SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2015, Vol. 58 032105:1–032105:14
doi: 10.1007/s11432-014-5219-0

The DBlock family of block ciphers

WU WenLing[1,2*], ZHANG Lei[1] & YU XiaoLi[1]

[1]*Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing* 100190*, China;*
[2]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190*, China*

Received March 11, 2014; accepted September 23, 2014; published online January 19, 2015

Abstract In this paper, we propose a new family of block ciphers named DBlock. It consists of three variants which are specified as DBlock-128, DBlock-192, and DBlock-256, respectively. DBlock-n has the equal n-bit block length and key length. The structure of DBlock successfully combines the advantages of Feistel and Type-2 generalized Feistel structures together. Also, its design of round function employs different linear transforms operating on various word-sizes, which efficiently improve the diffusion effect. For key schedule of DBlock, it basically employs the same module used in encryption, except the choice of different byte permutations, which can improve its suitability for various implementation environments and also enhance its security against many cryptanalytic techniques. Our preliminary evaluation shows that DBlock can achieve enough security margin against known attacks, and it can also obtain excellent performances on various software and hardware platforms.

Keywords block cipher, structure, round function, key scheduling, security, performance

Citation Wu W L, Zhang L, Yu X L. The DBlock family of block ciphers. Sci China Inf Sci, 2015, 58: 032105(14), doi: 10.1007/s11432-014-5219-0

1 Introduction

Developments of new cryptanalysis and design techniques are the two most important aspects of the study on block cipher. Recently, these techniques have achieved great improvement and many new cryptanalytic ideas and techniques have been proposed against block cipher, which improved the attack results greatly. In the last few years, a number of cryptanalytic results were proposed on the security analysis of advanced encryption standard (AES) [1], such as a series of related-key type attacks on AES proposed in [2–5], and the single-key attack on AES proposed in [6]. Also, research on the combination of several cryptanalysis techniques together has also enriched the development of attack methods, including multiple differential cryptanalysis, zero-correlation linear cryptanalysis, and multidimensional linear attack. On the other hand, research on the design of block ciphers mainly focused on lightweight-type which aimed at being suitable for extremely constrained environment. In recent years, a number of lightweight block ciphers, such as PRESENT [7], HIGHT [8], KTANTAN [9], LBlock [10], LED [11], and Piccolo [12], have been proposed. In their design, the primary concern is the hardware implementation cost, which is different from the design goal of general block cipher, such as AES. In contrast to the propositions of various

* Corresponding author (email: wwl@tca.iscas.ac.cn)

lightweight ciphers, only a few general block ciphers were proposed after the projects of AES and NESSIE competitions, including FOX [13], ARIA [14], and CLEFIA [15].

Therefore, according to the rapid development in cryptanalysis and design techniques of block cipher, we believe that it is a good time to design a new general block cipher that is suitable for various environments and also has stronger security margin against newly developed attacks. Also, recently published cryptanalysis results shed light on the design rationale of block cipher. For example, we believe that the design of key scheduling should be paid more attention and it can be considered to be as important as the encryption design. Simple key scheduling without enough nonlinear transforms may seriously damage the overall security and many recent cryptanalysis results just obtained breakthroughs from this point. Also, new encryption structure and basic module researches are interesting aspects of the design of block cipher. Nowadays, modules of block cipher such as round functions of AES have also been used in the design of hash functions frequently. Therefore, proposition of new block cipher will be a meaningful trial and will benefit the design theory.

In this paper, we propose a new family of block ciphers called DBlock, with three variants denoted as DBlock-128, DBlock-192, and DBlock-256, respectively. According to the analysis in [16], a general meet-in-the-middle attack can always be mounted against any practical block ciphers, and they suggest that when the number of rounds is fixed, it is better to take a key size equal to the block size. Therefore, unlike many other general encryption standards, such as AES and Camellia, we choose equal block size and key size for DBlock family. For example, DBlock-n has the same n-bit block size and key size. Note that the cipher design with large block size has already been studied in Rijndael [1], 3D [17], and Threefish [18], and it is getting increasingly reasonable with the popularization of 64-bit processor and wide applications in the design of hash function.

The design of DBlock family mainly employs a Feitel-type structure with improved diffusion effect, which also combines the advantages of Feistel and Type-2 generalized Feistel structures together. It has similar encryption and decryption structures, and its basic nonlinear module can be small and easily implemented in parallel, which makes DBlock to be implemented efficiently in both software and hardware, and even in some resource constraint environments. Also, compared with typical Type-2 generalized Feistel structure, DBlock can be implemented more efficiently on a 64-bit platform. On the other hand, DBlock employs different linear transforms operating on various word-sizes, including byte permutation, linear function on 32-bit, and structure XOR operation, which efficiently improve the diffusion effect. Therefore, it can achieve full diffusion in less rounds and guarantee more differential and linear active S-boxes. Also, it can reduce the possible number of rounds for impossible differential and integral distinguishers, and hence improve its security against various byte-oriented cryptanalytic techniques.

For key scheduling of DBlock, it basically employs the same modules used in the encryption procedure to reduce hardware implementation cost so as to be suitable for resource constraint environment. The main difference between key scheduling and encryption are the byte position permutation transforms used. The choices of different byte position permutations in each kind of key scheduling are based on the consideration of combining encryption and key scheduling together to evaluate its number of related-key differential active S-boxes so as to guarantee the security of DBlock against related-key type attacks. Also, the byte permutations make DBlock achieve enough diffusion in key scheduling so that there is no simple relation between the master key and round subkeys.

The rest of this paper is organized as follows. Section 2 describes the specification of DBlock. Section 3 explains our design rationale in detail. Then, Sections 4 and 5 present our security analysis and performance evaluation of DBlock, respectively. Finally, Section 6 concludes the paper.

2 Specification of DBlock

DBlock has three variants denoted as DBlock-128, DBlock-192, and DBlock-256, respectively. DBlock-n has the n-bit block size and the n-bit key size. The specification of DBlock consists of three parts: encryption procedure, decryption procedure, and key scheduling algorithm.

