Chapter 3

The Block Cipher IDEA

The block cipher IDEA (for International Data Encryption Algorithm) was first presented by us in [32]; its previous version PES (for Proposed Encryption Standard) was proposed in [31]. In both ciphers, the plaintext and the ciphertext are 64 bit blocks, while the secret key is 128 bits long. Both ciphers were based on the new design concept of "mixing operations from different algebraic groups". The required "confusion" was achieved by successively using three "incompatible" group operations on pairs of 16-bit subblocks and the cipher structure was chosen to provide the necessary "diffusion". The cipher structure was further chosen to facilitate both hardware and software implementations. The IDEA cipher is an improved version of PES and was developed to increase the security against differential cryptanalysis.

3.1 Description of IDEA

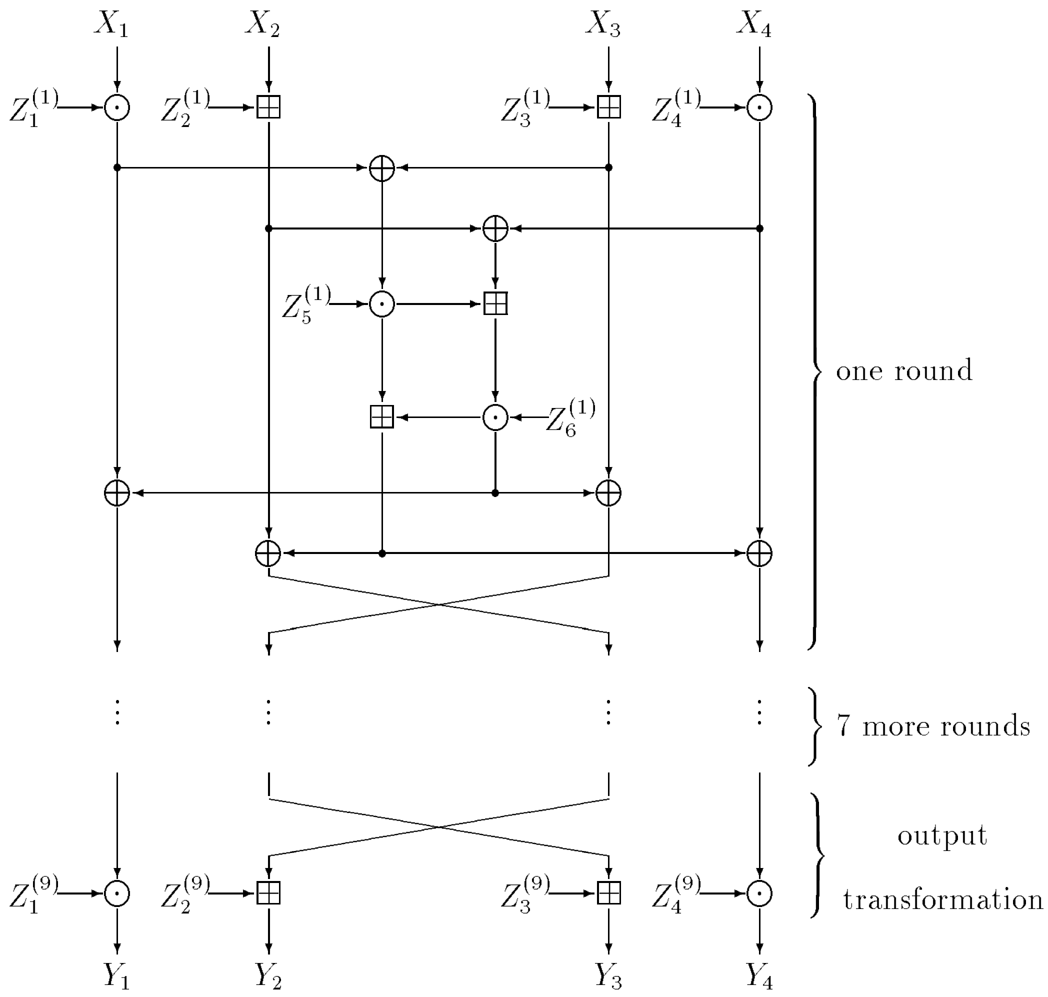
The cipher IDEA is an iterated cipher consisting of 8 rounds followed by an output transformation. The complete first round and the output transformation are depicted in the computational graph shown in Fig.3.1.

3.1.1 The encryption process

In the encryption process shown in Fig.3.1, 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 16-bit subblock is treated as the usual radix-two representation of an integer; the resulting operation is denoted as $\boxplus$;



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^{16}

Figure 3.1: Computational graph for the encryption process of the IDEA cipher.

– multiplication of integers modulo $2^{16}+1$ where the 16-bit subblock is treated as the usual radix-two representation of an integer except that the all-zero subblock is treated as representing 2^{16}; the resulting operation is denoted as $\odot$.

