Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
February 2014, 21(1): 104–108
www.sciencedirect.com/science/journal/10058885
The Journal of China Universities of Posts and Telecommunications
http://jcupt.xsw.bupt.cn

Eight-sided fortress: a lightweight block cipher

LIU Xuan[1], ZHANG Wen-ying[1,2,4] (✉), LIU Xiang-zhong[3], LIU Feng[1]

1. School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
2. Science and Technology on Information Assume Laboratory, Beijing 100072, China
3. No.2 Middle School Attached to Shnadong Normal University, Jinan 250014, China
4. State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Abstract

In this paper, we present a new lightweight block cipher named eight-sided fortress (ESF), which is suitable for resource-constrained environments such as sensor networks and low-cost radio rrequency identification (RFID) tags. Meanwhile, we present the specification, design rationale and evaluation results in terms of the hardware implementation. For realizing both efficiency and security in embedded systems, similar to the other lightweight block ciphers, ESF is 64 bits block length and key size is 80 bits. It is inspired from existing block cipher, PRESENT and LBlock. The encryption algorithm of ESF is based on variant Feistel structure with SPN round function, used Feistel network as an overall structure with the purpose of minimizing computational resources.

Keywords block cipher, lightweight, RFID tags, efficiency, cryptography design

1 Introduction

With the development of low resource devices such as RFID tags, sensor nodes, smart cards, protection of privacy and so on, which are regarded as critical technologies in digital information society based on hardware and Internet. However resource-efficient cryptographic primitives are extremely critical on performance for security and efficiency in wireless communication and embedded systems. Lightweight cryptography has become a hot pot, research on designing and analyzing lightweight block ciphers have attracted a lot of attention. In fact, the applications of these technology have some properties, such as small storage space, weak computation ability, extremely power constraints etc. However, conventional algorithms such as AES although quite secure, are not suitable for extremely constrained environments. The filed of lightweight cryptography deals with designing ciphers for such environments [1]. In lightweight cryptography there is trade-off between security, cost, and performance, but it is highly difficult to optimize all the three, yet at the same time the design of new lightweight block cipher is also carefully considered the trade-off between hardware level and software level optimization, this allows us to make a trade-off and even though we use block cipher design techniques [2].

Recently, the lightweight cryptography for resource-constraint applications has attracted much attention. A series of lightweight block ciphers have been proposed in the literature, such as LBlock [3], PRESENT [1], HIGHT [4], mCrypton [5], DESL [6], KATAN and KTANTAN [7], CGEN [8], MIBS [9], TWIS [10], SEA [11] etc. All of these ciphers are designed and aimed at specifically for low-cost environments. Among them, LBlock [3] employs a variant of Festal structure and the encryption algorithm is four bits oriented which can be implemented efficiently in both hardware and software platform. Hardware implementation requires about 1 320 GE on 0.18 μm technology. PRESENT [1] is an example of an SP-network and consists of 31 rounds, a block size of 64 bits, and a key size of 80 bits or 128 bits. Its structure favors repetition and hence it can be compactly implemented in

Received date: 10-08-2013
Corresponding author: ZHANG Wen-ying, E-mail: wenyingzh@sohu.com
DOI: 10.1016/S1005-8885(14)60275-2