2.1 Notations

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

- P: n-bit plaintext;
- C: n-bit ciphertext;
- K: n-bit master key;
- K_i: $n/2$-bit round subkey;
- F_n: round function operates on $n/2$-bit;
- P_n, P_n^*: byte permutations operate on $n/2$-bit;
- s: 8×8 S-box;
- T: nonlinear transformation operates on 32-bit;
- S: S-box layer consists of four s in parallel;
- A: linear transformation operates on 32-bit;
- $\oplus$: bitwise exclusive-OR operation;
- $\lll 8$: 8-bit left rotation operation;
- $||$: concatenation of two binary strings.

2.2 Encryption procedure

The overall structure of DBlock in encryption is a kind of Feistel-type structure. Figure A1 in the Appendix A illustrates the encryption procedure in detail, where F_n denotes the round function of DBlock-n. The number of iterated rounds is 20 for all three variants. Let $P = X_1||X_0$ denote the n-bit plaintext, and $(K_1, K_2, \ldots, K_{20})$ denote the $n/2$-bit subkeys used in each round. Then, the encryption procedure of DBlock can be expressed as follows:

1. For $i = 2, 3, \ldots, 20, 21$, compute

$$X_i = F_n(X_{i-1} \oplus K_{i-1}) \oplus X_{i-2}.$$

2. Output $C = X_{20}||X_{21}$ as the n-bit ciphertext.

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

(1) Round function F_n.

F_n is defined as the combination of two functions P_n and G_n as follows:

$$\begin{aligned} F_n: \{0,1\}^{n/2} &\longrightarrow \{0,1\}^{n/2}, \\ X_i &\longrightarrow U = G_n(P_n(X_i)). \end{aligned}$$

(2) Byte permutation P_n.

P_n is a simple byte position permutation that operates on $n/16$ bytes. The specific P_n used in DBlock-n are defined, respectively, as the following expressions:

$$\begin{aligned} P_{128}: \{0,1\}^{64} &\longrightarrow \{0,1\}^{64}, \\ Y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0) &\longrightarrow Z = (z_7, z_6, z_5, z_4, z_3, z_2, z_1, z_0), \end{aligned}$$

$$\begin{aligned} z_7 = y_6,\ \ z_6 = y_5,\ \ z_5 = y_3,\ \ z_4 = y_1, \\ z_3 = y_4,\ \ z_2 = y_7,\ \ z_1 = y_0,\ \ z_0 = y_2. \end{aligned}$$

$$\begin{aligned} P_{192}: \{0,1\}^{96} &\longrightarrow \{0,1\}^{96}, \\ Y = (y_{11}, y_{10}, ..., y_3, y_2, y_1, y_0) &\longrightarrow Z = (z_{11}, z_{10}, ..., z_3, z_2, z_1, z_0), \end{aligned}$$

$$\begin{aligned} z_{11} = y_9,\ \ z_{10} = y_6,\ \ z_9 = y_4,\ \ z_8 = y_3, \\ z_7 = y_5,\ \ z_6 = y_{11},\ \ z_5 = y_0,\ \ z_4 = y_2, \\ z_3 = y_{10},\ \ z_2 = y_7,\ \ z_1 = y_8,\ \ z_0 = y_1. \end{aligned}$$

$$P_{256} : \{0,1\}^{128} \longrightarrow \{0,1\}^{128},$$
$$Y = (y_{15}, y_{14}, ..., y_3, y_2, y_1, y_0) \longrightarrow Z = (z_{15}, z_{14}, ..., z_3, z_2, z_1, z_0),$$

$$\begin{aligned} z_{15} &= y_{10}, \quad z_{14} = y_5, \quad z_{13} = y_0, \quad z_{12} = y_{15}, \\ z_{11} &= y_6, \quad z_{10} = y_{11}, \quad z_9 = y_{12}, \quad z_8 = y_1, \\ z_7 &= y_{13}, \quad z_6 = y_8, \quad z_5 = y_7, \quad z_4 = y_2, \\ z_3 &= y_4, \quad z_2 = y_9, \quad z_1 = y_{14}, \quad z_0 = y_3. \end{aligned}$$

(3) Function G_n.

G_n is the main nonlinear transform in encryption and it consists of $n/64$ identical 32-bit nonlinear function T in parallel.

$$G_{128} : \{0,1\}^{64} \longrightarrow \{0,1\}^{64},$$
$$Z = (z_7, z_6, z_5, z_4, z_3, z_2, z_1, z_0) \longrightarrow U = (u_7, u_6, u_5, u_4, u_3, u_2, u_1, u_0),$$

$$\begin{aligned} (u_3, u_2, u_1, u_0) &= T(z_3, z_2, z_1, z_0), \\ (u_7, u_6, u_5, u_4) &= T(z_7, z_6, z_5, z_4). \end{aligned}$$

$$G_{192} : \{0,1\}^{96} \longrightarrow \{0,1\}^{96},$$
$$Z = (z_{11}, z_{10}, ..., z_3, z_2, z_1, z_0) \longrightarrow U = (u_{11}, u_{10}, ..., u_3, u_2, u_1, u_0),$$

$$\begin{aligned} (u_3, u_2, u_1, u_0) &= T(z_3, z_2, z_1, z_0), \\ (u_7, u_6, u_5, u_4) &= T(z_7, z_6, z_5, z_4), \\ (u_{11}, u_{10}, u_9, u_8) &= T(z_{11}, z_{10}, z_9, z_8). \end{aligned}$$

$$G_{256} : \{0,1\}^{128} \longrightarrow \{0,1\}^{128},$$
$$Z = (z_{15}, z_{14}, ..., z_3, z_2, z_1, z_0) \longrightarrow U = (u_{15}, u_{14}, ..., u_3, u_2, u_1, u_0),$$

$$\begin{aligned} (u_3, u_2, u_1, u_0) &= T(z_3, z_2, z_1, z_0), \\ (u_7, u_6, u_5, u_4) &= T(z_7, z_6, z_5, z_4), \\ (u_{11}, u_{10}, u_9, u_8) &= T(z_{11}, z_{10}, z_9, z_8), \\ (u_{15}, u_{14}, u_{13}, u_{12}) &= T(z_{15}, z_{14}, z_{13}, z_{12}). \end{aligned}$$

(4) Nonlinear function T.

