TWIS – A Lightweight Block Cipher

Shri Kant Ojha[1], Naveen Kumar[2], Kritika Jain[2], and Sangeeta[2]

[1] Joint Cipher Bureau, Department of Defence R & D, Metcalfe House, Delhi-110054, India
[2] Department of Computer Science, University of Delhi, Delhi-110007, India

Abstract. A new 128-bit block cipher, TWIS is proposed. It uses key size of 128-bits. The design targets to software environment for resource constrained applications. It is inspired from existing block cipher, CLEFIA. Although the proposed design uses less resources as compared to CLEFIA, it compares favorably with CLEFIA in terms of security provided.

Keywords: Cipher, CLEFIA, lightweight cryptography, and S-Box.

1 Introduction

Cryptographic techniques are used to protect sensitive and valuable information against any undesirable third party. In cryptography, we encrypt plaintext i.e. original message to be sent, using a key, which produces ciphertext i.e encrypted message. The encrypted message is decrypted by the receiver to get plaintext. There are in general two types of cryptographic algorithms, secret key (or symmetric key) and public-key (or asymmetric key) algorithms. While symmetric key algorithms use the same secret key to encrypt and decrypt a message, asymmetric algorithms use different keys for encryption and decryption. The symmetric key algorithms are further classified as block ciphers and stream ciphers whereas asymmetric key algorithms work only as block ciphers.

While block ciphers encrypt a block of data at a time, stream ciphers produce a series of data bits called keystream bits (a sequence of bits used as a key). In case of stream ciphers, encryption is accomplished by xoring the plaintext bits with key stream bits. Generation of keystream bits is independent of the plaintext. Hence, the ciphertext produced by a stream cipher for the same unit of plaintext may differ depending on where it appears. In contrast, the ciphertext produced by a block cipher depends on plaintext and secret key only. Hence they will always produce same output after encryption when provided with particular pair of plaintext block and secret key. Some of the common examples of stream cipher include Geffe, Grain, Trivium, Lex, Salsa20. DES, AES, RSA, are some common examples of block ciphers. Whereas former two are symmetric block ciphers, the latter is an asymmetric block cipher.

Conventional algorithms such as AES although quite secure, are not suitable for the extremely resource constrained applications such as RFID tags and sensor networks. The field of lightweight cryptography deals with designing ciphers for such environments. It deals with refining existing cryptographic algorithms or discovering new algorithms for providing high security in constrained environments [2],[1]. In lightweight cryptography there is trade-off between security, cost, and performance; and it is highly difficult to optimize all the three [2]. One also needs to carry out the trade off between - hardware level and software level optimization. For example, bit permutations can be implemented in hardware without any cost but can slow down performance in software, and large substitution tables are good to implement in software but they are relatively difficult to implement in hardware [2].

A number of symmetric lightweight block ciphers have been proposed in the literature - for example DESL [5], PRESENT [1], HIGHT [3], CLEFIA [10], LCASE [9], Cobra-H64 [6] and Cobra-H128 [6]. DESL [2],[5] is the modification of well known cipher DES. It modifies DES by replacing 8-S boxes of DES by one cryptographically stronger S-box [5]. This lightweight variant uses approximately 20% smaller chip than DES and 1,850 gate equivalents(GEs) against 2,310 GEs used in DES. PRESENT [1] is a new lightweight block cipher based on SPN with 32 rounds, a block size of 64 bits, and a key size of 80 or 128 bits. Design of PRESENT is very simple and it provides good performance in both hardware and software. Its structure favors repetition and hence it can be compactly implemented in hardware. As it requires only 1570 GEs its hardware requirements are competitive with today's leading stream ciphers. HIGHT [3] uses 64-bit block lenght and 128-bit key length.It is a hardware oriented cipher, based on 32-round iterative strucutre which is modification of Generalized Fiestel structure. HIGHT uses vary simple operations such as XOR, addition mod 2^8, and left bitwise rotation, and can be implmented with 3048 GE's on $0.25\mu m$ technology. Hence it is suitable for low-cost, low-power, and ultra-light implementation. CLEFIA [11] is another lightweight block cipher using variable block length and variable key size of 128, 192 and 246 bits. It is based on 4-branch compact but Generalized Feistel structure. Diffusion matrices and two S-box system is used to provide high security. LCASE [9] exploits inherent parallelism of the cellular automata for designing a high speed cipher. It improves over ICEBERG and AES significantly in terms of hardware complexity. Cobra-H64 [6] and Cobra-H128 [6] are two new high speed ciphers that make use of cryptographic primitives based on data-dependent permutations. They support variable plaintext lengths of size 64-bit and 128-bit. The ciphers exploit switchable operations to prevent weak keys. The authors have shown through hardware implementation of the ciphers that they are suitable for high-speed networks [6].