hardware. It requires only 1 570 GE and its hardware requirements are competitive with today's leading stream ciphers. HIGHT [4] uses 64 bits block length and 128 bits key length. It is a hardware oriented cipher, based on 32 rounds iterative structure which is modification of Generalized Feistel structure. HIGHT uses very simple operations such as XOR, addition mod 28, and left bit-wise rotation, and can be implemented with 3 048 GE on 0.25μm technology. mCrypton [5] has a precise hardware assessment and requires 2 949 GE. DESL [6] is based on the classical data encryption standard (DES) [12] design, but unlike DES it uses special case of a single *S*-box repeated eight times and hardware implementation results of DESL requires around 1 848 GE. KATAN and KTANTAN [7] are a family of lightweight block ciphers which contain six variants altogether. To ensure sufficient mixing, 254 rounds of the cipher are executed and the plaintext is loaded into two registers. In each round, several bits are taken from the registers and enter into two nonlinear Boolean functions. The output of the Boolean functions is loaded to the least significant bits of the registers. KATAN and KTANTAN are among the most hardware-efficient, requiring less than 1 000 GE. CGEN [8] is compact algorithm and built closely around the principles underlying the AES [13]. MIBS [9] is based on Feistel structure with SPN round function, used Feistel network as an overall structure with the purpose of minimizing computational resources, and selected the SPN for round function. The hardware implementation of MIBS-64 requires 1 400 GE on 0.18 μm technology. TWIS [10] is inspired from existing block cipher, CLEFIA [14–15], while SEA [11] with parameters requires around 2 280 GE.

In this paper, we propose a new lightweight block cipher called ESF. It is 64 bits block cipher and uses key of size 80 bits. Its consists of two parts: Key scheduling part and Data processing part, which is based on variant Feistel structure with SPN round function, and used Feistel network as an overall structure with the purpose of minimizing computational resources, addition a round subkey in each round function. The encryption algorithm of ESF consists of a 31 rounds intermediate structure. Furthermore, ESF performance is efficiently not only in software platforms but also in resource-contained hardware environments.

The rest of this paper is organized as follows. The specification of ESF is given in Sect. 2. Sect. 3 describes the design rationale. Sect. 4 provide results on hardware implementation. Finally, we conclude in Sect. 5.

2 Specification of ESF

This section provides the specification of ESF. The test vectors of ESF can be found in Appendix A.

2.1 Notations

The following notations are used throughout this paper.

M: 64 bits plaintext
C: 64 bits ciphertext
K: 80 bits master key
F: Round function
K_r: 32 bits round subkey
P: Permutation
$<<<7$: 7 bits cyclic left shift
S: 4×4 S-box
S: Substitution layer consists of eight 4×4 S-box in parallel
$||$: Two binary strings concatenation
$\oplus$: Bit-wise XOR
$[i]_2$: Binary form of an round_counter i

2.2 Data processing

The data processing part of ESF consists of an encryption part and a decryption part. Encryption and decryption are based on the 2-branch variant Feistel structure with SPN round function. Round functions composed of a key addition, an *S*-box layer and a permutation. The particular of this cipher is that the permutation and the substitution layer are reversible, meaning that the same primitive is used for both encryption and decryption process. Let $M,C\in\{0,1\}^{64}$ be a plaintext and corresponding ciphertext respectively, and let $M=L_0||R_0$ denote a 64 bits plaintext. Despite the design principle of ESF structure is adopted from the design principle of LBlock structure, the permutation layer is different from the LBlock, which applied on bit-wise permutation. The structure of ESF reduces restriction of designing inner auxiliary functions. Compared to SP-like structure, the round function is light. Since encryption process is simply converted into decryption process, implementation of the circuit supporting both encryption and decryption processes does not require much more cost

than the encryption-only circuit. The 80-bit register used in the key schedule algorithm contains the master key value both before and after running the algorithm. So, only one 80-bit register is required for both encryption and decryption processes.

2.2.1 Encryption algorithm

Each of the 32 rounds consists of an XOR operation to introduce a round key K_r for $1 \leqslant r \leqslant 32$, a linear permutation and a non-linear substitution layer. The non-linear layer uses eight 4×4 *S*-box *S* in parallel in each. The encryption procedure of ESF is described in Fig. 1, and each part is now specified in turn.