T is the basic nonlinear module used in every variant of DBlock. It operates on $\{0,1\}^{32}$ and computes as follows, where S denotes a nonlinear layer consisting of four identical 8-bit S-boxes s in parallel and A denotes a 32-bit linear function.

$$T : \{0,1\}^{32} \longrightarrow \{0,1\}^{32},$$
$$(u_3, u_2, u_1, u_0) \longrightarrow A(S(u_3, u_2, u_1, u_0)).$$

(5) S-box s.

The 8-bit S-box s is a nonlinear confusion function operating on $\{0,1\}^8$, and its contents are listed in Table 1.

(6) Linear function A.

A is a simple 32-bit linear diffusion function and it consists of rotation and XOR operations only.

$$A : \{0,1\}^{32} \longrightarrow \{0,1\}^{32},$$
$$X \longrightarrow A(X) = X \oplus (X \lll 8) \oplus (X \lll 10) \oplus (X \lll 18) \oplus (X \lll 26).$$

Table 1 Contents of the S-box used in DBlock

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0	51	36	93	53	d9	4a	fc	58	e4	2e	0d	14	da	9d	91	69
0x1	ef	72	03	c6	15	8d	5c	62	3f	b9	45	70	13	a3	95	6f
0x2	84	db	b8	89	8a	6e	d4	7b	40	dc	9b	0c	50	8e	ee	6a
0x3	88	3b	0f	6b	85	d3	54	a8	20	df	b5	1b	32	7c	56	64
0x4	74	fa	c7	2d	96	17	ae	cd	b4	f5	57	8c	f1	bc	d8	fe
0x5	27	06	e1	a9	1a	0e	5b	08	f4	9f	4b	ed	73	b7	ac	76
0x6	23	ca	16	ba	a7	00	8b	46	41	d5	7e	f2	05	f6	63	67
0x7	61	8f	3d	c8	1c	5a	b0	79	38	81	aa	33	97	e6	2c	01
0x8	22	87	4f	be	24	71	35	9c	b1	ad	c5	1d	80	3e	75	b3
0x9	28	68	2a	a0	bf	2f	b2	c4	ce	19	d7	cf	af	02	a4	a5
0xA	7a	39	d2	04	ab	f7	60	2b	4c	ec	4d	10	90	12	fb	78
0xB	82	4e	37	47	d6	a2	d1	86	b6	c1	e9	dd	a1	f8	55	de
0xC	98	7d	e5	30	fd	e2	cc	3a	ea	d0	0a	29	e8	e3	eb	f0
0xD	9a	5d	3c	21	c0	48	6d	1e	e7	1f	c9	44	34	18	83	f9
0xE	59	5f	42	92	6c	11	a6	52	ff	9e	49	26	07	43	bd	c3
0xF	99	f3	77	0b	5e	cb	09	31	e0	c2	65	7f	25	94	bb	66

2.3 Decryption procedure

Decryption of DBlock is the inverse of the encryption procedure, and it consists of 20 rounds. Let $C = X_{20}||X_{21}$ denote an n-bit ciphertext, and the decryption procedure can be expressed as follows:

1. For $j = 19, 18, \ldots, 1, 0$, compute

$$X_j = F_n(X_{j+1} \oplus K_{j+1}) \oplus X_{j+2};$$

2. Output $P = X_1||X_0$ as the n-bit plaintext.

2.4 Key scheduling

In the key scheduling of DBlock, we need 18 $n/2$-bit constants to generate the round subkeys from master key. The constants are denoted as $Ck_i = (a_{i,m} \ldots a_{i,0})$, where $m = n/16 - 1$, and each constant can be generated as follows:

$$a_{i,j} = (16i + j) \times 7 \ \ \mathbf{mod} \ \ 256, \qquad i = 1, \ldots, 18, \ \ j = 0, \ldots, m.$$

Figure A2 in Appendix A illustrates the key scheduling and the encryption procedure in detail. For the n-bit master key, it is first split into two halves and denoted as $K_2||K_1$. Output K_1 and K_2 are the round subkeys used in the first two rounds, respectively. Then, compute the ith round subkey as follows:

$$K_i = G_n(P_n^*(K_{i-1} \oplus Ck_{i-2})) \oplus K_{i-2}, \qquad i = 3, 4, \ldots, 20,$$

where function G_n is defined exactly the same as in encryption, and P_n^* is a different byte position permutation operating on $n/16$ bytes, which is defined as follows:

$$\begin{aligned} P_{128}^* : \{0,1\}^{64} &\longrightarrow \{0,1\}^{64}, \\ Y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0) &\longrightarrow Z = (z_7, z_6, z_5, z_4, z_3, z_2, z_1, z_0), \end{aligned}$$

$$\begin{aligned} z_7 = y_1, \ \ z_6 = y_0, \ \ z_5 = y_7, \ \ z_4 = y_6, \\ z_3 = y_5, \ \ z_2 = y_4, \ \ z_1 = y_2, \ \ z_0 = y_3. \end{aligned}$$

$$\begin{aligned} P_{192}^* : \{0,1\}^{96} &\longrightarrow \{0,1\}^{96}, \\ Y = (y_{11}, y_{10}, ..., y_3, y_2, y_1, y_0) &\longrightarrow Z = (z_{11}, z_{10}, ..., z_3, z_2, z_1, z_0), \end{aligned}$$

