

LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors

Deukjo Hong¹(✉), Jung-Keun Lee¹, Dong-Chan Kim¹, Daesung Kwon¹,
Kwon Ho Ryu¹, and Dong-Geon Lee²

¹ Attached Institute of ETRI, Seoul, Korea

{hongdj,jklee,dongchan,ds.kwon,jude}@ensec.re.kr

² Information Security & IoT Laboratory, Pusan National University,
Busan, South Korea
guneez@pusan.ac.kr

Abstract. We propose a new block cipher LEA, which has 128-bit block size and 128, 192, or 256-bit key size. It provides a high-speed software encryption on general-purpose processors. Our experiments show that LEA is faster than AES on Intel, AMD, ARM, and ColdFire platforms. LEA can be also implemented to have tiny code size. Its hardware implementation has a competitive throughput per area. It is secure against all the existing attacks on block ciphers.

Keywords: LEA · Block cipher · Fast encryption

1 Introduction

CPUs and operating systems are continuously developing, and many computing devices work much better than before, with such powerful resources. For example, smart portable devices like smart phones and tablet PCs do not only replace mobile phones but also allow to enjoy various cloud computing and social network services. With those applications, the amount of the private data which people create for their business and life will be significantly increasing. Another example is a smart meter, which is a basic unit of an advanced metering infrastructure in a smart grid, recording consumption of electric energy, gathering data for remote reporting, and communicating with the utility for monitoring and billing purpose. For the convenience of management, smart meters are often implemented to perform tasks in software with small CPUs [18].

Those data mentioned in the above examples are usually important information which must be protected from various threats in networks. It implies that the wide use of software applications significantly causes the necessity of cryptographic systems on software platforms. With this consideration, we have been interested in a software encryption. Software encryptions are easier to deploy and more cost-effective than hardware ones in many cases. In particular, when a new encryption service is required for already deployed computing environments, a software encryption is more suitable than a hardware one.

A block cipher is one of the most widely used cryptographic primitives. It is applied to data encryption, message authentication, random bit generation, message hashing and so on. Presumably, the most widely used block cipher in the world is AES [27] which has been established as various international standards. AES shows good performance figures on most software and hardware platforms and is generally considered to be secure after surviving about 15 years of comprehensive cryptanalysis though some weaknesses have been found. Since AES, many block ciphers have been designed for hardware lightweight implementation. Some of them were standardized as ISO lightweight cryptography (ISO/IEC 29192-2). The main feature of the lightweight block ciphers is the efficient hardware implementation with low resource. In order to achieve that goal, most of them use simple structures with small block sizes and large number of rounds. However, those design approaches usually lead to low performance, and is far from our consideration for software encryption. Consequently, we have designed a new block cipher providing a fast encryption on common software platforms.

1.1 Contribution

We propose a new block cipher LEA. It has the block size of 128 bits and the key size of 128, 192, or 256 bits. We denote the algorithms with 128-bit, 192-bit, and 256-bit keys by LEA-128, LEA-192, and LEA-256, respectively. The structure of LEA has the following features.

1. LEA consists of only ARX (modular Addition, bitwise Rotation, and bitwise XOR) operations for 32-bit words. Those operations are well-supported and fast in many 32-bit and 64-bit platforms. Moreover, we suppose that the usage of 32-bit and 64-bit processors will grow rapidly compared to 8-bit or 16-bit ones.
2. The ARX operations contribute to the encryption and key schedule procedures in efficient and parallel way. Our arrangement of operations does not only lead to fast software encryption and small size code, but also strong resistance against the attacks using the properties of a particular operation.
3. The last round function of LEA is the same as other round functions, while many block cipher including DES and AES have special last round functions which are somewhat different from other round functions. This is for the encryption speed in both software and hardware because we think the block cipher encryption is more frequently used than decryption.
4. The key schedule of LEA has a simple structure without any interleaving between 32-bit key words. It is good for the efficiency, and does not cause any weakness.

Security. Our goal for the security of LEA is to get the resistance against all the existing attacks for block ciphers and to provide enough security margin. To achieve this goal, we firstly found the minimum number R of rounds for LEA to

resist against all the known cryptanalytic techniques for each key size. Then we determined the number of rounds of LEA as around $3R/2$ to prepare for the unknown attacks to appear in future.

Efficiency. LEA provides a fast encryption on many platforms. Our experiments measuring the speed for one-block encryption on the platforms of Intel, AMD, ARM and ColdFire show that even a C level implementation of LEA is very fast. It implies that the evaluation of LEA encryption requires the light overhead to CPUs. Note that the light overhead can lead to the low power consumption which is useful for the devices based on batteries. The optimized implementation of LEA-128 for one-block encryption is faster than those of AES-128 publicly reported [25, 47], on our test platforms. To objectivity, we used the announced facts for comparison instead of implementing AES.

LEA can be implemented with SIMD operations supported by Intel and AMD CPUs such that it encrypts 4 blocks simultaneously. It is useful for the highly fast encryption with ECB or CTR modes under a powerful environment like a server-based computing. Our experiments on Intel Core 2 Quad Q6600 and Intel Core i7-800 show that the speed of the 4-block SIMD implementation of LEA-128 is about 2 times and 1.7 times faster than the best records of the multi-block encryption codes of AES-128 [35], respectively.

We also found that LEA is implemented with a small code-size. The small-size implementation is useful in a memory-limited environment. LEA-128 is implemented with less than 600 and 750 bytes on the platforms of ARM926EJ-S and ColdFire MCF5213, respectively, while AES-128 is known to be implemented with around 2,400 and 960 bytes on the platforms of ARM7TDMI and ColdFire v2, respectively.

Comparison with Other Ciphers. We compare LEA to other ciphers in order to explain why it is meaningful to propose this new block cipher.

- **AES.** AES was designed based on design and analysis techniques by 2000, and cryptanalysis of block ciphers has been continuously researched and developed. Recent several attacks have pointed out some weaknesses for AES. In [11], Biryukov et al. presented a chosen-key distinguisher for full 14-round AES-256 and converted it to a key-recovery attack for a weak key class with the complexities of 2^{131} time and 2^{65} memory. In [10], Biryukov and Khovratovich presented related-key boomerang attacks on full 14-round AES-256 with $2^{99.5}$ time and data complexities and AES-192 with 2^{176} time and 2^{123} data complexities. In [14], Bogdanov et al. used biclique techniques to make key recovery attacks on full AES-128, AES-192, and AES-256 with time complexities $2^{126.1}$, $2^{189.7}$, and $2^{254.4}$, respectively. LEA is designed based on the latest design and analysis techniques and we checked that LEA is secure and has sufficient security margin against all the existing attacks.

Furthermore, as we already mentioned, LEA provides better software encryptions in speed and size on many platforms than AES.

- **Block ciphers with ARX structure.** TEA [56] and XTEA [46] are Feistel block ciphers with simple round function and key schedule. Their encryption speeds are not fast because they have the block length of 64 bits shorter than LEA and 64 rounds more than LEA. Additionally, there are full-round attacks [37, 58] on TEA and XXTEA [57], which is the third algorithm of TEA family.

At the AES competition, RC6 [49] was regarded as faster than Rijndael [21], which is the AES winner, on many software platforms. However, parallelism offered by modern CPUs is not exploited well with RC6, and the performance of recent implementation of AES exceeds that of RC6.

HIGHT [31] is a lightweight block cipher based on 8-bit ARX operations. So, it is not suitable for fast encryption on 32-bit CPUs. Recently, full-round attacks on HIGHT have been published [32, 41].

Hash functions often adopt the ARX structure for the high performance on various platforms, similarly to our design goal [2, 26]. Most of them have block ciphers as a component for building compression and hash functions. They are even secure against attacks for block ciphers. However, hash functions and block ciphers are different in the usage. In particular, most block ciphers in the hash functions have much larger block and key sizes than those usually required for the security and application of block ciphers.

Recently, NSA published two block cipher families SPECK and SIMON [3]. SPECK is a typical ARX cipher and SIMON consists of ANDs, rotations, and XORs. They have various parameters. The algorithms with 128-bit block are comparable with LEA. The Performance of LEA is faster than SIMON in both 32- and 64-bit processors. Since SPECK uses 64-bit addition with 128-bit block, its performance exceeds that of LEA only in 64-bit processors but LEA is more suitable for most 32-bit processors.

