndcon_gen(STRT_D,rcon) ;

mu(a) ;
for( i=0 ; i<NMBR ; i++ )
   {
   a[0] ^= ki[0] ^ (rcon[i]<<16) ;
   a[1] ^= ki[1] ;
   a[2] ^= ki[2] ^ rcon[i] ;
   rho(a) ;
   }
a[0] ^= ki[0] ^ (rcon[NMBR]<<16) ;
a[1] ^= ki[1] ;
a[2] ^= ki[2] ^ rcon[NMBR] ;
theta(a) ;
mu(a) ;
}
```

Testprogram

```
#include <stdio.h>
#include <stdlib.h>
#include "threewayref.c"

void printvec(word32 *a)
{
printf("%08x %08x %08x\n",a[2],a[1],a[0]) ;
}

main()
{
word32 a[3], k[3] ;

scanf("%x %x %x %x %x %x",a+2,a+1,a,k+2,k+1,k) ;
printf("key        : ") ; printvec(k) ;
printf("plaintext  : ") ; printvec(a) ; encrypt(a,k) ;
printf("ciphertext : ") ; printvec(a) ; decrypt(a,k) ;
/* printf("checking   : ") ; printvec(a) ;  */
}
```

Testvalues

```
key        : 00000000 00000000 00000000
plaintext  : 00000001 00000001 00000001
ciphertext : ad21ecf7 83ae9dc4 4059c76e

key        : 00000004 00000005 00000006
plaintext  : 00000001 00000002 00000003
ciphertext : cab920cd d6144138 d2f05b5e

key        : bcdef012 456789ab def01234
plaintext  : 01234567 9abcdef0 23456789
ciphertext : 7cdb76b2 9cdddb6d 0aa55dbb

key        : cab920cd d6144138 d2f05b5e
plaintext  : ad21ecf7 83ae9dc4 4059c76e
ciphertext : 15b155ed 6b13f17c 478ea871
```