$$z_{11} = y_2, \quad z_{10} = y_{11}, \quad z_9 = y_4, \quad z_8 = y_1,$$
$$z_7 = y_{10}, \quad z_6 = y_9, \quad z_5 = y_0, \quad z_4 = y_7,$$
$$z_3 = y_5, \quad z_2 = y_8, \quad z_1 = y_3, \quad z_0 = y_6.$$

$$P^*_{256} : \{0,1\}^{128} \longrightarrow \{0,1\}^{128},$$
$$Y = (y_{15}, y_{14}, ..., y_3, y_2, y_1, y_0) \longrightarrow Z = (z_{15}, z_{14}, ..., z_3, z_2, z_1, z_0),$$

$$z_{15} = y_{11}, \quad z_{14} = y_7, \quad z_{13} = y_3, \quad z_{12} = y_{15},$$
$$z_{11} = y_6, \quad z_{10} = y_2, \quad z_9 = y_{14}, \quad z_8 = y_{10},$$
$$z_7 = y_1, \quad z_6 = y_5, \quad z_5 = y_9, \quad z_4 = y_{13},$$
$$z_3 = y_8, \quad z_2 = y_{12}, \quad z_1 = y_0, \quad z_0 = y_4.$$

3 Design rationale

3.1 Structure

The overall structure of DBlock can be expressed as Feistel-PF, where P denotes the byte position permutation before round function F. Its design combines the advantages of Feistel and Type-2 generalized Feistel structures together. Therefore, its decryption structure can be exactly the same as encryption and its basic nonlinear module T is small and parallelable, which makes DBlock to be implemented efficiently in both software and hardware, and even in some resource constraint environments.

Compared with the traditional Feistel structure, the diffusion effect of Type-2 generalized Feistel structure is slower and hence more rounds are needed to achieve enough security margin. To avoid this kind of disadvantage, we propose the Feistel-PF structure in DBlock. Its main feature is that we add a novel word permutation P before the round function F and its word size equals to the size of S-box. Also, addition of P will not increase any cost in hardware implementations. Obviously, different choices of P may affect the security property of this kind of structure, and in our design of DBlock all the permutations are chosen carefully through computer searches and security evaluation tests.

Compared with Type-2 generalized Feistel structure, this kind of structure used in DBlock can achieve better diffusion effect. For example, Type-2 generalized Feistel structure can achieve full diffusion in 5 rounds, whereas DBlock structure only needs 4 rounds, where round function uses SP structure and branch number of A is 5. Also, for Type-2 generalized Feistel structure there always exist 9-round impossible differential and integral distinguishers, whereas for DBlock structure the numbers of rounds of the best impossible differential and integral distinguishers are both equal to 8 rounds. Considering the guaranteed number of differential and linear active S-boxes, DBlock structure also has more advantages than the Type-2 generalized Feistel structure.

Compared with the improved Type-2 GFS [19] when $k = 4$, the DBlock structure has no significant improvement. For k up to 6 or 8, the DBlock structure can achieve better diffusion effect and shorter impossible differential and integral distinguisher. For improved 6(8)-partition Type-2 GFS, there always exist 9(10)-round impossible differential and 10(11) integral distinguishers and needs 5(6) rounds to achieve full diffusion, respectively.

3.2 Nonlinear function T

Nonlinear function T is the basic module of DBlock, and it is used repeatedly in both encryption and key schedule. Therefore, its security and implementation performances may determine the final evaluation result of DBlock. The choice of SA structure in function T represents our consideration on performance and security trade-off. For example, the combination of four 8-bit S-boxes in parallel together with 32-bit linear function can be implemented efficiently on both 32/64-bit software processors.

The 8-bit S-box s is designed based on inverse function on $GF(2^8)$. Its main advantages include efficient hardware implementation technique and excellent security properties. For example, s can be implemented

in about 247 GE, and it fulfills the following conditions: completed, not involution, no fix point, best nonlinearity 2^{-4}, best differential probability 2^{-6}, and best algebraic order 7. Also, linear function A achieves best branch number 5 and its hardware area cost is rather small. Also, the combination of s and A can be efficiently implemented by table look-ups on 8-bit microprocessor platforms too.

3.3 Diffusion layer

The diffusion layer of DBlock consists of two parts, namely byte permutation P_n and linear function A. Considering that P_n is a simple byte position permutation, its software and hardware implementations both achieve optimum, which require no additional cost. Linear function A can be combined with S-boxes in round function which are implemented easily as table look-ups in software environments, such as 8-bit and 32-bit microprocessor platforms. Also, the combination of P_n and A can efficiently improve the security of this kind of variant structure used in DBlock. For example, there already are at least 23 active S-boxes for 11-round DBlock-128, 32 active S-boxes for 12-round DBlock-192, and 45 active S-boxes for 13-round DBlock-256.

3.4 Key scheduling

Our main design goals of key schedule include the following three parts. First, its speed is not slower than encryption procedure. Second, its hardware implementation should require as less additional area cost as possible so as to be suitable for resource constraint environment. Last, its security against related-key type attacks is our most important concern, and we also pay attention to the relations between master key and different subkeys.

To reduce the hardware area cost of DBlock, we use the method of generating subkeys when needed in the design of key schedule. Also, we try to reuse the basic modules of encryption, such as nonlinear function T so as to save the hardware cost of DBlock for resource constraint environment. For key schedule of DBlock-n, the main difference between key scheduling and encryption procedure is the byte permutation transforms P_n and P_n^*. How to choose appropriate transforms P_n^* is our main consideration, and they should satisfy the following properties. First, these choices are based on the consideration of combining encryption and key schedule together and to evaluate its number of related-key differential active S-boxes so as to guarantee the security of DBlock against related-key type attacks. Second, byte permutations P_n^* should achieve enough diffusion in key scheduling so that there is no simple relation between master key and round subkeys. Therefore, it will be very hard for the attacker to reduce the attack complexity by using relations between different subkeys. Finally, each kind of key schedule uses the same constant generation method, which can save additional area costs too.