In this paper, we propose a new lightweight block cipher, TWIS. The proposed cipher is inspired from CLEFIA [10,11]. TWIS is a 128-bit block cipher which uses key of size 128 bit. It consists of two parts: Key scheduling part and Data processing part. It employs a 2-branch Generalized Fiestel structure which employs key whitening parts at the beginning and at the end of the cipher. The number of rounds taken by TWIS before producing a ciphertext is 10. TWIS also uses an S-Box and a diffusion matrix that enables good diffusion properties [4] while generating keystream.

TWIS is designed to employ good balance between three fundamental features: security, speed and cost of implementation. The salient features of TWIS are as under:

1. G-function used is same for both encryption and decryption.
2. We have used only 11 round keys. Each round key is of 4 bytes, totaling only 44 bytes.
3. Each of the 10 rounds during encryption or decryption process uses two round keys. G-function as defined in section 2.3 is called twice in one iteration, each call to G-function uses a single round key.

The paper is organized as follows: section 2 gives the design criteria and discusses the algorithm of the cipher. Section 3 describes the cryptanalysis of the cipher and the complexity of the cipher is discussed in section 4. Finally we conclude the paper in section 5.

2 Design Criteria

The design of the cipher is inspired from CLEFIA with an aim to make it lighter without compromising the security. The security requirements correspond to a computational complexity of 2^{128}, equivalent to exhaustive key search. For making the cipher lighter than CLEFIA, we have to take into account both time and space taken by the cipher in software.

The proposed cipher has 10 rounds for encryption/decryption. The repeated use of 64-bit G-function is to have a strong encryption algorithm, a single round of encryption might constitute a weak algorithm [7]. Each round of encryption/decryption uses two rounds of Fiestel network to scramble all bits of text block. The S-Box is used in the cipher to have good diffusion properties [4] while generating a ciphertext. It uses diffusion matrix [11] while generating round keys. The cipher also makes use of key whitening steps to increase the difficulty of key search attacks against the remainder of the cipher.

An overview of the different blocks of cipher for encryption/ decryption is shown in Fig.2 and Fig.3. The cipher uses a 2-branch Generalized Fiestel structure with additional key whitening of the input and output. It is divided into two parts namely, key scheduling and data processing.

2.1 Key Scheduling

The key scheduling part of the cipher deals with generation of round keys used in data processing part. Let K be the 128-bit initial key, which will produce 11 round keys, denoted by RK_i $(0 \leq i \leq 10)$, of 32-bit each. The key, K is divided into 16 bytes, denoted by $K[i]$ for $(1 \leq i \leq 16)$.

The complete algorithm for Key scheduling is as under:

```
1. for $round\_counter \leftarrow 1$ to $number\_rounds + 1$ do the following:
    (a) $K = K <<< 3$   ▷ left rotate the key by 3 bits
    (b) $K[round\_counter] \leftarrow S(K[round\_counter].0x3f)$
    (c) $K[15] \leftarrow S(K[15].0x3f)$
    (d) $K[16] \leftarrow K[16] \oplus round\_counter$
    (e) $(y_0, y_1, y_2, y_3) \leftarrow (K[13], K[14], K[15], K[16])$
    (f) $RK^t_{round\_counter-1} \leftarrow M(y_0, y_1, y_2, y_3)^t$
```

S is nonlinear 8-bit S-box, and M is 4×4 matrix defined later in subsequent sections.

