

A Proposal for a New Block Encryption Standard

Xuejia Lai James L. Massey

Institute for Signal and Information Processing
Swiss Federal Institute of Technology
CH-8092 Zürich, Switzerland

Abstract

A new secret-key block cipher is proposed as a candidate for a new encryption standard. In the proposed cipher, the plaintext and the ciphertext are 64 bit blocks, while the secret key is 128 bit long. The cipher is based on the design concept of "mixing operations from different algebraic groups". The cipher structure was chosen to provide confusion and diffusion and to facilitate both hardware and software implementations.

1 Introduction

A new secret-key block cipher is proposed herein as a candidate for a new encryption standard. In the proposed cipher, the plaintext and the ciphertext are 64 bit blocks, while the secret key is 128 bit long. The cipher is based on the design concept of "mixing operations from different algebraic groups". The required confusion is achieved by successively using three different group operations on pairs of 16-bit subblocks and the cipher structure was chosen to provide the necessary diffusion. The cipher is so constructed that the deciphering process is the same as the enciphering process once the decryption key subblocks have been computed from the encryption key subblocks. The cipher structure was chosen to facilitate both hardware and software implementations.

The cipher is described in Section 2. Section 3 considers the relation of the three chosen different operations to one another and the effect of their "mixing". The design principles for the cipher are discussed in Section 4. Section 5 discusses

the implementation of the cipher in software as well as in hardware. A C-language program of the cipher is given in Appendix B together with examples that can be used to test the correctness of implementations.

2 Description of the Proposed Cipher

The computational graph of the encryption process is shown in Fig.1. The process consists of 8 similar rounds followed by an output transformation. The complete first round and the output transformation are depicted in Fig.1.

In the encryption process shown in Fig.1, three different group operations on pairs of 16-bit subblocks are used, namely,

- bit-by-bit exclusive-OR of two 16-bit subblocks, denoted as \oplus ;
- addition of integers modulo 2^{16} where the subblock is treated as the usual radix-two representation of an integer, the resulting operation is denoted as \boxplus ;
- multiplication of integers modulo $2^{16} + 1$ where the subblock is treated as the usual radix-two representation of an integer except that the all-zero subblock is treated as representing 2^{16} , and the resulting operation is denoted as \odot .

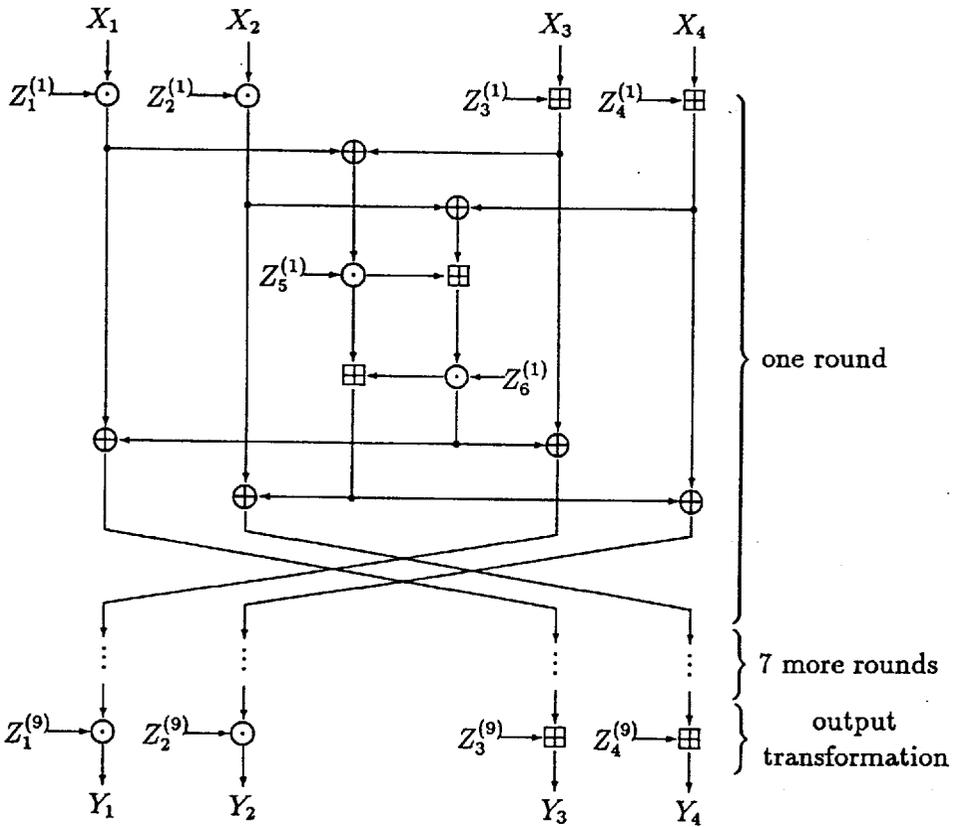
For example,

$$(0, \dots, 0) \odot (1, 0, \dots, 0) = (1, 0, \dots, 0, 1)$$

because

$$2^{16} 2^{15} \bmod (2^{16} + 1) = 2^{15} + 1.$$

The 64-bit plaintext block is partitioned into four 16-bit subblocks, the i -th of which is denoted as X_i in Fig.1. The four plaintext subblocks are then transformed into four 16-bit ciphertext subblocks, Y_1, Y_2, Y_3 and Y_4 , under the control of 52 key subblocks of 16 bits, where the six key subblocks used in the r -th ($r=1, \dots, 8$) round are denoted as $Z_1^{(r)}, \dots, Z_6^{(r)}$ and where the four key subblocks used in the output transformation are denoted as $Z_1^{(9)}, Z_2^{(9)}, Z_3^{(9)}, Z_4^{(9)}$.