4 Security evaluation

In this section, we provide results on security analysis for DBlock.

4.1 Differential/linear cryptanalysis

For differential cryptanalysis [20] and linear cryptanalysis [21], we adopt the regular method of searching the least number of active S-boxes to evaluate the security of DBlock. We search the guaranteed number of differential (linear) active S-boxes of DBlock by computer program, and the results are listed in Table 2. There are at least 23(32, 45) active S-boxes for 11(12, 13)-round DBlock-128(192, 256), and the best differential probability and linear bias of s are equal to 2^{-6} and 2^{-4}, respectively. This means that there is no useful 13-round differential characteristic and linear approximation for DBlock. As a result, we believe that the full-round DBlock is secure against differential/linear cryptanalysis.

Table 2 Guaranteed number of differential(linear) active S-boxes of DBlock

Rounds	2	3	4	5	6	7	8	9	10	**11**	12	13	14	15	16	17	18	19	20
DBlock-128	1	2	6	10	12	14	16	17	21	23	26	28	30	32	35	38	41	43	44
DBlock-192	1	2	6	10	14	18	22	24	26	29	32	36	40	42	46	48	51	54	58
DBlock-256	1	2	6	10	16	22	27	31	35	37	41	45	51	54	61	64	67	71	75

Table 3 8-Round integral distinguisher of DBlock

Rounds	Integral characteristics for DBlock-128			
0	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$
1	$\mathcal{A}\mathcal{C}\mathcal{C}\mathcal{A}$	$\mathcal{C}\mathcal{A}\mathcal{C}\mathcal{A}$	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$
2	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{A}\mathcal{C}\mathcal{C}\mathcal{A}$	$\mathcal{C}\mathcal{A}\mathcal{C}\mathcal{A}$
3	$\mathcal{A}\mathcal{C}\mathcal{C}\mathcal{A}$	$\mathcal{C}\mathcal{A}\mathcal{C}\mathcal{A}$	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$
4	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$	$\mathcal{A}\mathcal{C}\mathcal{C}\mathcal{A}$	$\mathcal{C}\mathcal{A}\mathcal{C}\mathcal{A}$
5	$\mathcal{B}\mathcal{A}\mathcal{A}\mathcal{B}$	$\mathcal{A}\mathcal{B}\mathcal{A}\mathcal{B}$	$\mathcal{C}\mathcal{C}\mathcal{C}\mathcal{C}$	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$
6	$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$	? ? ? ?	$\mathcal{B}\mathcal{A}\mathcal{A}\mathcal{B}$	$\mathcal{A}\mathcal{B}\mathcal{A}\mathcal{B}$
7	$A(\mathcal{B}\mathcal{B}??)\|\|A(\mathcal{B}\mathcal{B}??)$		$\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}$	? ? ? ?
8	? ? ? ?	? ? ? ?	$A(\mathcal{B}\mathcal{B}??)\|\|A(\mathcal{B}\mathcal{B}??)$	

4.2 Impossible differential cryptanalysis

Impossible differential cryptanalysis [22] may be one of the most powerful attacks against reduced round DBlock. Our evaluation shows that the best impossible differential distinguisher of DBlock-n can all achieve 8 rounds, and they can be expressed as the following forms for DBlock-128, DBlock-192, and DBlock-256, respectively:

$$(\underbrace{0\ldots0}_{8}, 0000000\alpha) \overset{8r}{\nrightarrow} (0000000\beta, \underbrace{0\ldots0}_{8}), \tag{1}$$

$$(\underbrace{0\ldots0}_{12}, 00000000\alpha000) \overset{8r}{\nrightarrow} (00000000\beta000, \underbrace{0\ldots0}_{12}), \tag{2}$$

$$(\underbrace{0\ldots0}_{16}, 000000000000000\alpha) \overset{8r}{\nrightarrow} (000000000000000\beta, \underbrace{0\ldots0}_{16}), \tag{3}$$

where α and β represent non-zero differences in input difference and output difference, respectively.

We evaluate the security of DBlock against impossible differential attack using the 8-round distinguishers above. By adding two additional rounds before and after the distinguisher, we can present impossible differential attack on 12-round for DBlock.

4.3 Integral attack

Because DBlock is a word-oriented block cipher, we also consider integral attack [23]. We use the general inside-out method to construct an integral distinguisher, and the best integral characteristic found is an 8-round integral distinguisher consisting of a normal 5-round propagation in the encryption direction combined with a 3-round propagation in the decryption direction for three versions of DBlock. Table 3 shows one of this kind of 8-round integral characteristics in detail, and by changing the position of $\mathcal{C}$ in plaintext we can obtain similar integral distinguishers easily.

In integral attack, $\mathcal{C}$ denotes a constant byte, $\mathcal{A}$ denotes an active byte, $\mathcal{B}$ denotes a balance byte, and $A(\mathcal{B}\mathcal{B}??)||A(\mathcal{B}\mathcal{B}??)$ denotes a value whose output after transform $A^{-1}||A^{-1}$ is balanced at $j(=2,3,6,7)$th byte, respectively. Based on this kind of 8-round integral distinguisher, we can mount an integral attack on 11-round DBlock.

4.4 Related-key attack

Recently, the combination of related-key [24] and traditional cryptanalysis has become one of the most powerful attacks, and its application to some block ciphers has improved the cryptanalytic results significantly. Therefore, we have searched for the possible related-key differential characteristic of DBlock so as to evaluate its security against related-key type attacks. Based on test results in related-key environment, there are at least 22(32, 45) active S-boxes for 13-round DBlock-128(192, 256), respectively. Therefore, full-round DBlock is secure against related-key type attacks.

4.5 Biclique cryptanalysis

Biclique cryptanalysis [6] is a new powerful technique proposed against full-round AES in Asiacrypt 2011, and then it has also been applied to many other block ciphers successfully. Although most of these attacks had only a constant factor improvement over exhaustive search, biclique technique is still believed to be an important aspect for the security evaluation of new block cipher designs. Therefore, in this subsection we analyze the security of DBlock family block ciphers against biclique cryptanalysis.

