Hindawi
Security and Communication Networks
Volume 2020, Article ID 1457419, 26 pages
https://doi.org/10.1155/2020/1457419

WILEY | Hindawi

Research Article

A Key Selected S-Box Mechanism and Its Investigation in Modern Block Cipher Design

Jiqiang Lu[1] **and Hwajung Seo**[2]

[1]*Beihang University, Beijing, China*
[2]*Hansung University, Seoul, Republic of Korea*

Correspondence should be addressed to Jiqiang Lu; lvjiqiang@hotmail.com

Received 19 February 2019; Revised 15 December 2019; Accepted 4 May 2020; Published 26 May 2020

Academic Editor: Kaitai Liang

The block cipher is an important means to provide data confidentiality in reality, and the S-box is an essential part in most of modern block cipher designs. In 1973, Feistel used a key selected S-box mechanism in his early block cipher designs, whose idea is to let each S-box have two different states and use a key bit to select which of the two states is to be used in an encryption or decryption operation. However, this key selected S-box mechanism has not got much attention in modern block cipher design with the DES block cipher published in 1977. In this paper, we revisit Feistel's key selected S-box mechanism, give a generalised version of Feistel's key selected S-box mechanism, compare it with existing close notions, and design the LBC example cipher to demonstrate that the generalised key selected S-box mechanism can be advantageous over the ordinary S-box mechanism in modern block cipher design for improving security and/or performance without intensifying computational effort and space in some application environments.

1. Introduction

The block cipher is an important primitive in secret-key cryptography. A block cipher is an algorithm that transforms a fixed-length data block, called a plaintext block, into another data block of the same length, called a ciphertext block, under the control of a secret user key. One main purpose of a block cipher is to provide confidentiality for data transmitted in insecure communication environments. A block cipher typically involves two types of operations, one is for the confusion property, which aims to make an involved relationship between ciphertext and plaintext/key, and the other is for the diffusion property, which aims to dissipate the statistical structure of plaintext over ciphertext. The diffusion operation is usually a linear permutation operation, and the confusion operation is usually made up of a nonlinear substitution box (S-box for short). An S-box takes as input a certain number of data bits and transforms them into a certain number of output bits in a nonlinear way, which is usually implemented as a lookup table. In modern block cipher designs such as DES [1] and AES [2], the S-box is usually an essential part and plays an important role in securing the ciphers.

In 1973, Feistel (the inventor of so-called Feistel ciphers) used a key selected S-box mechanism in his early block cipher designs [3, 4]. Feistel's key selected S-box mechanism to let each S-box have two different states and use a key bit to select which of the two states is to be used in an encryption or decryption operation. However, this key selected S-box mechanism has not got much attention since DES was published in 1977 and has not been investigated in modern block cipher design, although there is occasionally an application [5] with the S-box replaced with such key selected S-boxes in AES.

In this paper, we revisit Feistel's key selected S-box mechanism. First, we generalise Feistel's key selected S-box mechanism, and the generalised key selected S-box mechanism is to store several specific S-boxes (with the same dimension sizes) into a table and use certain key (or subkey) bits to select which of the S-boxes should be used

in each S-box position of the S-box layer of a round of a cipher in an encryption or decryption operation. Then, we compare the generalised key selected S-box mechanism with existing close notions and find that the generalised key selected S-box mechanism can offer extra security without intensifying computational effort and space, by producing many key-dependent choices for the round function of a cipher; in particular, it is well resistant against not only conventional cryptanalysis methods such as differential cryptanalysis [6] and linear cryptanalysis [7] but also recently emerging sophisticated variants such as multiple differential cryptanalysis [8], multiple linear cryptanalysis [9], and multidimensional linear cryptanalysis [10]. The extra security gain can allow us to reduce the number of rounds for the sake of performance, as long as the overhead caused by the key selected S-box mechanism in comparison with the ordinary S-box mechanism is negligible when compared with the gain resulted from the reduced number of rounds. Finally, we design the LBC cipher as an example to demonstrate that the generalised key selected S-box mechanism can be advantageous over the ordinary S-box mechanism in some application environments, where we define the combined difference distribution table and the combined bias distribution table for a generalised key selected S-box and describe frameworks to analyse the security of a block cipher with a generalised key selected S-box against differential and linear cryptanalysis. For this example cipher, the key selected S-box mechanism offers a software speedup of around 12% on a lightweight ARM NEON processor and a software speedup of around 16% on a general-purpose Intel i3 processor and offers a hardware speedup of around 22% in a parallel hardware implementation with one cycle per round, although it requires slightly more gate equivalents (GEs) than the ordinary S-box mechanism in this particular parallel case. However, nevertheless, note that, like most of block cipher designs, we only consider the algorithmic security in the black-box model and do not consider the physical security of its implementations, such as side-channel attacks [11], which work in the gray-box model (that assumes an attacker having more power than the black-box model) and usually need additional resistance countermeasures; we note that an implementation of the (generalised) key selected S-box mechanism may be more vulnerable to side-channel attacks; however, the applicability of side-channel attacks is completely dependent on application environments and no single cipher design can be optimal in all application environments.