X_i : 16-bit plaintext subblock

Y_i : 16-bit ciphertext subblock

$Z_i^{(r)}$: 16-bit key subblock

\oplus : bit-by-bit exclusive-OR of 16-bit subblocks

\boxplus : addition modulo 2^{16} of 16-bit integers

\odot : multiplication modulo $2^{16} + 1$ of 16-bit integers
with the zero subblock corresponding to 2^n

Figure 1: Computational graph for encryption

Decryption

The computational graph of the decryption process is essentially the same as that of the encryption process, the only change being that the decryption key subblocks are computed from the encryption key subblocks as shown in the following table, where Z^{-1} denotes the multiplicative inverse (modulo $2^{16} + 1$) of Z , i.e., $Z \odot Z^{-1} = 1$ and $-Z$ denotes the additive inverse (modulo 2^{16}) of Z , i.e., $-Z \boxplus Z = 0$.

Encryption key subblocks

Decryption key subblocks

1-st round	$Z_1^{(1)} Z_2^{(1)} Z_3^{(1)} Z_4^{(1)}$ $Z_5^{(1)} Z_6^{(1)}$	1-st round	$Z_1^{(9)-1} Z_2^{(9)-1} - Z_3^{(9)} - Z_4^{(9)}$ $Z_5^{(8)} Z_6^{(8)}$
2-nd round	$Z_1^{(2)} Z_2^{(2)} Z_3^{(2)} Z_4^{(2)}$ $Z_5^{(2)} Z_6^{(2)}$	2-nd round	$Z_1^{(8)-1} Z_2^{(8)-1} - Z_3^{(8)} - Z_4^{(8)}$ $Z_5^{(7)} Z_6^{(7)}$
3-rd round	$Z_1^{(3)} Z_2^{(3)} Z_3^{(3)} Z_4^{(3)}$ $Z_5^{(3)} Z_6^{(3)}$	3-rd round	$Z_1^{(7)-1} Z_2^{(7)-1} - Z_3^{(7)} - Z_4^{(7)}$ $Z_5^{(6)} Z_6^{(6)}$
4-th round	$Z_1^{(4)} Z_2^{(4)} Z_3^{(4)} Z_4^{(4)}$ $Z_5^{(4)} Z_6^{(4)}$	4-th round	$Z_1^{(6)-1} Z_2^{(6)-1} - Z_3^{(6)} - Z_4^{(6)}$ $Z_5^{(5)} Z_6^{(5)}$
5-th round	$Z_1^{(5)} Z_2^{(5)} Z_3^{(5)} Z_4^{(5)}$ $Z_5^{(5)} Z_6^{(5)}$	5-th round	$Z_1^{(5)-1} Z_2^{(5)-1} - Z_3^{(5)} - Z_4^{(5)}$ $Z_5^{(4)} Z_6^{(4)}$
6-th round	$Z_1^{(6)} Z_2^{(6)} Z_3^{(6)} Z_4^{(6)}$ $Z_5^{(6)} Z_6^{(6)}$	6-th round	$Z_1^{(4)-1} Z_2^{(4)-1} - Z_3^{(4)} - Z_4^{(4)}$ $Z_5^{(3)} Z_6^{(3)}$
7-th round	$Z_1^{(7)} Z_2^{(7)} Z_3^{(7)} Z_4^{(7)}$ $Z_5^{(7)} Z_6^{(7)}$	7-th round	$Z_1^{(3)-1} Z_2^{(3)-1} - Z_3^{(3)} - Z_4^{(3)}$ $Z_5^{(2)} Z_6^{(2)}$
8-th round	$Z_1^{(8)} Z_2^{(8)} Z_3^{(8)} Z_4^{(8)}$ $Z_5^{(8)} Z_6^{(8)}$	8-th round	$Z_1^{(2)-1} Z_2^{(2)-1} - Z_3^{(2)} - Z_4^{(2)}$ $Z_5^{(1)} Z_6^{(1)}$
output transform.	$Z_1^{(9)} Z_2^{(9)} Z_3^{(9)} Z_4^{(9)}$	output transform.	$Z_1^{(1)-1} Z_2^{(1)-1} - Z_3^{(1)} - Z_4^{(1)}$

The key schedule

The 52 key subblocks used in the encryption process are generated from the 128 bit user-selected key as follows:

The 128 bit user-selected key is partitioned into 8 subblocks that are directly used as the first eight key subblocks, where the ordering of key subblocks is as follows: $Z_1^{(1)}, Z_2^{(1)}, \dots, Z_6^{(1)}, Z_1^{(2)}, \dots, Z_6^{(2)}, \dots, Z_1^{(8)}, \dots, Z_6^{(8)}, Z_1^{(9)}, Z_2^{(9)}, Z_3^{(9)}, Z_4^{(9)}$.

