LBlock: A Lightweight Block Cipher*

Wenling Wu and Lei Zhang

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, P. R. China
E-mail:{wwl, zhanglei1015}@is.iscas.ac.cn

Abstract. In this paper, we propose a new lightweight block cipher called LBlock. Similar to many other lightweight block ciphers, the block size of LBlock is 64-bit and the key size is 80-bit. Our security evaluation shows that LBlock can achieve enough security margin against known attacks, such as differential cryptanalysis, linear cryptanalysis, impossible differential cryptanalysis and related-key attacks etc. Furthermore, LBlock can be implemented efficiently not only in hardware environments but also in software platforms such as 8-bit microcontroller. Our hardware implementation of LBlock requires about 1320 GE on 0.18 μm technology with a throughput of 200 Kbps at 100 KHz. The software implementation of LBlock on 8-bit microcontroller requires about 3955 clock cycles to encrypt a plaintext block.

Key words: Block cipher, Lightweight, Hardware efficiency, Design, Cryptanalysis.

1 Introduction

With the development of electronic and communication applications, RFID technology has been used in many aspects of life, such as access control, parking management, identification, goods tracking etc. In this kind of new cryptography environment, the applications of RFID technology and sensor networking both have similar features, such as weak computation ability, small storage space, and strict power constraints. Therefore, traditional block ciphers such as AES are not suitable for this kind of extremely constrained environment. Hence, in recent years, research on lightweight ciphers has received a lot of attention. Compared with traditional block ciphers, lightweight ciphers have the following three main properties. Firstly, applications for constrained devices are unlikely to require the encryption of large amounts of data, and hence there is no requirement of high throughput for lightweight ciphers. Secondly, in this cryptography environment, attackers are lack of data and computing ability, which means lightweight ciphers only need to achieve moderate security. Lastly, lightweight ciphers are usually

* This paper was first published at ACNS 2011, LNCS 6715, pp. 327-344. Unfortunately, there are some errors in the contents of S-box table in page 332, and here we provide a revision of this paper.

implemented in hardware environment, and small part of them are also implemented on software platforms such as 8-bit microcontroller. Therefore, hardware performance will be the primary consideration for lightweight ciphers. Hardware efficiency can be measured in many different ways: the length of the critical path, latency, clock cycles, power consumption, throughput, area requirements, and so on. Among them area requirement is the most important parameter, since small area requirement can minimize both the cost and the power consumption efficiently. Therefore, it has become common to use the term hardware efficient as a synonym for small area requirements, and the area requirements are usually measured as gate equivalents (GE). At present, for the hardware implementation of lightweight cipher, area requirements are usually dominated by the registers storing the data state and the key, since registers typically consist of flipflops which have a rather high area and power demand. For example, when using the standard cell library it requires between 6 and 12 GE to store a single bit [26]. Therefore, in the design of lightweight block ciphers, 64-bit block size and 80-bit key size are popular parameters.

While there is a growing requirement of ciphers suited for resource-constraint applications, a series of lightweight block ciphers have been proposed recently, e.g. PRESENT[9], HIGHT[14], mCrypton[21], DESL[19], CGEN[28], MIBS[15], KATAN & KTANTAN[10], TWIS[23], SEA[30] etc. All of these ciphers are designed and targeted specifically for extremely constrained environments such as RFID tags and sensor networks. Among them, PRESENT is supposed to be very competitive, since its hardware requirement is comparable with today's leading compact stream ciphers, and it is called an ultra-lightweight block cipher. Since its publication, only a few cryptanalytic results have been proposed against PRESENT, including the related-key rectangle attack on 17-round PRESENT in [24] and the side-channel attacks described in [27, 35]. HIGHT has a 32-round generalized Feistel structure. Its main feature is the compact round function which contains no S-box and all the operations are simple computations such as XOR, rotation, and addition operating on 8-bit input. In respect of cryptanalysis, a related-key attack on full-round HIGHT was presented in ICISC2010, and an impossible differential attack on 26-round HIGHT were presented in [24]. mCrypton can be considered as a miniature of the block cipher Crypton[20], and a related-key rectangle attack on 8-round mCrypton has been reported in [25]. DESL and DESXL are lightweight modified versions of the well-known DES, and they adopt only one single S-box in order to minimize the hardware implementation. CGEN employs a compact round function called *mixtable* operation, and the main design strategies include using a fixed and per-device seed key which reduces the key scheduling and the decryption operation is not needed either. MIBS is a 32-round Feistel cipher, and its round function employs SP-network with XOR operations as diffusion layer, whose hardware requirements are more expensive than the bitwise permutation used in PRESENT etc. KATAN and KTANTAN are a family of lightweight block ciphers which contain six variants altogether. The KATAN family of ciphers all employ the same components, whose design strategy exploits some features of stream cipher [11]. Meet-in-the-

middle attacks to the KTANTAN family with a key of 80 bits were presented in [36]. TWIS is inspired from the existing block cipher CLEFIA [29]. However, a differential distinguisher with probability 1 for full-round TWIS was presented in [31]. SEA is a Feistel cipher with scalable block and key sizes, and its round function only consists of rotation, XOR, and a single 3-bit S-box operations. TEA [33] and XTEA [34] are lightweight block ciphers proposed several years earlier.