2.2 Data Processing

The data processing part of TWIS consists of an encryption part, ENC_r and a decryption part, DEC_r for decryption. ENC_r and DEC_r are based on the 2-branch Generalized Feistal structure. Let $P, C \in \{0,1\}^{128}$ be a plaintext and corresponding ciphertext respectively, and let $P_i, C_i \in \{0,1\}^{32}$ $(0 \le i < 4)$ be divided plaintext and ciphertext where $P = P_0|P_1|P_2|P_3$ and $C = C_0|C_1|C_2|C_3$, and $RK_i \in \{0,1\}^{32}$ $(0 \le i \le 10)$ be round keys provided by the key scheduling part. Then 10-round encryption function ENC_r is defined as follows and is as shown in figure 2:

ENC_r :

```
1. $T_0|T_1|T_2|T_3 \leftarrow (P_0 \oplus RK_0)|P_1|P_2|(P_3 \oplus RK_1)$
2. for $round\_counter \leftarrow 1$ to 10, repeat steps (a) to (g)
    (a) $X_0|X_1 \leftarrow G - Function(RK_{round\_counter-1}, T_0, T_1)$
    (b) $T_2 \leftarrow X_0 \oplus T_2$
        $T_3 \leftarrow X_1 \oplus T_3$
    (c) $T_1 \leftarrow T_1 <<< 8$
        $T_3 \leftarrow T_3 >>> 1$
    (d) $T_0|T_1|T_2|T_3 \leftarrow T_2|T_3|T_0|T_1$
    (e) $X_0|X_1 \leftarrow G - Function(RK_{round\_counter}, T_0, T_3)$
    (f) $T_1 \leftarrow X_0 \oplus T_1$
        $T_2 \leftarrow X_1 \oplus T_2$
    (g) $T_2 \leftarrow T_2 >>> 1$
        $T_3 \leftarrow T_3 <<< 8$
3. $C_0|C_1|C_2|C_3 \leftarrow (T_0 \oplus RK_2)|T_1|T_2|(T_3 \oplus RK_3)$
```

The 10-round decryption function is just reverse of encryption process and is defined as follows and is shown in figure 3:

DEC_r :

```
1. $T_0|T_1|T_2|T_3 \leftarrow (C_0 \oplus RK_2)|C_1|C_2|(C_3 \oplus RK_3)$
2. for $round\_counter \leftarrow 10$ down to 1, repeat steps (a) to (g)
    (a) $T_2 \leftarrow T_2 <<< 1$
        $T_3 \leftarrow T_3 >>> 8$
    (b) $X_0|X_1 \leftarrow G - Function(RK_{round\_counter}, T_0, T_3)$
    (c) $T_1 \leftarrow X_0 \oplus T_1$
        $T_2 \leftarrow X_1 \oplus T_2$
    (d) $T_0|T_1|T_2|T_3 \leftarrow T_2|T_3|T_0|T_1$
    (e) $T_1 \leftarrow T_1 >>> 8$
        $T_3 \leftarrow T_3 <<< 1$
    (f) $X_0|X_1 \leftarrow G - Function(RK_{round\_counter-1}, T_0, T_1)$
    (g) $T_2 \leftarrow X_0 \oplus T_2$
        $T_3 \leftarrow X_1 \oplus T_3$
3. $P_0|P_1|P_2|P_3 \leftarrow (T_0 \oplus RK_0)|T_1|T_2|(T_3 \oplus RK_1)$
```

2.3 TWIS Building Blocks

In this subsection we will describe the building blocks i.e. G-function, F-function, S-box, and diffusion matrix 'M' used in the cipher.