The 128 bit user-selected key is then cyclic shifted to the left by 25 positions, after which the resulting 128 bit block is again partitioned into eight subblocks that are

taken as the next eight key subblocks. The obtained 128 bit block is cyclic shifted again to the left by 25 positions to produce the next eight key subblocks, and this procedure is repeated until all 52 key subblocks have been generated.

3 Group Operations and Their Interaction

The cipher is based on the design concept of "mixing operations from different algebraic groups having the same number of elements". Group operations were chosen because the statistical relation of random variables X, Y, Z related by a group operation as $Y = X * Z$ has the desired "perfect secrecy" property, i.e., if one of the random variables is chosen equally likely to be any group element, then the other two random variables are statistically independent. The interaction of different group operations contributes to the "confusion" required for a secure cipher. In this section, the interaction of different operations will be considered in terms of isotopism of quasigroups and in terms of polynomial expressions.

The three operations as quasigroup operations

Def. Let S be a set and let $*$ denote an operation from pairs (a, b) of elements of S to an element $a * b$ of S . Then $(S, *)$ is said to be a *quasigroup* if, for any $a, b \in S$, the equations $a * x = b$ and $y * a = b$ both have exactly one solution in S . A *group* is a quasigroup in which the operation is associative, i.e., for which $a * (b * c) = (a * b) * c$ for all a, b and c in S . Quasigroups $(S_1, *_1), (S_2, *_2)$ are said to be *isotopic* if there are bijective mappings $\theta, \phi, \psi : S_1 \rightarrow S_2$, such that, for all $x, y \in S_1$, $\theta(x) *_2 \phi(y) = \psi(x *_1 y)$. Such a triple (θ, ϕ, ψ) is then called an *isotopism* of $(S_1, *_1)$ upon $(S_2, *_2)$. Two groups are said to be *isomorphic* if they are isotopic as quasigroups for which the isotopism is (θ, θ, θ) .

It can be shown that two groups are isomorphic if and only if they are isotopic [3].

Let n be one of the integers 1, 2, 4, 8 or 16 so that the integer $2^n + 1$ is a prime, and let Z_{2^n} denote the ring of integers modulo 2^n . Let $(Z_{2^n+1}^*, \cdot)$ denote the multiplicative group of the non-zero elements of the field Z_{2^n+1} , let $(Z_{2^n}, +)$ denote the additive group of the ring Z_{2^n} , and let (F_2^n, \oplus) denote the group of n -tuples over F_2 under the bitwise exclusive-or operation. Then the following theorem states some of the "incompatibility" properties of these groups.

Theorem 1 For $n \in \{1, 2, 4, 8, 16\}$:

- 1) Quasigroups (F_2^n, \oplus) and $(Z_{2^n}, +)$ are not isotopic for $n \geq 2$.
- 2) Quasigroups $(Z_{2^n+1}^*, \cdot)$ and (F_2^n, \oplus) are not isotopic for $n \geq 2$.
- 3) (θ, ϕ, ψ) is an isotopism of $(Z_{2^n+1}^*, \cdot)$ upon $(Z_{2^n}, +)$ if and only if there exist constants $c_1, c_2 \in Z_{2^n}$ and a primitive element a of the field $Z_{2^n+1}^*$ such that, for all x in Z_{2^n} ,

$$\theta(x) - c_1 = \phi(x) - c_2 = \psi(x) - (c_1 + c_2) = \log_a(x), \quad (1)$$

i.e., any isotopism between these groups is essentially the logarithm. Moreover, if (θ, ϕ, ψ) is an isotopism, none of these maps will be the "mixing mapping" m from $Z_{2^n+1}^*$ to Z_{2^n} defined by $m(i) = i$, for $i \neq 2^n$ and $m(2^n) = 0$ when $n \geq 2$.

Proof.

- 1) For $n \geq 2$, the groups (F_2^n, \oplus) and $(Z_{2^n}, +)$ are not isomorphic because $(Z_{2^n}, +)$ is a cyclic group while (F_2^n, \oplus) is not. Thus, they are not isotopic as quasigroups.
- 2) $(Z_{2^n+1}^*, \cdot)$ and $(Z_{2^n}, +)$ are isomorphic groups for $n = 1, 2, 4, 8, 16$ because both groups are cyclic. Thus, $(Z_{2^n+1}^*, \cdot)$ is isotopic to (F_2^n, \oplus) if and only if $(Z_{2^n}, +)$ is isotopic to (F_2^n, \oplus) .
- 3) If (1) holds, then for any x, y in Z_{2^n+1} ,

$$\psi(x \cdot y) = \log_a(x \cdot y) + c_1 + c_2 = \log_a(x) + \log_a(y) + c_1 + c_2 = \theta(x) + \phi(y).$$