For DBlock-128, we have found a biclique attack on up to 9 rounds. We use the independent biclique technique and construct an eight-dimensional biclique for 2 rounds of DBlock-128. Hence, we can split the key space into 2^{112} groups of keys, and each group contains 2^{16} keys which are indexed as elements of a $2^8 \times 2^8$ matrix: $K[i, j]$. Then, for each group of keys we can construct a 2-round biclique, which covers the 8th and 9th rounds. Based on this initial structure, we can mount a meet-in-the-middle attack for the remaining rounds and choose the matching point as the rightmost 32 bits of X_4. Overall, the attack requires about 2^{32} chosen ciphertexts, and the time complexity can be estimated by the number of sbox computations, which is about $2^{126.93}$. Similar biclique attacks can also be applied to 9-round DBlock-192 and 10-round DBlock-256, and the time complexities are $2^{190.82}$ and $2^{254.97}$, respectively.

4.6 Other attacks

We also consider other attacks, including slide attack, higher order differential attack, truncated differential attack, boomerang attack, rectangle attack, meet-in-the-middle attack, and algebraic attack. Although the details of the evaluations for those attacks are omitted because of the page limitation, consequently, we consider that those attacks do not threaten the full-round DBlock.

5 Performance evaluation

In the design of DBlock, we pay much consideration to its performance on both hardware and 8/32/64-bit software platforms. In this section, we will introduce the optimized implementation of DBlock on various platforms. Since the basic modules of the three variants are similar, we will only describe the case of DBlock-128.

5.1 Hardware implementations

For the 0.13-μm CMOS application specific integrated circuit (ASIC) library, one gate equivalent (GE) is equivalent to the area of a two-way NAND with the lowest drive strength.

5.1.1 *Normal implementation*

For normal hardware implementation of DBlock-128, we can use 128-bit width datapath and realize two nonlinear functions T in parallel. Then, we can perform one round encryption in 1 clock cycle, and in key schedule each round subkey can be generated in 1 clock cycle. Therefore, encryption of 128-bit plaintext with 128-bit key occupies about 6552 GE and requires 20 clock cycles.

Specifically, in this kind of implementation, the area requirement is mainly occupied by flip-flops for storing the key and data state. Storage of 128-bit key requires about $128 \times 4 = 512$ GE, and storage of 128-bit data state also requires 512 GE. For the round function F in encryption, it consists of three parts.

KeyAddition is a 64-bit XOR operation which requires about $64 \times 1.5 = 96$ GE. Byte permutation P can be realized by simple wiring which costs no area. Each nonlinear function T consists of four 8×8 S-boxes in parallel and a linear diffusion function A, which requires about $247 \times 4 + 3 \times 32 \times 1.5 = 1132$ GE. At last, in the end of each round another 64-bit XOR operation of two halves is needed, which requires about $64 \times 1.5 = 96$ GE. Also, except the above modules of encryption, another 5-bit counter is needed in key schedule, which requires about 20 GE. Also, additional control logics are needed which require about 596 GE. Therefore, the normal hardware implementation of DBlock-128 will require about 6552 GE.

5.1.2 *Optimized implementations*

For area optimized implementation, we can give a more compact implementation of DBlock-128 by reusing the registers and basic module T. Its key schedule can reuse the 64-bit register in encryption and module T only needs to be realized once. Therefore, in this kind of area optimized implementation of DBlock-128, it only needs about 2640 GE. Because the register is reused in both key schedule and encryption, the generation of each round subkey will need 2 clock cycles and encryption procedure will need 40 clock cycles. Therefore, encryption of 128-bit plaintext with 128-bit key will need 80 clock cycles in total.

To improve the throughput of DBlock-128, we can encrypt 2 rounds in 1 clock cycle with the cost of 70% increment of area requirement. Therefore, in this kind of speed optimized implementation it needs about 12,000 GE. Then, in key schedule two round subkeys can be generated in 1 clock cycle and the encryption procedure needs 10 clock cycles. Therefore, encryption of a 128-bit plaintext with 128-bit key requires only 10 clock cycles in total.

5.2 Software implementations

5.2.1 8-*bit platform*

Considering that the main module used in both encryption and key scheduling is the nonlinear function T, we will mainly describe the optimized implementation of T on 8-bit platform in this section. Denote the input of T as (x_3, x_2, x_1, x_0), and then the output $(y_3, y_2, y_1, y_0) = A(S(x_3, x_2, x_1, x_0))$ can be computed as follows.

First, define two new 8×8 S-boxes based on the origin S-box of DBlock:

$$s_1(x) = s(x) \oplus s(x) \ll 2,$$
$$s_2(x) = s(x) \lll 2.$$

Then, compute the intermediate values by eight table look-ups:

$$a_1 = s_1(x_0), \quad a_2 = s_2(x_0), \quad a_3 = s_1(x_1), \quad a_4 = s_2(x_1),$$
$$a_5 = s_1(x_2), \quad a_6 = s_2(x_2), \quad a_7 = s_1(x_3), \quad a_8 = s_2(x_3).$$

At last, the output of T is computed as follows:

$$a_1 = a_1 \oplus a_4, \quad a_3 = a_3 \oplus a_6,$$
$$a_5 = a_5 \oplus a_8, \quad a_7 = a_7 \oplus a_2,$$

$$y_3 = a_4 \oplus a_5 \oplus a_7,$$
$$y_2 = a_2 \oplus a_3 \oplus a_5,$$
$$y_1 = a_1 \oplus a_3 \oplus a_8,$$
$$y_0 = a_1 \oplus a_6 \oplus a_7.$$

Therefore, implementation of function T on 8-bit platform needs 8 table look-ups and 12 XOR operations. In each round encryption of DBlock-128, there are two functions of T, one round subkey addition and one XOR operation on 8 bytes, and hence it will need about 16 table look-ups and 40 XOR operations altogether on 8-bit platform.