As an example of these group operations, note that

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

because

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

The 64-bit plaintext block X is partitioned into four 16-bit subblocks X_1, X_2, X_3, X_4, i.e., $X = (X_1, X_2, X_3, X_4)$. The four plaintext subblocks are then transformed into four 16-bit ciphertext subblocks Y_1, Y_2, Y_3, Y_4 [i.e., the ciphertext block is $Y = (Y_1, Y_2, Y_3, Y_4)$] under the control of 52 key subblocks of 16 bits that are formed from the 128-bit secret key in a manner to be described below. For $r = 1, 2, \ldots, 8$, the six key subblocks used in the r-th round will be denoted as $Z_1^{(r)}, .., Z_6^{(r)}$. Four 16-bit key subblocks are used in the output transformation; these subblocks will be denoted as $Z_1^{(9)}$, $Z_2^{(9)}$, $Z_3^{(9)}$, $Z_4^{(9)}$.

3.1.2 The decryption process

The computational graph of the decryption process is essentially the same as that of the encryption process (cf. Sec.3.4.1), the only change being that the decryption key subblocks $K_i^{(r)}$ are computed from the encryption key subblocks $Z_i^{(r)}$ as follows:

$$\begin{aligned}
(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) &= (Z_1^{(10-r)^{-1}}, -Z_3^{(10-r)}, -Z_2^{(10-r)}, Z_4^{(10-r)^{-1}}) && \text{for } r\text{=2,3,..,8;}\\
(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) &= (Z_1^{(10-r)^{-1}}, -Z_2^{(10-r)}, -Z_3^{(10-r)}, Z_4^{(10-r)^{-1}}) && \text{for } r\text{=1 and 9;}\\
(K_5^{(r)}, K_6^{(r)}) &= (Z_5^{(r)}, Z_6^{(r)}) && \text{for } r\text{=1,2,..,8;}
\end{aligned}$$

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., $-Z \boxplus Z = 0$.

The computation of decryption key subblocks from the encryption key subblocks is also shown in table 3.1.

3.1.3 The key schedule

The 52 key subblocks of 16 bits 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 the key subblocks is defined as follows: $Z_1^{(1)}, Z_2^{(1)}, .., Z_6^{(1)}$, $Z_1^{(2)}, .., Z_6^{(2)}, ..,$

Encryption key subblocks

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

Decryption key subblocks

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

Table 3.1: The encryption and decryption key subblocks.

$Z_1^{(8)},..,Z_6^{(8)}$, $Z_1^{(9)},Z_2^{(9)},Z_3^{(9)},Z_4^{(9)}$. 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 again cyclic shifted 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.

3.2 Group Operations and their Interaction

The IDEA cipher is based on the new design concept of mixing operations from different algebraic groups having the same number of elements. Group operations were chosen because the statistical relation of any three random variables U, V, W related by a group operation as $W = U * V$ has the "perfect secrecy" property that if any one of the three random variables is chosen independently of the others and equally likely to be any group element, then the other two random variables are statistically independent. The interaction of the different group operations contributes to the "confusion" required for a secure cipher, as will be explained in the following two sections.

The interaction of the different group operations will now be considered in terms of isotopism of quasigroups and in terms of polynomial expressions. To generalize the discussion beyond the case of 16-bit subblocks, 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 $\mathbb{Z}_{2^n}$ denote the ring of integers modulo 2^n. Let $(\mathbb{Z}^*_{2^n+1}, \cdot)$ denote the multiplicative group of the non-zero elements of the field $\mathbb{Z}_{2^n+1}$, let $(\mathbb{Z}_{2^n}, +)$ denote the additive group of the ring $\mathbb{Z}_{2^n}$, and let $(\mathbb{F}_2^n, \oplus)$ denote the group of n-tuples over $\mathbb{F}_2$ under the bitwise exclusive-or operation. Define the *direct* mapping d from $\mathbb{Z}^*_{2^n+1}$ onto $\mathbb{Z}_{2^n}$ as

$$d(i) = i \text{ for } i \neq 2^n \text{ and } d(2^n) = 0. \tag{3.1}$$

3.2.1 The three operations as quasigroup operations

Let S be a non-empty 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 all a and 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. The quasigroups $(S_1, *_1)$ and $(S_2, *_2)$ are said to be *isotopic* if there are bijective mappings $\theta, \phi, \psi : S_1 \rightarrow S_2$, such that,

$$\theta(x) *_2 \phi(y) = \psi(x *_1 y) \qquad \text{for all } x \text{ and } y \text{ in } S_1.$$

Such a triple (θ, ϕ, ψ) of bijections is called an *isotopism* of $(S_1, *_1)$ onto $(S_2, *_2)$. Two groups are said to be *isomorphic* if they are isotopic as quasigroups and the isotopism has the form (θ, θ, θ). It can be shown that two groups are isomorphic if and only if they are isotopic [18]. Note that every isomorphism between two groups is also an isotopism, but the converse is not true in general. In general for two isomorphic groups, there will be many more isotopisms between these groups than there will be isomorphisms. For this reason, we consider isotopisms rather than isomorphisms although our objects are all groups. The following theorem states some "incompatibility" properties of the three groups $(\mathbb{F}_2^n, \oplus)$, $(\mathbb{Z}_{2^n}, +)$ and $(\mathbb{Z}^*_{2^n+1}, \cdot)$ when $n \geq 2$.