The remainder of the paper is organised as follows. In Section 2, we give the abbreviations and notation used throughout this paper. In Section 3, we generalise Feistel's key selected S-box mechanism and compare it with existing similar notions. We specify the LBC block cipher in Section 4, discuss its design rationale in Section 5, and evaluate the security and performance gain of the key selected S-box mechanism over the ordinary S-box mechanism under the LBC example cipher in Sections 6 and 7, respectively. Section 8 concludes this paper.

2. Abbreviations and Notations

In all descriptions we assume that the bits of an n-bit value are numbered from 0 to $n-1$ from right to left, with the most significant bit being the $(n-1)$th, a number without a prefix expresses a decimal number unless stated otherwise, and a number with prefix $0x$ expresses a hexadecimal number. We use the following abbreviations and notations throughout this paper.

GE: gate equivalent

SIMD: single instruction multiple data

SPN: substitution-permutation network

$\oplus$: bitwise logical exclusive OR (XOR) operation

$\lll i$: left rotation (of a bit string) by i bits

: string concatenation

$\circ$: functional composition; when composing functions X and Y, $X \circ Y$ denotes the function obtained by first applying X and then Y

$\langle X \rangle$: X in binary (base 2) notation

$|X|$: the bit length of a value X

$X^{(i_0, i_1, \ldots, i_j)}$: bits $(i_0, i_1, \ldots, i_j)$ of an n-bit value X, $(0 \le i_0, i_1, \ldots, i_j, j \le n-1)$

3. The Generalised Key Selected S-Box Mechanism

In this section, we generalise Feistel's key selected S-box mechanism and compare it with existing close notions by discussing their similarities and differences.

3.1. Definition

Definition 1. A two-variable function $\mathcal{F}: \{0,1\}^m \times \{0,1\}^v \longrightarrow \{0,1\}^q$ (for specific values of m, v, and q) is called a key selected S-box if there are 2^v ordinary (that is, one-variable) $m \times q$-bit S-boxes with indexes from 0 to $2^v - 1$ and, for each fixed v-bit value V, that is, some v bits of key material (e.g., a round key), $\mathcal{F}(\cdot, V)$ refers to the Vth $m \times q$-bit S-box.

We call V the selection vector and write $\mathcal{F}_V(\cdot)$ as $\mathcal{F}(\cdot, V)$ for any fixed V or simply write $\mathcal{F}_V$.

3.2. A Comparison with Key-Dependent S-Boxes.

Key-dependent S-boxes [12, 13] are a class of S-boxes whose input bits include some key (or subkey) bits.

At a high level, the key selected S-box mechanism can be treated as a simplified version or a special case of notion of the key-dependent S-box:

(i) The key selected S-box has two input parameters, one is of course the key parameter and the other is what we usually refer to as the data parameter, so is the key-dependent S-box.

(ii) Like the key-dependent S-box, if designed carefully, the key selected S-box may also result in a better performance with a reduced number of rounds by

providing a greater security than the ordinary S-box mechanism and making the resulting cipher particularly resistant to multiple differential cryptanalysis [8], multiple linear cryptanalysis [9], and multidimensional linear cryptanalysis [10]. Because different differential characteristics or linear approximations usually require different sets of key (or subkey) bits under the key selected S-box, an attacker needs to specify the corresponding selecting key bits when establishing a differential characteristic or linear approximation, which shrinks the remaining key space that can be guessed in the key recovery phase. By contrast, under the ordinary S-box mechanism, a differential characteristic or linear approximation generally works under a random key, and an attacker does not need to specify the corresponding key bits when establishing a differential characteristic or linear approximation, and different differential characteristics or linear approximations can presumably work under the same key, and all these facts leave the full key space that can be guessed in the key recovery phase. As a consequence, under the key selected S-box mechanism, we do not need to additionally increase the number of rounds of a cipher due to the effect of multiple differential cryptanalysis, multiple linear cryptanalysis, and multidimensional linear cryptanalysis, which may produce a performance gain.

However, the key selected S-box is slightly different from the key-dependent S-box.

(i) The current key-dependent S-box construction methods such as [12, 13] generally involve a number of interactions (at least 2, which is from the key-dependent S-box built from an ordinary S-box in [14], one XOR with the input of the ordinary S-box and one XOR with its output) between the key parameter and the data parameter, which is costly. While in the key selected S-box, the key parameter serves simply as the index to the associated ordinary S-boxes and then produces the output after only one simple interaction with the data parameter. In other words, the key selected S-box is usually much less computation intensive than the key-dependent S-box.