In this paper we propose a new lightweight block cipher called LBlock. The design of its structure and components, such as S-box layer, P permutation layer etc, all represent the trade-off between security and performance. Our security analysis shows that full-round LBlock can provide enough security margin against known cryptanalytic techniques, such as differential cryptanalysis, linear cryptanalysis, impossible differential cryptanalysis, related-key attack etc. Furthermore, the performance evaluation of LBlock shows that not only hardware efficiency but also software implementations on 8-bit/32-bit platforms are ultra lightweight. The rest of this paper is organized as follows. Sect. 2 presents the specification of LBlock. Sect. 3 introduces the design rationale briefly. Sect. 4 and Sect. 5 describe the security analysis and performance evaluation of LBlock respectively. Finally, Sect. 6 concludes the paper.

2 Specification of LBlock

The block length of LBlock is 64-bit, and the key length is 80-bit. It employs a variant Feistel structure and consists of 32 rounds. The specification of LBlock consists of three parts: encryption algorithm, decryption algorithm and key scheduling.

2.1 Notations

In the specification of LBlock, we use the following notations:

- M: 64-bit plaintext
- C: 64-bit ciphertext
- K: 80-bit master key
- K_i: 32-bit round subkey
- F: Round function
- s: 4×4 S-box
- S: S-box layer consists of eight s in parallel
- P, P_1: Permutations operate on 32-bit
- $\bigoplus$: Bitwise exclusive-OR operation
- $<<< 8$: 8-bit left cyclic shift operation
- $>>> 8$: 8-bit right cyclic shift operation
- $||$: Concatenation of two binary strings
- $[i]_2$: Binary form of an integer i

2.2 Encryption Algorithm

The encryption algorithm of LBlock consists of a 32-round iterative structure which is a variant of Feistel network. The encryption procedure is illustrated in Fig. 1. Let $M = X_1||X_0$ denote a 64-bit plaintext, and then the data processing procedure can be expressed as follows.

1. For $i = 2, 3, \ldots, 33$, do

$$X_i = F(X_{i-1}, K_{i-1}) \oplus (X_{i-2} <<< 8)$$

2. Output $C = X_{32}||X_{33}$ as the 64-bit ciphertext

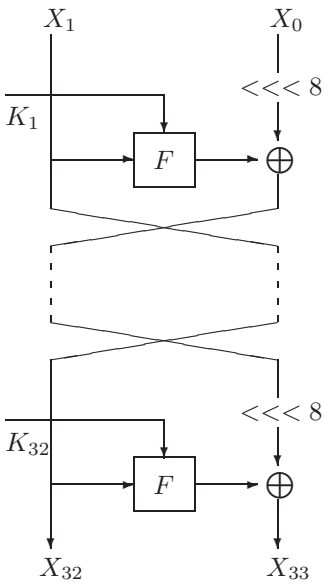

Fig. 1. Encryption procedure of LBlock

Specifically, the components used in each round are defined as follows.

(1) Round function F

The round function F is defined as follows, where S and P denote the confusion and diffusion functions which will be defined later.

$$\begin{array}{rccl} F: & \{0,1\}^{32} \times \{0,1\}^{32} & \longrightarrow & \{0,1\}^{32} \\ & (X, K_i) & \longrightarrow & U = P(S(X \oplus K_i)) \end{array}$$

Fig. 2 illustrates the structure of round function F in detail.

(2) Confusion function S

Confusion function S denotes the non-linear layer of round function F, and it consists of eight 4-bit S-boxes s_i in parallel.

$$\begin{array}{c} S: \{0,1\}^{32} \longrightarrow \{0,1\}^{32} \\ Y = Y_7||Y_6||Y_5||Y_4||Y_3||Y_2||Y_1||Y_0 \longrightarrow Z = Z_7||Z_6||Z_5||Z_4||Z_3||Z_2||Z_1||Z_0 \end{array}$$

$$\begin{array}{c} Z_7 = s_7(Y_7), \quad Z_6 = s_6(Y_6), \quad Z_5 = s_5(Y_5), \quad Z_4 = s_4(Y_4), \\ Z_3 = s_3(Y_3), \quad Z_2 = s_2(Y_2), \quad Z_1 = s_1(Y_1), \quad Z_0 = s_0(Y_0). \end{array}$$

The contents of eight 4-bit S-boxes are listed in Table 1.

(3) Diffusion function P

Diffusion function P is defined as a permutation of eight 4-bit words, and it can be expressed as the following equations.

$$\begin{array}{rcl} P: \{0,1\}^{32} & \longrightarrow & \{0,1\}^{32} \\ Z = Z_7||Z_6||Z_5||Z_4||Z_3||Z_2||Z_1||Z_0 & \longrightarrow & U = U_7||U_6||U_5||U_4||U_3||U_2||U_1||U_0 \end{array}$$

$$\begin{array}{c} U_7 = Z_6, \quad U_6 = Z_4, \quad U_5 = Z_7, \quad U_4 = Z_5, \\ U_3 = Z_2, \quad U_2 = Z_0, \quad U_1 = Z_3, \quad U_0 = Z_1. \end{array}$$

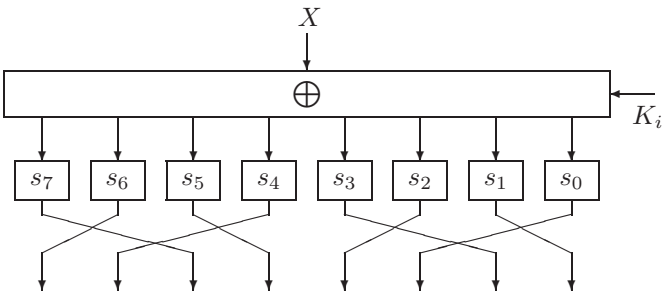

Fig. 2. Round function F

2.3 Decryption Algorithm

The decryption algorithm of LBlock is the inverse of encryption procedure, and it consists of a 32-round variant Feistel structure too. Let $C = X_{32}||X_{33}$ denotes a 64-bit ciphertext, and then the decryption procedure can be expressed as follows.