Now suppose that (θ, ϕ, ψ) is an isotopism. Then for all $x, y \in Z_{2^n+1}^*$, $\theta(x) + \phi(y) = \psi(x \cdot y)$. Let $\theta_1(x) = \theta(x) - \theta(1)$, $\phi_1(x) = \phi(x) - \phi(1)$ and $\psi_1(x) = \psi(x) - \psi(1)$, then $(\theta_1, \phi_1, \psi_1)$ is also an isotopism and $\psi_1(1) = \theta_1(1) = \phi_1(1) = 0$. In the equation $\theta_1(x) + \phi_1(y) = \psi_1(x \cdot y)$, setting first x and then y to 1 results in $\theta_1(y) = \phi_1(y) = \psi_1(y)$ so that $\psi_1(x \cdot y) = \psi_1(x) + \psi_1(y)$. Let a be the element of $Z_{2^n+1}^*$ such that $\psi_1(a) = 1$, then $\psi_1(a^i) = i$ for $i = 1, 2, \dots, 2^n - 1$ and $\psi_1(a^{2^n}) = 0$. This implies that a is a primitive element of Z_{2^n+1} . Thus, for each $x \in Z_{2^n+1}^*$, there exists a $t \in Z_{2^n}$ such that,

$$\psi_1(x) = \psi_1(a^t) = t = t \log_a(a) = \log_a(a^t) = \log_a(x).$$

Letting $c_1 = \theta(1)$, $c_2 = \phi(1)$, we arrive at equation (1).

Finally, suppose that the mixing mapping m is an isotopism. Then $m(x) = \log_a(x) + c$ implies $1 = m(1) = \log_a(1) + c = c$, $2 = m(2) = \log_a(2) + 1$, which implies that $a = 2$, and $0 = m(2^n) = \log_2(2^n) + 1 = n + 1$, which implies that $n=1$.

Polynomial expressions for multiplication and addition

Under the mixing mapping m , multiplication modulo $2^n + 1$, which is a bilinear function over the field Z_{2^n+1} , is a two variable function over the ring Z_{2^n} , which we denote by $x \odot y$. Similarly, under the inverse mixing mapping m^{-1} , addition modulo 2^n , which is an affine function in each argument over the ring Z_{2^n} , is a two variable function over the field Z_{2^n+1} , which we denote by $F(X, Y)$. Here and hereafter in this section, we denote arguments with lower-case letters when we consider them to be elements of Z_{2^n} and with upper-case letters when we consider them to be elements of Z_{2^n+1} . For example, when $n = 1$, we have

$$x + y \text{ mod } 2 \longleftrightarrow F(X, Y) = 2XY \text{ mod } 3,$$

$$XY \text{ mod } 3 \longleftrightarrow x \odot y = x + y + 1 \text{ mod } 2.$$

Theorem 2 For $n \in \{2, 4, 8, 16\}$:

1. For any fixed $X \neq 2^n$ (i.e., $x \neq 0$), the function $F(X, Y)$, corresponding to addition $x + y \pmod{2^n}$ in Z_{2^n} , is a polynomial in Y over $Z_{2^{n+1}}$ with degree $2^n - 1$. Similarly, for any fixed $Y \neq 2^n$, $F(X, Y)$ is a polynomial in X over $Z_{2^{n+1}}$ with degree $2^n - 1$.
2. For any fixed $x \neq 0, 1$ (i.e., $X \neq 2^n, 1$), the function $x \odot y$, corresponding to multiplication $XY \pmod{2^n + 1}$ in $Z_{2^{n+1}}$ cannot be written as a polynomial in y over Z_{2^n} . Similarly, for any fixed $y \neq 0, 1$, $x \odot y$ is not a polynomial in x over Z_{2^n} .

Example 1

For $n = 2$, in Z_5 , the function $F(X, Y)$ corresponding to $x + y \pmod{4}$ is

$$F(X, Y) = 3(X^3Y^2 + X^2Y^3) + 3(X^3Y + XY^3) + 2X^2Y^2 + 4(X^2Y + XY^2).$$

Proof.

1. In any finite field $GF(q)$, we have

$$\prod_{j \neq i, \alpha_j \neq 0} (x - \alpha_j)(-\alpha_i) = \begin{cases} \prod_{j(j \neq i), \alpha_j \neq 0} (\alpha_i - \alpha_j)(-\alpha_i) = -\prod_{\alpha_i \neq 0} \alpha_i = 1 & x = \alpha_i \\ 0 & x \neq \alpha_i, 0. \end{cases}$$

Thus, every function $f(x)$ from the set $GF(q) - \{0\}$ to $GF(q) - \{0\}$ can be written as a polynomial over $GF(q)$ of degree at most $q - 2$ as follows:

$$f(x) = \sum_{\alpha_i \in GF(q) - \{0\}} f(\alpha_i) \prod_{\alpha_j \neq \alpha_i, 0} (x - \alpha_j)(-\alpha_i). \quad (2)$$

The function $F(A, X)$ corresponding to $a + x \pmod{2^n}$ is a function from $GF(2^n + 1) - \{0\}$ to $GF(2^n + 1) - \{0\}$ in X . According to (2), this function can be written as

$$\begin{aligned} F(A, X) &= \begin{cases} A + X & 1 \leq X \leq 2^n - A \\ A + X + 1 & 2^n - A < X \leq 2^n \end{cases} \\ &= \sum_{I=1}^{2^n - A} (A + I) \prod_{\substack{J \neq I \\ 1 \leq J \leq 2^n}} (X - J)(-I) + \sum_{I=2^n - A + 1}^{2^n} (A + I + 1) \prod_{\substack{J \neq I \\ 1 \leq J \leq 2^n}} (X - J)(-I) \\ &= \sum_{I=1}^{2^n} (A + I) \prod_{\substack{J \neq I \\ 1 \leq J \leq 2^n}} (X - J)(-I) + \sum_{I=2^n - A + 1}^{2^n} \prod_{\substack{J \neq I \\ 1 \leq J \leq 2^n}} (X - J)(-I). \end{aligned}$$