(ii) In the current key-dependent S-boxes, the key parameter usually has the same role as and the same dimension size as the data parameter for a good randomness, and the key-dependent S-box can usually produce a relatively large number of instantiations over the key parameter space. By comparison, in the key selected S-box, the key parameter has a different role with the data parameter and usually has a smaller dimension size than the data parameter, as we use next in LBC.

3.3. A Comparison with DES (-like) S-Boxes. The notion of key selected S-box is similar to the notion of a DES (or DES-like) S-box [1], which is an ordinary (6×4 bit) S-box involving only one input parameter, the data parameter, rather than a key-dependent S-box involving the data and key parameters, but a DES S-box uses two bits of the data parameter as the index to the four rows of the S-box table each of which can be treated as an ordinary (4×4 bit) S-box. However, the key selected S-box is different from a DES S-box in which the key selected S-box has the other input parameter, the key parameter, which causes a distinction from the two bits of a DES S-box used for the index to the four rows, although they both serve as an index, for example,

(i) When applying the differential cryptanalysis method [6] at an S-box level, for a DES S-box, we can generate its difference distribution table and then use it under the general assumption that data is distributed uniformly at random, but for a key selected S-box, although we can generate the difference distribution tables of the associated ordinary S-boxes, we have to guess the specific value of the key parameter in order to determine which difference distribution table should be used, since the key parameter is fixed once a user key is provided and thus is not distributed uniformly at random for the data produced with the user key.

(ii) When applying the differential cryptanalysis method at a cipher level, the differential behaviors of the rounds of a cipher using a DES-like S-box are simply iterations of the differential behavior of a round since data is distributed uniformly at random; however, for a cipher using a key selected S-box, although we can make a guess for the values of the key parameters of a few rounds, the guessed values of the key parameters of the few rounds will shrink the space of possible user keys, and eventually the space of possible user keys will become very small or empty after a number of rounds, which would make it no sense to cryptanalyse the cipher any more.

In short, like the key-dependent S-box, the key selected S-box makes more difficulty than an ordinary S-box for an attacker to apply differential cryptanalysis. The same situation holds for linear cryptanalysis [7], multiple differential cryptanalysis, multiple linear cryptanalysis, multidimensional linear cryptanalysis, etc.

3.4. A Comparison with Lucifer S-Box Mechanism. The DES precursor Lucifer [15] also uses a key bit to control which of its two S-boxes is to be used as follows. Suppose S_0 and S_1 are two four-bit S-boxes, X and Y are four-bit nibbles, and some key bit is a so-called Interchange Control Bit (ICB). When ICB is equal to 0, then X will go through S_0 and Y will go through S_1; when ICB is equal to 1, then X will go through S_1 and Y will go through S_0.

Lucifer S-box mechanism is different from the key selected S-box mechanism, which is best illustrated by the simple example with only two S-boxes in Figure 1. In the Lucifer S-box mechanism, the outputs of S_0 and S_1 are dependent. If X goes through S_0, then Y must go through S_1,

and vice versa. However, in the key selected S-box mechanism, whether X will go through S_0 or S_1 is independent from whether Y will go through S_0 or S_1, since the two selection key bits are independent. In this simple example, the Lucifer S-box mechanism can produce two possible output patterns: $(S_0(X), S_1(Y))$ and $(S_1(X), S_0(Y))$, while the key selected S-box mechanism can produce four possible output patterns: $(S_0(X), S_1(Y))$, $(S_1(X), S_1(Y))$, $(S_1(X), S_0(Y))$, and $(S_0(X), S_0(Y))$. Other small distinctions include (1) the relative positions of X and Y are variable in the output of the Lucifer S-box mechanism, while the relative positions of X and Y are fixed in the output of the key selected S-box mechanism and (2) the relative positions of S_0 and S_1 are fixed in the output of the Lucifer S-box mechanism, while the relative positions of S_0 and S_1 are indeterminate in the output of the key selected S-box mechanism.

3.5. A Comparison with Key-Dependent S-Box Layers. In 1994, when discussing how to strength the DES block cipher, Biham and Biryukov [14] mentioned the idea of using several sets of S-boxes (for the S-box layer of the DES round function) and using additional key bits to control which set is used (in an encryption/decryption operation), by writing that 'One can compute several different sets of S-boxes according to the design principles of DES and use additional key bits to control which set is used.' In 1999, Harris and Adams [16] mentioned a slightly different idea, which uses several S-boxes in a key-dependent order (also for the S-box layer of the round function of a cipher), by writing 'Another possibility is to order the s-boxes in a key-dependent way.' However, neither Biham and Biryukov nor Harris and Adams implemented their idea, and they mentioned that the security gain is small when the number of (the sets of) S-boxes is small; specifically, Biham and Biryukov mentioned that the scheme is strengthened by a factor smaller than the number of the sets of S-boxes, and Harris and Adams mentioned that it is not particularly useful with only four S-boxes, (since there are only 4! = 24 possible orders, adding less than five bits of entropy to the key space). In other words, Biham and Biryukov's and Harris and Adams's mechanisms would require a large number of S-boxes in order to produce a large security gain in practice.