G-Function
It takes two inputs, 32 bit round key and 64 bit data. It calls F-function, that takes input of 32-bit round key and 32-bit intermediate ciphertext or plaintext. Mathematically, G-function can be written as in equation 1:

$$G - function = \begin{cases} \{0,1\}^{32} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64} \\ (RK_{(32)}, X_{(64)}) \mapsto Y_{(64)} \end{cases} \tag{1}$$

The algorithm employed by G-function is as under:

```
1. $T_0|T_1 \leftarrow X$
2. $T_0 \leftarrow T_1 \oplus F(RK, T_0)$
3. $Y_0|Y_1 \leftarrow T_1|T_0$
```

F-Function
It takes two 32-bit inputs and produces 32-bit output. The input/output functions are defined by equation 2 [10].

$$F = \begin{cases} \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32} \\ (RK_{(32)}, x_{(32)}) \mapsto y_{(32)} \end{cases} \tag{2}$$

The F-function used takes 32-bit round key RK and 32-bit intermediate plaintext or ciphertext, depending on encryption or decryption, denoted by X. The 32-bit output is returned in Y. The algorithm used by F-function is as under:

```
1. $T_0|T_1|T_2|T_3 \leftarrow RK \oplus X$
2. $T_0 \leftarrow S(T_0.0x3f)$
   $T_1 \leftarrow S(T_1.0x3f)$
   $T_2 \leftarrow S(T_2.0x3f)$
   $T_3 \leftarrow S(T_3.0x3f)$
3. $Y_0|Y_1|Y_2|Y_3 \leftarrow T_2|T_3|T_0|T_1$
```

Table 1. S-box - S used

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	90	49	d1	c6	2f	33	74	fb	95	6d	82	ea	0e	b0	a8	1c
1.	28	d0	4b	92	5c	ee	85	b1	c4	0a	76	3d	63	f9	17	af
2.	bf	bf	19	65	f7	7a	32	20	16	ce	e4	83	9d	5b	4c	d8
3.	ee	99	2e	f8	d4	9b	0f	13	29	89	67	cd	71	dd	b6	f4

S-box
The cipher employs an S-box with 6-bit input which yields 8-bit output. The S-Box used by TWIS is shown in Table 1.

Diffusion Matrix
The diffusion matrix is employed at the round key generation step to ensure diffusion at the key level making it difficult for the cryptanalyst to know exact key. The diffusion matrix M [10] is defined as follows:

$$M = \begin{pmatrix} 0x01 & 0x02 & 0x04 & 0x06 \\ 0x02 & 0x01 & 0x06 & 0x04 \\ 0x04 & 0x06 & 0x01 & 0x02 \\ 0x06 & 0x04 & 0x02 & 0x01 \end{pmatrix}$$

3 Cryptanalysis

In this section, we consider some general attacks that are possible on block ciphers and investigate to what extent TWIS is resistant to them. In ideal situation there should be no attack faster than exhaustive key search having computational complexity 2^{128}.

3.1 Statistical Testing

NIST Statistical Test Suite, SP800-22 [12] is widely used testing software for pseudo random sequence generator. It is provided with 16 tests which test the random nature of bits generated by any cipher algorithm[1]. The results of statistical test on TWIS are shown in Table 2.

We used 100 files each containing 10^7 bits sequence, using default parameters. Table 2 indicates that when we convert the block cipher to stream cipher, the bits produced are random in nature.

3.2 Avalanche Effect

Avalanche effect states that if there is single bit change in key or plaintext, then there must be at least 50% change in number of ciphertext/ round key bits.

[1] The details of the tests can be referred in [12] and test suite can be downloaded from http://csrc.nist.gov/groups/ST/toolkit/rng/index.html

Table 2. Statistical Results of TWIS

Statistical Tests	P-value (TWIS)
Frequency	0.727003
Block Frequency (m = 128)	0.880692
Cumulative Sum - Forward	0.577018
Cumulative Sum - Backward	0.318680
Runs	0.711364
Long Runs of Ones (M = 10000)	0.753960
Rank	0.331621
Spectral DFT	0.654639
Non-overlapping Templates (m = 9 , B = 000000001)	0.970310
Overlapping Template	0.299256
Universal (L = 7, Q = 1280)	0.517380
Approximate Entropy (m = 10)	0.677380
Random Excursions (x = +1)	0.078587
Random Excursions Variant (x = -1)	0.732206
Linear Complexity (M = 500)	0.489456
Serial (m = 5, $\nabla\psi_m^2$)	0.130827

Effect on Round Keys on Changing Single Bit of Key

When a single bit of key is changed, there should be change in at least half of the bits of the round keys generated from the initial key. When this test is applied to TWIS, we found that there is at least 50% change of bits in round key on single bit change of key.