Theorem 2 *For $n \in \{1, 2, 4, 8, 16\}$:*
1) The quasigroups $(\mathbb{F}_2^n, \oplus)$ and $(\mathbb{Z}_{2^n}, +)$ are not isotopic for $n \geq 2$.
2) The quasigroups $(\mathbb{F}_2^n, \oplus)$ and $(\mathbb{Z}^*_{2^n+1}, \cdot)$ are not isotopic for $n \geq 2$.
3) The triple (θ, ϕ, ψ) of bijections from $\mathbb{Z}^*_{2^n+1}$ to $\mathbb{Z}_{2^n}$ is an isotopism of $(\mathbb{Z}^*_{2^n+1}, \cdot)$ onto $(\mathbb{Z}_{2^n}, +)$ if and only if there exist c_1 and c_2 in $\mathbb{Z}_{2^n}$ and a generator α of the cyclic group $\mathbb{Z}^*_{2^n+1}$ such that, for all x in $\mathbb{Z}^*_{2^n+1}$,

$$\theta(x) - c_1 = \phi(x) - c_2 = \psi(x) - (c_1 + c_2) = \log_\alpha(x), \tag{3.2}$$

i.e., any isotopism between these groups is essentially the discrete logarithm. Moreover, when $n \geq 2$, none of the three bijections in an isotopism (θ, ϕ, ψ) from $\mathbb{Z}^*_{2^n+1}$ onto $\mathbb{Z}_{2^n}$ can be the direct mapping d defined in (3.1).

Proof.
1) For $n \geq 2$, the groups $(\mathbb{F}_2^n, \oplus)$ and $(\mathbb{Z}_{2^n}, +)$ are not isomorphic because $(\mathbb{Z}_{2^n}, +)$ is a cyclic group while $(\mathbb{F}_2^n, \oplus)$ is not. Thus, they are not isotopic as quasigroups.
2) $(\mathbb{Z}^*_{2^n+1}, \cdot)$ and $(\mathbb{Z}_{2^n}, +)$ are isomorphic groups for $n = 1, 2, 4, 8, 16$ because both groups are cyclic. Thus, $(\mathbb{Z}^*_{2^n+1}, \cdot)$ is isotopic to $(\mathbb{F}_2^n, \oplus)$ if and only if $(\mathbb{Z}_{2^n}, +)$ is isotopic to $(\mathbb{F}_2^n, \oplus)$, which is not the case for $n \geq 2$.
3) Suppose that (θ, ϕ, ψ) satisfies (3.2) for all x in $\mathbb{Z}^*_{2^n+1}$, then for every x and y in $\mathbb{Z}^*_{2^n+1}$,

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

Thus, (θ, ϕ, ψ) is indeed an isotopism.

Conversely, if (θ, ϕ, ψ) is an isotopism from $(\mathbb{Z}^*_{2^n+1}, \cdot)$ onto $(\mathbb{Z}_{2^n}, +)$, then for all x and y in $\mathbb{Z}^*_{2^n+1}$, $\theta(x) + \phi(y) = \psi(x \cdot y)$. 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 from $(\mathbb{Z}^*_{2^n+1}, \cdot)$ onto

$(\mathbb{Z}_{2^n}, +)$ as is easily checked. Moreover, $\theta_1(1) = \phi_1(1) = \psi_1(1) = 0$. In the isotopism equation

$$\theta_1(x) + \phi_1(y) = \psi_1(x \cdot y), \tag{3.3}$$

setting x to 1 results in $\phi_1(y) = \psi_1(y)$ for all y in $\mathbb{Z}^*_{2^n+1}$, and then setting y to 1 in (3.3) results in $\theta_1(x) = \psi_1(x)$ for all x in $\mathbb{Z}^*_{2^n+1}$. Thus, the three mappings θ_1, ϕ_1 and ψ_1 are identical. Equation (3.3) can thus be written as

$$\psi_1(x \cdot y) = \psi_1(x) + \psi_1(y). \tag{3.4}$$

Let α be the element of $\mathbb{Z}^*_{2^n+1}$ such that $\psi_1(\alpha) = 1$, then (3.4) implies that $\psi_1(\alpha^i) = i$ for $i = 1, 2, ..2^n - 1$ and $\psi_1(\alpha^{2^n}) = 0$. This implies that α is a generator of the cyclic group $(\mathbb{Z}^*_{2^n+1}, \cdot)$ and that $\psi_1(x) = \log_\alpha(x)$ for all x in $\mathbb{Z}^*_{2^n+1}$. Letting $c_1 = \theta(1)$ and $c_2 = \phi(1)$, we arrive at (3.2).