- **Lightweight block ciphers.** Many lightweight block ciphers like HIGHT [31], PRESENT [13], LED [30], and Piccolo [51] have short block size and large number of rounds and their software encryptions are usually not fast. Although [43] provides fast bitslice implementation of PRESENT and Piccolo, SIMD implementation of LEA is faster than them. Furthermore, a short block size is not proper for encrypting huge data because some modes of operation can allow security leakage like a ciphertext-matching attack.

KLEIN [28] is designed to be faster than AES on 8-bit and 16-bit platforms, while our targets are 32-bit and 64-bit platforms. CLEFIA [52] has the same block and key size as AES and the performance of its software encryption is close to that of AES on AMD Athlon TM Processor 4000+. However, as far as we know, it does not claim higher software performance than AES. Recently, PRINCE [16] was proposed as a low-latency block cipher which has good performance in software and hardware implementations, but its security goal is somewhat different from that for the general-purpose block ciphers.

- **Stream ciphers.** Several stream ciphers such as Salsa20 [4] are based on ARX operations, but we think the block cipher is not totally comparable to the stream cipher because they do not always have the same applications.

1.2 Organization

The remaining part is organized as follows. Section 2 describes the specification of LEA. In Sect. 3, we introduce design principles. In Sect. 4, we present the security analysis results for existing cryptanalytic techniques. In Sect. 5, we explain the implementation results. Section 6 is the conclusion of our paper.

2 Specification of LEA

LEA is a block cipher with 128-bit block. Key size is 128-bit, 192-bit, and 256-bit. The number of rounds is 24 for 128-bit keys, 28 for 192-bit keys, and 32 for 256-bit keys. In Sect. 2.1, we introduce notations which are often used in this paper. We explain how the key schedule generates round keys from the master key in Sect. 2.3. We explain how the encryption procedure converts a plaintext to a ciphertext in Sect. 2.4. We omit the description of the decryption procedure because it is simply considered as the inverse of the encryption procedure.

2.1 Notations

- P : a 128-bit plaintext, consisting of four 32-bit words $P = (P[0], P[1], P[2], P[3])$
- C : a 128-bit ciphertext, consisting of four 32-bit words $C = (C[0], C[1], C[2], C[3])$
- X_i : a 128-bit intermediate value (an input of i -th round in the encryption function), consisting of four 32-bit words $X_i = (X_i[0], X_i[1], X_i[2], X_i[3])$
- $\text{Len}(x)$: the bit-length of a string x
- K : a master key. It is denoted as a concatenation of 32-bit words. $K = (K[0], K[1], K[2], K[3])$ when $\text{Len}(K) = 128$; $K = (K[0], K[1], \dots, K[5])$ when $\text{Len}(K) = 192$; $K = (K[0], K[1], \dots, K[7])$ when $\text{Len}(K) = 256$
- r : the number of rounds. $r = 24$ when $\text{Len}(K) = 128$; $r = 28$ when $\text{Len}(K) = 192$; $r = 32$ when $\text{Len}(K) = 256$
- RK : the concatenation of all round keys, defined by $RK = (RK_0, RK_1, \dots, RK_{r-1})$ where RK_i is the 192-bit round key for the i -th round. Each RK_i consists of six 32-bit words $RK_i = (RK_i[0], RK_i[1], \dots, RK_i[5])$
- $x \oplus y$: XOR (eXclusive OR) of bit strings x and y with same length
- $x \boxplus y$: Addition modulo 2^{32} of 32-bit strings x and y
- $\text{ROL}_i(x)$: the i -bit left rotation on a 32-bit value x
- $\text{ROR}_i(x)$: the i -bit right rotation on a 32-bit value x

2.2 State Representation

Let $a[0], a[1], \dots$, be representation of arrays of bytes. The bytes and the bit ordering within bytes are derived from the 128-bit input sequence $input_0, input_1, \dots$ as follows:

$$a[i] = \{input_{8i}, input_{8i+1}, \dots, input_{8i+7}\}.$$

All the operations in the LEA algorithm are 32-bit-word-oriented. The 128-bit plaintext P of LEA is represented as an array of four 32-bit words $P[0], P[1], P[2], P[3]$. Each $P[i]$ is taken for the input bytes $a[0], a[1], \dots, a[15]$ as follows:

$$P[i] = a[4i + 3] \parallel a[4i + 2] \parallel a[4i + 1] \parallel a[4i] \quad \text{for } 0 \leq i \leq 3.$$

The key K of LEA is also represented as an array of 32-bit words $K[0], K[1], \dots$, and taken for the input bytes in the same way. Table 1 shows how bits and bytes in the word indexed by 0 are numbered.

Table 1. Representations for words, bytes, and bits

Input bit sequence	24	...	31	16	...	23	8	...	15	0	...	7
Word number	0											
Byte number	3			2			1			0		
Bit numbers in word	31	...										0

2.3 Key Schedule

The key schedule generates a sequence of 192-bit round keys RK_i as follows.

Constants. The key schedule uses several constants for generating round keys, which are defined as

$$\begin{aligned} \delta[0] &= 0xc3efe9db, & \delta[1] &= 0x44626b02, \\ \delta[2] &= 0x79e27c8a, & \delta[3] &= 0x78df30ec, \\ \delta[4] &= 0x715ea49e, & \delta[5] &= 0xc785da0a, \\ \delta[6] &= 0xe04ef22a, & \delta[7] &= 0xe5c40957. \end{aligned}$$

They are obtained from hexadecimal expression of $\sqrt{766995}$, where 76, 69, and 95 are ASCII codes of ‘L,’ ‘E,’ and ‘A.’

Key Schedule with a 128-Bit Key. Let $K = (K[0], K[1], K[2], K[3])$ be a 128-bit key. We set $T[i] = K[i]$ for $0 \leq i < 4$. Round key $RK_i = (RK_i[0], RK_i[1], \dots, RK_i[5])$ for $0 \leq i < 24$ are produced through the following relations:

$$\begin{aligned} T[0] &\leftarrow \text{ROL}_1(T[0] \oplus \text{ROL}_i(\delta[i \bmod 4])), \\ T[1] &\leftarrow \text{ROL}_3(T[1] \oplus \text{ROL}_{i+1}(\delta[i \bmod 4])), \\ T[2] &\leftarrow \text{ROL}_6(T[2] \oplus \text{ROL}_{i+2}(\delta[i \bmod 4])), \\ T[3] &\leftarrow \text{ROL}_{11}(T[3] \oplus \text{ROL}_{i+3}(\delta[i \bmod 4])), \\ RK_i &\leftarrow (T[0], T[1], T[2], T[1], T[3], T[1]). \end{aligned}$$

Key Schedule with a 192-Bit Key. Let $K = (K[0], K[1], \dots, K[5])$ be a 192-bit key. We set $T[i] = K[i]$ for $0 \leq i < 6$. Round key $RK_i = (RK_i[0], RK_i[1], \dots, RK_i[5])$ for $0 \leq i < 28$ are produced through the following relations:

$$\begin{aligned} T[0] &\leftarrow \text{ROL}_1(T[0] \oplus \text{ROL}_i(\delta[i \bmod 6])), \\ T[1] &\leftarrow \text{ROL}_3(T[1] \oplus \text{ROL}_{i+1}(\delta[i \bmod 6])), \\ T[2] &\leftarrow \text{ROL}_6(T[2] \oplus \text{ROL}_{i+2}(\delta[i \bmod 6])), \\ T[3] &\leftarrow \text{ROL}_{11}(T[3] \oplus \text{ROL}_{i+3}(\delta[i \bmod 6])), \\ T[4] &\leftarrow \text{ROL}_{13}(T[4] \oplus \text{ROL}_{i+4}(\delta[i \bmod 6])), \\ T[5] &\leftarrow \text{ROL}_{17}(T[5] \oplus \text{ROL}_{i+5}(\delta[i \bmod 6])), \\ RK_i &\leftarrow (T[0], T[1], T[2], T[3], T[4], T[5]). \end{aligned}$$