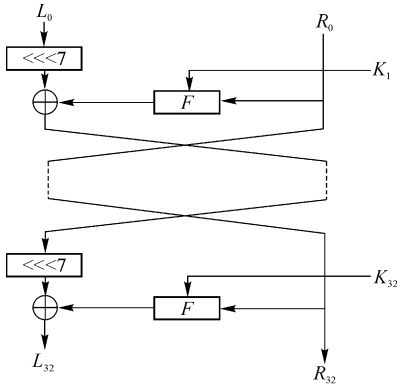

Fig. 1 Encryption procedure of ESF

ESF encryption procedure can be expressed as follows:

Step 1 ESF takes a 64 bits plaintext $M=L_0||R_0$ as the inputs, and the right part 32 bits R_0 enter into round function and the left part 32 bits L_0 deal with 7 bits left cyclic shift.

Step 2 For r=1,2,3,…,31, do $R_{r+1} = (L_{r-1} <<<7) \oplus \mathrm{F}(R_r, K_r)$, $L_{r+1} = R_r$.

Step 3 Finally, outputs a 64 bits data as a ciphertext $C=L_{32}||R_{32}$.

We selected the SPN for round function, and with regard to hardware efficiency and at the same time adequate security are guaranteed we used 4×4 *S*-boxes. Substitution layer is the only non-linear in ESF that a natural requirement is an optimal resistance against linear and differential attack.

Round function *F* The round function *F* is defined as follows, where *S* function is Substitution and *P* function is Permutation. Fig. 2 describes the structure of round function *F* in detail. The *F* function takes 32 bits round key and right half part of 64 bits intermediate state denoted by *x*, depending on encryption and decryption, the 32 bits output is returned in *y*. The input/output functions are defined by equation as the following.

$$F: \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$$
$$P[S(K_{r(32)}, x_{(32)})] \rightarrow y_{32}$$

Substitution layer Block ciphers employ substitution boxes (S-boxes) as important components. We must consider various factors in order to design good *S*-boxes, such as differential uniform, non-linearity, degrees of the coordinate Boolean functions and so on. Hence, we use the Serpent (http://www.cs.technion.ac.Il/biham/Reports/Serpent.) *S*-boxes, the nonlinear substitution layer which consists of eight different 4×4 *S*-box in parallel. Furthermore, the implementation cost of such a 4 bits *S*-box is much lower than that of an 8 bits *S*-box either by hardware, and we can also save the implementation costs for its inverse. For the current state 32 bits is considered as eight different 4 bits words and the output 32 bits provides the updated state values in the obvious way.

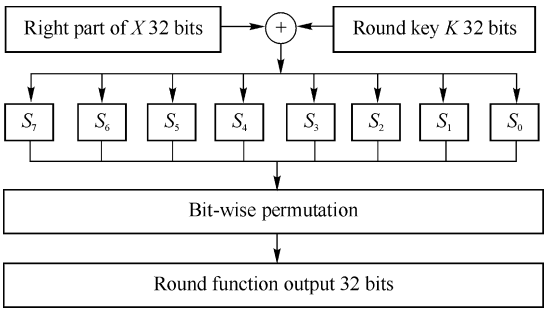

Fig. 2 Round function *F* of ESF

$$S: \{0,1\}^{32} \rightarrow \{0,1\}^{32}$$
$$a=a_7||a_6||a_5||a_4||a_3||a_2||a_1||a_0 \rightarrow b= b_7||b_6||b_5||b_4||b_3||b_2||b_1||b_0.$$
$$b_7 = S_7(a_7),\ b_6 = S_6(a_6),\ b_5 = S_5(a_5),\ b_4 = S_4(a_4),$$
$$b_3 = S_3(a_3),\ b_2 = S_2(a_2),\ b_1 = S_1(a_1),\ b_0 = S_0(a_0).$$

Permutation layer This stage is the bit-wise permutation layer. The inputs and outputs are 32 bits, respectively. We denote input one bit as *i*, each bit moved to the bit *P*[*i*] new position as output by permutation layer, which is given by Table 2 in detail. We directly to bit-wise permutation is similar to PRESENT [1] permutation layer. Since bit-wise permutation provides fast diffusion without hardware implementation cost, enhances security against impossible differential, saturation, meet-in-the-middle (MITM), etc, especially, we focus on hardware efficiency and achieve simplicity demands a linear layer that can be implemented with a minimum number of processing elements. Therefore, we have chosen a bit-wise permutation and it can be expressed as the following

equations.