Compared with Biham and Biryukov's and Harris and Adams's mechanisms [14, 16], the (generalised) key selected S-box mechanism can produce a much larger security gain at the expense of relatively more overhead, given a small number of S-boxes. Below, we discuss other similarities and differences between the (generalised) key selected mechanism and Biham and Biryukov's and Harris and Adams's mechanisms. These similarities and differences are better illustrated by the typical example in Figure 2, where S_0, S_1, S_2, and S_3 are four S-boxes with the same size, K is a user key, and $K_1, K_2, \ldots, K_m$ are round keys for some positive integer m:

(i) Storage space required: Biham and Biryukov's mechanism [14] requires storing a number of sets of permuted S-boxes for the S-box layer of the round function, while Harris and Adams's mechanism and the (generalised) key selected S-box mechanism require storing a number of S-boxes for the S-box layer of the round function. Thus, Biham and Biryukov's mechanism generally requires a larger storage space than Harris and Adams's mechanism and the key selected S-box mechanism.

(ii) The number of choices on the S-box layer of the round function of a cipher: Biham and Biryukov's mechanism produces the same number of choices as the sets of permuted S-boxes stored, Harris and Adams's mechanism produces all possible permutations of the S-boxes stored, while the key selected S-box mechanism produces all possible patterns of the S-boxes stored.

(iii) Under Biham and Biryukov's and Harris and Adams's mechanisms, the number of total choices for all the S-box layers of a cipher is equal to the number of total choices on an S-box layer of the cipher. While under the key selected S-box mechanism, the number of total choices for all the S-box layers of a cipher can be equal to the key space at maximum in theory.

(iv) Given a user key: Biham and Biryukov's and Harris and Adams's mechanisms use the same S-box layer for all rounds of a cipher, while the key selected S-box mechanism likely uses different S-box layers for different rounds. This significantly increases the security gain, albeit at the expense of relatively more implementation overhead if used under the same number of rounds, but nevertheless the security gain can allow for a reduced number of rounds so that a better overall performance may be possible, depending on specific cipher designs.

(v) When implemented in a parallel hardware with one cycle per round, the key selected S-box mechanism generally requires slightly more hardware area (or GEs) than its counterparts using Biham and Biryukov's and Harris and Adams's mechanisms. Anyway, when implemented in a serial hardware, the key selected S-box mechanism may produce a more compact implementation, depending on the reduced number of rounds owing to the security gain.

Particularly, coming back to the typical example in Figure 2, Biham and Biryukov mentioned that the security gain is small (i.e., 2 bits of entropy) when the number of the sets of S-boxes is small, and Harris and Adams mentioned that their mechanism is not particularly useful with only four S-boxes, since there are only 4! = 24 possible orders, adding less than five bits of entropy to the key space. However, the key selected S-box mechanism can produce a much larger security gain even with the small number of four S-boxes, and specific security gain depends on a specific cipher design that the key selected S-box mechanism applies to.

3.6. Summary. In summary, the key selected S-box is similar to but more or less different from existing close notions; it is simple to construct a key selected S-box from ordinary S-boxes,

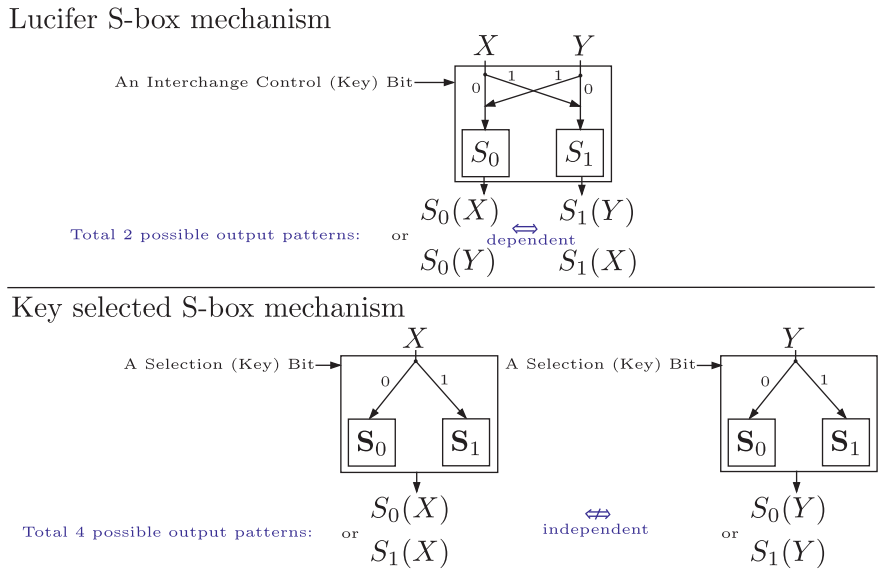


Figure 1: A comparison of the key selected S-box mechanism with Lucifer S-box mechanism.