Key Schedule with a 256-Bit Key. Let $K = (K[0], K[1], \dots, K[7])$ be a 256-bit key. We set $T[i] = K[i]$ for $0 \leq i < 8$. Round key $RK_i = (RK_i[0], RK_i[1], \dots, RK_i[5])$ for $0 \leq i < 32$ are produced through the following relations:

$$\begin{aligned} T[6i \bmod 8] &\leftarrow \text{ROL}_1(T[6i \bmod 8] \oplus \text{ROL}_i(\delta[i \bmod 8])), \\ T[6i + 1 \bmod 8] &\leftarrow \text{ROL}_3(T[6i + 1 \bmod 8] \oplus \text{ROL}_{i+1}(\delta[i \bmod 8])), \\ T[6i + 2 \bmod 8] &\leftarrow \text{ROL}_6(T[6i + 2 \bmod 8] \oplus \text{ROL}_{i+2}(\delta[i \bmod 8])), \\ T[6i + 3 \bmod 8] &\leftarrow \text{ROL}_{11}(T[6i + 3 \bmod 8] \oplus \text{ROL}_{i+3}(\delta[i \bmod 8])), \\ T[6i + 4 \bmod 8] &\leftarrow \text{ROL}_{13}(T[6i + 4 \bmod 8] \oplus \text{ROL}_{i+4}(\delta[i \bmod 8])), \\ T[6i + 5 \bmod 8] &\leftarrow \text{ROL}_{17}(T[6i + 5 \bmod 8] \oplus \text{ROL}_{i+5}(\delta[i \bmod 8])), \\ RK_i &\leftarrow (T[6i \bmod 8], T[6i + 1 \bmod 8], T[6i + 2 \bmod 8], \\ &\quad T[6i + 3 \bmod 8], T[6i + 4 \bmod 8], T[6i + 5 \bmod 8]). \end{aligned}$$

2.4 Encryption Procedure

The encryption procedure of LEA consists of 24 rounds for 128-bit keys, 28 rounds for 192-bit keys, and 32 rounds for 256-bit keys. For r rounds, it encrypts a 128-bit plaintext $P = (P[0], P[1], P[2], P[3])$ to a 128-bit ciphertext $C = (C[0], C[1], C[2], C[3])$.

Initialization. Set the 128-bit intermediate value X_0 to the plaintext P . Run the key schedule to generate r round keys.

Iterating Rounds. The 128-bit output $X_{i+1} = (X_{i+1}[0], \dots, X_{i+1}[3])$ of the i th round for $0 \leq i \leq r - 1$ is computed as

$$\begin{aligned} X_{i+1}[0] &\leftarrow \text{ROL}_9((X_i[0] \oplus RK_i[0]) \oplus (X_i[1] \oplus RK_i[1])), \\ X_{i+1}[1] &\leftarrow \text{ROR}_5((X_i[1] \oplus RK_i[2]) \oplus (X_i[2] \oplus RK_i[3])), \\ X_{i+1}[2] &\leftarrow \text{ROR}_3((X_i[2] \oplus RK_i[4]) \oplus (X_i[3] \oplus RK_i[5])), \\ X_{i+1}[3] &\leftarrow X_i[0]. \end{aligned}$$

Finalization. The ciphertext C is produced from the finally obtained X_r after round iteration in the following way:

$$C[0] \leftarrow X_r[0], C[1] \leftarrow X_r[1], C[2] \leftarrow X_r[2], \text{ and } C[3] \leftarrow X_r[3].$$

3 Design Principles

We explain the design principles for LEA (Fig. 1).

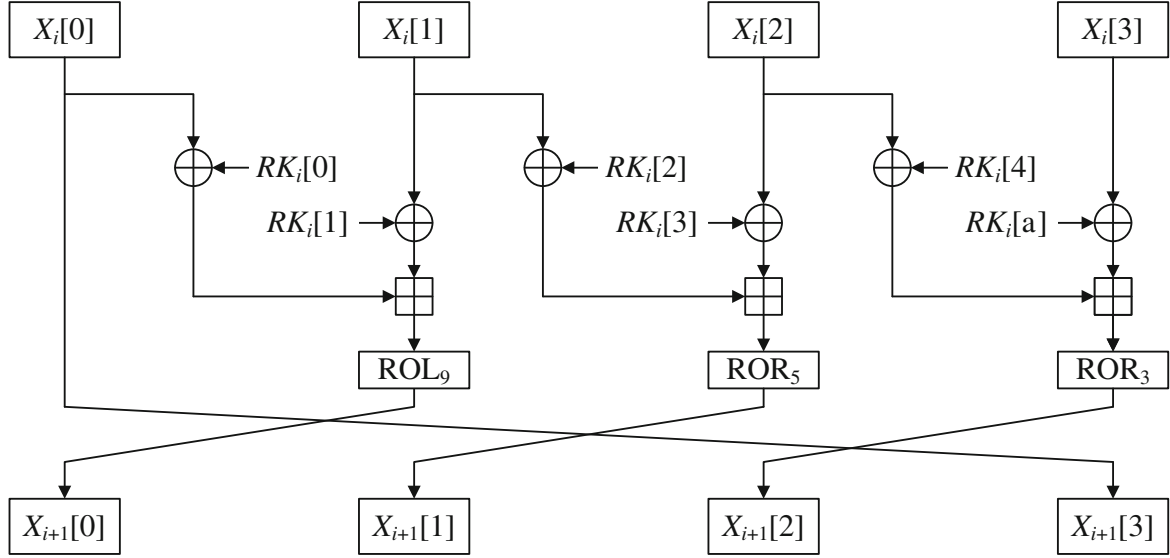


Fig. 1. i th round function

Efficient round structure with 32-bit ARX operations. The round function of LEA consists of ARX operations. Especially, we used 32-bit ARX operations instead of 8-bit ones because 32-bit operations are more popular than 8-bit ones and we think that most processors will be developed to support 32-bit operations even in resource-constrained devices. It has just three internal computation modules including two key XORs, one addition, and one bitwise rotation. We adopt the addition modulo 2^{32} as a nonlinear function with two 32-bit inputs and one 32-bit output¹. Round key XORs are used for randomizing the inputs of the nonlinear functions, the bitwise rotations and the word-wise swap are used for diffusion. The simple and efficient structure of LEA provides both tiny-size code and high-speed code. In spite of its simplicity, it has nice nonlinearity and diffusion effect to give a proper number of rounds for good performance.

Encryption is more useful than decryption. Unexpectedly, there are not many modes of operation which need the decryption function. For example, ISO/IEC 9797-1, ISO/IEC 10116, and ISO/IEC 19772 specify 6 message authentication

¹ On the other hand, round key XORs and rotations work as nonlinear functions for the adversary using add-differences.

modes, 5 encryption modes, and 6 authenticated-encryption modes of block ciphers, respectively. However, only ECB, CBC, and OCB modes need both encryption and decryption functions². It implies that the block cipher encryption is more widely and frequently used than the block cipher decryption. With this consideration, we did not care for the balance of the speed between encryption and decryption. Note that most block ciphers usually have the special last round different from other rounds for efficiency in the implementation of decryption, while the last round of LEA is not special but has the same structure as the other rounds for efficiency of the encryption-only modes. Nevertheless, the decryption speed of LEA is still competitive with most block ciphers.