P: $\{0,1\}^{32} \rightarrow \{0,1\}^{32}$

$b= b_7||b_6||b_5||b_4||b_3||b_2||b_1||b_0 \rightarrow c= c_7||c_6||c_5||c_4||c_3||c_2||c_1||c_0$.

For $0 \leqslant i<8$, do $b_{4i}||b_{4i+1}||b_{4i+2}||b_{4i+3} \rightarrow c_i||c_{i+8}||c_{i+16}||c_{i+24}$

2.2.2 Decryption algorithm

The decryption algorithm of ESF is the inverse of encryption procedure. Key schedule generates the subkeys in the reverse order. The decryption process can be made to have almost the same architecture as encryption process by using the transformation round keys, it consists of a 32 rounds variant Feistel structure with SPN round function, and it has >>> instead of <<< with the opposite direction to that in the encryption process.

The decryption procedure can be expressed as follows.

Step 1 Let C denotes a 64 bits ciphertext, for r=31, 30, …, 0, do $R_r= L_{r+1}, L_r=[F(L_{r+1},K_r) \oplus R_{r+1}]>>>7$.

Step 2 Finally, outputs a 64-bit data as a plaintext M.

2.3 Key scheduling

The design principle of ESF key schedule is adopted from the design principle of PRESENT key schedule. Our key scheduling generates 32 bits round subkey from 80 bits master key K. In particular, the master key is described as $K=k_{79}k_{78}…k_0$, which is stored in a key register. The key state for each round is updated as the following.

For i=1, 2, …, 31, update the key register K as follows.

Step 1 $K<<<13$;

Step 2 $[k_{79}k_{78}k_{77}k_{76}] = S_0[k_{79}k_{78}k_{77}k_{76}]$; $[k_{75}k_{74}k_{73}k_{72}] = S_0[k_{75}k_{74}k_{73}k_{72}]$;

Step 3 $[k_{47}k_{46}k_{45}k_{44}k_{43}] = [k_{47}k_{46}k_{45}k_{44}k_{43}] \oplus$ round_counter i;

Step 4 Output the leftmost 32 bits of current content of register K as round subkey K_{i+1}.

Firstly, the key register is rotated by 13 bits to the left. Secondly, let the current register state content of ESF the leftmost 8 bits $k_{79}k_{78}k_{77}k_{76}$, $k_{75}k_{74}k_{73}k_{72}$ are passed though the ESF S_0 respectively, and at the same time the round-counter i is XOR with $k_{47}k_{46}k_{45}k_{44}k_{43}$ 5-bit. Lastly, the round subkey is the leftmost 32 bits of the update current content of register K.

3 Design rationale

The design of the cipher is inspired from PRESENT and LBlock with an aim to make it lighter with compromising the security. For making the cipher lighter, we have to take into account both time and space taken by the cipher in implement. At present the designing of the block cipher mainly focuses on the nonlinear S-box design, the choice of permutation method and the better key schedule. ESF has 32 rounds for encryption or decryption, and repeated use of 32 bits F function is to have a strong encryption, a single round of encryption might constitute a weak algorithm [16]. Moreover, encryption and decryption are based on the 2-branch variant Feistel structure with SPN round function. Round functions are composed of a key addition, an S-box layer and a permutation. Furthermore, the Serpent S-box is used in the cipher to have good diffusion properties while generating a ciphertext. It uses diffusion layer while generating round keys [17]. Meanwhile, permutation layer is a linear operation, its implementation with not at any cost, only execute bit-wise permutation. In particular, the cipher also makes use of left 7 bits rotation steps to increase the difficulty of key search attacks against the remainder of the cipher. In the aspect of implement, the design of ESF consider lots of factors, especially security and area requirement implement in hardware. ESF uses variant Feistel structure with an SPN round function, since this structure construction operates on half of the block length in each iteration. Therefore the size of code or circuitry required to implement it is nearly halved, thus we use this structure with the purpose of minimizing computational resources, and it show that success reduce area, cost and performance. In the aspect of security requirement, we chose the bit-wise permutation in order that ESF can acquire the better diffusion in the first rounds. Moreover, the numbers of active S-boxes increase quickly. Hence, 2-branch variant Feistel structure seems a proper choice, which certainly is one of the most important considerations in hardware design for tiny ubiquitous devices.