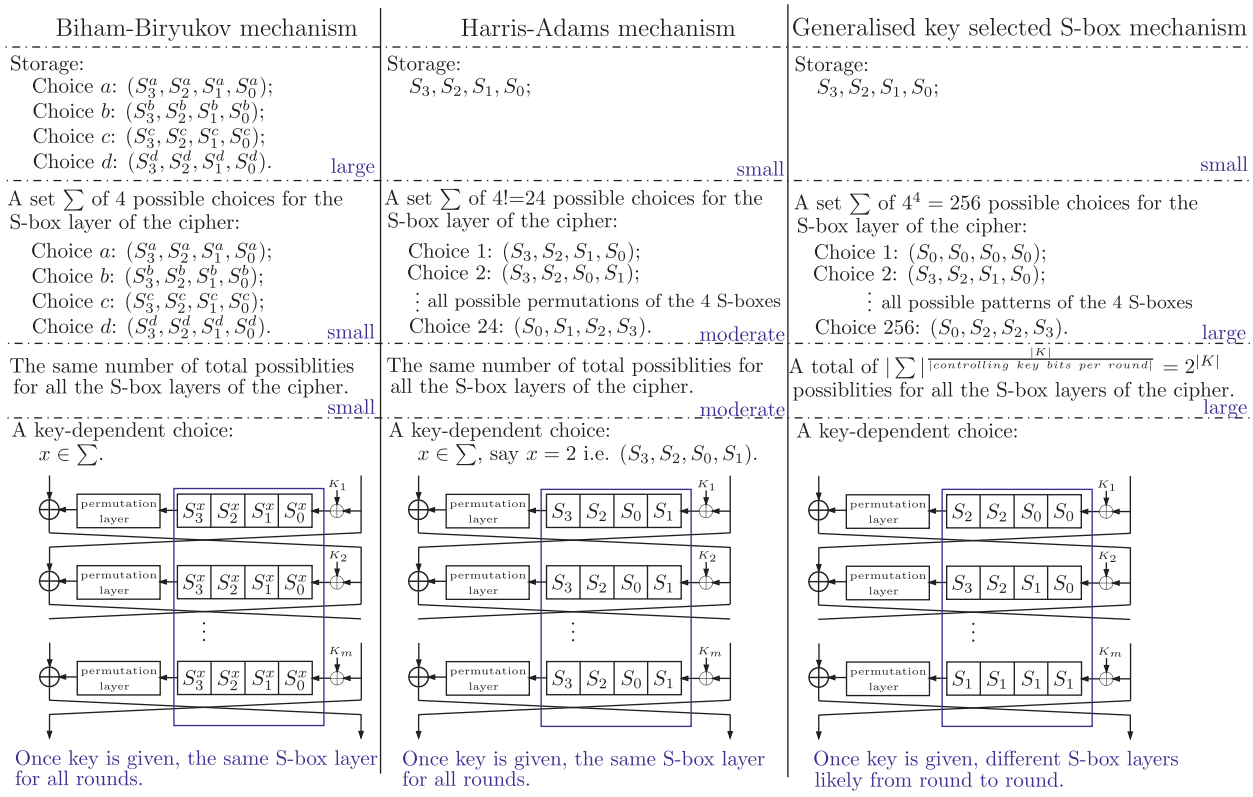


Figure 2: A comparison of the (generalised) key selected S-box mechanism with Biham and Biryukov's and Harris and Adams's mechanisms.

and it produces greater security improvement. A modern block cipher can gain extra security by using the (generalised) key selected S-box mechanism and can gain a better performance by reducing the number of rounds according to the extra security gain, as long as the overhead caused by the key selected S-box mechanism in comparison with the ordinary S-box mechanism is negligible when compared with the gain resulted from the reduced number of rounds.

4. The LBC Block Cipher

In this section, we specify the LBC block cipher, which employs a Feistel structure with a 64-bit block size, a variable length user key from 96 to 128 bits and a total of 25 rounds and takes advantage of the key selected S-box mechanism to achieve a good security and performance. LBC uses two elementary operations and involves three subalgorithms,

namely, a key schedule algorithm, an encryption algorithm, and a decryption algorithm.

Below, we first describe the two elementary operations used in LBC, then the round function, the key schedule algorithm, the encryption algorithm, the decryption algorithm, and finally several test vectors of LBC.

