Chapter 2

CIPHERUNICORN-E

CIPHERUNICORN-E was designed by NEC Corporation in 2000.

2.1 Design Properties

2.1.1 Basic Properties

The design includes logical operations (bitwise XOR, AND, shifts, ...), arithmetic additions on 32-bit words, and table lookups. There are four 8-bit to 8-bit substitution boxes defined by tables, which leads to 1KB of table.

Several errors were found in the design description:

- Figure 3.7 have $\oplus$ operations with a single input. They should be removed.
- Figure 3.11 represent several outputs at a place which is not consistent with Figure 3.8 and Figure 3.10. It should be corrected.

2.1.2 Upper Level

Encryption is a cascade of L operations and Feistel schemes with round functions F. The L operation is linear. Depending on the subkey bits in LK, L takes each of the bits of the two 32-bit halves bitwise and perform one of the four following operations.

1. nothing,
2. XOR of the left bit to the right bit,
3. XOR of the right bit to the left bit,

4. swap of the two bits.

We notice that XORing left bits onto the right ones can be included in the F function of the next Feistel round. Similarly, XORing the right bit onto the left one can be included in the F function of the previous Feistel round. Thus, all the first three operations do not alter the Feistel scheme.

Swapping the bits can however be viewed as a merge of the previous and the next Feistel rounds. Typically, if $\mathrm{LK}^{\{i\}} = \texttt{0xffffffffffffffff}$, the round $2i-1$ and $2i$ can be merged with a single round function which is the XOR of $F^{\{2i-1\}}$ and $F^{\{2i\}}$. Additionally, if $\mathrm{SK}^{\{2i-1\}} = \mathrm{SK}^{\{2i\}}$ and $\mathrm{FK}^{\{2i-1\}} = \mathrm{FK}^{\{2i\}}$, this XOR is zero, so the two rounds vanish and we merge the round $2i-2$ and $2i+1$. Therefore this operation decreases the number of effective Feistel rounds.

2.1.3 The F Function

The round function is a tricky cascade of T functions with additional XOR with data-dependent temporary keys and a structure permuted according to the temporary key. The T functions are of four types: T_0, T_1, T_2, T_3. They perform XOR on all four bytes with a value taken from the tables which depends on a single byte. It should be noticed that two consecutive T_i operations may cancel each other if the ith input byte is the same. Strictly speaking, two consecutive T_i operations cannot cancel because the ith byte is modified by the first T_i operation through a function which has purposely no fixed points. This however occurs if the ith byte is additionally modified by an extra operation in between the two T_i operations, like a K_i operation which simply XOR a temporary key onto the ith byte.

The cascade of T_i functions provides a good diffusion of byte information.

The temporary key generation mechanism is another tricky cascade of T_i functions with an additional $Y_{i,j,k}$ function. This $Y_{i,j,k}$ function can be considered like a

$$x \to x.(1+2^i).(1+2^j).(1+2^k) \bmod 2^{32}$$

function. This propagates lower order bytes though higher order bytes. On the other hand, T_0 propagates the higher order byte onto all other bytes. Finally, only the 4 higher order bits and the 16 lower level bits are used in order to define the temporary key.

2.1.4 Relationship between Substitution Tables

The four substitution tables have been designed in order to have a maximal resistance against linear and differential cryptanalysis. Their design core is

an inverse function over $GF(2^8)$ in combination with an affine transformation. The four different substitution tables are thus generated by the inverse function in four different representations of $GF(2^8)$; each one of these representations is defined by an irreducible polynomial. More formally, each S-box S_i is defined by the formula

$$S_i(x) = \sigma_i \left((x + c_i)^{-1} \bmod g_i \right) + d_i$$

where c_i's and d_i's are 8-bit constants not equal to zero, g_i are irreducible polynomials of degree 8 over $GF(2)$ and σ_i are matrices defining a linear application over $GF(2)^8$.

We noticed that the output of two S-boxes can be linearly related in some cases. More precisely, it is well known that two different representations of $GF(2^8)$ are isomorphic.

Let's take as example two different representations of $GF(2^3)$:

$$\begin{aligned} \mathbb{F} &= GF(2)[x]/(x^3 + x + 1) \\ \mathbb{F}' &= GF(2)[x]/(x^3 + x^2 + 1). \end{aligned}$$