1. For $j = 31, 30, \ldots, 0$, do

$$X_j = (F(X_{j+1}, K_{j+1}) \oplus X_{j+2}) >>> 8$$

2. Output $M = X_1||X_0$ as the 64-bit plaintext.

2.4 Key Scheduling

The 80-bit master key K is stored in a key register and denoted as $K = k_{79}\, k_{78}\, k_{77}\, k_{76}\, ...\, k_1 k_0$. Output the leftmost 32 bits of current content of register K as round subkey K_1, and then operate as follows:

1. For $i = 1, 2, \ldots, 31$, update the key register K as follows:
   (a) $K <<< 29$
   (b) $[k_{79}\, k_{78}\, k_{77}\, k_{76}] = s_9[k_{79}\, k_{78}\, k_{77}\, k_{76}]$
   $[k_{75}\, k_{74}\, k_{73}\, k_{72}] = s_8[k_{75}\, k_{74}\, k_{73}\, k_{72}]$
   (c) $[k_{50} k_{49} k_{48} k_{47} k_{46}] \oplus [i]_2$
   (d) Output the leftmost 32 bits of current content of register K as round subkey K_{i+1}.

where s_8 and s_9 are two 4-bit S-boxes, and they are defined in Table 1.[1]

Table 1. Contents of the S-boxes used in LBlock

s_0	14, 9, 15, 0, 13, 4, 10, 11, 1, 2, 8, 3, 7, 6, 12, 5
s_1	4, 11, 14, 9, 15, 13, 0, 10, 7, 12, 5, 6, 2, 8, 1, 3
s_2	1, 14, 7, 12, 15, 13, 0, 6, 11, 5, 9, 3, 2, 4, 8, 10
s_3	7, 6, 8, 11, 0, 15, 3, 14, 9, 10, 12, 13, 5, 2, 4, 1
s_4	14, 5, 15, 0, 7, 2, 12, 13, 1, 8, 4, 9, 11, 10, 6, 3
s_5	2, 13, 11, 12, 15, 14, 0, 9, 7, 10, 6, 3, 1, 8, 4, 5
s_6	11, 9, 4, 14, 0, 15, 10, 13, 6, 12, 5, 7, 3, 8, 1, 2
s_7	13, 10, 15, 0, 14, 4, 9, 11, 2, 1, 8, 3, 7, 5, 12, 6
s_8	8, 7, 14, 5, 15, 13, 0, 6, 11, 12, 9, 10, 2, 4, 1, 3
s_9	11, 5, 15, 0, 7, 2, 9, 13, 4, 8, 1, 12, 14, 10, 3, 6

3 Design Rationale

3.1 Structure

The structure of LBlock is a variant of Feistel network, and its design decisions contain a lot of considerations about security and efficient implementations (such as area, cost and performance etc.). In the aspect of implementation, the most important consideration is the area requirement when implemented in hardware. Therefore, we try to reduce the number of S-boxes used in each round and also minimize the size of each S-box used. Hence a Feistel-type structure seems a

[1] There are some errors in the contents of s_8 and s_9 in the original paper, and we have corrected them here. Note that the other parts of this paper remain unchanged (including test vectors in Appendix I). The errors are only introduced in our typos.

proper choice. Furthermore, for all kinds of generalized Feistel structures which operate less bits in each round, to achieve enough security margin they must take more rounds iteration which will affect its performance (such as speed and throughput). Therefore, in each round of LBlock, we choose only half of the data to go through round function F, and the other half applies a simple rotation operation. In the diffusion layer, we also choose to use permutation which can be implemented with no cost in hardware. However, instead of the bitwise permutation usually used, we apply a 4-bit word-wise permutation which can be implemented cheaply not only in hardware but also in software environments such as 8-bit microprocessor platforms. For example, the word-wise permutation in round function F can be combined with the S-box layer to form 8×8 table lookups. Moreover, we specifically choose the rotation offsets of right half in each round as 8 bits which can be omitted in 8-bit platform implementation. On the other hand, in the aspect of security requirement, we choose the word-wise permutation carefully so that the structure of LBlock satisfies that in both encryption and decryption directions it can achieve best diffusion [32] in 8 rounds. Furthermore, the number of differential and linear active S-boxes both increase quickly, and the following Table 2 lists the guaranteed number of active S-boxes before 20 rounds.

Table 2. Guaranteed number of active S-boxes of LBlock

Rounds	**DS**	**LS**	**Rounds**	**DS**	**LS**
1	0	0	**11**	22	22
2	1	1	**12**	24	24
3	2	2	**13**	27	27
4	3	3	**14**	30	30
5	4	5	**15**	32	32
6	6	6	**16**	35	35
7	8	8	**17**	36	36
8	11	11	**18**	39	39
9	14	14	**19**	41	41
10	18	18	**20**	44	44

3.2 Diffusion Layer

The diffusion permutation of LBlock consists of two parts, namely the word-wise permutation in round function which is denoted as P, and the rotation of right half data in each round which is denoted as P_1. Both of these permutations can be implemented by wiring in hardware which needs no additional area cost. For software environments such as 8-bit and 32-bit microprocessor platforms, P can be combined with the S-box layer in round function as table lookups

and P_1 (8-bit rotation) can be implemented quite easily. Therefore, the diffusion permutations of LBlock can be implemented efficiently both in hardware and in software environments. Furthermore, the combination of P and P_1 can guarantee the best diffusion rounds and the least number of active S-boxes of LBlock. For example, there already exist at least 32 active S-boxes for 15-round LBlock.