Finally, if (θ, ϕ, ψ) is an isotopism from $(\mathbb{Z}^*_{2^n+1}, \cdot)$ onto $(\mathbb{Z}_{2^n}, +)$ and one of the mappings θ, ϕ and ψ is the direct mapping d, then there exist c in $\mathbb{Z}_{2^n}$ and α in $\mathbb{Z}^*_{2^n+1}$ such that

$$d(x) = \log_\alpha(x) + c \text{ for all } x \text{ in } \mathbb{Z}^*_{2^n+1}. \tag{3.5}$$

But then $d(1) = 1$ implies that $1 = \log_\alpha(1) + c = c$ so that $d(x) = \log_\alpha(x) + 1$. Moreover, for $n \geq 2$, $d(2) = 2$, which implies that $2 = \log_a(2) + 1$ so that $\alpha = 2$. But then $d(2^n) = 0$ implies that $\log_2(2^n) + 1 = n + 1 = 0$ which is a contradiction because $n < 2^n - 1$ for $n \geq 2$. Thus, none of the mappings θ, ϕ and ψ can be the direct mapping d if $n \geq 2$. □

3.2.2 Polynomial expressions for multiplication and addition

In the encryption process of the cipher IDEA, multiplication modulo $2^n + 1$ and addition modulo 2^n are related via the direct mapping d and its inverse d^{-1}. More precisely, multiplication modulo $2^n + 1$ induces the function $g : \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n}$ defined by

$$g(x, y) = d[(d^{-1}(x) \cdot d^{-1}(y)) \bmod (2^n + 1)] \quad \text{for all } x \text{ and } y \text{ in } \mathbb{Z}_{2^n}. \tag{3.6}$$

Note that $g(x, y)$ is the function that we denoted as $x \odot y$ in Section 3.1.1. Similarly, addition modulo 2^n (the operation $\boxplus$) induces a function $f^* : \mathbb{Z}^*_{2^n+1} \times \mathbb{Z}^*_{2^n+1} \to \mathbb{Z}^*_{2^n+1}$ defined as

$$f^*(x, y) = d^{-1}[(d(x) + d(y)) \bmod 2^n] \quad \text{for all } x \text{ and } y \text{ in } \mathbb{Z}^*_{2^n+1}. \tag{3.7}$$

We can and do extend the function f^* to a function $f : \mathbb{Z}_{2^n+1} \times \mathbb{Z}_{2^n+1} \to \mathbb{Z}_{2^n+1}$ as follows:

$$f(x,y) = \begin{cases} d^{-1}[(d(x)+d(y)) \bmod 2^n] & \text{for all } x \text{ and } y \text{ in } \mathbb{Z}^*_{2^n+1} \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

For example, when $n = 1$, the function f induced by addition modulo 2 is

$$f(x,y) = 2xy \bmod 3 \quad \text{for all } x \text{ and } y \text{ in } \mathbb{Z}_3.$$

Similarly, the function g induced by multiplication modulo 3 is

$$g(x,y) = x + y + 1 \bmod 2 \quad \text{for all } x \text{ and } y \text{ in } \mathbb{Z}_2.$$

In what follows in this section, we show the "nonlinearity" of the function f over the field $\mathbb{Z}_{2^n+1}$ and the "nonlinearity" of the function g over the ring $\mathbb{Z}_{2^n}$ in terms of their polynomial expressions when $n \geq 2$.

Theorem 3 *For $n \in \{2, 4, 8, 16\}$:*
For every a in $\mathbb{Z}_{2^n+1} - \{0, 2^n\}$, the function $f(a,y)$ is a polynomial in y over the field $\mathbb{Z}_{2^n+1}$ with degree $2^n - 1$. Similarly, for every a in $\mathbb{Z}_{2^n+1} - \{0, 2^n\}$, the function $f(x,a)$ is a polynomial in x over $\mathbb{Z}_{2^n+1}$ with degree $2^n - 1$.

Example 3 *For $n = 2$, the function $f(x,y)$ over $\mathbb{Z}_5$ induced by addition modulo 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 of Theorem 3. For any finite field $\mathbb{F} = GF(q)$ and for every α in $\mathbb{F}^* = GF(q) - \{0\}$,

$$(-\alpha) \prod_{\beta \in \mathbb{F}^* - \{\alpha\}} (x - \beta) = \begin{cases} 1 & x = \alpha \text{ or } x = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.9}$$

as follows from the fact that, in any finite field, the product of all non-zero elements equals -1 so that

$$(-\alpha) \prod_{\beta \in \mathbb{F}^* - \{\alpha\}} (\alpha - \beta) = - \prod_{\beta \in \mathbb{F}^*} \beta = 1.$$

Thus, every function $h(\cdot)$ from $\mathbb{F}$ to $\mathbb{F}$ can be written as a polynomial over $\mathbb{F}$ of degree at most $q - 1$ as follows:

$$h(x) = \sum_{\alpha \in \mathbb{F}^*} h(\alpha)(-\alpha) \prod_{\beta \in \mathbb{F}^* - \{\alpha\}} (x - \beta) + (x^{q-1} - 1)[h(0) - \sum_{\alpha \in \mathbb{F}^*} h(\alpha)]. \tag{3.10}$$

Note that $f(0, y) = 0$ for all y in $\mathbb{F}$, and that $f(a, \cdot)$ for every $a \neq 0$ is a bijection from $\mathbb{F}^*$ to $\mathbb{F}^*$, so that

$$\sum_{\alpha \in \mathbb{F}^*} f(a, \alpha) = \sum_{\alpha \in \mathbb{F}^*} \alpha = 0 \quad \text{for every } a \neq 0 \text{ and for } \mathbb{F} \neq GF(2).$$