Table 4 Software performances comparison of AES and DBlock

Algorithm	Key length		
	128	192	256
AES	16 c/B	19 c/B	22 c/B
DBlock	22.2 c/B	18.3 c/B	15.4 c/B

5.2.2 32-*bit platform*

For the implementation of DBlock-128 on a 32-bit platform, we can use the normal technique of realizing function T by table look-ups. According to the definitions of S-box s and permutation A, we can construct four 8×32 tables as follows:

$$T_3(x) = A(s(x), 0, 0, 0),$$
$$T_2(x) = A(0, s(x), 0, 0),$$
$$T_1(x) = A(0, 0, s(x), 0),$$
$$T_0(x) = A(0, 0, 0, s(x)).$$

Therefore, for each round encryption of DBlock-128, it only needs 8 table look-ups and 10 XOR operations altogether. To store these tables, it needs about 4KB memory. Also, considering that there are simple relations between these tables, we can store only one 8×32 table and use rotation operations to optimize the implementation for memory constraint environment.

5.2.3 64-*bit platform*

Similarly, for the implementation of DBlock-128 on a 64-bit platform, we can use eight 8×64 tables to realize the round function.

Therefore, for each round encryption of DBlock-128, it only needs 8 table look-ups and 9 XOR operations altogether. To store these tables, it needs about 16KB memory. Also, based on the relations between these tables, we can store only four 8×64 tables and use rotation operations to optimize the implementation for memory constraint environment.

5.2.4 *Performance comparison*

Finally, we give a simple software implementation performance comparison of DBlock family with the well-known encryption standard AES. The test benchmark used is an Intel(R) Core(TM) i5-3210M CPU clocked at 2.50 GHz. For comparison, we use the optimized table-based implementation for both DBlock and AES, with the same kind of 8×32 table look-ups. We evaluate their encryption speed in cycles per byte and the results are listed in Table 4. Table B1 in Appendix B shows test vectors for DBlock.

6 Conclusion

In this paper, we propose a new family of block cipher named DBlock. Its structure combines the advantages of Feistel and Type-2 generalized Feistel structures together. Therefore, its decryption structure can be exactly the same as encryption and its basic nonlinear module T is small and parallelable, which can be implemented efficiently in both software and hardware. DBlock also employs different linear transforms operating on various word-sizes, which can improve its diffusion effect significantly. For key scheduling of DBlock, it mainly reuses the basic modules of encryption, and choices of different byte position permutations can enhance the security of DBlock with ignorable cost. We have evaluated the security of DBlock against known cryptanalytic techniques carefully and our analysis results show that DBlock can achieve enough immunity against known attacks. Of course, we also strongly encourage any security analysis of DBlock and helpful comments.

Acknowledgements

This work was supported by the National Basic Research Program of China (973 Program) (Grant No. 2013CB338002) and National Natural Science Foundation of China (Grant Nos. 61272476, 61232009 and 61202420).

References

1 Daemen J, Rijmen V. The design of Rijndael. In: Information Security and Cryptography. Berlin: Springer-Verlag, 2002
2 Biryukov A, Khovratovich D. Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui M, ed. Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, 2009. 5912: 1–18
3 Biryukov A, Khovratovich D, Nikolic I. Distinguisher and related-key attack on the full AES-256. In: Halevi S, ed. Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, CA, USA, 2009. 5677: 231–249
4 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. Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, 2010. 6110: 322–344
5 Zhang W, Wu W, Zhang L, et al. Improved related-Key impossible differential attacks on reduced-round AES-192. In: Biham E, Youssef A M, eds. Proceedings of the 13th International Workshop on Selected Areas in Cryptography-SAC, Montreal, Canada, 2006. 4356: 15–27
6 Bogdanov A, Khovratovich D, Rechberger C. Biclique Cryptanalysis of the Full AES. In: Lee D H, Wang X, eds. Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, 2011. 7073: 344–371
7 Bogdanov A, Knudsen L, Leander G, et al. PRESENT: An ultra-lightweight block cipher. In: Paillier P, Verbauwhede I, eds. Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 2007. 4727: 450–466
8 Hong D, Sung J, Hong S, et al. HIGHT: A new block cipher suitable for low-resource device. In: Goubin L, Matsui M, eds. Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems, Yokohama, Japan, 2006. 4249: 46–59
9 Canniere C, Dunkelman O, Knezevic M. KATAN and KTANTAN-A family of small and efficient hardware-oriented block ciphers. In: Clavier C, Gaj K, eds. Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems, Lausanne, Switzerland, 2009. 5747: 272–288
10 Wu W, Zhang L. LBlock: A lightweight block cipher. In: Lopez J, Tsudik G, eds. Proceedings of the 9th International Conference on Applied Cryptography and Network Security, Nerja, Spain, 2011. 6715: 327-344
11 Guo J, Peyrin T, Poschmann A, et al. The led block cipher. In: Preneel B, Takagi T, eds. Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems, Nara, Japan, 2011. 6917: 326–341
12 Shibutani K, Isobe T, Hiwatari H, et al. Piccolo: An ultra-lightweight blockcipher. In: Preneel B, Takagi T, eds. Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems, Nara, Japan, 2011. 6917: 342–357
13 Junod P, Vaudenay S. FOX: A new family of block ciphers. In: Handschuh H, Hasan M A, eds. Proceedings of the 11th International Workshop on Selected Areas in Cryptography, Waterloo, Canada, 2004. 3357: 131–146
14 Kwon D, Kim J, Park S, et al. New block cipher: ARIA. In: Lim J I, Lee D H, eds. Proceedings of the 6th International Conference on Information Security and Cryptology, Seoul, Korea, 2003. 2971: 432–445
15 Shirai T, Shibutani K, Akishita T, et al. The 128-bit blockcipher CLEFIA (Extended Abstract). In: Biryukov A, ed. Proceedings of the 14th International Workshop on Fast Software Encryption, Luxembourg, Luxembourg, 2007. 4593: 181–195
16 Huang J, Lai X. What is the effective key length for a block cipher: An attack on every block cipher. http://eprint.iacr.org/2012/677.pdf (2012)
17 Nakahara Jr J. 3D: A three-dimensional block cipher. In: Franklin M K, Hui L C K, Wong D S, eds. Proceedings of the 7th International Conference on Cryptology and Network Security, Hong-Kong, China, 2008. 5339: 252–267
18 Ferguson N, Lucks S, Schneier B, et al. The skein hash function family. Submission to NIST, Round 3, 2010, http://www.skein-hash.info
19 Suzaki T, Minematsu K. Improving the generalized Feistel. In: Hong S, Iwata T, eds. Proceedings of the 17th International Workshop Fast Software Encryption, Seoul, Korea, 2010. 6147: 19–39
20 Biham E, Shamir A. Differential Cryptanalysis of the Data Encryption Standard. Berlin: Springer-Verlag, 1993
21 Matsui M. Linear cryptoanalysis method for DES cipher. In: Helleseth T, ed. Proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, 1993. 765: 386–397
22 Biham E, Biryukov A, Shamir A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. J