4 Hardware implementation

The main goal of design can achieve efficiency in restricted-resource environment. Each component function is carefully designed with hardware implementation in mind. We implemented ESF in VHDL and synthesized it in order to check the hardware complexity on 0.18 μm CMOS technology. Fig. 4 shows data path as hardware implementation of ESF. The circuit consists of three parts: round function, key schedule and control logic. Each round function consists of key addition, substitution layer,

permutation layer, and data XOR. The substitution layer consists of eight 4×4 *S*-boxes which are used in parallel. 4×4 *S*-box is implemented with simple combination logic of 4 bits permutation, which is the same as the Serpent *S*-box. The key addition and data XOR are implemented as bit-wise XOR. The permutation layer is a simple bit-wise permutation with not at any cost. The master key is 80 bits through key schedule generates 32 bits round subkey, and round keys applied for each round function can be generated on-the-fly. Therefore, there is no need to store all the round keys, control logic controls round function and key schedule to process ESF algorithm. Our circuit processes one round encryption per one clock cycle. The estimated area for ESF with 80 bits master key requires about 1 300 gates on 0.18 μm technology (you can see Ref. [3] computed in details employ on GE). The 80 bits register used in the key schedule algorithm contains the master key value both before and after running the algorithm. Hence, only one 80 bits register is required for both encryption and decryption processes.

The overall merit of ESF are high efficiency and flexibility in both software and hardware implementations. The data process of ESF can be efficiently implemented using lookup tables by precomputing eight *S*-boxes, each of which contains of 4 bits and is realized as eight 4 bits lookup tables.Furthermore, we also need the original *S*-box for key schedule. For decryption processing, there is no need of storing *S*-boxes in more resource-restricted 8 bits platforms, since the same as *S*-boxes in the encryption processing. The eight *S*-boxes are stored more compactly only using 32 bits of storage. ESF needs a set of 32-round keys of 32 bits for data processing, corresponding to a storage of 80 bits user key and can achieve efficiency in hardware and software platform compared with other known lightweight block ciphers. A comparison for the hardware efficiency of ESF and other lightweight block ciphers is shown in Table 1.

Table 1 Comparsion of lightweight cipher implementations

Block ciphers	Block size	Key size	Throughput at 100 kHz/ (kbit·s^{-1})	Logic process/μm	Area (GE)
ESF	64	80	200	0.18	1 300
LBlock	64	80	200	0.18	1 320
PRESENT-80	64	80	200	0.18	1 570
MIBS-64	64	64	200	0.18	1 396
mCRYPTON	64	128	492.3	0.13	2 500
HIGHT	64	128	188.2	0.25	3 048
DES	64	56	44.4	0.18	2 300
DESXL	64	184	44.4	0.18	2 168
KATAN	64	80	25.1	0.13	1 054
KTANTAN	64	80	25.1	0.13	688