From the definition of $f(x, y)$ and from equation (3.10), the function $f(a, y)$ can be written for every $a \neq 0$ as

$$\begin{aligned} f(a, y) &= \begin{cases} a + y & 1 \leq y \leq 2^n - a \\ a + y + 1 & 2^n - a < y \leq 2^n \end{cases} \\ &= \sum_{i=1}^{2^n - a} (a+i)(-i) \prod_{\substack{j \neq i \\ 1 \leq j \leq 2^n}} (y - j) + \sum_{i=2^n - a + 1}^{2^n} (a+i+1)(-i) \prod_{\substack{j \neq i \\ 1 \leq j \leq 2^n}} (y - j) \\ &= \sum_{i=1}^{2^n} (a+i)(-i) \prod_{\substack{j \neq i \\ 1 \leq j \leq 2^n}} (y - j) + \sum_{i=2^n - a + 1}^{2^n} (-i) \prod_{\substack{j \neq i \\ 1 \leq j \leq 2^n}} (y - j). \end{aligned}$$

That is, the function $f(a, y)$ is a polynomial in y with degree at most $2^n - 1$. Moreover, the coefficient of $y^{2^n - 1}$ in $f(a, y)$ is

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

which is zero if and only if $a = 0$ or $a = -1 = 2^n$, which cases are excluded by hypothesis. (We have used the facts that $\sum_{i=1}^{2^n} i = 0 \bmod (2^n + 1)$, that $\sum_{i=1}^{2^n} i^2 = \frac{1}{6} 2^n (2^n + 1)(2 \times 2^n + 1)$, that $2 | 2^n$, and that $3 | 2 \times 2^n + 1$ for $n \in \{2, 4, 8, 16\}$ so that $\sum_{i=1}^{2^n} i^2 = 0 \bmod (2^n + 1)$.) Thus, we have shown that the degree of the polynomial $f(a, y)$ is indeed $2^n - 1$.

Note that $f(x, y) = f(y, x)$ for all x and y in $\mathbb{Z}_{2^n + 1}$ so that, for every $a \notin \{0, 2^n\}$, $f(x, a)$ is a polynomial in x of degree $2^n - 1$. □

Theorem 4 *If $n \in \{2, 4, 8, 16\}$, then, for every a in $\mathbb{Z}_{2^n} - \{0, 1\}$, the function $g(a, x) = a \odot x = x \odot a$* **cannot** *be written as a polynomial in x over the ring $\mathbb{Z}_{2^n}$.*

We show first the following lemma:

Lemma 1 *If $p(x)$ is a polynomial over $\mathbb{Z}_{2^n}$, then, for all β in $\mathbb{Z}_{2^n}$,*

$$p(2\beta) \bmod 2 = p(0) \bmod 2.$$

Proof. Let $p(x) = a_k x^k + a_{k-1} x^{k-1} + \cdots + a_1 x + a_0$.
Then for all β in $\mathbb{Z}_{2^n}$,

$$p(2\beta) = a_k(2\beta)^k + a_{k-1}(2\beta)^{k-1} + \cdots + a_1 2\beta + a_0.$$

Thus, $p(2x) \bmod 2 = a_0 \bmod 2 = f(0) \bmod 2$. □

Proof of Theorem 4 Let $n > 1$, then for every integer a, $1 < a < 2^n$, there exists an integer $x_0 \in \{1, 2, \ldots, 2^n\}$ such that the following three inequalities all satisfied:

$$2^n + 1 < 2ax_0 < 2(2^n + 1), \tag{3.11}$$

$$0 \leq 2a(x_0 - 1) < 2^n + 1 \tag{3.12}$$

and

$$0 \leq 2x_0 \leq 2^n. \tag{3.13}$$

Inequality (3.11) is equivalent to the inequality $0 < 2ax_0 - (2^n + 1) < 2^n + 1$ with the condition that $2ax_0 - (2^n + 1)$ is an odd integer. Because of (3.13) and from the definition of the function g,

$$g(a, 2x_0) = a \odot (2x_0) = 2ax_0 - (2^n + 1).$$

Thus,

$$g(a, 2x_0) \bmod 2 = (2ax_0 - (2^n + 1)) \bmod 2 = 1.$$

On the other hand, inequality (3.12) implies that $2a(x_0 - 1)$ is an even integer in $\{0, 1, .., 2^n\}$ so that

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

Hence, it follows from Lemma 1 that $g(a, x) = a \odot x$ is not a polynomial over $\mathbb{Z}_{2^n}$.
□

3.3 Security Features of IDEA

In this section, we state some provable security features of the IDEA cipher. The security of the IDEA cipher against differential cryptanalysis will be discussed in detail in Chapter 5.

3.3.1 Confusion

The confusion (see page 12) required for a secure cipher is achieved in the IDEA cipher by mixing three incompatible group operations. In the computational graph of the encryption process for IDEA, 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 instance, for the operations $\odot$ and $\boxplus$, there exist a, b, and c in $\mathbb{F}_2^{16}$, such that,

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

For example, when $a = b = c = 1 = (0, 0, .., 0, 1)$, the left side of the above inequality is $2 = (0, 0, .., 0, 1, 0)$, while the right side equals $4 = (0, 0, .., 0, 1, 0, 0)$.