3.3 S-Box Layer

On the pursuit of hardware efficiency, we use 4×4 S-boxes $s : F_2^4 \rightarrow F_2^4$ in LBlock. Compared with the regular 8×8 S-box, small S-box has much more advantage when implemented in hardware. For example, to implement the S-box of AES in hardware more than 200 GE are needed. On the other hand, for the 4×4 S-boxes used in LBlock, all of them can be implemented in hardware with only about 22 GE. Furthermore, in the aspect of security, the S-boxes used in LBlock are carefully chosen so that they all fulfill the following conditions: no fix point, completed, best non linearity, best differential probability, and good algebraic order etc.

3.4 Key Scheduling

Similar to many other lightweight block ciphers, the key scheduling of LBlock is also designed in a stream cipher way. We only apply simple rotation and non-linear operations to generate the round subkeys. First of all, the operation of 29-bit left rotation can be implemented freely in hardware, and it can also break the 4-bit word structure, which helps to improve the security of LBlock against related-key attacks. Secondly, we choose to use two 4×4 S-boxes as the non-linear operation which represents a trade-off between security and performance. Lastly, the exact values of rotation offset, constants and positions of constant addition are carefully chosen, so as to avoid weak relations between round subkeys.

4 Security Evaluation

4.1 Differential Cryptanalysis

For differential cryptanalysis, we adopt an approach to count the number of active S-boxes of differential characteristics. This is a regular method to evaluate the security against differential attack, which were adopted by many other block ciphers, such as AES [12], Camellia [1] and CLEFIA [29] etc. We found the guaranteed number of differential active S-boxes of LBlock by computer program, and the results before 20-round are listed in Table 2. Considering that there are at least 32 active S-boxes for 15-round LBlock and the best differential probabilities of s_i are all equal to 2^{-2}, then the maximum probability of differential characteristics for 15-round LBlock satisfies $DCP_{\max}^{15r} \leq 2^{32 \times (-2)} = 2^{-64}$. This means there is no useful 15-round differential characteristic for LBlock, since the block length of LBlock is only 64-bit. Therefore, we believe that the full 32-round LBlock is secure against differential cryptanalysis.

4.2 Linear Cryptanalysis

We also apply the method of counting active S-boxes for the evaluation of LBlock against linear cryptanalysis. Since there are at least 32 active S-boxes for 15-round LBlock and the best linear bias of each s_i is 2^{-2}, the maximum bias of linear approximations for 15-round LBlock satisfies $LCP_{\max}^{15r} \leq 2^{32-1} \cdot 2^{32\times(-2)} = 2^{-33}$. Therefore, according to the complexity estimation of linear cryptanalysis, we can conclude that it is difficult to find useful 15-round linear-hulls which can be used to distinguish LBlock from a random permutation. As a result, we believe that the full 32-round LBlock has enough security margin against linear cryptanalysis.

4.3 Impossible differential Cryptanalysis

Impossible differential attack [3] is one of the most powerful cryptanalytic techniques, and its applications to many block ciphers (such as Camellia and CLEFIA etc.) represent the best cryptanalytic results obtained so far. We search for the impossible differential characteristic of LBlock using the algorithm proposed by Kim et al. [16]. The best distinguisher found is the following 14-round impossible differential characteristic:

$$(00000000, 00\alpha00000) \overset{14r}{\not\rightarrow} (0\beta000000, 00000000), \tag{1}$$

where $\alpha, \beta \in \{0,1\}^4 \backslash \{0\}$ represent non-zero differences. Note that by changing the positions of α, β, we can construct other 14-round impossible differential characteristics in a similar way.

Based on the 14-round impossible differential distinguishers, we can mount a key recovery attack on 20-round LBlock. The attack procedure can be described as follows.

1. Choose a set of 2^{12} plaintexts to construct a structure, where the 4-bit words $X_{0,1}$, $X_{0,3}$ and $X_{1,2}$ take all possible values and all the other words take constants. Then each structure can generate about 2^{23} plaintext pairs satisfying the input difference $(\Delta X_1, \Delta X_0) = (00000 * 00, 0000 * 0 * 0)$. Choose 2^{51} different structures which can generate about 2^{74} candidate plaintext pairs.
2. For each corresponding ciphertext structure after 20-round encryption, choose the pairs satisfying the output difference $(\Delta X_{21}, \Delta X_{20}) = (**00**0*, 000*0**0)$, where $*$ denotes non-zero difference. After this test, there remains about $2^{74} \times 2^{-32} = 2^{42}$ candidate pairs.
3. For every guess of 28-bit subkey $K_{20,0}, K_{20,1}, K_{20,2}, K_{20,4}, K_{20,5}, K_{20,6}, K_{20,7}$, partially decrypt Round 20 to check if the pairs satisfying $(\Delta X_{20}, \Delta X_{19}) = (000*0**0, 00*0000*)$. After this test, there remains about $2^{42} \times 2^{-12} = 2^{30}$ pairs.
4. For every guess of the 16-bit subkey $K_{19,0}, K_{19,2}, K_{19,3}, K_{19,5}$, partially decrypt Round 19 to check if the pairs satisfying $(\Delta X_{19}, \Delta X_{18}) = (00 * 0000*, *0000000)$. After this test, there remains $2^{30} \times 2^{-8} = 2^{22}$ pairs.