Choice of rotations. We chose the rotations in encryption procedure such that it has the strong diffusion property. Firstly, for the parameters (a, b, c) , we set the round function with input X_i , output X_{i+1} , and round key RK_i as follows.

$$\begin{aligned} X_{i+1}[0] &\leftarrow \text{ROL}_a((X_i[0] \oplus RK_i[0]) \boxplus (X_i[1] \oplus RK_i[1])), \\ X_{i+1}[1] &\leftarrow \text{ROL}_b((X_i[1] \oplus RK_i[2]) \boxplus (X_i[2] \oplus RK_i[3])), \\ X_{i+1}[2] &\leftarrow \text{ROL}_c((X_i[2] \oplus RK_i[4]) \boxplus (X_i[3] \oplus RK_i[5])), \\ X_{i+1}[3] &\leftarrow X_i[0]. \end{aligned}$$

Then, we linearized the LEA encryption algorithm by replacing the additions with XORs, and searched the XOR differential characteristics (XDCs) of the linearized structure for all possible $\binom{32}{3}$ candidates of (a, b, c) . Note that $\text{ROL}_b = \text{ROR}_{32-b}$ and $\text{ROL}_c = \text{ROR}_{32-c}$. One of our searching strategies is to start from a middle round with low Hamming weight of differences. As a result, we found that for each candidate of (a, b, c) there exists a 11-round XDC whose probability is not lower than 2^{-128} . The probability is estimated under the assumption that every addition is independent. Note that this assumption is not stronger than any other block ciphers because each addition can be regarded as a nonlinear function with two 32-bit inputs and a 32-bit output and because XORing subkeys with the inputs of nonlinear functions is the most popular way to combine key materials to encryption body. We found 32 candidates of (a, b, c) which have 12-round XDCs with the probability of 2^{-128} or 2^{-129} as best ones. We optimized these characteristics such that both of the first and last rounds are not linearized. We chose $(9, 27, 29)$ because it made only differential characteristics with the probability 2^{-128} as best ones, and because the number of such characteristics is fewer than any other candidates.

Additionally, we considered short characteristics for the boomerang attack, and found that the maximum number of rounds having differential characteristic with the probability greater than 2^{-32} is 7 over all (a, b, c) , and so does it for the case $(a, b, c) = (9, 27, 29)$.

As a different approach for the same goal, we can also regard the linearized rounds as a linear code. So, we tried to get good differential characteristics by

² We leave the fifth mode, ‘Encrypt-then-MAC’ in [33] out of the discussion because it uses general notions of encryption and MAC.

applying Canteaut-Chabaud method [17] to search code words with minimum weight code words, but we could not find better differential characteristics than the first approach.

Simple key schedule. We adopt a very simple structure for the key schedule. It does not mix the words of the key and has no avalanche effect in key bits at all. Nevertheless, our security analysis show that it protects LEA from the attacks such as slide attack [12], related-key attack [5], related-key boomerang attack [10, 11], biclique attack [14], rotational attack [38] and so on. The simplicity of the key schedule provides efficiency in small-size hardware and software implementations.

4 Security Analysis

We analyzed the security of LEA for existing cryptanalytic techniques by searching, constructing, or exploiting various characteristics such as differential and linear trails. For each attack, firstly, we found the maximum number N of rounds where there exists an available characteristic, and then constructed the best N -round characteristic. We determined the number of rounds making the algorithm secure against each attack, considering the difference propagation of the round function and the arrangement of the round key words, as follows.

1. If the characteristic is N -round and holds with the probability between 0 and 1, then the secure number of rounds is $N + 3$ for 128-bit keys, $N + 4$ for 192-bit keys, and $N + 5$ for 256-bit keys.
2. If the characteristic is N -round and holds with the probability 0 or 1, then the secure number of rounds is $N + 4$ for 128-bit keys, $N + 5$ for 192-bit keys, and $N + 6$ for 256-bit keys.

COSIC made an evaluation report for LEA, too, independently of us [20]. We will explain some of their security analysis results. The whole main analysis results are summarized at Table 2. We also discuss other attacks not listed in Table 2.

Table 2. Security of LEA against several main attacks

Attack type	Round # of Characteristic	Probability of Characteristic	Secure # of rounds		
			LEA-128	LEA-192	LEA-256
Differential [9]	11	$p = 2^{-98}$	14	15	16
Truncated Differential [39]	11	$p = 2^{-91.9}$	14	15	16
Linear [44]	11	$ p - 1/2 = 2^{-62}$	14	15	16
Zero Correlation [15]	7	$ p - 1/2 = 0$	11	12	13
Boomerang [54]	14	$p^2 q^2 = 2^{-108}$	17	18	19
Impossible Differential [6]	10	$p = 0$	14	15	16
Integral [40]	6	$p = 1$	10	11	12
Differential-Linear [8]	14	$ p - 1/2 < 2^{-57}$	17	18	19

4.1 Differential Attack

As we mentioned in Sect. 3, the probability of the best 12-round differential characteristic which we have found is estimated at most as 2^{-128} . Since it is not available for the attack, we searched 11-round differential characteristic with the same way; firstly find the XOR-linearized differential characteristics with high probabilities and then optimize it by removing the linearity in the differential paths of the first and the last rounds. As a result, the best found ones of 11-round differential characteristics have the probability 2^{-98} and the following form:

- Input difference: $80000234 \alpha 0402214 \beta 0401205 \gamma 0400281$, where $\alpha \in \{4, c\}$, $\beta \in \{4, c\}$, and $\gamma = \beta \oplus 1$,
- Output difference: $\eta 800000a \ 88aaa00a \ 220202\zeta 0 \ 00200050$, where $\eta \in \{4, c\}$ and $\zeta \in \{2, 6\}$.

We can apply one of these characteristics to 11 rounds from Round 0 to Round 10, and attack 12 rounds for 128-bit keys. This attack recovers 96 bits of the round key RK_{11} in the last round, Round 11 with very high signal-to-noise ratio, and requires around 2^{100} plaintexts, 2^{84} encryptions, and the memory for 2^{76} bytes. Extending it to 13-round attack is not successful because

- If one applies the 11-round characteristic to the first 11 rounds from Round 0 to Round 10 and tries to recover partial bits of RK_{12} , the round key of Round 12, he will be in trouble with the poor filtering and it leads to the bad signal-to-noise ratio.
- If one applies the 11-round characteristic to Round 1 to Round 11 and tries to recover partial bits of RK_0 , the round key of Round 0 and RK_{12} , the round key of Round 12, he will face too much guessed key bits or too weak filtering to attack.

We consider the possibility of 13-round attack for 192-bit keys and 14-round attack for 256-bit keys, respectively.

Using a set of many differential characteristics with relatively high probabilities instead of a best one, we can increase the probability from 2^{-98} to $2^{-91.9}$. This is a kind of truncated differential characteristic [39], which can be used for reducing some of complexities for the above differential attack, but not be helpful for increasing the number of the attacked rounds. Analyses with other types of differences [20] have been tried but not found any critical weaknesses.

4.2 Linear Attack

A linear approximation has the following form:

$$\Gamma_P \cdot P \oplus \Gamma_C \cdot C = \Gamma_{RK} \cdot RK, \quad (1)$$

where RK is a vector composed of all round keys. We denote the probability that (1) is satisfied, by p , and let $\varepsilon = p - 1/2$. ε is called the bias of (1). A linear attack using a linear approximation has the data complexity of $O(\varepsilon^{-2})$.

It is not easy to find a good linear approximation for long rounds of LEA. Wallén’s work [55] shows that in the masks of a linear approximation for modular additions the absolute value of the bias tends to decrease as the highest nonzero bit of the masks is close to the most significant. The combination of the bitwise rotations in LEA encryption significantly disturbs the appearance of linear approximations with good biases. We searched the linear approximations in such a way that the propagation of linear masks is suppressed as strong as possible. Consequently, we found 10-round linear approximation with $\varepsilon = 2^{-46}$ and 11-round linear approximation with $\varepsilon = 2^{-63}$. We can use Matsui’s algorithm 1 and the 11-round linear approximation to get 1-bit information about round keys for 11 rounds with $O(2^{126})$ known plaintexts, and we can use Matsui’s algorithm 2 and the 10-round linear approximation to make a 11-round key recovery attack with $O(2^{92})$ known plaintexts.

4.3 Zero Correlation Attack

Recently, the attacks using zero correlation approximations have been introduced [15], which is a counter part of the impossible differential attack in linear cryptanalysis. The best key recovery attacks in single-key setting based on zero correlation approximations have been made for TEA and XTEA. Since LEA has the use of ARX operations in common with TEA and XTEA, one may suspect the vulnerability of LEA against zero correlation attack. However, we found that a 7-round zero correlation approximation is constructed from 3-round forward and 4-round backward approximations, and it is difficult to construct much longer zero correlation approximations than 7 rounds. Based on the 7-round zero correlation approximations, we consider the possibility of 9-round attack for 128-bit keys, 10-round attack for 192-bit keys, and 11-round attack for 256-bit keys, respectively.

4.4 Boomerang Attack

The best differential probability for 7 rounds is 2^{-27} . The best 7-round one has the following differences of input and output.