Crypt, 2005, 18: 291–311
23 Knudsen L, Wagner D. Integral cryptanalysis. In: Daemen J, Rijmen V, eds. Proceedings of the 9th International Workshop on Fast Software Encryption, Leuven, Belgium, 2002. 2365: 112–127
24 Biham E. New types of cryptanalytic attacks using related keys. J Crypt, 1994, 7: 229–246

Appendix A Figures

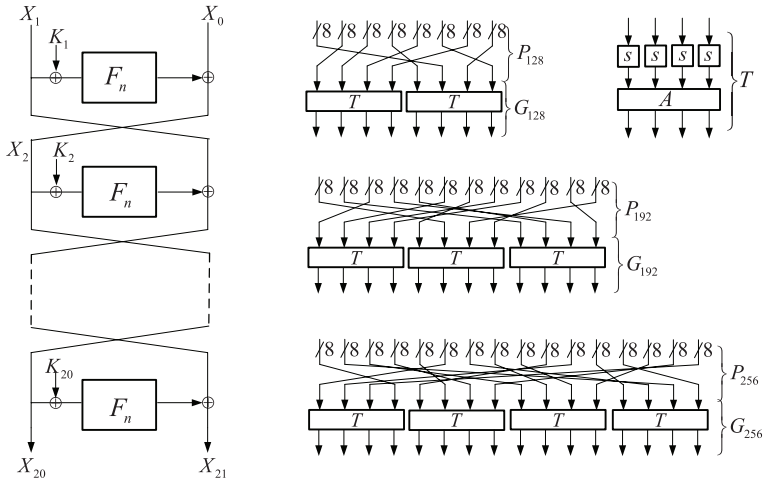

Figure A1 Encryption procedure and round function F_n.

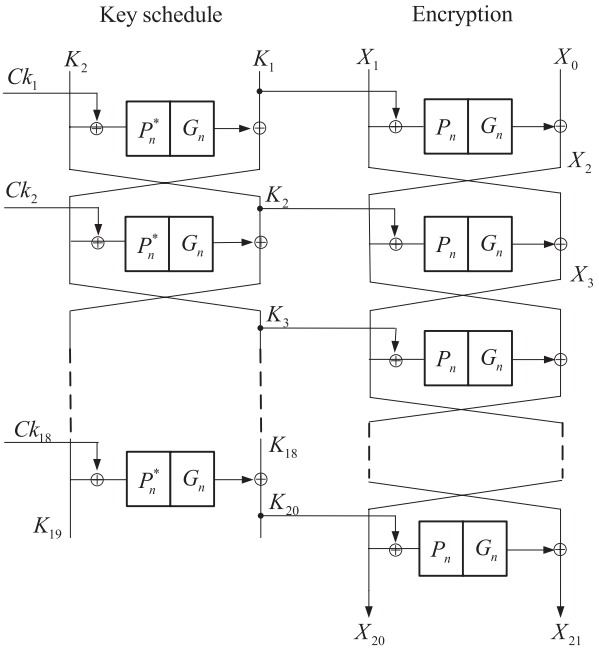

Figure A2 Key scheduling and encryption of DBlock-n.

Appendix B Test vectors

Test vectors for DBlock are shown in hexadecimal notation as follows (Table B1).

Table B1 Test vectors for DBlock

	Plaintext	Key	Ciphertext
DBlock-128	01 23 45 67 89 AB CD EF	01 23 45 67 89 AB CD EF	BE D2 EB 8E E0 DA 0C 55
	FE DC BA 98 76 54 32 10	FE DC BA 98 76 54 32 10	D5 78 0B 6D 94 06 BE CA
DBlock-192	01 23 45 67 89 AB CD EF	01 23 45 67 89 AB CD EF	C3 65 B5 67 B6 B8 FF CB
	FE DC BA 98 76 54 32 10	FE DC BA 98 76 54 32 10	E8 97 68 6B 42 C8 B1 0B
	01 23 45 67 89 AB CD EF	01 23 45 67 89 AB CD EF	45 62 2B 60 BE 9F E8 FE
DBlock-256	01 23 45 67 89 AB CD EF	01 23 45 67 89 AB CD EF	9F EB 4B 91 63 79 91 BD
	FE DC BA 98 76 54 32 10	FE DC BA 98 76 54 32 10	A1 82 98 09 FF F4 B5 DE
	01 23 45 67 89 AB CD EF	01 23 45 67 89 AB CD EF	6D 88 DE 79 56 96 77 88
	FE DC BA 98 76 54 32 10	FE DC BA 98 76 54 32 10	E3 A6 98 1A DC D1 85 92