5. For every guess of the 8-bit subkey $K_{18,1}, K_{18,7}$, partially decrypt Round 18 to check if the candidate pairs satisfying $(\Delta X_{18}, \Delta X_{17}) = (*0000000, 0 * 000000)$. After this test, there remains about $2^{22} \times 2^{-4} = 2^{18}$ pairs.
6. For every guess of the 4-bit subkey $K_{17,6}$, partially decrypt Round 17 to check if the candidate pairs satisfying $(\Delta X_{17}, \Delta X_{16}) = (0 * 000000, 00000000)$. After this test, there remains about $2^{18} \times 2^{-4} = 2^{14}$ pairs.
7. For every guess of the 8-bit subkey $K_{1,2}$, $K_{1,7}$, partially encrypt Round 1 to check if the candidate pairs satisfying $(\Delta X_2, \Delta X_1) = (00 * 00000, 00000 * 00)$. After this test, there remains about $2^{14} \times 2^{-4} = 2^{10}$ pairs.
8. For every guess of the 4-bit subkey $K_{2,5}$, partially encrypt Round 2 to check if the candidate pairs satisfying the following equation:

$$(\Delta X_3, \Delta X_2) = (00000000, 00 * 00000).$$

9. If there still remains a pair satisfying the impossible differential, then the 68-bit subkey guessed must be wrong. Delete it from the candidate subkey table. If the table of candidate subkey is not empty after analyzing all the remaining pairs, output the subkey remained in table as correct subkey.

For each of the candidate pair in Step 8, the probability that it satisfies the filtering condition is about 2^{-4}. Therefore, for a wrong subkey guess, the probability of its remaining after Step 8 is about $(1 - 2^{-4})^{2^{10}} \approx 2^{-95}$. Then we can expect that after all these filtering, there remains about $2^{68} \times 2^{-95} \approx 2^{-27}$ wrong subkey guess, and only the correct subkey will be output.

The data and time complexities of above attack can be estimated as follows. First of all, we choose 2^{51} structures and the data complexity is $2^{51} \times 2^{12} = 2^{63}$ chosen plaintexts. The time complexity is dominated by Step 7 to Step 8, and each step needs about 2^{78} S-box operations. Therefore, the time complexity of the attack is about $2 \times 2 \times 2^{78} \times \frac{1}{8} \times \frac{1}{20} \approx 2^{72.7}$ 20-round encryptions. According to the complexities of impossible differential attack on 20-round LBlock, we expect that the full 32-round LBlock has enough security margin against this attack.

4.4 Integral Attack

Since LBlock is a 4-bit word oriented cipher, we also consider that integral attack [18] may be one of the most powerful attacks against LBlock. The best integral characteristic found is the 15-round distinguisher. Table 3 illustrates one of the 15-round integral distinguisher in detail, where C denotes a constant word, A denotes an active word and B denotes a balanced word respectively. Note that by changing the position of C in plaintext, we can obtain similar integral distinguishers easily.

Based on the 15-round integral distinguisher, we can mount a key recovery attack up to 20-round LBlock. For simplicity, we first give the integral attack on 18-round LBlock, and the attack procedure is as follows.

1. Choose a set of 2^{60} plaintexts to construct a structure, where only 4-bit word takes a constant and all the other words take all the possible values

Table 3. 15-Round integral distinguisher of LBlock

Rounds	Integral characterisitcs			
0	$AAAC$	$AAAA$	$AAAA$	$AAAA$
1	$AAAC$	$ACAC$	$AAAC$	$AAAA$
2	$CCCC$	$AAAC$	$AAAC$	$ACAC$
3	$ACAC$	$CCCC$	$CCCC$	$AAAC$
4	$CCCC$	$ACCC$	$ACAC$	$CCCC$
5	$ACCC$	$CCCC$	$CCCC$	$ACCC$
6	$CCCC$	$CCCC$	$ACCC$	$CCCC$
7	$CCCC$	$CCAC$	$CCCC$	$CCCC$
8	$CCCC$	$CCCA$	$CCCC$	$CCAC$
9	$CCCC$	$AACC$	$CCCC$	$CCCA$
10	$CCCC$	$AAAC$	$CCCC$	$AACC$
11	$CCAA$	$ACAA$	$CCCC$	$AAAC$
12	$CAAB$	$AAAA$	$CCAA$	$ACAA$
13	$B?AA$	$BBAA$	$CAAB$	$AAAA$
14	$?B?B$	$?B?B$	$B?AA$	$BBAA$
15	$????$	$????$	$?B?B$	$?B?B$

of $\{0,1\}^{60}$. Obtain the corresponding ciphertext after 18-round encryption. Count the number of value $X_{18,6}, X_{18,4}, X_{18,1}, X_{19,6}, X_{19,0}$ occurs, and discard the values which occur even times.

2. Guess corresponding subkeys to decrypt the ciphertexts.
   (a) For every guess of the 8-bit subkey $(K_{18,1}, K_{18,4})$, partially decrypt Round 18 to compute $X_{17,4} = s_4(X_{18,4} \oplus K_{18,4}) \oplus X_{19,6}$ and $X_{17,6} = s_1(X_{18,1} \oplus K_{18,1}) \oplus X_{19,0}$.
   (b) For every guess of the 4-bit subkey $K_{17,4}$, partially decrypt Round 17 to compute $X_{16,4} = s_4(X_{17,4} \oplus K_{17,4}) \oplus X_{18,6}$.
   (c) For every guess of the 4-bit subkey $K_{16,4}$, partially decrypt Round 16 to compute $X_{15,4} = s_4(X_{16,4} \oplus K_{16,4}) \oplus X_{17,6}$.
3. Check if the equation $\bigoplus_{l} X_{15,4} = 0$ is satisfied, where l is the number of plaintexts. If the equation is satisfied, then $X_{15,4}$ is a balance word. Otherwise, guess another subkey and repeat until we get the correct subkey.