5 Conclusions

In this paper we present a new lightweight block cipher ESF with a 64 bits block size and an 80 bits key. we present the specification, design rationale and evaluation results in terms of the hardware implementation. For realizing both security and efficiency in resource-constrained applications, such as low-cost RFID tags and sensors, we make a trade-off and even though we use block cipher design techniques. ESF is based on the proven architecture of LBlock, which employs variant Feistel structure with SPN round function. The substitution layer *S*-boxes is the Serpent *S*-boxes that consists of eight 4×4 *S*-boxes in parallel. The permutation layer use bit-wise permutation, both of them are implemented efficiently in hardware and software. Meanwhile, we also demonstrated through hardware simulation that ESF is well-suited for our target applications. Our hardware implementation of ESF requires about 1 300 GE on 0.18 μm technology. As possible future work, we could perform validation of software efficiency used in typical sensor nodes and further cryptanalysis efforts are consequently required.

Acknowledgements

This work was supported by the the National Science Foundation of China (61272434), the Natural Science Foundation of Shandong Province (ZR2012FM004, ZR2013FQ021), and the Project of Senior Visiting Scholar of Shandong Province and Foundation of Science and Technology on Information Assume Laboratory (KJ-13-004).

Appendix A Test vectors of ESF

Test vectors of ESF for each key length are given here. The data are expressed in hexadecimal notation as follows.

Plaintext	Key	Ciphertext
00000000 00000000	00000000 00000000 0000	7766CE423E8C238F
00000000 00000000	FFFFFFFF FFFFFFFF FFFF	9A67F2AC F62F01C8
FFFFFFFF FFFFFFFF	00000000 00000000 0000	7317130E F71D52FF
FFFFFFFF FFFFFFFF	FFFFFFFF FFFFFFFF FFFF	3B006D54D48743F7
01234567 89ABCDEF	01234567 89ABCDEFFEDC	59AD89EA6C229DCC

References

1. Bogdanov A, Knudsen L R, Leander G, et al. PRESENT: an ultra-lightweight block cipher. Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'07), Sep 10–13, 2007, Vienna, Austria. LNCS 4727. Heidelberg, Germany: Springer, 2007: 450–466
2. Eisenbarth T, Paar C, Poschmann A, et al. A survey of lightweight cryptography Implementations. IEEE Design & Test of Computers, 2007, 24(6): 522–533

To p. 128

3. Huang Y H, Liao Q Z, Wei S M, et al. Dynamic modeling and analysis of a front-wheel drive bicycle robot moving on a slope. Proceedings of the 2010 IEEE International Conference on Automation and Logistics (ICAL'10), Aug 16–20, 2010, Hong Kong/Macau, China. Piscataway, NJ, USA: IEEE, 2010: 43–48
4. Shen T L. The robust control basis of robot. Beijing, China: Tsinghua University Press, 2000: 205–213 (in Chinese)
5. Zhang J, Hu H X. Dynamics modeling and simulation verification for space robot. Aerospace Control and Application, 2009, 35(6): 6–12 (in Chinese)
6. Li G J, Yi G, Lin Y. A method of bidirectional dynamic modeling for humanoid robot. Journal of Guangxi University: Natural Science Edition, 2011, 36(6): 1009–1015 (in Chinese)
7. Zhang G L, Tao L, Song H T, et al. Inverse kinematics analysis and emulation of the legs joint of the humanoid robot. Proceedings of the 2nd International Symposium on Test Automation & Instrumentation (ISTAI'08), Nov 17–21, 2008, Beijing, China. Beijing China: International Academic Publishers, 2008: 311–315
8. Tian X S. Hmanoid robots. Beijng, China: Tsinghua University Press, 2007: 87–89 (in Chinese)
9. Luo X M. The research of human body dynamics modeling and simulation based on ADAMS. Master Dissertation. Guangzhou, China: Guangdong University of Technology , 2006: 22–32 (in Chinese)

(Editor: ZHANG Ying)