The coefficient of $X^{2^n - 1}$ in $F(A, X)$ is

$$\begin{aligned} &\sum_{I=1}^{2^n} (A + I)(-I) + \sum_{I=2^n - A + 1}^{2^n} (-I) = -A \sum_{I=1}^{2^n} I - \sum_{I=1}^{2^n} I^2 + \sum_{I=-A}^{-1} (-I) \\ &= \sum_{I=1}^A I = \frac{A(A+1)}{2}, \end{aligned}$$

which is zero if and only if $A = 0$ or $A = -1 = 2^n$, which cases are excluded by the assumption. Thus, $\deg F(X, A) = 2^n - 1$.

2. We show first the following lemma:

Lemma 1 If $f(x)$ is a polynomial over Z_{2^n} , then for all $x \in Z_{2^n}$,

$$f(2x) \bmod 2 = f(0) \bmod 2.$$

Proof. Let $f(x) = a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$.
Then for all $x \in Z_{2^n}$,

$$f(2x) = a_k (2x)^k + a_{k-1} (2x)^{k-1} + \dots + a_1 2x + a_0.$$

Taking both sides of this equation modulo 2^n , we have $f(2x) = 2e + a_0 \bmod 2^n$, where e is an element in Z_{2^n} . Thus, $f(2x) \bmod 2 = a_0 \bmod 2 = f(0) \bmod 2$.

Now let $n > 1$, then for every integer a , $1 < a < 2^n$, there exists an integer $x_0 \in \{1, 2, \dots, 2^n\}$ such that $2^n + 1 < 2ax_0 < 2(2^n + 1)$ and $0 \leq 2a(x_0 - 1) < 2^n + 1$. The first inequality is equivalent to that $0 < 2ax_0 - (2^n + 1) < 2^n + 1$ and $2ax_0 - (2^n + 1)$ is an odd integer. Hence the function $f_a(x) = a \odot x$ corresponding to $AX \bmod (2^n + 1)$ satisfies

$$f_a(2x_0) \bmod 2 = (2ax_0 - (2^n + 1)) \bmod 2 = 1.$$

On the other hand, the inequality $0 \leq 2a(x_0 - 1) < 2^n + 1$ implies that $2a(x_0 - 1)$ is an even integer in $\{0, 1, \dots, 2^n\}$ so that

$$f_a(2(x_0 - 1)) \bmod 2 = 2a(x_0 - 1) \bmod 2 = 0.$$

By the Lemma, $f_a(x)$ is not a polynomial over Z_{2^n} .

4 Design Principles for the Proposed Cipher

Confusion

Confusion (see [1,2]) means that the ciphertext depends on the plaintext and key in a complicated and involved way.

The confusion is achieved by mixing three different group operations. In the computational graph of the encryption process, the three different group operations are so arranged that the output of an operation of one type is **never** used as the input to an operation of the same type.

The three operations are incompatible in the sense that:

1. No pair of the 3 operations satisfies a distributive law. For example,

$$a \boxplus (b \odot c) \neq (a \boxplus b) \odot (a \boxplus c).$$

2. No pair of the 3 operations satisfies an associative law. For example,

$$a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c.$$

3. The 3 operations are combined by the mixing mapping m , which inhibits isotopisms as was shown in Theorem 1. Thus, using any bijections on the operands, it is impossible to realize any one of the three operations by another operation.

4. Under the mixing mapping, multiplication modulo $2^{16} + 1$, which is a bilinear function over $Z_{2^{16}+1}$, corresponds a non-polynomial function over $Z_{2^{16}}$; Under the inverse mixing mapping, addition modulo 2^{16} , which is an affine function in each argument over $Z_{2^{16}}$, corresponds a two variable polynomial of degree $2^{16} - 1$ in each variable over $Z_{2^{16}+1}$.

Diffusion

The *diffusion* requirement on a cipher is that each plaintext bit should influence every ciphertext bit and each key bit should influence every ciphertext bit(see [1,2]).

For the proposed cipher, a check by computer has shown that the diffusion requirement is satisfied after the first round, i.e., each output bit of the first round depends on every bit of the plaintext and on every bit of the key used for that round.

Diffusion is provided by the transformation called the multiplication-addition (MA) structure. The computational graph of the MA structure is shown in Fig.2. The MA structure transforms two 16 bit subblocks into two 16 bit subblocks controlled by two 16 bit key subblocks. This structure has the following properties:

- for any choice of the key subblocks Z_5 and Z_6 , $MA(\cdot, \cdot, Z_5, Z_6)$ is an invertible transformation; for any choice of U_1 and U_2 , $MA(U_1, U_2, \cdot, \cdot)$ is also an invertible transformation;
- this structure has a "complete diffusion" effect in the sense that each output subblock depends on every input subblock, and
- this structure uses the least number of operations (four) required to achieve such complete diffusion (see Appendix A for the proof).