The complexity of this attack can be estimated as follows. Step 1 needs about 2^{60} plaintexts which requires 2^{60} encryptions. For the five words counted in Step 1, there are at most 2^{20} values. Therefore, the time complexity of Step 1 to Step 3 are less than $2^{20} \times 2^{16}$ encryptions. For a wrong subkey guess, the probability that equation $\bigoplus_{l} X_{15,4} = 0$ is satisfied is about 2^{-4}. Therefore, to discard all the wrong 16-bit subkey guesses, we need about five plaintext structures. Therefore, the total data and time complexities of this attack are both 5×2^{60}.

Moreover, we can mount an integral attack on 20-round LBlock based on the 15-round integral distinguisher. The attack procedure is similar with the attack

on 18-round LBlock, and we add two additional rounds in the end. Therefore, 12 subkey words need to be guessed and the data and time complexities will increase to about $13 \times 2^{60} \approx 2^{63.7}$.

4.5 Related-Key Attacks

Recently, the combination of related-key [2, 17] and traditional cryptanalysis has become one of the most powerful attacks, and its application to some ciphers has improved the cryptanalytic results significantly [4, 6–8, 13]. Therefore, we have studied the possible related-key differential characteristic of LBlock so as to evaluate the security of LBlock against related-key attacks. In order to get related-key differential characteristic with high probability, we have to control the number of active S-boxes. Therefore, we first choose the output differences of 10 S-boxes (8 S-boxes in round function and 2 S-boxes in key scheduling) in Round i all have hamming weight less than 2. Then we search for the related-key differential before Round i in the decryption direction and after Round i in the encryption direction respectively, and count the total number of active S-boxes. The best related-key differential obtained so far is a 13-round distinguisher with 26 active S-boxes, and its probability is $(2^{-2})^{25} \cdot (2^{-3}) = 2^{-53}$. For the 14-round related-key differential obtained, there are 32 active S-boxes and its probability is less than $(2^{-2})^{31} \cdot (2^{-3}) = 2^{-65}$. Table 4 illustrates the propagation of 14-round related-key differential of LBlock in detail.

Table 4. 14-Round related-key differential characteristic of LBlock

Rounds	ΔX_L	ΔRK	ΔI_S	ΔO_P	ΔX_R
1	01200101	00000000	01200101	20012100	01222121
2	02200001	00000000	02200001	20010100	01200101
3	00000001	02000000	02000001	20000100	02200001
4	00000002	00000000	00000002	00000100	00000001
5	00000000	00000008	00000008	00000200	00000002
6	00000000	00000000	00000000	00000000	00000000
7	00000000	00000000	00000000	00000000	00000000
8	00000000	00000400	00000400	00001000	00000000
9	00001000	00000000	00001000	00000010	00000000
10	00000010	00000000	00000010	00000002	00001000
11	00100002	00020000	00120002	01010100	00000010
12	01011100	00000000	01011100	21002010	00100002
13	31002210	00000000	31002210	20102012	01011100
14	21012013	02000000	23012013	11200212	31002210

5 Performance Evaluation

5.1 Hardware Performance

We implemented LBlock in VHDL and synthesized it on $0.18\mu m$ CMOS technology to check for its hardware complexity. Figure 3 in Appendix III shows the datapath of an parallelization implementation of LBlock, which performs one round in one clock cycle. In this optimized implementation, we use a 64-bit width datapath and implement the eight S-boxes of round function in parallel. Then, to encrypt 64-bit plaintext with an 80-bit key occupies about 1320 GE and requires 32 clock cycles. Table 5 compares the hardware performances of LBlock with other lightweight block ciphers.

Table 5. Comparison of lightweight block cipher implementations

Algorithm	Block Size	Key Size	Area #GE	Speed kbps@100KHz	Logic Process
XTEA	64	128	3490	57.1	0.13 μm
HIGHT	64	128	3048	188.2	0.25 μm
mCrypton	64	128	2500	492.3	0.13 μm
DES	64	56	2300	44.4	0.18 μm
DESXL	64	184	2168	44.4	0.18 μm
KATAN	64	80	1054	25.1	0.13 μm
KTANTAN	64	80	688	25.1	0.13 μm
PRESENT	64	80	1570	200	0.18 μm
LBlock	64	80	1320	200	0.18 μm

Specifically, in the above implementation the area requirement is occupied by flip-flops for storing the key and the data state. To store the 80-bit key requires about 480 GE and to store the 64-bit data state requires two 32-bit registers (denoted as *memleft* and *memright*) which are about 384 GE. For round function F, it is consisted of the following three parts. The KeyAddition is a 32-bit XOR operation which requires about 87 GE. The S-box layer consists of eight 4×4 S-boxes in parallel, which requires about $21.84 \times 8 = 174.8$ GE. The diffusion layer P can be implemented by simple wiring and costs no area. Then in the end of each round, another 32-bit XOR operation of two halves is needed which requires about 87 GE. Furthermore, another two 4×4 S-boxes and a 5-bit XOR operation are needed in key scheduling which require at most $43.7 + 13.5 \approx 57.2$ GE. Moreover, control logic and other counters require about 50 GE. Therefore, the hardware implementation of LBlock requires an estimated area of 1320 GE.

We can give a more compact implementation of LBlock with a serialization design. For example, in the key scheduling we can reuse the 32-bit register and generate each subkey by several operations. Then the area requirement of key

register can be reduced to 212 GE, while additional RAM is needed. Furthermore, the data state in encryption can also reuse the 32-bit key register and the area requirements can be reduced to 192 GE. Then the control logic and other counters need about 70 GE. Therefore, this area-optimized implementation of LBlock only needs about 866.3 GE with additional RAM. Since the register is reused in both key scheduling and encryption, the generation of each round subkey will need 12 clock cycles, and the encryption procedure will need 192 clock cycles. Therefore, to encrypt 64-bit plaintext with 80-bit key needs about 576 clock cycles in total. Table 6 in Appendix II summarizes the area requirement of LBlock in detail.