From p. 108

3. Wu W L, Zhang L. LBlock: a lightweight block cipher. Proceedings of the 9th International Workshop on Applied Cryptography and Network Security (ACNS'11), Jun 7–10, 2011, Nerja, Spain. LNCS 6715. Heidelberg, Germany: Springer, 2011: 327–344
4. Hong D, Sung J, Hong S, et al. HIGHT: a new block cipher suitable for low-resource device. Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'06), Oct 10–13, 2006, Yokohama, Japan. LNCS 4249. Heidelberg, Germany: Springer, 2006: 46–59
5. Lim C H, Korkishko T. mCrypton: a lightweight block cipher for security of low-cost RFID tags and sensors. Proceedings of the 6th International Workshop on Information Security Applications (WISA'05), Aug 22–24, 2005, Jeju Island,Republic of Korea. LNCS 3786. Springer, Germany: Heidelberg, 2006: 243–258
6. Leander G, Paar C, Poschmann A. New lightweight DES variants. Proceedings of the 14th International Conference on Fast Software Encryption (FSE'07), Mar 26–28, 2007, Luxembourg. LNCS 4593. Heidelberg, Germany: Springer, 2007: 196–210
7. Canniere C, Dunkelman O, Knezevic M. KATAN and KTANTAN: a family of small and efficient hardware-oriented block ciphers. Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'09), Sep 6–9, 2009, Lausanne, Switzerland. LNCS 5747. Springer, Germany: Heidelberg, 2009: 272–288
8. Robshaw M J B. Searching for compact algorithms: CGEN. Progress in Cryptology: Proceedings of the 1st International Conference on Cryptology (VIETCRYPT'06), Sep 25–28, 2006, Hanoi, Vietnam. LNCS 4341. Heidelberg, Germany: Springer, 2006: 37–49
9. Izadi M, Sadeghiyan B, Sadeghian S, et al. MIBS: a new lightweight block cipher. Proceedings of International Conference on Cryptology and Network Security (CANS'09), Dec 12–14, 2009, Kanazawa, Japan. LNCS 5888. Heidelberg, Germany: Springer, 2009: 334–348
10. Ojha S, Kumar N, Jain K, et al. TWIS: a lightweight block cipher. Proceedings of the 5th International Conference on Information Systems Security (ICISS'09), Dec 14–18, 2009, Kolkata, India. LNCS 5905. Heidelberg, Germany: Springer, 2009: 280–291
11. Standaert F X, Piret G, Gershenfeld N, et al. SEA: a scalable encryption algorithm for small embedded applications.Proceedings of the 7th Smart Card Research and Advanced Application IFIP Conference (CARDIS'06), Apr 19–21, 2006, Tarragona, Spain. LNCS 3928. Heidelberg, Germany: Springer, 2006: 222–236
12. Coppersmith C. The data encryption standard (DES) and its strength against attacks. Technical Report rc 186131994. IBM Thomas J. Watson Research Center, 1994
13. Daemen J, Rijmen V. The design of rijndael. Berlin, Germany: Springer-Verlag, 2002
14. Shirai T, Shibutani K, Akishita T, et al. The 128-bit block cipher CLEFIA (Extended Abstract). Proceedings of the 14th International Conference on Fast Software Encryption (FSE'07), Mar 26–28, 2007, Luxembourg. LNCS 4593. Heidelberg, Germany: Springer, 2007: 181–195
15. Zhang W, Han J. Impossible differential analysis of reduced round CLEFIA. Proceedings of the 4th International Conference on Information Security and Cryptology (Inscrypt'08), Dec 14–17, 2008, Beijing, China. Heidelberg, Germany: Springer, 2008:181–191
16. Scheiner B, Kelsey J, Whiting D, et al. A 128-bit block cipher. Berkeley, CA, USA: Counterpane System, 1998
17. Kim K. Construction of DES-like S-box based on Boolean functions satisfying the SAC. Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'91), Nov 11–14, 1991, Fujiyoshida, Japan. LNCS 739. Heidelberg, Germany: Springer, 1993: 59–72

(Editor: ZHANG Ying)