4.1. Elementary Operations. LBC mainly uses two elementary operations: a confusion operation **S** and a diffusion operation **L**, which are defined as follows:

(i) **S**: $\{0,1\}^{32} \times \{0,1\}^{16} \longrightarrow \{0,1\}^{32}$ is a nonlinear substitution operation, constructed by applying a key selected S-box $\mathcal{S}$: $\{0,1\}^4 \times \{0,1\}^2 \longrightarrow \{0,1\}^4$ eight times in parallel to the inputs. The four general 4×4-bit S-boxes involved in the key selected S-box $\mathcal{S}$ are $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, which we chose according to the most recent work on 4-bit optimal S-boxes owing to Zhang et al. [17], whose specifications are given in Table 1. If $X = (X_3, X_2, X_1, X_0)$ is a 32-bit block represented as four bytes which are further arranged as a 4×8-bit array:

$$\begin{pmatrix} X_0^{(7)} & X_0^{(6)} & X_0^{(5)} & X_0^{(4)} & X_0^{(3)} & X_0^{(2)} & X_0^{(1)} & X_0^{(0)} \\ X_1^{(7)} & X_1^{(6)} & X_1^{(5)} & X_1^{(4)} & X_1^{(3)} & X_1^{(2)} & X_1^{(1)} & X_1^{(0)} \\ X_2^{(7)} & X_2^{(6)} & X_2^{(5)} & X_2^{(4)} & X_2^{(3)} & X_2^{(2)} & X_2^{(1)} & X_2^{(0)} \\ X_3^{(7)} & X_3^{(6)} & X_3^{(5)} & X_3^{(4)} & X_3^{(3)} & X_3^{(2)} & X_3^{(1)} & X_3^{(0)} \end{pmatrix}, \tag{1}$$

and $V = (V^{(15)}, \ldots, V^{(1)}, V^{(0)})$ is a 16-bit block; then, $S(X, V)$ is defined to equal a 32-bit value represented as four bytes (Y_3, Y_2, Y_1, Y_0) that are further arranged as a 4×8-bit array:

$$\mathbf{S}(X,V) = (Y_3, Y_2, Y_1, Y_0) = \begin{pmatrix} Y_0^{(7)} & Y_0^{(6)} & Y_0^{(5)} & Y_0^{(4)} & Y_0^{(3)} & Y_0^{(2)} & Y_0^{(1)} & Y_0^{(0)} \\ Y_1^{(7)} & Y_1^{(6)} & Y_1^{(5)} & Y_1^{(4)} & Y_1^{(3)} & Y_1^{(2)} & Y_1^{(1)} & Y_1^{(0)} \\ Y_2^{(7)} & Y_2^{(6)} & Y_2^{(5)} & Y_2^{(4)} & Y_2^{(3)} & Y_2^{(2)} & X_2^{(1)} & Y_2^{(0)} \\ Y_3^{(7)} & Y_3^{(6)} & Y_3^{(5)} & Y_3^{(4)} & Y_3^{(3)} & Y_3^{(2)} & Y_3^{(1)} & Y_3^{(0)} \end{pmatrix}, \tag{2}$$

where $\langle Y_3^{(i)}, Y_2^{(i)}, Y_1^{(i)}, Y_0^{(i)} \rangle = \mathcal{S}_{\langle V^{(2i+1)}, V^{(2i)} \rangle}(\langle X_3^{(i)}, X_2^{(i)}, X_1^{(i)}, X_0^{(i)} \rangle)$, for $i = 0, 1, \ldots, 7$.

(ii) **L**: $\{0,1\}^{32} \longrightarrow \{0,1\}^{32}$ is a linear transformation. If $X = (X_3, X_2, X_1, X_0)$ is a 32-bit block represented as four bytes; then, $\mathbf{L}(X)$ is defined as

$$\begin{aligned} \mathbf{L}(X) = (&X_3 \oplus (X_3 \lll 2) \oplus (X_3 \lll 5), \\ &X_1 \oplus (X_1 \lll 2) \oplus (X_1 \lll 5), \\ &X_2 \oplus (X_2 \lll 2) \oplus (X_2 \lll 5), \\ &X_0 \oplus (X_0 \lll 2) \oplus (X_0 \lll 5)). \end{aligned} \tag{3}$$

Note that there are several equivalent descriptions for the **L** operation.

4.2. Round Function. The round function of LBC is built mainly on the nonlinear substitution operation **S** and the linear **L** operation, which takes two 32-bit blocks as inputs and outputs a 32-bit block.

If X and Y are 32-bit blocks, then the round function **F**: $\{0,1\}^{32} \times \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$ of LBC is defined as follows:

$$\mathbf{F}(X,Y) = \mathbf{L}\left(\mathbf{S}\left(X \oplus Y, Y^{(0,1,4,5,8,9,12,13,16,17,20,21,24,25,28,29)}\right)\right). \tag{4}$$

4.3. Key Schedule Algorithm. The key schedule algorithm of LBC takes a k-bit user key as input and outputs the required twenty-five 32-bit round subkeys, where k can be a variable between 96 and 128 bits and typically $k = 96$. The key schedule algorithm is as follows:

(1) A k-bit user key is stored in a key register K; $K = (K^{(k-1)}, \ldots, K^{(1)}, K^{(0)})$.

(2) Output the leftmost 32 bits of the current content of the key register K as the first round subkey K_1.

(3) For $i = 1$ to 24,

(a) Rotate the key register to the left by 29 bits, that is, $K = (K \lll 29)$.