2. No pair of the 3 operations satisfies a "generalized associative" law. For instance, for the operations $\boxplus$ and $\oplus$, there exist a, b, and c in $\mathbb{F}_2^{16}$, such that,

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

For example, for $a = b = c = 1 = (0, 0, ...0, 1)$ in $\mathbb{F}_2^{16}$, the left side of the above inequality is $1 = (0, 0, ...0, 1)$, while the right side equals $3 = (0, 0, .., 0, 1, 1)$. Thus, one cannot arbitrarily change the order of operations to simplify analysis.

3. The 3 operations are connected by the direct mapping d and its inverse, which inhibits isotopisms as was shown in Theorem 2. The cryptographic significance of this fact is that, if there were an isotopism between two operations, then one could replace one operation with the other by applying bijective mappings on the inputs and on the output. It follows from Theorem 2 that $(\mathbb{F}_2^{16}, \odot)$ and $(\mathbb{F}_2^{16}, \oplus)$ are not isotopic and that $(\mathbb{F}_2^{16}, \boxplus)$ and $(\mathbb{F}_2^{16}, \oplus)$ are not isotopic. The isotopism from $(\mathbb{F}_2^{16}, \odot)$ onto $(\mathbb{F}_2^{16}, \boxplus)$ is essentially the discrete logarithm, which , as shown in Theorem 2, cannot be the direct mapping d. Moreover, the discrete logarithm is generally considered to be a "complex" function.

4. Under the direct mapping d and its inverse d^{-1}, it is possible to consider the operations $\odot$ and $\boxplus$ as acting on the same set (either in the ring $\mathbb{Z}_{2^n}$ or in the field $\mathbb{Z}_{2^n+1}$). However, by doing so, we must analyze some highly non-linear functions in the sense that multiplication modulo $2^{16}+1$, which is a bilinear function over $\mathbb{Z}_{2^{16}+1}$, corresponds to a non-polynomial function over $\mathbb{Z}_{2^{16}}$, as was shown in Theorem 4. Similarly, addition modulo 2^{16}, which is an affine function in each argument over $\mathbb{Z}_{2^{16}}$, corresponds to a two variable polynomial of degree $2^{16}-1$ in each variable over $\mathbb{Z}_{2^{16}+1}$, as was shown in Theorem 3. [Note that every function h from $\mathbb{Z}_{2^{16}+1}$ to $\mathbb{Z}_{2^{16}+1}$ is a polynomial of degree at most 2^{16}. Moreover, if such a function is invertible then its degree is at most $2^{16}-1$ as follows from (3.10) and from the facts that function $h(x)$ is invertible if and only if function $h(x)-h(0)$ is invertible and that these two functions have the same degree].

3.3.2 Diffusion

A check by direct computation has shown that the round function is "complete", i.e., that 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. This diffusion is provided in the IDEA cipher by the transformation called the multiplication-addition (MA) structure whose computational graph is shown in Fig.3.2. 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:

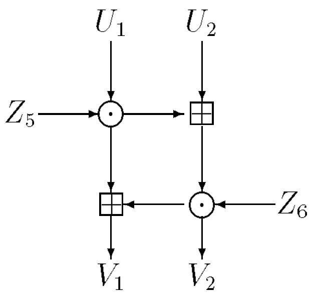


Figure 3.2: Computational graph of the MA structure.

– 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. [To give a formal proof of this property, we need the following definitions.

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

Consider a function having the form

$$(Y_1, Y_2) = E(X_1, X_2, Z_1, Z_2), \qquad X_i, Y_i \in \mathbb{F}_2^n, \quad Z_i \in \mathbb{F}_2^k \tag{3.14}$$

and 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 *2-block cipher.* A 2-block cipher will be said to have *complete diffusion* if each of its output variable depends non-idly on every input variable.

Lemma 2 *If a 2-block cipher of the form (3.14) has complete diffusion, then the computational graph determined by any algorithm that computes the cipher function contains at least 4 operations.*

Proof. Let $Y_1 = E_1(X_1, X_2, Z_1, Z_2)$, and $Y_2 = E_2(X_1, X_2, Z_1, Z_2)$. Because E_1 has complete diffusion, its computational graph contains at least 3 operations because this function has four input variables. Suppose E_1 contains exactly 3 operations. The invertibility of the 2-block cipher 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 the computational graph of E_2. This proves the lemma.]

3.3.3 Perfect secrecy for a "one-time" key

Perfect secrecy (see page 7) in the sense of Shannon is obtained in each round of encryption if a "one-time" key (see page 7) 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 $\mathbb{F}_2^{64}$, there are exactly 2^{32} different choices of the key subblocks $(Z_1, .., Z_6)$ 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).

3.4 Implementations of the Cipher

The cipher IDEA can be easily implemented in software because only basic operations on pairs of 16-bit subblocks are used in the encryption process. A C-language program implementing the cipher and some sample data for checking the correctness of implementation are given in Section 3.4.3. This C-program can achieve data-rates from about 200 Kbits per second on an IBM-PC to about 3.2 Mbits per second on a VAX-9000.