- Input difference: 80000014 80400014 80400004 80400080,
- Output difference: 00001200 28000200 80800800 00000008.

We construct a 14-round boomerang characteristic from the best 7-round differential characteristic. There are some round-skip techniques maximizing the number of rounds of the boomerang characteristic [10, 24], but they do not work for LEA. It is the best one which we have found ever. For 128-bit keys, we can use it to make an attack on at most 15 rounds with $2^{116.3}$ plaintexts. We could not find a proper attack on 16 rounds due to increased data complexity and worsened filtering. The amplified boomerang [36] or rectangle attacks [7] do not seem to improve our attacks significantly. We consider the possibility of 16-round attack for 192-bit keys and 17-round attack for 256-bit keys, respectively.

4.5 Impossible Differential Attack

Impossible differential attack [6] uses differential characteristics with probability of 0. They are usually constructed from miss-in-the-middle combination with forward and backward truncated differential characteristics with probability of 1. For LEA, the best impossible differential characteristics are 10 rounds, constructed with 6-round forward and 4-round backward truncated differential characteristics with probability of 1, which is reported in [20].

For 128-bit keys, we can use the 10-round impossible differential characteristics to make a 11-round attack to derive a partial information of the last round key. one may make a 12-round attack by using a set of specially chosen plaintexts or constructing a key-recovering process. We consider the possibility of 13-round attack for 192-bit keys, and 14-round attack for 256-bit keys, respectively.

4.6 Integral Attack

Integral attack [40] for LEA uses a 6-round integral characteristic, which is reported at [20]. A 6-round integral characteristic of LEA is reported at [20]. It shows that if the 3-th word $P[3]$ of the plaintext P is active, which takes all 32-bit values for one time, and other words of P are constants, then the least significant bit of the 1-th word $X[1]$ of the output X after 6 rounds is ADD-balanced. For 128-bit keys, we can use the 6-round integral characteristic to make a 9-round attack to derive a partial information of round keys. Adding rounds to the characteristic at top is impossible because it requires a code book of all plaintexts. We consider the possibility of 10-round attack for 192-bit keys, and 11-round attack for 256-bit keys, respectively. We suppose higher order differential characteristic [39] is also constructed for 6 rounds at most.

4.7 Differential-Linear Attack

Differential-linear attack [8] uses a combined characteristic from short-round differential characteristics and linear approximations. A $(r_1 + r_2)$ -round differential-linear characteristic based on one r_1 -round differential characteristic with the probability p_d and two r_2 -round linear approximations with same masks and the probability $p_l = 1/2 + \varepsilon$ holds with the probability $p = 1/2 + 2p_d\varepsilon^2$. Our analysis for differential and linear attacks on LEA implies that the available differential-linear characteristics for LEA can be constructed up to 14 rounds and that the biasour searching program can find 14-round differential-linear characteristics with the bias at most 2^{-57} . However, this reasoning is based on the best results which we can find for differential and linear trails, and so we suppose that the actually found differential-linear characteristics be much shorter than 14 rounds or have the bias whose absolute value is significantly smaller than 2^{-57} .

4.8 Attacks Using Weakness of Key Schedule

Slide attack [12] uses a self-similarity in the block cipher. The key schedule of LEA obstructs it by adding the rotated constants to the key materials.

For instance, when the key size is 128 bits, $\text{ROL}_i(\delta[i \bmod 4])$ is added for the leftmost 32 bits of the i -th round key $RK_i[0]$. Although only several 32-bit constants are used, rotations depending on i make the effects of adding different round constants for every round. Therefore, there is no self-similarity which can be exploited for any attacks on LEA.

Related-key differential attack [34] and related-key boomerang attack [10, 11] is the most popular ones among the attacks using related keys [5]. In the similar way to differential cryptanalysis, we searched how many rounds there exists a key difference having differential characteristics with the probability $> 2^{-128}$ up to. The best related-key differential characteristics which we found ever are 11-round one for 128-bit keys, and 12-round one for 192 and 256-bit keys. However, those characteristics cannot be used straightforwardly for any attacks because they hold with only small part of the key space.

Bogdanov et al. [14] has introduced the key recovery attacks in single-key setting, based on biclique techniques with two attack approaches. The first approach is to use the bicliques constructed from independent related-key differentials and to search the right key with partial computations based on precomputation. We checked that it is hard to construct such bicliques for more than one round of LEA for the key sizes of 128 and 192 bits and for more than two round for the key size of 256 bits, because LEA uses 192-bit round keys and all key materials are wasted in one round for 128 and 192-bit keys and in two rounds for 256-bit keys, and because all additions in the same round are active within two rounds in backward direction for any key difference. Therefore, the time complexity of the key recovery attacks based on the first approach would have a negligible difference with that of exhaustive search. The second approach is to use the bicliques constructed from interleaving related-key differential trails and to apply a basic meet-in-the-middle technique for key recovery. Such bicliques would not be constructed for more than 8 rounds because the propagation of the difference inserted at key is fast in the encryption of LEA in spite of its simple structure. Furthermore, the basic meet-in-the-middle technique of the second approach is applicable to only short rounds. So, the attack based on second approach can work for only small reduced variants with much less rounds than recommended.

4.9 Other Attacks

Recently, some kinds of meet-in-the-middle attacks have made impressive cryptanalytic results for block ciphers and hash functions. We checked that meet-in-the-middle attack techniques are not applicable to LEA very well. A basic meet-in-the-middle attack [23] is disturbed since there is no separation of long rounds. The meet-in-the-middle pseudo-preimage attack [1, 50] does not work for even half rounds. The partial-matching and initial-structure techniques are not efficient in LEA.

Rotational cryptanalysis [38] is attractively available on ARX-based structures. We examined the resistance of LEA against rotational cryptanalysis for the single-key model and the related-key model in which two keys form a rotational pair. We found that key XORs in the encryption procedure and constant XORs

in the key schedule prevent rotational characteristics from being constructed for long rounds.

Algebraic attack [19] forms an overdefined system of equations derived from the block cipher. Several algorithms are proposed for solving it, but they fail to find a right solution for existing block ciphers. We think they hardly work for LEA, too.

4.10 Security Margin

We have studied various existing cryptanalytic techniques for block ciphers in order to analyze the security of LEA. Although some characteristics we mentioned can be somewhat upgraded by new technologies, it is unlikely to find a new attack to improve significantly the results in Table 2, as long as we did not miss critical weakness of LEA. We determined the number of rounds for LEA-128 based on the above security analysis such that the security margin to the whole rounds ratio is greater than 30 %. For LEA-192 and LEA-256, we added 4 and 8 rounds, respectively to the rounds of LEA-128, considering the difference of key schedules and security criteria.

5 Implementation

5.1 Software Implementation

We have implemented LEA on various 32-bit and 64-bit software platforms. We have focused on LEA-128 since the speed decreases almost in proportion to the number of rounds.

On ARM platforms, we can implement LEA without register-spilling and most of the bit rotations can be processed without costing any clock cycle thanks to the barrel shifter. Thus we get remarkably high throughput compared to other block ciphers both in encryption and decryption.

On Intel/AMD platforms, we can also implement LEA without register-spilling and, due to the highly parallel structure of LEA round function, we also get high encryption speed. Moreover, by utilizing SIMD(Single Instruction Multiple Data) instructions inherent in most of recent Intel/AMD platforms, we can get even higher throughput for parallel modes of LEA.

On ARM and ColdFire platforms, we have measured the compactness of LEA. Since the round function of LEA consists of a small number ARX operations without S-box, the code size of LEA on these platforms is quite small compared to other block ciphers with the same block size.

We have also estimated the efficiency of LEA on some 8-bit platforms and confirmed that LEA has sound performance on these platforms.

ARM platforms. ARM processors are the most widely used 32-bit embedded processors. They support rotate, multiple load/store instructions as well as most arithmetic and logical ones. Comparison with the speed-optimized implementation of AES on comparable platforms is given in Table 3.

Table 4 shows the comparison with the code-size-optimized implementation of AES.

Intel and AMD Platforms. Most of recent Intel/AMD CPUs have 3 pipelines. Since LEA consists of 24 rounds and each round can be expressed as a sequence of 16 instructions, the minimal cycle cost of LEA encryption is expected to be around 128. Comparison with 32-bit implementation of AES is given in Table 5.