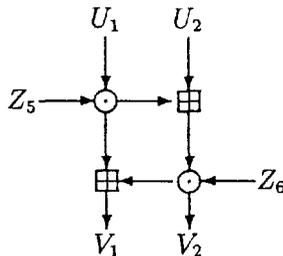


Figure 2: Computational graph of the MA structure

Similarity of encryption and decryption

The *similarity* of encryption and decryption means that decryption is essentially the same process as encryption, the only difference being that different key subblocks are used. This similarity results from

- using the output transformation in the encryption process so that the effect of (Z_1, Z_2, Z_3, Z_4) can be cancelled by using inverse key subblocks $(Z_1^{-1}, Z_2^{-1}, -Z_3, -Z_4)$ in the decryption process,

and from

- using an involution (i.e., a self-inverse function) with a 64 bit input and a 64 bit output controlled by a 32 bit key within the cipher. The involution used in the cipher is shown in Fig.3. The self-inverse property is a consequence of the fact that the exclusive-OR of (S_1, S_2) and (S_3, S_4) is equal to the exclusive-OR of (T_1, T_2) and (T_3, T_4) ; Thus, the input to the MA structure in Fig.3 is unchanged when S_1, S_2, S_3 and S_4 are replaced by T_1, T_2, T_3 and T_4 . Thus, if T_1, T_2, T_3 and T_4 are the inputs to the involution, the left half of the output is

$$\begin{aligned} & (T_3, T_4) \oplus MA((T_1, T_2) \oplus (T_3, T_4), Z_5, Z_6) \\ &= (S_1, S_2) \oplus MA((S_1, S_2) \oplus (S_3, S_4), Z_5, Z_6) \oplus MA((S_1, S_2) \oplus (S_3, S_4), Z_5, Z_6) \\ &= (S_1, S_2). \end{aligned}$$

Similarly, the right half of the output is (S_3, S_4) .

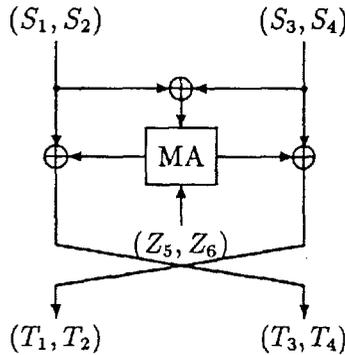


Figure 3: Computational graph of the involution

Perfect secrecy for a "one-time" key

Perfect secrecy in the sense of Shannon is obtained in each round of encryption if a "one-time" key is used. In fact, such perfect secrecy is achieved at the input transformation in the first round because each operation is a group operation. In addition,

for every choice of (p_1, p_2, p_3, p_4) and of (q_1, q_2, q_3, q_4) in F_2^{64} , there are exactly 2^{32} different choices of the key subblocks (Z_1, \dots, Z_6) such that the first round of the cipher transforms (p_1, p_2, p_3, p_4) into (q_1, q_2, q_3, q_4) .

5 Implementations of the Cipher

The cipher structure was chosen to facilitate both hardware and software implementations. In the encryption process, a regular modular structure was chosen so that the cipher can be easily implemented in hardware. A VLSI implementation of the cipher is being carried out at the Integrated Systems Laboratory of the ETH, Zürich. The estimated data rate of this chip varies from 45 Mbits per second to 115 Mbits per second, depending on the complexity of the architecture chosen.

The cipher can also be easily implemented in software because only operations on pairs of 16-bit subblocks are used in the encryption process.

The most difficult part in the implementation, multiplication modulo $(2^{16} + 1)$, can be implemented in the way suggested by the following lemma.

Lemma 2 *Let a, b be two n -bit non-zero integers in Z_{2^n+1} , then*

$$ab \bmod (2^n + 1) = \begin{cases} (ab \bmod 2^n) - (ab \operatorname{div} 2^n) & \text{if } (ab \bmod 2^n) \geq (ab \operatorname{div} 2^n) \\ (ab \bmod 2^n) - (ab \operatorname{div} 2^n) + 2^n + 1 & \text{if } (ab \bmod 2^n) < (ab \operatorname{div} 2^n) \end{cases}$$

where $(ab \operatorname{div} 2^n)$ denotes the quotient when ab is divided by 2^n .

Note that $(ab \bmod 2^n)$ corresponds to the n least significant bits of ab , and $(ab \operatorname{div} 2^n)$ is just the right-shift of ab by n bits. Note also that $(ab \bmod 2^n) = (ab \operatorname{div} 2^n)$ implies that $ab \bmod (2^n + 1) = 0$ and hence cannot occur when $2^n + 1$ is a prime.

Proof. For any non-zero a and b in Z_{2^n+1} , there exist unique integers q and r such that

$$ab = q(2^n + 1) + r, \quad 0 \leq r < 2^n + 1, \quad 0 \leq q < 2^n.$$

Moreover, $q + r < 2^{n+1}$. Note that $r = ab \bmod (2^n + 1)$. We have

$$(ab \operatorname{div} 2^n) = \begin{cases} q & \text{if } q + r < 2^n \\ q + 1 & \text{if } q + r \geq 2^n \end{cases}$$

and

$$(ab \bmod 2^n) = \begin{cases} q + r & \text{if } q + r < 2^n \\ q + r - 2^n & \text{if } q + r \geq 2^n. \end{cases}$$

Thus

$$r = \begin{cases} (ab \bmod 2^n) - (ab \operatorname{div} 2^n) & \text{if } q + r < 2^n \\ (ab \bmod 2^n) - (ab \operatorname{div} 2^n) + 2^n + 1 & \text{if } q + r \geq 2^n. \end{cases}$$

But $q + r < 2^n$ if and only if $(ab \bmod 2^n) \geq (ab \operatorname{div} 2^n)$. This proves the Lemma.