(b) Update the leftmost 32 bits of the key register as follows:

$$\begin{aligned} &\left(K^{(k-1)}, \ldots, K^{(k-31)}, K^{(k-32)}\right) \\ &= \mathbf{L}\left(\mathbf{S}\left(\left(K^{(k-1)}, \ldots, K^{(k-31)}, K^{(k-32)}\right), \langle k \rangle\right)\right) \oplus \langle i \rangle, \end{aligned} \tag{5}$$

where $\langle k \rangle$ and $\langle i \rangle$ represent, respectively, the binary representations of k and i with the left side being extended by concatenating as many zeros as required to reach the required bit length.

(c) Output the leftmost 32 bits of the current content of the key register K as the $(i+1)$th round subkey K_{i+1}.

Note that the key schedule uses the key selected S-box in an abused way, where the selection vector for the key selected S-box is not the key material but rather the key length.

Table 1: The 4×4-bit S-boxes $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ of LBC, where the values are in hexadecimal notation.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\mathcal{S}_0(x)$	6	F	D	A	9	3	2	4	5	8	B	1	0	E	C	7
$\mathcal{S}_1(x)$	1	0	3	2	5	6	D	8	4	B	9	C	7	F	A	E
$\mathcal{S}_2(x)$	3	2	0	4	8	A	9	5	E	D	C	6	F	1	7	B
$\mathcal{S}_3(x)$	7	3	1	0	B	2	C	F	6	E	5	9	D	4	8	A

4.4. Encryption Algorithm. The encryption algorithm of LBC transforms a 64-bit data block, called a plaintext (block), into a pseudorandom data block of the same length, called a ciphertext (block), under the control of a secret user key.

The encryption algorithm takes as input a 64-bit plaintext block p and has a total of 25 rounds. The encryption procedure is as follows, where L_j and R_j are 32-bit variable $(j = 0, 1, \ldots, 25)$ (K_i $(1 \le i \le 25)$ is a round subkey generated from a user key by the key schedule algorithm of LBC).

(1) Let $(L_0 \| R_0) = P$.

(2) For $i = 1$ to 25,

(i) $R_i = L_{i-1} \oplus \mathbf{F}(R_{i-1}, K_i)$.
(ii) $L_i = R_{i-1}$.

(3) Ciphertext $= (L_{25} \| R_{25})$.

Figure 3 illustrates an encryption round of LBC.

4.5. Decryption Algorithm. The decryption algorithm of LBC is the inverse of the encryption algorithm, and it decrypts a ciphertext to obtain the original plaintext, under the control of the same user key as in the encryption process. It takes a 64-bit ciphertext block C as input and works as follows:

(1) Let $(L_{25} \| R_{25}) = C$.

(2) For $i = 25$ to 1,

(i) $L_{i-1} = R_i \oplus \mathbf{F}(L_i, K_i)$.
(ii) $R_{i-1} = L_i$.

(3) Plaintext $= (L_0 \| R_0)$.

4.6. Test Vectors. Three test vectors of LBC with a 96 bit key $0x89ABCDEFFEDCBA9876543210$ are as follows: (Plain text $= 0x0000000000000000$, Cipher text $= 0xC289DEBA2BD0BD92$), (Plain text $= 0x8765432112345678$, Cipher text $= 0x50305777CE4759C8$), and (Plain text $= 0xFFFFFFFFFFFFFFFF$, Cipher text $= 0xE6897583\ D57A75F8$).