Table 3. Speed of LEA-128 and AES-128 on ARM platform

Algorithm	Speed (cycles/byte)	Platform
LEA-128	20.06	ARM926EJ-S
AES-128 [47]	34.00	StrongARM SA-1110

Table 4. Code size of LEA-128 and AES-128 on ARM platform

Algorithm	ROM size (bytes)	RAM size (bytes)	Speed (cycles/byte)	Platform
LEA-128	590	32	326.94	ARM926EJ-S
AES-128 [22]	2,164	304	460.50	ARM7TDMI

Decryption is slower than encryption since decryption is processed rather serially. We note that AES is faster than all other well-known block ciphers with similar block and key size on these platforms.

Most of recent Intel/AMD processors support SIMD extensions at least up to SSE2. Thus, basic 32-bit operations like XOR, ADD, SHIFT can be performed very efficiently in parallel. Moreover, the latency and throughput of SIMD instructions are close to those of corresponding 32-bit-wise instructions on recent processors. Since LEA is described as a combination of XOR, ADD, and ROTATE, it is straightforward to implement parallel modes of LEA using SSE2 to process 4 or 8 blocks simultaneously.

Comparison with SIMD implementations of AES (not using AES instruction set) is given in Table 6.

ColdFire platforms. ColdFire processors are 32-bit microprocessors targeted towards embedded systems. LEA shows lower performance here than on ARM platforms since load/store and rotate operation are performed less efficiently: They do not support rotate, multiple load/store instructions and the shift instruction can shift only by up to 8 bits. We have implemented speed-optimized and size-optimized LEA on MCF5213. Comparison with implementation of AES on comparable platform is given in Table 7. We note that LEA runs faster than hardware-accelerated AES.

8-bit and 16-bit Platforms. Though LEA is designed to achieve high performance in 32-bit platforms. We have also analyzed the performance of LEA on Advanced

Table 5. Speed (cycles/byte) of LEA-128 and AES-128 on 32-bit Intel/AMD platforms

Platform	LEA-128		AES-128
	Encryption	Decryption	Encryption
Intel Core 2 Quad Q6600	9.29	14.83	12.20 [25]
Intel Core i5-2500	9.29	14.52	11.35 [25]
AMD Phenom II X4 965	8.85	14.50	10.35 [25]
AMD Opteron 6176 SE	8.55	14.05	N/A

Table 6. SIMD implementations of LEA-128 and AES-128

Platform	LEA CTR	AES CTR
Intel Core 2 Quad Q6600	4.51	9.32 [35]
Intel Core i7-860	4.19	6.92 [35]
AMD Opteron 6176SE	4.50	N/A

Virtual RISC(AVR), which are among the most favorable 8-bit platforms. LEA is estimated to run at around 3,040 cycles for encryption on AVR AT90USB82/162 where AES best record is 1,993 cycles [47]. We suppose that the performance of LEA is comparable to that of AES on low-end 8-bit or 16-bit platforms, both in speed and code size.

5.2 Hardware Implementation

We have implemented LEA-128 with Verilog HDL and synthesized to ASIC with fully verifying the correctness of front-end and back-end design. For HDL implementation and verification of our design, we have used Mentor Modelsim 6.5f for RTL simulation and Synopsys Design Compiler Ver. B-2008.09-SP5 for its synthesis. Our RTL level design result of LEA is synthesized to ASIC with the UMC 0.13 μ m standard cell library and 100 MHz operating frequency.

Since the LEA consists of the small number of simple operations such as bit XOR, rotation and 32-bit adder without complex operations such as S-box, it can be implemented with low hardware resources. The LEA can also achieve high

Table 7. Implementations of LEA-128 and AES-128 on ColdFire Platform

Algorithm	ROM size (bytes)	RAM size (bytes)	Speed (cycles/byte)	Platform
LEA-128	9,674	832	103.59	MCF5213
LEA-128	704	32	829.25	MCF5213
AES-128 [48]	7,996		1,403.51	ColdFire v2
AES-128 [48]	960		160.00	ColdFire v2 with CAU [†]

[†]Cryptographic Acceleration Unit

performance for its short critical path characteristics. The operational blocks for the round function and key scheduling are so regular that we can achieve these operations with low hardware resources by using its basic operational blocks repetitively.

Table 8 shows the hardware complexity of two different implementations of LEA-128 encryption module: One is the area-optimized and the other is the FOM-optimized (throughput/area). The area-optimized implementation of LEA has 3,826 GE and 168 clock cycles, and the FOM-optimized has 5,426 GE and 24 clock cycles. We can see that the LEA encryption algorithm has relatively lightweight key scheduling and encryption block (Round Function) from this table.

Table 9 compares our hardware implementation results of LEA-128 encryption to other 128-bit key block ciphers with view point of FOM.

Table 8. Hardware feature of LEA-128 encryption module

Block	Area(GE)	
	Area-optimized	FOM-optimized
Constants generation	970	964
Control unit	75	54
Key scheduling	400	695
State register	920	1,037
Key register	998	1,037
Round function	450	1,080
Others	23	559
Total block	3,826	5,426

Table 9. Hardware implementation of LEA-128 encryption algorithm and its comparison to that of other 128-bit key block ciphers

Algorithm	Size(bits)		Cycles	T.put [†]	Tech.	Area	FOM [‡]
	Key	block				(μm)	(GE)
LED [30]	128	64	1,872	3.42	0.18	1,265	0.26
CLEFIA [52]	128	128	328	39	0.09	2,488	1.56
PICCOLO [51]	128	64	528	12.12	0.13	758	1.59
LEA-128 ¹	128	128	168	76.19	0.13	3,826	1.9
AES [45]	128	128	226	56.64	0.13	2,400	2.35
HIGHT [31]	128	64	34	188.24	0.25	3,048	6.17
TWINE [53]	128	64	36	178	0.09	1,866	9.53
LEA-128 ²	128	128	24	533.33	0.13	5,426	9.82
PRESENT [13]	128	64	32	200	0.18	1,570	12.73

[†]Throughput@100KHz (Kbps), [‡]FOM : (Throughput/Area) $\times 10^2$

¹ : Area-optimized implementation of LEA-128

² : FOM-optimized implementation of LEA-128

6 Conclusion

We have proposed a new block cipher LEA, which has 128-bit block size and 128, 192, or 256-bit key size. LEA provides a high-speed software encryption on general-purpose processors. It can be also implemented to have tiny code size. Its hardware implementation has a competitive throughput per area. It is secure against all the existing attacks. In spite of the remarkable implementation results presented in this paper, we believe that they have room for further optimizations.

A Differential Characteristic

Let ΔX_i be the XOR difference of X_i , and let p_i be the probability of $\Delta X_i \rightarrow \Delta X_{i+1}$. The probability p of an r -round differential characteristic is computed as $p = \prod_{i=0}^{r-1} p_i$.

Table 10 shows the 11-round differential characteristic with the probability of 2^{-98} . The differences in the table are denoted in hexadecimal.

Table 10. 11-round differential characteristic with the probability of 2^{-98}

i	ΔX_i				p_i
0	80000234	α 0402214	β 0401205	γ 0400281	2^{-22}
1	80400080	8a000080	82000210	80000234	2^{-14}
2	80000014	80400014	80400004	80400080	2^{-9}
3	80000000	80000000	80000010	80000014	2^{-3}
4	00000000	80000000	80000000	80000000	1
5	00000100	00000000	00000000	00000000	2^{-1}
6	00020000	00000000	00000000	00000100	2^{-2}
7	04000000	00000000	00000020	00020000	2^{-4}
8	00000008	00000001	00004004	04000000	2^{-8}
9	00001200	28000200	80800800	00000008	2^{-12}
10	00200050	05440050	10100101	00001200	2^{-23}
11	η 800000a	88aaa00a	220202 ζ 0	00200050	

The 7-round differential characteristic with the probability of 2^{-27} , discarding the first two rounds and the last two rounds is used for constructing a 14-round boomerang characteristic.