A C-program of the cipher together with some examples for checking the correctness of the implementation are given in Appendix B.

6 Conclusion

The cipher described above is proposed as a possible candidate for a new encryption standard. The cipher is based on the design concept of "mixing 3 different group operations" to achieve the required confusion and diffusion. Confusion is achieved by arranging the operations in a way that no pair of successive operations are of the same type and by the fact that operations of different types are incompatible. The structure of the cipher is so chosen that diffusion can be achieved using a small number of operations.

Enciphering and deciphering are essentially the same process, but with different key subblocks. Because of the use of 16-bit operations and a regular modular structure, the cipher can be implemented efficiently in both hardware and software. In particular, bit-level permutations are avoided in the encryption process because such permutations are difficult to implement in software.

In all of the statistical testings conducted up to now, we have not found any significant difference between the permutation of F_2^{64} determined by the cipher with a randomly chosen key and a permutation equiprobably chosen from all possible permutations of F_2^{64} .

The security of the proposed cipher needs further intensive investigation. The authors hereby invite interested parties to attack this proposed cipher and will be grateful to receive the results (positive or negative) of any such attacks.

References

- [1] C. E. Shannon, "Communication Theory of Secrecy Systems", B.S.T.J., Vol. 28, pp.656-715, Oct. 1949.
- [2] J. L. Massey, "An Introduction to Contemporary Cryptology", *Proc. IEEE*, Vol. 76, No. 5, pp. 533-549, May 1988.
- [3] J.Dénes, A.D.Keedwell, *Latin squares and their applications*, Akadémiai Kiadó, Budapest 1974.

Appendix A: Number of Operations Required for Diffusion

An *operation* is a mapping from two variables to one variable. A *computational graph* is a directed graph in which the vertices are operations, the edges entering a vertex are the inputs to the operation, the edges leaving a vertex are the outputs of the operation, the edges entering no vertex are the graph outputs, and the edges leaving no vertex are the graph inputs. An algorithm to compute a function determines a computational graph where the graph inputs are the inputs of the algorithm and the graph outputs are the outputs of the algorithm.

Consider a function having the form

$$(Y_1, Y_2) = E(X_1, X_2, Z_1, Z_2), \quad X_i, Y_i \in F_2^m, \quad X_i \in F_2^k \quad (3)$$

such that, for every choice of (Z_1, Z_2) , $E(\cdot, \cdot, Z_1, Z_2)$ is invertible. Such a function will be called a *cipher function*. A cipher function is said to have *complete diffusion* if each of its output variable depends non-idly on every input variable.

Theorem 3 *If a cipher function of the form (3) has complete diffusion, then the computational graph of any algorithm that computes the function contains at least 4 operations.*

Proof. Let $Y_1 = E_1(X_1, X_2, Z_1, Z_2)$, and $Y_2 = E_2(X_1, X_2, Z_1, Z_2)$. Then if E_1 has complete diffusion, it contains at least 3 operations because there are four input variables. Suppose E_1 contains 3 operations. The invertibility of the cipher function implies that $E_2 \neq E_1$ and complete diffusion requires that E_2 not equal any intermediate result that appears in E_1 . Thus, at least one operation not appearing in E_1 is required in E_2 . This proves the theorem.

It is easy to check that the MA structure shown in Fig.2, which has four operations, is a cipher function with complete diffusion, i.e., that each of the output variables depends non-idly on all four input variables.

Appendix B: C-program of the Cipher and Sample Data

```
/* C - program blockcipher */

# define maxim 65537
# define fuyi 65536
# define one 65535
# define round 8
void cip(unsigned IN[5], unsigned OUT[5], unsigned Z[7][10]);
void key( short unsigned uskey[9], unsigned Z[7][10] );
void de_key(unsigned Z[7][10], unsigned DK[7][10]);
unsigned inv(unsigned xin);
unsigned mul(unsigned a, unsigned b);
```

```

main()
{
    unsigned Z[7][10], DK[7][10], XX[5], TT[5], YY[5];
    short unsigned uskey[9];

    /*          to generate encryption key blocks */
    key(uskey,Z);
    /*          to generate decryption key blocks */
    de_key(Z,DK);
    /*          to encipher plaintext XX to ciphertext YY by key blocks Z */
    cip(XX,YY,Z);
    /*          to decipher ciphertext YY to TT by decryption key blocks DK */
    cip(YY,TT,DK);
}

/* encryption algorithm */
void cip(unsigned IN[5],unsigned OUT[5],unsigned Z[7][10])
{
    unsigned r, x1,x2,x3,x4,kk,t1,t2,a;
    x1=IN[1]; x2=IN[2]; x3= IN[3]; x4=IN[4];
    for (r= 1; r<= 8; r++) {
        x1 =mul(x1,Z[1][r]);
        x2 =mul(x2,Z[2][r]);
        x3 =( x3+ Z[3][r] ) & one;
        x4 =( x4 + Z[4][r] ) & one;
        kk = mul( Z[5][r], ( x1^x3 ) );
        t1 = mul( Z[6][r], ( kk+ ( x2^x4 )& one);
        t2 = ( Kk+ t1 ) & one;
        a = x1^t1;      x1 = x3^t1;      x3 = a;
        a = x2^t2;      x2 = x4^t2;      x4 = a;
    }
    OUT[1] = mul( x1,Z[1][round+1] );
    OUT[2] = mul( x2,Z[2][round+1] );
    OUT[3] = ( x3 + Z[3][round +1] )& one;
    OUT[4] = ( x4 + Z[4][round+1] ) & one;
}

/* the multiplication */
unsigned mul(unsigned a, unsigned b)
{
    long int p;
    long unsigned q;
    if (a==0) p = maxim-b;
    else if ( b==0 ) p = maxim-a; else
        { q=a*b; p=( q & one) - (q>>16); if (p<=0) p= p+maxim; }
    return (unsigned)(p & one);
}

/* compute multiplication inverse of integer xin
   by Euclidean gcd algorithm */
unsigned inv(unsigned xin)
{
    long n1,n2,q,r,b1,b2,t;
    if ( xin == 0 ) b2 = 0;
}