Let $\alpha = x$ and $\gamma = x$ be generators of the multiplicative group of $\mathbb{F}$ and $\mathbb{F}'$, respectively. Then α and $\beta = \gamma^3$ have both $x^3 + x + 1$ as minimal polynomial (the minimal polynomial of an element $x \in GF(p)$ is the lowest degree monic polynomial $M(x)$ with coefficient over $GF(p)$ such that $M(x) = 0$); this defines the isomorphism between the two representations: we map $a + b\alpha + c\alpha^2$ represented as (c, b, a) onto $a + b\beta + c\beta^2$ represented with $1 = (0, 0, 1)$, $\gamma = (0, 1, 0)$ and $\gamma^2 = (1, 0, 0)$. In our example, the matrix of the linear application δ is given by

$$\delta = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

A similar matrix can be found for each combination of two S-boxes. Let us consider

$$\begin{aligned} \mathrm{matrixA}_1^{-1}(S_1(x \oplus c_1) \oplus d_1) &= x^{-1} \bmod g_1 \\ \mathrm{matrixA}_2^{-1}(S_2(y \oplus c_2) \oplus d_2) &= y^{-1} \bmod g_2 \end{aligned}$$

We know that the modulo g_1 and modulo g_2 representations of $\mathrm{GF}(2^8)$ are isomorphic. If γ is an element of $\mathrm{GF}(2)[x]/g_2$ of minimal polynomial g_1 (namely, a primitive root of g_1), then we can define the matrix matrixP which maps x^i onto γ^i for $i = 0, \ldots, 7$. This is a field isomorphism, thus

$$(\mathrm{matrixP}(x))^{-1} \bmod g_2 = \mathrm{matrixP}(x^{-1} \bmod g_1).$$

By taking $y = \text{matrixP}(x)$, we obtain that

$$\begin{aligned}&\text{matrixA}_2^{-1}(S_2(\text{matrixP}(x) \oplus c_2) \oplus d_2) = \\ &\text{matrixP} \times \text{matrixA}_1^{-1}(S_1(x \oplus c_1) \oplus d_1)\end{aligned}$$

which summarizes as

$$S_2(z) = v \oplus \text{matrixA}_2 \times \text{matrixP} \times \text{matrixA}_1^{-1}\left(S_1(\text{matrixP}^{-1}(z \oplus u))\right)$$

with

$$\begin{aligned}u &= c_2 \oplus \text{matrixP}(c_1) \\ v &= d_2 \oplus \text{matrixA}_2 \times \text{matrixP} \times \text{matrixA}_1^{-1}(d_1).\end{aligned}$$

Finally, we can write

$$S_i(z) = v_i \oplus \text{matrixQ}_i\left(S_1(\text{matrixR}_i(z \oplus u_i))\right)$$

for $i = 1, 2, 3$. This means that all S_i are affine transformations of each other. It is however not clear if this fact can be exploited in an effective attack against the cipher.

2.1.5 Substitution Tables

Substitution tables have other surprising properties. Namely, for any i, there exist two affine transformation f and g such that $S_i \circ f = g \circ S_i$.

For instance, let us take $i = 0$. We let

$$f(x) = (x^2 \bmod g_0) + c_0 + (c_0^2 \bmod g_0)$$

which is an affine transformation. We have

$$\begin{aligned}S_0 \circ f(x) &= \text{matrixA}_0((x + c_0)^{-2} \bmod g_0) + d_0 \\ &= \text{matrixL} \times (S_0(x) + d_0) + d_0\end{aligned}$$

with

$$\text{matrixL} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

so

$$g(x) = \text{matrixL}(x + d) + d.$$

2.1.6 The K Function

The K function XOR a byte with wk1 or wk2. We notice that if one of these bytes is zero, the corresponding K function does nothing.

We also notice that the K(sh[wk0][1]) function commutes with the four previous T functions, and the K(sh[wk0][0]) function commutes with the three previous T functions.

2.1.7 Key Scheduler

The key scheduler is a Feistel scheme in which the round function sequence has a period of four. It generates alternately an LK-type subkey, an SK-type subkey, and an FK-type subkey. Due to the periodic structure of the key scheduler, there is no chance that the generated subkeys have a period of three, six or nine. We can still expect to have a period of twelve, which would mean that the LK subkeys sequence would have a period of four, as well as the SK subkeys sequence and the FK subkeys sequence. This would give a square structure to the encryption: we could write it $\sigma \circ \sigma$. It can be furthermore written with

$$\sigma = L^4 \circ \pi^2 \circ L^3 \circ \pi^1 \circ L^2 \circ \pi^2 \circ L^1 \circ \pi^1$$

where π^1 and π^2 are two 2-round Feistel schemes and the L^i's are four L functions with four different LK subkeys.

2.2 Differential and Linear Cryptanalysis

The S_i substitution boxes make it hard to get good differential and linear cryptanalysis since inversion in Galois fields is well known to be highly non-linear. Diffusion of byte information in Feistel round functions is also quite good. We can still hope to be able to use the mentioned design properties of F (like cancellation of consecutive T_i functions) and of the overall structure (like cancellation of consecutive rounds). This will however occur with low probability, and the number of rounds (16 minus the number of canceled rounds) is still large enough to protect against these attacks.