5.2 Software Implementations

For some resource-constraint environments, such as smart card and sensor networking system, the embedded CPU is usually 8-bit oriented. Therefore, in the design of LBlock, we consider the implementation performance of LBlock not only in hardware environment but also in software platform such as 8-bit microcontroller. The choices of 4-bit word permutation in round function and 8-bit rotation in right half of each round are suitable for both hardware and software platforms. For example, in case of 8-bit oriented software implementation, the eight S-boxes and 4-bit word permutation P in round function can be combined together and realized as four 8-bit lookup tables. Our software implementation of LBlock on 8-bit microcontroller only requires about 3955 clock cycles to encrypt a plaintext block. Hence, LBlock can achieve competitive hardware and software performances compared with other known lightweight block ciphers.

6 Conclusion

In this paper we propose a new lightweight block cipher LBlock, whose block size is 64-bit and key size is 80-bit. Our design goal is to provide cryptography security for resource-constraint environments, e.g. RFID tags and sensor networks etc. Moreover, compared with other lightweight block ciphers, the proposal should achieve better hardware performance and also have good software efficiency on 8-bit microcontroller. Therefore, in the design of LBlock, we employ a variant Feistel structure and the encryption algorithm is 4-bit oriented which can be implemented efficiently in both hardware and software. Furthermore, the round function employs a SP-network, whose confusion layer consists of small 4×4 S-boxes and diffusion layer consists of a simple 4-bit word permutation. All of these components are designed with the consideration of both security and implementation efficiency in mind. Our hardware implementation of LBlock requires about 1320 GE on 0.18 μm technology, which satisfies the regular limitation of 2000 GE in RFID applications. Furthermore, in an area-optimized implementation, LBlock requires only 866.3 GE with additional RAM. We also evaluate the security of LBlock and our cryptanalytic results show that LBlock achieves enough security margin against known attacks. In the end, we strongly encourage the security analysis of LBlock and helpful comments.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (No.60873259), and the Knowledge Innovation Project of The Chinese Academy of Sciences. Moreover, the authors are very grateful to the anonymous referees for their comments and editorial suggestions.

References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39-56. Springer, Heidelberg (2001)
2. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. Journal of Cryptology, 7(4): 229-246. Springer, Heidelberg (1994)
3. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. EUROCRYPT 1999. LNCS, vol. 3027, pp. 12-23. Springer, Heidelberg (1999)
4. Biham, E., Dunkelman, O., Keller, N.: A Related-Key Rectangle Attack on the Full KASUMI. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 443-461. Springer, Heidelberg (2005)
5. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Berlin: Springer-Verlag (1993)
6. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1-18. Springer, Heidelberg (2009)
7. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231-249. Springer, Heidelberg (2009)
8. Biryukov, A., Nikolic, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322-344. Springer, Heidelberg (2010)
9. Bogdanov, A., Knudsen, L.R., Leander, G., Parr, 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)
10. De Canniere, C., Dunkelman, Orr., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Effcient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272-288. Springer, Heidelberg (2009)
11. De Canniere, C., Preneel, B.: Trivium Specifications. eSTREAM submission, available online at http://www.ecrypt.eu.org/stream/triviump3.html.
12. Daemen, J., Rijmen, V.: The Design of Rijndael. Berlin: Springer-Verlag (2002)
13. Dunkelman, O., Keller, N., Shamir, A.: A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony. Faculty of Mathematics and Computer Science Weizmann Institute of Science P.O. Box 26, Rehovot 76100, Israel. (2010)
14. 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)

15. Izadi, M., Sadeghiyan, B., Sadeghian, S., Khanooki, H.: MIBS: A New Lightweight Block Cipher. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 334-348. Springer, Heidelberg (2009)
16. Kim, J., Hong, S., Sung, J., Lee, C., Lee, S.: Impossible differential cryptanalysis for block cipher structure. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82-96. Springer, Heidelberg (2003)
17. Knudsen, L.R.: Cryptanalysis of LOKI91. In: Seberry, J., Zhang,Y. (eds.) Auscrypt 1992. LNCS, vol. 718, pp. 196-208. Springer, Heidelberg (1993)
18. Knudsen, L., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V.(eds.) FSE 2002. LNCS, vol. 2365, pp. 112-127. Springer, Heidelberg (2002)
19. Leander, G., Paar, C., Poschmann, A.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196-210. Springer, Heidelberg (2007)
20. Lim, C.H.: A Revised Version of CRYPTON: CRYPTON v1.0. In: Knudsen, L.(ed.) FSE 1999. LNCS, vol. 1636, pp. 31-45. Spinger, Heidelberg (1999)
21. Lim, C., Korkishko, T.: mCrypton - A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors. In: Song, J., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243-258. Springer, Heidelberg (2006)
22. Matsui, M.: Linear Cryptoanalysis Method for DES Cipher. In: Hellenseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386-397. Springer, Heidelberg (1993)
23. Ojha, S., Kumar, N., Jain, K., Sangeeta.: TWIS - A Lightweight Block Cipher. In: Prakash, A., Gupta, I. (eds.) ICISS 2009. LNCS, vol. 5905, pp. 280-291. Springer, Heidelberg (2009)
24. Ozen, O., Varici, K., Tezcan, C., Kocair, C.: Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In: Boyd, C., Gonzalez Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 90-107. Springer, Heidelberg (2009)
25. Park, J.: Security Analysis of mCrypton Proper to Low-cost Ubiquitous Computing Devices and Applications. International Journal of Communication Systems, 22(8): 959-969. (2009)
26. Parr, C., Poschmann, A., Robshaw, M.J.B.: New Designs in Lightweight Symmetric Encryption. In: Kitsos, P., Zhang, Y. (eds.) RFID Security: Techniques, Protocols and System-on-Chip Design, pp. 349-371. Springer, Heidelberg (2008)
27. Renauld, M., Standaert, F.-X.: Algebraic Side-Channel Attacks. Cryptology ePrint Archive, report 2009/179. http://eprint.iacr.org/2009/279
28. Robshaw, M.J.B.: Searching for Compact Algorithms: CGEN. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 37-49. Springer, Heidelberg (2006)
29. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit Block-cipher CLEFIA (Extended Abstract). FSE 2007. LNCS, vol. 4593, pp. 181-195. Springer, Heidelberg (2007)
30. Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 222-236. Springer, Heidelberg (2006)
31. Su, B., Wu, W., Zhang, L., Li, Y.: Full-Round Differential Attack on TWIS Block Cipher. In: Y. Chung and M. Yung (eds.) WISA 2010. LNCS, vol. 6513, pp.234-242. Springer, Heidelberg (2010)
32. Suzaki, T., Minematsu, K.: Improving the Generalized Feistel. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 19-39. Springer, Heidelberg (2010)
33. Wheeler, D., Needham, R.: TEA, a Tiny Encryption Algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363-366. Springer, Heidelberg (1995)