```

```

else {
    n1=maxim; n2 = xin; b2= 1; b1= 0;
    do { r = (n1 % n2); q = (n1-r)/n2 ;
        if (r== 0) {if ( b2<0 ) b2 = maxim+b2; }
        else { n1= n2; n2= r; t = b2; b2= b1- q*b2; b1= t; }
    } while (r != 0);
}
return (unsigned)b2;
}

/* generate key blocks Z[i][r] from user key uskey */
void key( short unsigned uskey[9], unsigned Z[7][10] )
{
    short unsigned S[55];
    int i,j,r;
    /* shifts */
    for (i = 1; i<9; i++) S[i-1] = uskey[i];
    for (i = 8; i< 55; i++ ) {
        if ( (i+2)%8 == 0 ) /* for S[14],S[22],... */
            S[i] = ( S[i-7]<<9 )^( S[i-14]>>7 );
        else if ( (i+1)%8 ==0 ) /* for S[15],S[23],... */
            S[i] = ( S[i-15]<<9 )^( S[i-14]>>7 );
        else /* for other S[i] */
            S[i] = ( S[i-7]<<9 )^( S[i-6]>>7 );
    }
    /* get endencryption key blocks */
    for (r= 1; r<=round+1; r++) for(j= 1;j<7; j++)
        Z[j][r] = S[6*(r-1) + j-1];
}

/* compute decryption key blocks DK[i][r]
   from encryption key blocks Z[i][r] */
void de_key(unsigned Z[7][10],unsigned DK[7][10])
{
    int j;
    for (j = 1; j<=round+1; j++) {
        DK[1][round-j+2] = inv(Z[1][j]);
        DK[2][round-j+2] = inv(Z[2][j]);
        DK[3][round-j+2] = ( fuyi-Z[3][j] ) & one;
        DK[4][round-j+2] = ( fuyi-Z[4][j] ) & one;
    }
    for (j= 1;j<=round+1;j++)
        { DK[5][round+1-j] = Z[5][j]; DK[6][round+1-j] = Z[6][j];}
}

```

Some sample data for checking the correctness of implementations are given below. The numbers are 16-bit integers in the decimal form.

```

the eight 16-bit user key subblocks:  uskey[i]
                                     1      2      3      4      5      6      7      8
encryption key subblocks Z[i][r]

```

	Z[1][r]	Z[2][r]	Z[3][r]	Z[4][r]	Z[5][r]	Z[6][r]
1-st round	1	2	3	4	5	6
2-nd round	7	8	1024	1536	2048	2560
3-rd round	3072	3584	4096	512	16	20
4-th round	24	28	32	4	8	12
5-th round	10240	12288	14336	16384	2048	4096
6-th round	6144	8192	112	128	16	32
7-th round	48	64	80	96	0	8192
8-th round	16384	24576	32768	40960	49152	57345
output transf	128	192	256	320	--	--

decryption key subblocks	DK[i][r]					
1-st round	65025	43350	65280	65216	49152	57345
2-nd round	65533	21843	32768	24576	0	8192
3-rd round	42326	64513	65456	65440	16	32
4-th round	21835	65529	65424	65408	2048	4096
5-th round	13101	43686	51200	49152	8	12
6-th round	19115	53834	65504	65532	16	20
7-th round	43670	28069	61440	65024	2048	2560
8-th round	18725	57345	64512	64000	5	6
output transf	1	32769	65533	65532	--	--

plaintext XX	0	1	2	3
after 1-st round	177	202	180	207
after 2-nd round	5054	10696	5085	10583
after 3-rd round	42790	64040	25583	15559
after 4-th round	16281	58571	61463	33861
after 5-th round	62321	51187	1399	59053
after 6-th round	37668	1126	6125	42057
after 7-th round	49700	61227	19644	21245
after 8-th round	2688	12695	2372	1339
YY =cip(XX,Z)	16379	12571	2628	1659

TT=cip(YY,DK) 0 1 2 3