The cipher produces 11 round keys, each of 32 bits. Thus, having 352 bits in total. As we can see from the graph in figure 1, for 100 executions of the cipher for different single bit change in key, there is approximately 50% change in bits of round key. Thus, the avalanche effect requirement is satisfied by TWIS.

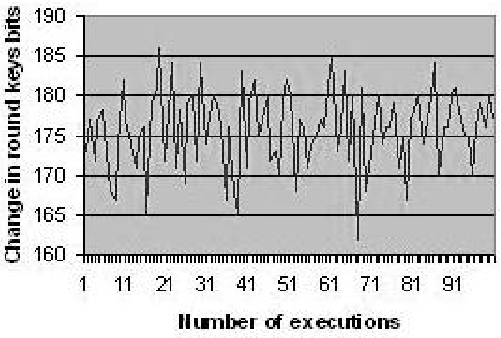

Fig. 1. Effect on round key

Effect on Ciphertext on Changing 1 bit of key keeping Plaintext Constant
A set of 1000 keys were generated randomly. For each of 1000 128-bit random key, 128-bit ciphertext is obtained keeping the plaintext fixed. We changed a single bit of key, and again obtained corresponding ciphertext. We found the number of bits changed in two ciphertexts by xoring them. We organized the the number of bits changed in five classes; 0-58, 59-62, 63-65, 66-69, and 70-128 [8]. The observations are shown in Table 3.

Table 3. Key/Ciphertext Avalanche Effect keeping plaintext fixed

Class	Frequency for Number of bits uncorrelated
0-58	156
59-62	231
63-65	211
66-69	235
70-128	167

From Table 3, it is concluded that in TWIS 61.3% times atleast half of the bits of key/keystream are uncorrelated, satisfying avalanche effect.

Effect on Ciphertext on Changing 1 bit of Plaintext keeping Key Constant
A set of 1000 random plaintexts are generated. For each of 1000 128-bit random plaintexts, 128-bit ciphertext is obtained keeping key fixed. We changed a single bit of plaintext, and again computed corresponding ciphertext. We find the number of bits changed in two ciphertexts by xoring them. After finding change in bits, we classified them into into 5 classes; 0-58, 59-62, 63-65, 66-69, and 70-128 [8]. The observations are shown in Table 4.

Table 4. Plaintext/Ciphertext Avalanche Effect keeping key fixed

Class	Frequency for Number of bits uncorrelated
0-58	156
59-62	217
63-65	234
66-69	230
70-128	163

From Table 4, it is concluded that 62.7% of times atleast half of the bits of key/keystream are uncorrelated, satisfying avalanche effect.

4 Complexity

The major goal behind the design of the cipher is to make it lighter so that it can be used in extremely resource-constrained environment like RFID tag and sensor