We have not been able to find good differential or linear attacks. We still think that internal properties of the design may still hide unexpected weaknesses.

2.3 Other Attacks

2.3.1 Side Channel Attacks

Like most of block ciphers, CIPHERUNICORN-E is vulnerable against simple models of power analysis. Assuming we can get the Hamming weight of all CPU registers throughout the computation, we can recover the whole secret key with a few chosen plaintexts. This is however a powerful power analysis model.

The T (and K) functions which depend on temporary keys may induce additional weaknesses against power analysis of timing attacks, depending on how it is implemented. For instance, T may require to use data dependent rotations, which are known to introduce important weaknesses against timing attacks. More generally, if the implementation makes the ordering of the CPU instructions change, we may have important weaknesses against power analysis since power analysis will be able to see the actual sequence of instructions which is performed. Tricky implementations may be able to thwart these problems, but any careless implementation may be highly vulnerable.

The substitution tables induce weaknesses against differential fault analysis: if we speed up the clock signal, part of the memory will become out of reach. If the unreachable memory contains the substitution tables, analyzing the differences may lead to information on internal computations.

CIPHERUNICORN-E may thus be more vulnerable against side channel attacks than usual because of the internal structure (T, S).

2.3.2 Weak Keys

If we try to find keys for which the key scheduler has a period of four (which is the "natural period" of the scheduler), we obtain that the same 64-bit string must go though two 2-round Feistel schemes with T_0 and T_1, and with T_2 and T_3, so that, if x denotes its 32 rightmost bits, we have

$$T_0(x) = T_2(x).$$

This ends up with the equations

$$\begin{aligned} S_0^{-1} \circ S_2 \circ S_1^{-1} \circ S_3(x_0) &= x_0 \\ S_1^{-1} \circ S_3 \circ S_1^{-1} \circ S_3(x_0) &= x_0 \\ S_2^{-1} \circ S_0 \circ S_1^{-1} \circ S_3(x_0) &= x_0. \end{aligned}$$

This system has no solution, thus it is not possible to find keys with such a property.

Even if this was possible, it is not quite clear how weak the generated subkeys would be since the period of four conflicts with the period of the subkey type sequence which is three. We can also wonder about keys for which the period is twelve. We expect to get about one keys with this property. For this, as already mentioned, the encryption is a square permutation. It may be possible to distinguish square permutations from other ones, which can be considered as a weakness.

There may thus exist about one weak key, but it is not clear how weak it is, nor how to identify it.

2.3.3 Decorrelation

The F function provides too much entropy. Actually, it is likely that we cannot efficiently distinguish f from a random function with only two chosen inputs. With four chosen inputs, we obtain enough information in order to reconstruct the round subkey. We estimated the decorrelation of order four to be at least $2 \times e^{-1} \approx 0.74$. Although it is hard to estimate the actual decorrelation, it is likely to be quite small, and well known techniques may prove that no efficient characteristic can be found against nine rounds.

2.4 Available Literature

The only available literature about CIPHERUNICORN-E is to our knowledge the specification document provided by NEC and the given IECE references [19, 20]. We haven't found any independent cryptanalysis or third party comments on CIPHERUNICORN-E in the academic literature.

2.5 Conclusion

CIPHERUNICORN-E is a recent block cipher with conservative design. Although the original documents still have errors, it looks like a strong cipher since no attacks has been found so far. Publication of CIPHERUNICORN-E is however restricted to domestic area, and no public analysis were done so far.

We also outlined a few internal properties which may ultimately lead to some attacks which are unknown so far: cancellation of internal rounds, cancellation of T functions, affine transformations between substitution boxes, symmetries in the key scheduler. Careless implementation may also be more vulnerable than usual against side channel attacks.

Diffusion and confusion are however quite good, and decorrelation seems to suggest strong resistance against variants of linear and differential attacks.

Finally, we think that the internal mathematical structure is so rich that we did not found all interesting properties so far. We recommend to have deeper analysis.

Here are our conclusion about CIPHERUNICORN-E.

1. **Discovery of unexpected internal properties: "−".** We discovered an unexpected relationship between substitution boxes.
2. **Randomness provided by the key schedule: "++".** The key schedule seems to provide pretty good randomness.
3. **Resistance against differential and linear cryptanalysis: "+".** The design seems to trickyly resist to these attacks.
4. **Resistance against side channel attacks: "−".** The complexity of round functions may provide some weaknesses.
5. **Maturity of the algorithm: ".".** Although quite young, this algorithm looks pretty mature. We did not found any international academic reference so far.
6. **Overall security confidence: ".".** We believe that deeper analysis is required.
7. **Beauty of the design: "+".** The design is quite conservative and looks quite tricky.