The regular modular structure of the cipher facilitates hardware implementations. The similarity of encryption and decryption for the IDEA cipher, shown in next section, makes it possible to use the same device in both encryption and decryption. An algorithm for computing the operation $\odot$ is described in Section 3.4.2.

3.4.1 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. Thus, the same device can be used for both encryption and decryption, the only "extra" cost being the pre-computation of the key subblocks from the 128-bit secret key. In the following we show that the round function of the IDEA cipher has the form (2.2) on page 17, that is, the round function consists of a group cipher followed by an involution cipher plus an involutary permutation which is an automorphism of the group $(\mathbb{F}_2^{64}, \otimes)$. Then it follows from Theorem 1 (see page 17) that IDEA cipher has similarity of encryption and decryption.

For the encryption process of the IDEA cipher shown in Fig.3.1, define

$$X \otimes Z_A = (X_1 \odot Z_1, X_2 \boxplus Z_2, X_3 \boxplus Z_3, X_4 \odot Z_4),$$

then it is easy to see that $(\mathbb{F}_2^{64}, \otimes)$ is a group.

Let $P_I(X)$ be the permutation on X that interchanges the subblocks X_2 and X_3 of $X = (X_1, X_2, X_3, X_4)$ at the end of each round. It is obvious that P_I is an involution and that $P_I(X \otimes Z_A) = P_I(X) \otimes P_I(Z_A)$, so that P_I is an automorphism of the group $(\mathbb{F}_2^{64}, \otimes)$.

It remains to show that the function $In(\cdot, Z_B)$, shown in Fig.3.3, with the 64-bit input (S_1, S_2, S_3, S_4) and the 64-bit output (T_1, T_2, T_3, T_4) controlled by the 32-bit key $Z_B = (Z_5, Z_6)$, is an involution. That is, for any fixed Z_B, the inverse of the function $In(\cdot, Z_B)$ is itself. This 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. 3.2 is unchanged when S_1, S_2, S_3 and S_4 are replaced by T_1, T_2, T_3 and T_4.

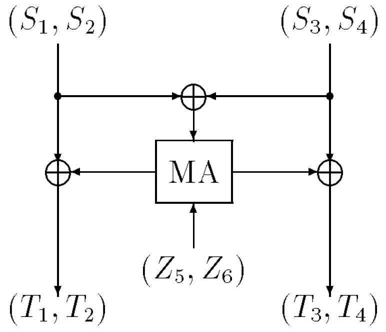


Figure 3.3: Computational graph of the involution $In(\cdot, Z_B)$.

3.4.2 Low-High algorithm for multiplication

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

Lemma 3 [Low-High algorithm for $\odot$] *Let a, b be two n-bit non-zero integers in $\mathbb{Z}_{2^n+1}$, 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 $\mathbb{Z}_{2^n+1}$, there exist unique integers q and r such that

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

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

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

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

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

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

Remark. There are of course other ways to compute the operation $\odot$. For example, based on the fact that

$$x \cdot y = \alpha^{(\log_\alpha(x) + \log_\alpha(y) \bmod 2^n)} \bmod (2^n + 1)$$

for all x and y in $\mathbb{Z}^*_{2^n+1}$ where α is a generator of the cyclic group $\mathbb{Z}^*_{2^n+1}$, one can compute $\odot$ by using $\boxplus$ together with look-up tables for computing $\log_\alpha(\cdot)$ and $\alpha^{(\cdot)}$. For small n, i.e., for $n = 2, 4$ or 8, this is more efficient than the Low-High algorithm. However, for $n = 16$, this method requires more memory. More details can be found in [10].