B Linear Approximation

Let ΓX_i be the mask of X_i , and let $\varepsilon_i = p_i - 1/2$ be the bias of the linear approximation

$$\Gamma X_i \cdot X_i \oplus \Gamma X_{i+1} \cdot X_{i+1} = \Gamma K_i \cdot RK. \quad (2)$$

Table 11. 11-round linear approximation with the bias $\varepsilon = 2^{-62}$

$\Gamma X_0 =$ 0aff33f0 470032b0 735801c0	15f00080
$(\alpha_0^0, \alpha_1^0, \alpha_2^0) = (0a0033f0, 0f0033b0, 0a0033b0)$	$\varepsilon_{\alpha^0} = 2^{-7}$
$(\beta_0^0, \beta_1^0, \beta_2^0) = (48000100, 6c000100, 48000180)$	$\varepsilon_{\beta^0} = -2^{-4}$
$(\gamma_0^0, \gamma_1^0, \gamma_2^0) = (1f5800c0, 15f00080, 15500080)$	$\varepsilon_{\gamma^0} = 2^{-7}$
$\Gamma X_1 =$ 00676014 0240000c 02aa0010	00ff0000
$(\alpha_0^1, \alpha_1^1, \alpha_2^1) = (00600014, 00400014, 0040001e)$	$\varepsilon_{\alpha^1} = 2^{-4}$
$(\beta_0^1, \beta_1^1, \beta_2^1) = (02000018, 02000010, 03000010)$	$\varepsilon_{\beta^1} = -2^{-3}$
$(\gamma_0^1, \gamma_1^1, \gamma_2^1) = (00aa0000, 00ff0000, 00aa0000)$	$\varepsilon_{\gamma^1} = 2^{-5}$
$\Gamma X_2 =$ 80003c00 80180000 00154000	00076000
$(\alpha_0^2, \alpha_1^2, \alpha_2^2) = (80000000, 80000000, c0000000)$	$\varepsilon_{\alpha^2} = -2^{-2}$
$(\beta_0^2, \beta_1^2, \beta_2^2) = (00180000, 00100000, 00100000)$	$\varepsilon_{\beta^2} = 2^{-2}$
$(\gamma_0^2, \gamma_1^2, \gamma_2^2) = (00054000, 00076000, 0005c000)$	$\varepsilon_{\gamma^2} = -2^{-4}$
$\Gamma X_3 =$ 00000180 00008000 0000b800	00003c00
$(\alpha_0^3, \alpha_1^3, \alpha_2^3) = (00000000, 00000000, 00000000)$	$\varepsilon_{\alpha^3} = 2^{-1}$
$(\beta_0^3, \beta_1^3, \beta_2^3) = (00008000, 00008000, 0000c000)$	$\varepsilon_{\beta^3} = -2^{-2}$
$(\gamma_0^3, \gamma_1^3, \gamma_2^3) = (00003800, 00003c00, 00003800)$	$\varepsilon_{\gamma^3} = 2^{-3}$
$\Gamma X_4 =$ 00000000 00000600 00000700	00000180
$(\alpha_0^4, \alpha_1^4, \alpha_2^4) = (00000000, 00000000, 00000000)$	$\varepsilon_{\alpha^4} = 2^{-1}$
$(\beta_0^4, \beta_1^4, \beta_2^4) = (00000600, 00000600, 00000600)$	$\varepsilon_{\beta^4} = 2^{-2}$
$(\gamma_0^4, \gamma_1^4, \gamma_2^4) = (00000100, 00000180, 00000100)$	$\varepsilon_{\gamma^4} = 2^{-2}$
$\Gamma X_5 =$ 00000000 00000030 00000020	00000000
$(\alpha_0^5, \alpha_1^5, \alpha_2^5) = (00000000, 00000000, 00000000)$	$\varepsilon_{\alpha^5} = 2^{-1}$
$(\beta_0^5, \beta_1^5, \beta_2^5) = (00000030, 00000020, 00000020)$	$\varepsilon_{\beta^5} = 2^{-2}$
$(\gamma_0^5, \gamma_1^5, \gamma_2^5) = (00000000, 00000000, 00000000)$	$\varepsilon_{\gamma^5} = 2^{-1}$
$\Gamma X_6 =$ 00000000 00000001 00000000	00000000
$(\alpha_0^6, \alpha_1^6, \alpha_2^6) = (00000001, 00000001, 00000001)$	$\varepsilon_{\alpha^6} = 2^{-1}$
$(\beta_0^6, \beta_1^6, \beta_2^6) = (00000000, 00000000, 00000000)$	$\varepsilon_{\beta^6} = 2^{-1}$
$(\gamma_0^6, \gamma_1^6, \gamma_2^6) = (00000000, 00000000, 00000000)$	$\varepsilon_{\gamma^6} = 2^{-1}$
$\Gamma X_7 =$ 00000200 00000000 00000000	00000001
$(\alpha_0^7, \alpha_1^7, \alpha_2^7) = (00000001, 00000001, 00000001)$	$\varepsilon_{\alpha^7} = 2^{-1}$
$(\beta_0^7, \beta_1^7, \beta_2^7) = (00000001, 00000001, 00000001)$	$\varepsilon_{\beta^7} = 2^{-1}$
$(\gamma_0^7, \gamma_1^7, \gamma_2^7) = (00000001, 00000001, 00000001)$	$\varepsilon_{\gamma^7} = 2^{-1}$
$\Gamma X_8 =$ 00000200 08000000 20000000	00000201
$(\alpha_0^8, \alpha_1^8, \alpha_2^8) = (28000201, 38000201, 2c000301)$	$\varepsilon_{\alpha^8} = 2^{-4}$
$(\beta_0^8, \beta_1^8, \beta_2^8) = (30000201, 20000301, 20000201)$	$\varepsilon_{\beta^8} = 2^{-3}$
$(\gamma_0^8, \gamma_1^8, \gamma_2^8) = (00000301, 00000201, 00000201)$	$\varepsilon_{\gamma^8} = 2^{-2}$
$\Gamma X_9 =$ 00060258 09000010 20000040	28000001
$(\alpha_0^9, \alpha_1^9, \alpha_2^9) = (01000059, 01000051, 01800071)$	$\varepsilon_{\alpha^9} = 2^{-4}$
$(\beta_0^9, \beta_1^9, \beta_2^9) = (08000041, 0c000041, 08000061)$	$\varepsilon_{\beta^9} = -2^{-3}$
$(\gamma_0^9, \gamma_1^9, \gamma_2^9) = (2c000001, 28000001, 38000001)$	$\varepsilon_{\gamma^9} = -2^{-3}$
$\Gamma X_{10} =$ 0000e203 08400003 27000000	01060201
$(\alpha_0^{10}, \alpha_1^{10}, \alpha_2^{10}) = (3c660203, 3c440202, 3c440302)$	$\varepsilon_{\alpha^{10}} = -2^{-7}$
$(\beta_0^{10}, \beta_1^{10}, \beta_2^{10}) = (34040201, 26040201, 24060301)$	$\varepsilon_{\beta^{10}} = 2^{-5}$
$(\gamma_0^{10}, \gamma_1^{10}, \gamma_2^{10}) = (01040201, 01060201, 01840301)$	$\varepsilon_{\gamma^{10}} = 2^{-4}$
$\Gamma X_{11} =$ 88060478 09203018 20308060	3c66e000