34. Wheeler, D., Needham, R.: TEA Extensions. October 1997. (Also Correction to XTEA. October 1998) Available via www.ftp.cl.cam.ac.uk/ftp/users/djw3/
35. Yang, L., Wang, M., Qiao, S.: Side Channel Cube Attack on PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 379-391. Springer, Heidelberg (2009)
36. Andrey Bogdanov and Christian Rechberger. Generalized Meet-in-the-Middle Attacks: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In: A. Biryukov, G. Gong and D. R. Stinson (eds.) SAC 2010. LNCS, vol. 6544, pp.228-238. Springer, Heidelberg (2010)

Appendix I: Test Vectors

Test vectors for LBlock are shown in hexadecimal notation as follows.

Plaintext	Key	Ciphertext
00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	c2 18 18 53 08 e7 5b cd
01 23 45 67 89 ab cd ef	01 23 45 67 89 ab cd ef fe dc	4b 71 79 d8 eb ee 0c 26

Appendix II

Table 6. Area requirement of LBlock

Module	Speed Optimized	Area Optimized
64-bit Data Register	384	192
Key Addition	87	87
S-box Layer	174.8	174.8
P Layer	0	0
32-bit XOR	87	87
80-bit Key Register	480	212
S-boxes (Key Scheule)	43.7	30
5-bit Constant XOR	13.5	13.5
Control Logic	50	70
Sum	1320 GE	866.3 GE (with RAM)

Appendix III

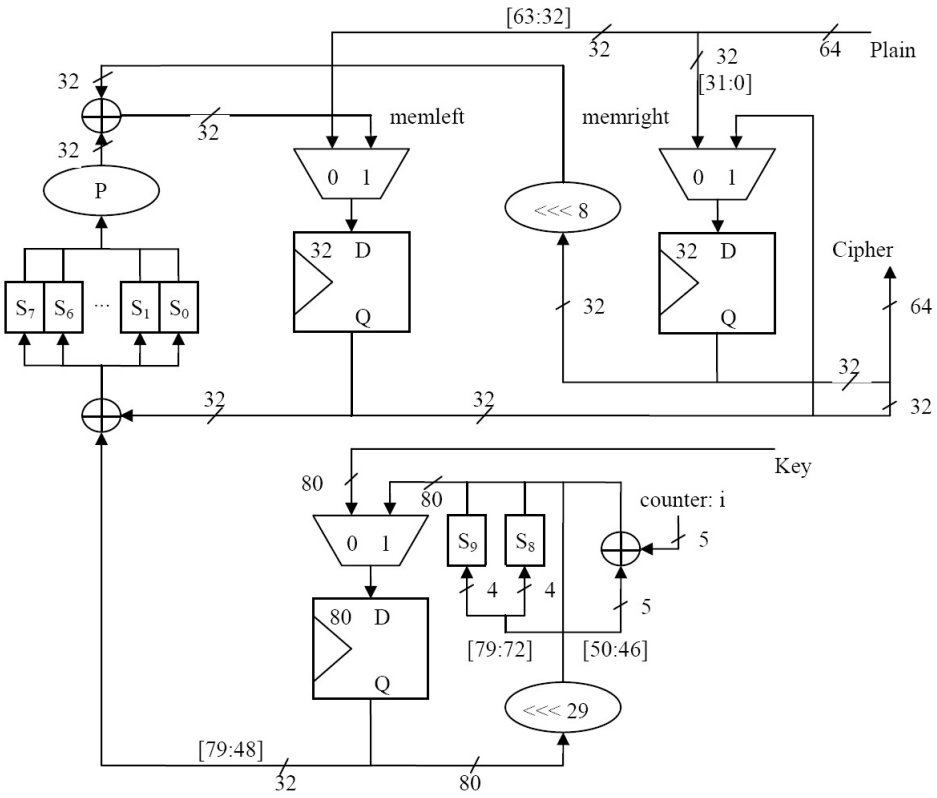

Fig. 3. The datapath of an area-optimized version of LBlock