3.4.3 C-program of IDEA cipher and sample data

```
/* C - program  of block cipher IDEA */

#include <stdio.h>
# 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()
{
  int       i, j, k, x;
  unsigned  Z[7][10], DK[7][10], XX[5],TT[5], YY[5];
  short unsigned uskey[9];
   for( i=1; i<=8; i++ ) uskey[i]= i;
   key(uskey,Z);                    /* generate encryption subkeys Z[i][r] */
```

```
   printf("\n encryption keys    Z1     Z2     Z3     Z4     Z5     Z6");
   for( j=1; j<=9; j++ ) {    printf("\n %3d-th round  ", j);
         if (j==9)  for( i=1; i<=4; i++ )     printf(" %6d",Z[i][j]);
             else   for( i=1; i<=6; i++ )     printf(" %6d",Z[i][j]);
     }
   de_key(Z,DK);                    /* compute decryption subkeys DK[i][r] */

   printf("\n \n decryption keys   DK1    DK2    DK3    DK4    DK5    DK6 ");
   for( j=1; j<=9; j++ ) {    printf("\n %3d-th round  ", j);
         if (j==9)  for( i=1; i<=4; i++ )     printf(" %6d",DK[i][j]);
             else   for( i=1; i<=6; i++ )     printf(" %6d",DK[i][j]);
        }
        for (x=1; x<=4; x++) XX[x]=x-1;
        printf("\n \n plaintext X  %6u  %6u  %6u  %6u  \n",
                                XX[1], XX[2], XX[3], XX[4]);

        cip(XX,YY,Z);              /* encipher XX to YY with key Z   */

        printf("\n \n ciphertext Y  %6u  %6u  %6u  %6u  \n",
                                YY[1], YY[2], YY[3], YY[4]);

        cip(YY,TT,DK);             /* decipher YY to TT with key DK */

        printf("\n \n result     T  %6u  %6u %6u  %6u  \n",
                                TT[1], TT[2], TT[3], TT[4]);
   }

    /* encryption algorithm  */
void  cip(unsigned IN[5],unsigned OUT[5],unsigned Z[7][10])
 {
    unsigned int 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++)                      /* the round function     */
      {
             /* the group operation on 64-bits block */
        x1 =mul(x1,Z[1][r]);        x4 =mul(x4,Z[4][r]);
        x2 =( x2 + Z[2][r] ) & one;   x3 =( x3 + Z[3][r] ) & one;
             /* the function of the MA structure */
        kk = mul( Z[5][r], ( x1^x3 ) );
        t1 = mul( Z[6][r], ( kk + ( x2^x4 ) )& one);
        t2 = ( kk + t1 ) & one;
             /*  the involutary permutation PI  */
        x1 = x1^t1;       x4 = x4^t2;
        a = x2^t2;       x2 = x3^t1;      x3 = a;
        printf("\n     %1u-th rnd  %6u  %6u  %6u  %6u  ", r, x1, x2, x3, x4);
      }
             /*   the output transformation      */
    OUT[1] = mul( x1,Z[1][round+1] );
    OUT[4] = mul( x4,Z[4][round+1] );
    OUT[2] =  ( x3 + Z[2][round +1] )& one;
    OUT[3] =  ( x2 + Z[3][round+1] ) & one;
 }
```

```
   /*  multiplication using the Low-High algorithm  */
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=(unsigned long)a*(unsigned long)b;
         p=( q & one) - (q>>16);  if (p<=0) p= p+maxim;
       }
  return (unsigned)(p & one);
}


         /*  compute inverse of xin by Euclidean gcd alg. */
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 encryption  subkeys Z's   */
void   key( short unsigned uskey[9], unsigned  Z[7][10] )
{
  short unsigned S[54];
  int  i,j,r;
 for (i = 1; i<9; i++)  S[i-1] = uskey[i];
   /*   shifts   */
 for (i = 8; i< 54; i++ )
  {
     if ( (i+2)%8 == 0 )                      /* for S[14],S[22],..  */
        S[i] = ( ( S[i-7] <<9 )^( S[i-14] >>7 ) ) & one;
     else  if ( (i+1)%8 ==0 )                 /* for S[15],S[23],..   */
        S[i] =( ( S[i-15] <<9 )^( S[i-14] >>7 ) ) & one ;
     else
        S[i] = ( ( S[i-7] <<9 )^( S[i-6] >>7 )) & one;
   }
    /* get subkeys     */
      for (r= 1; r<=round+1; r++)   for(j= 1;j<7; j++)
                Z[j][r] = S[6*(r-1) + j-1];
 }
```

```
     /* compute decryption subkeys DK's  */
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[4][round-j+2] = inv(Z[4][j]);
     if ( j==1 || j==round+1 ){
        DK[2][round-j+2] = ( fuyi-Z[2][j] ) & one;
        DK[3][round-j+2] = ( fuyi-Z[3][j] ) & one;
        }
     else {
        DK[2][round-j+2] = ( fuyi-Z[3][j] ) & one;
        DK[3][round-j+2] = ( fuyi-Z[2][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];}
 }
```

Sample Data. All the numbers are 16-bit integers with the leftmost bit being the most significant bit.

encryption keys	Z1	Z2	Z3	Z4	Z5	Z6
1-th round	1	2	3	4	5	6
2-th round	7	8	1024	1536	2048	2560
3-th 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
9-th round	128	192	256	320		

decryption keys	DK1	DK2	DK3	DK4	DK5	DK6
1-th round	65025	65344	65280	26010	49152	57345
2-th round	65533	32768	40960	52428	0	8192
3-th round	42326	65456	65472	21163	16	32
4-th round	21835	65424	57344	65025	2048	4096
5-th round	13101	51200	53248	65533	8	12
6-th round	19115	65504	65508	49153	16	20
7-th round	43670	61440	61952	65409	2048	2560
8-th round	18725	64512	65528	21803	5	6
9-th round	1	65534	65533	49153		

plaintext X	0	1	2	3
after				
1-th rnd	240	245	266	261
2-th rnd	8751	8629	62558	59737
3-th rnd	3974	14782	36584	4467
4-th rnd	22495	44120	50779	47693
5-th rnd	36481	47772	63359	14922
6-th rnd	26946	37897	57883	7268
7-th rnd	39376	51190	21297	25102
8-th rnd	2596	152	60523	18725
ciphertext Y	4603	60715	408	28133
after				
1-th rnd	55693	54065	10230	33464
2-th rnd	48205	57963	37961	42358
3-th rnd	2724	63471	55964	9443
4-th rnd	51782	65115	56408	4461
5-th rnd	29839	36616	14810	17868
6-th rnd	12902	1118	12213	45102
7-th rnd	1680	1290	253	7674
8-th rnd	0	5	3	12
result T	0	1	2	3