Table 12. 10-round impossible differential characteristic

i	ΔX_i in forward direction	ΔX_i in backward direction	i
$X[0]$	0 10000000000000000000000000000000		
$X[1]$	10000000000000000000000000000000		
$X[2]$	10000000000000000000000000000000		
$X[3]$	10000000000000000000000000000000		
$X[0]$	1 00000000000000000000000000000000		
$X[1]$	00000000000000000000000000000000		
$X[2]$	00000000000000000000000000000000		
$X[3]$	10000000000000000000000000000000		
$X[0]$	2 00000000000000000000000000000000	00000000000000000000000000000000	10
$X[1]$	00000000000000000000000000000000	00000000000000000000000000000000	
$X[2]$	00010000000000000000000000000000	00010000000000000000000000000000	
$X[3]$	00000000000000000000000000000000	00000000000000000000000000000000	
$X[0]$	3 00000000000000000000000000000000	00000000000000000000000000000000	9
$X[1]$	00000xxx100000000000000000000000	00000000000000000000000000000000	
$X[2]$	000xxx1000000000000000000000000	00000000000000000000000000000000	
$X[3]$	00000000000000000000000000000000	10000000000000000000000000000000	
$X[0]$	4 00000000000000000000000000000000	10000000000000000000000000000000	8
$X[1]$	00000xxxxxxx10000000000000000000	10000000000000000000000000000000	
$X[2]$	000xxxxxxx1000000000000000000000	10000000000000000000000000000000	
$X[3]$	00000000000000000000000000000000	10000000000000000000000000000000	
$X[0]$	5 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx	10000000000000000000000000000000	7
$X[1]$	00000xxxxxxxxxxxxxxxx10000000000000	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100000	
$X[2]$	000xxxxxxxxxxxxxxxx1000000000000000	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx10000	
$X[3]$	00000000000000000000000000000000	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100	
$X[0]$	6 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100	6
$X[1]$	00000xxxxxxxxxxxxxxxxxxxxxxxx100000000	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100	
$X[2]$	xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
$X[3]$	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	

Equation (2) is XOR-sum of the following approximations:

$$\alpha_0^i \cdot (X_i[0] \oplus RK_i[0]) \oplus \alpha_1^i \cdot (X_i[1] \oplus RK_i[1]) = \alpha_2^i \cdot ROR_9(X_{i+1}[0]),$$

$$p_{\alpha^i} = 1/2 + \varepsilon_{\alpha^i}, \quad (3)$$

$$\beta_0^i \cdot (X_i[1] \oplus RK_i[2]) \oplus \beta_1^i \cdot (X_i[2] \oplus RK_i[3]) = \beta_2^i \cdot ROL_5(X_{i+1}[1]),$$

$$p_{\beta^i} = 1/2 + \varepsilon_{\beta^i}, \quad (4)$$

$$\gamma_0^i \cdot (X_i[2] \oplus RK_i[4]) \oplus \gamma_1^i \cdot (X_i[3] \oplus RK_i[5]) = \gamma_2^i \cdot ROL_3(X_{i+1}[2]),$$

$$p_{\gamma^i} = 1/2 + \varepsilon_{\gamma^i}. \quad (5)$$

Let ε be the bias of an r -round linear approximation. Note that $\varepsilon_i = 4\varepsilon_{\alpha^i}\varepsilon_{\beta^i}\varepsilon_{\gamma^i}$ and $\varepsilon = 2^{r-1} \prod_{i=0}^{r-1} \varepsilon_i$ by Piling-Up Lemma [44].

Table 11 shows the 11-round linear approximation with the biases of 2^{-62} . The masks in the table are denoted in hexadecimal.

C Impossible Differential Characteristic

Table 12 shows one of three 10-round impossible differential characteristic reported in [20]. ‘1’ and ‘0’ mean the single bits 1 and 0 in the XOR difference. ‘x’ means an unknown bit.

References

1. Aoki, K., Sasaki, Y.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
2. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE. Submission to NIST (Round 3) (2010)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clar, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive. Report 2013/404 (2013)
4. Bernstein, D.J.: The salsa20 stream cipher. In: SKEW 2005 — Symmetric Key Encryption Workshop (2005)
5. Biham, E.: New types of cryptanalytic attacks using related keys. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994)
6. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
7. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
8. Biham, E., Dunkelman, O., Keller, N.: Enhancing differential-linear cryptanalysis. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 254–266. Springer, Heidelberg (2002)
9. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
10. 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)
11. Biryukov, A., Khovratovich, D., Nikolić, 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)
12. Biryukov, A., Wagner, D.: Slide attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
13. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, Ch., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsøe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbaauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
14. Bogdanov, A., Khovratovich, D., Rechberger, Ch.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
15. Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 29–48. Springer, Heidelberg (2012)

16. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S., Yalçın, T.: PRINCE - A low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
17. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: application to McEliece’s cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. Inf. Theory* **44**(1), 367–378 (1998)
18. Certicom White Paper Series. Critical infrastructure protection for AMI using a comprehensive security platform, February 2009
19. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
20. COSIC. Final Report: Security Evaluation of the Block Cipher LEA (2011)
21. Daemen, J., Rijmen, V.: The Design of Rijndael: AES. In: The Advanced Encryption Standard. Springer (2002)
22. Darnall, M., Kuhlman, D.: AES software implementations on ARM7TDMI. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 424–435. Springer, Heidelberg (2006)
23. Diffie, W., Hellman, M.: Exhaustive cryptanalysis of the NBS data encryption standard. *Computer* **10**(6), 74–84 (1977)
24. Dunkelman, O., Keller, N., Shamir, A.: A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 393–410. Springer, Heidelberg (2010)
25. eBACS: ECRYPT Benchmarking of Cryptographic Systems, bench.cr.yp.to
26. Ferguson, N., Lucks, S., Schneier, B., DougWhiting, Bellare, M., Tadayoshi Kohno, Callas, J., Jesse Walker, : The skein hash function family, Submission to NIST (Round 3) (2010)
27. ADVANCED ENCRYPTION STANDARD, (AES), Federal Information Processing Standards, Publication 197, 26 November 2001)
28. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A new family of lightweight block ciphers. In: Juels, A., Paar, Ch. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
29. Mukhopadhyay, D.: An improved fault based attack of the advanced encryption standard. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 421–434. Springer, Heidelberg (2009)
30. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
31. Hong, D., Sung, J., Hong, S.H., Lim, J.-I., Lee, S.-J., Koo, B.-S., Lee, C.-H., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J.-S., 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)
32. Hong, D., Koo, B., Kwon, D.: Biclique attack on the full HIGHT. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 365–374. Springer, Heidelberg (2012)
33. ISO/IEC 19772, Information technology — Security techniques — Authenticated encryption (2009)
34. Jakimoski, G., Desmedt, Y.: Related-key differential cryptanalysis of 192-bit key AES variants. In: Matsui, M., Zuccherato, R. (eds.) SAC 2004. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)

35. Käsper, E., Schwabe, P.: Faster and timing-attack resistant AES-GCM. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 1–27. Springer, Heidelberg (2009)
36. Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and serpent. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (2001)
37. Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-WAY, bihamDES, CAST, DES-X, newDES, RC2, and TEA. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 233–246. Springer, Heidelberg (1997)
38. Khovratovich, D., Nikolić, I.: Rotational cryptanalysis of ARX. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 333–346. Springer, Heidelberg (2010)
39. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
40. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
41. Koo, B., Hong, D., Kwon, D.: Related-key attack on the full HIGHT. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 49–67. Springer, Heidelberg (2011)
42. Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002)
43. Matsuda, S., Moriai, S.: Lightweight cryptography for the cloud: exploit the power of bitslice implementation. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 408–425. Springer, Heidelberg (2012)
44. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
45. Moradi, A., Poschmann, A., Ling, S., Paar, Ch., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
46. Needham, R.M., Wheeler, D.J.: TEA extensions. computer laboratory, University of Cambridge, Technical report, October 1997
47. Osvik, D.A., Bos, J.W., Stefan, D., Canright, D.: Fast software AES encryption. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 75–93. Springer, Heidelberg (2010)
48. <https://realtimelogic.com/products/sharkssl/Coldfire-80Mhz/>
49. Rivest, R.L., Robshaw, M.J.B., Sidney, R., Yin, Y.L.: Thr RC6 block cipher (1998)
50. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 282–296. Springer, Heidelberg (2009)
51. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)
52. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (Extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
53. Suzaki, T., Minematsu, K., Morioka, S., Kobayasi, E.: Twine: A lightweight, versatile block cipher. In: Proceedings of ECRYPT Workshop on Lightweight Cryptography (2011)
54. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

55. Wallén, J.: On the differential and linear properties of addition, Master's thesis, Helsinki University of Technology, Laboratory for Theoretical Computer Science, November 2003
56. Wheeler, D.J., Needham, R.M.: TEA, a tiny encryption algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)
57. Wheeler, D.J., Needham, R.M.: Correction of XTEA. Computer Laboratory, University of Cambridge, Technical report (October 1998)
58. Yarkov, E.: Cryptanalysis of XXTEA, IACR Cryptology ePrint Archive 2010/254 (2010)