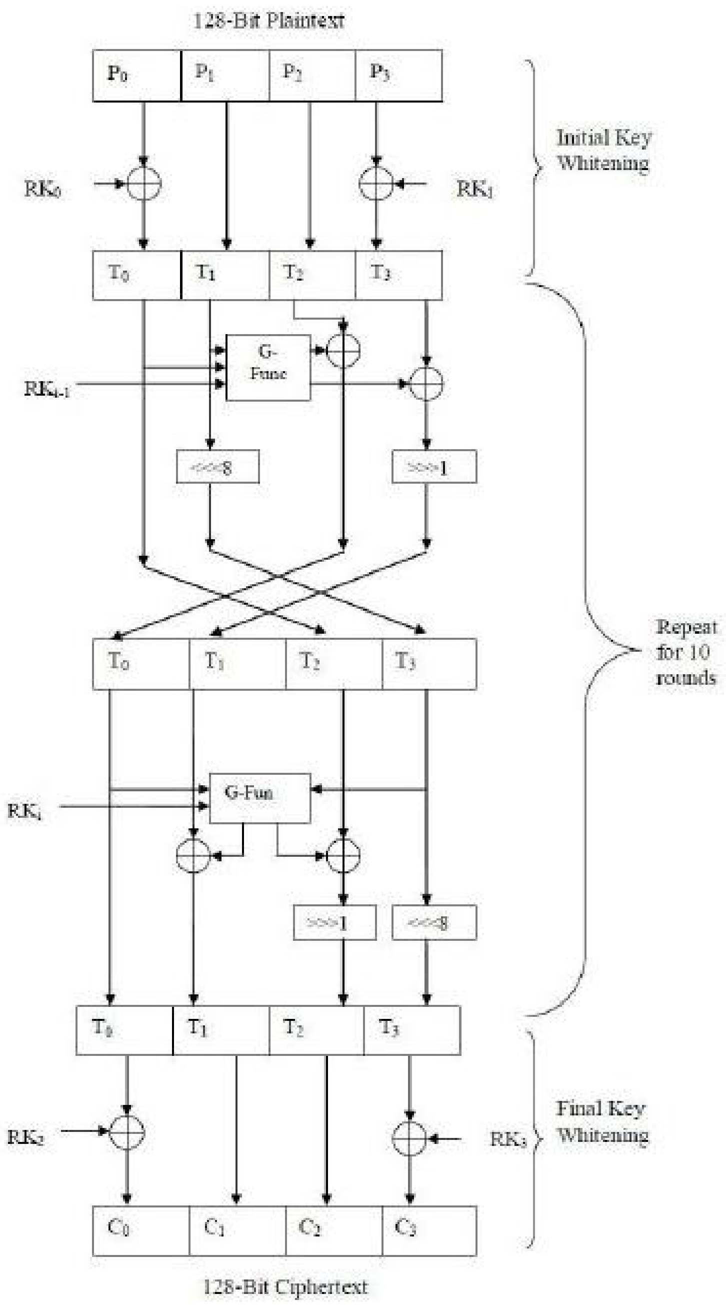

Fig. 2. The Encryption Process

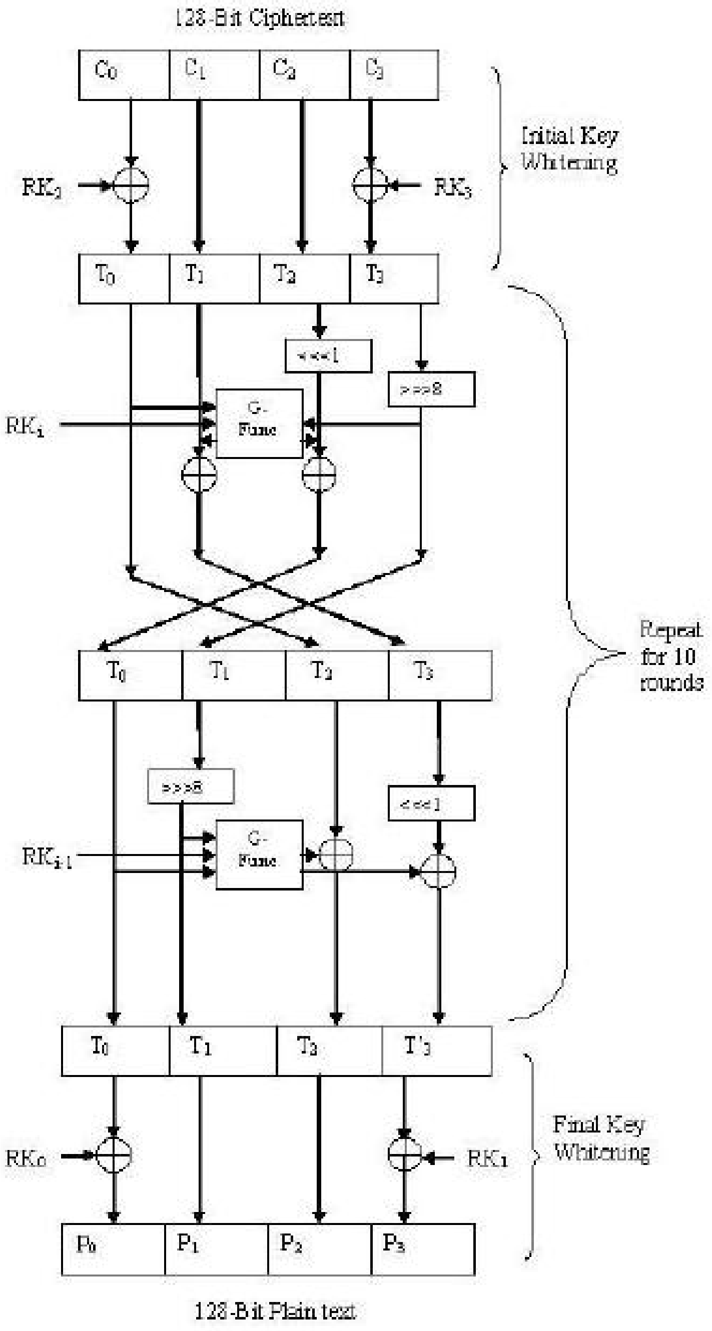

Fig. 3. The Decryption Process

networks. For this, we have modified the cipher CLEFIA and compared the time and memory it takes when implemented in software. By implementing both the ciphers on gcc compiler in Linux platform, it was found that for generating 10^7 bits using counter mode, which converts a block cipher to stream cipher, TWIS takes approximately 0.98 seconds against 1.98 seconds taken by CLEFIA. On examining the space, CLEFIA takes 121 bytes of memory while TWIS takes only 114 bytes of memory. However, both the ciphers take key length of 128 bits, having computational complexity of 2^{128}.

5 Conclusion

A new 128-bit lightweight block cipher, TWIS has been developed. It is designed with the motivation of building a highly secure cipher that can be used in devices used in extremely resource-constrained environment like RFID tags and sensor networks. As part of future work, we propose to study the cipher for linear and differential cryptanalysis.

Acknowledgment

The authors would like to acknowledge the support from research grant of University of Delhi: Dean(R)/R&D/2008/230.

References

1. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
2. Eisenbarth, T., Paar, C., Poschmann, A., Kumar, S., Uhsadel, L.: A Survey of Lightweight Cryptography Implementations, Copublished by the IEEE CS and the IEEE CASS, 0740-7475/07/. IEEE, Los Alamitos (2007)
3. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
4. Kim, K.: Construction of DES-like S-Box based on Boolean Functions satisfying the SAC. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 59–72. Springer, Heidelberg (1993)
5. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
6. Sklavos, N., Moldovyan, N.A., Koufopavlou, O.: High Speed Networking Security:Design and Implementation of Two New DDP-Based Ciphers. Mobile Networks and Applications 10, pp. 219–231. Springer, Heidelberg (2005)
7. Scheiner, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-bit Block Cipher. In: Counterpane System, Berkeley, California (1998)

8. Soumez, M., Doganaksoy, T.A., Calik, C.: Detailed Statistical Analysis of Synchronous Stream Ciphers. In: SASC 2006: Stream Cipher Revisited (2006)
9. Tripathy, S., Nandi, S.: LCASE: Lightweight Cellular Automata-based Symmetric-key Encryption. International Journal of Network Security 8(2), 243–252 (2009)
10. The 128-bit Block Cipher CLEFIA: Algorithm Specification. On-line document, 2007. Sony Corporation (2007)
11. The 128-bit Block Cipher CLEFIA: Design Rationale. Revision 1.0, 2007. Sony Corporation (2007)
12. NIST Special Publication 800-22. A Statistical Test for Random and Pseudorandom Number Generators for Cryptographic Application [EB/OL].[2001-05-15] (2001)