5. Design Rationale of LBC

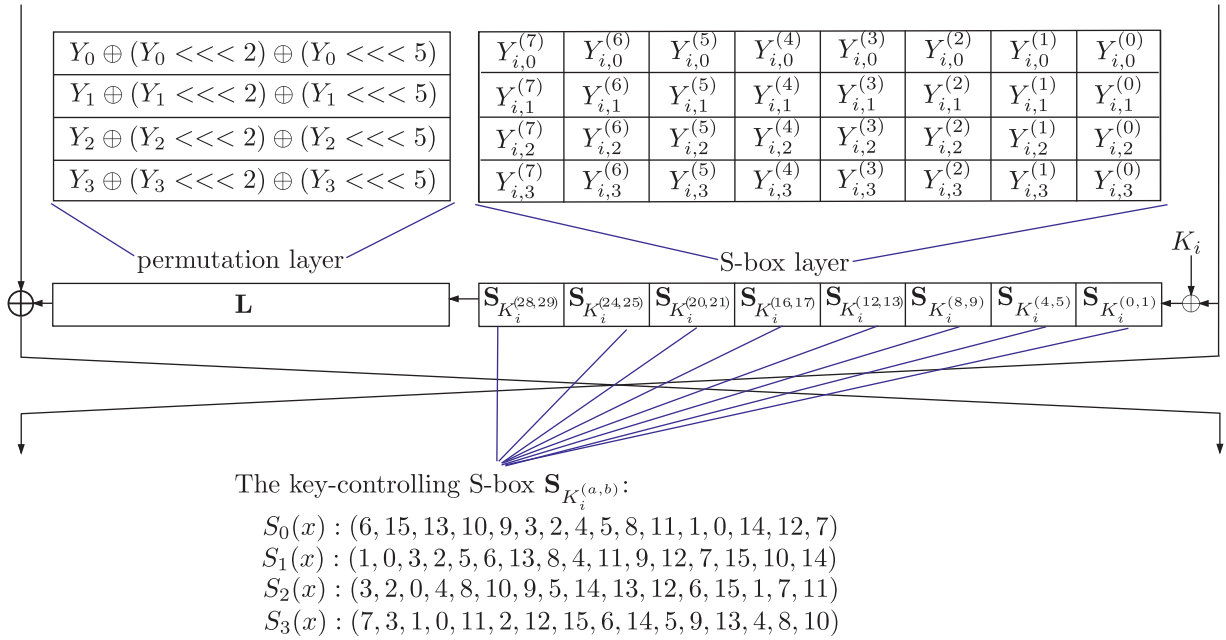
In this section, we give our design rationale for the structure, parameters, and components of LBC. At a high level, we feature the following distinctions when designing the LBC example cipher: (1) the novel notion of the key selected S-box is used to achieve a good performance and a sufficient security; (2) the Feistel structure is combined with simple substitution and permutation operations to achieve an efficient hardware implementation with a moderate amount of GEs and an efficient software implementation; (3) the same key schedule algorithm as well as the same encryption and decryption algorithms for different key length versions of a variable length user key is used to provide user friendliness and efficient resource utilization; and (4) the strong key schedule ensures partially that data authenticity is robustly provided when LBC is sometimes used to build or abused as a hash function in some applications.

5.1. Structure. There are mainly two types of structures for iterated block ciphers, one is the Feistel structure and the other is the Substitution-Permutation Network (SPN) structure.

LBC has a Feistel structure. Compared with an SPN structure, a Feistel structure with the same block size has the following merits: (1) there are more flexibilities to design its round function, for example, the linear or S-box operation does not need to be invertible; (2) the round function is generally lighter, partially due to the fact that the round function operates on a smaller number of bits; and (3) implementing the circuit for both encryption and decryption does not cost much more than implementing the circuit for encryption only, as decryption is (almost) identical to encryption. By contrast, for an SPN structure we need to implement the round function as well as its inverse for both encryption and decryption. Anyway, the Feistel structure may need a larger number of rounds to be secure, but nevertheless, this is not always the case, for example, the Feistel block cipher LBlock [18] has a comparable number of rounds with the SPN block cipher PRESENT [19] and has resisted extensive cryptanalysis. Moreover, LBC uses the novel notion of the key selected S-box as well as a good diffusion operation to achieve additional security protection.

5.2. Block Size. In reality, a general-purpose block cipher typically has a block size of 64 or 128 bits. LBC uses a block size of 64 bits, in order to meet the requirements of moderate application environments on memory, space, and performance. Although a 64-bit block size may be short in some applications due to the birthday bound paradox, it is still okay with appropriate block cipher modes of operation in many applications.

5.3. Key Length. In 2001, Lenstra and Verheul [20] estimated that, for a symmetric cipher, an 80-bit key size can provide a security margin until (around) 2012, a 96-bit key size can provide a security margin until 2034, and a 128-bit key size can provide a security margin until 2076. NIST



The key-controlling S-box $\mathbf{S}_{K_i^{(a,b)}}$:

$S_0(x) : (6, 15, 13, 10, 9, 3, 2, 4, 5, 8, 11, 1, 0, 14, 12, 7)$
$S_1(x) : (1, 0, 3, 2, 5, 6, 13, 8, 4, 11, 9, 12, 7, 15, 10, 14)$
$S_2(x) : (3, 2, 0, 4, 8, 10, 9, 5, 14, 13, 12, 6, 15, 1, 7, 11)$
$S_3(x) : (7, 3, 1, 0, 11, 2, 12, 15, 6, 14, 5, 9, 13, 4, 8, 10)$

Figure 3: An encryption round of LBC.

recommended not to use an 80-bit key in 2010 and disallowed an 80-bit key in 2012. In 2012, European ECRYPT II project remarked for a symmetric cipher that an 80-bit key size provides a security level of "Very short-term protection against agencies" and "$\leq$ 4 years protection," a 96-bit key size provides a security level of "Legacy standard level" and "$\approx$ 10 years protection," and a 128-bit key size provides a security level of "Long-term protection" and "$\approx$ 30 years (protection)". As Dinur [21] noted in 2015, the Bitcoin network [22] demonstrated that a computation of 2^{80} (cipher encryption operations) is (marginally) practical. In short, an 80-bit key is now considered to be too short to be secure in reality.

When designing the LBC cipher, we use a minimum key length of 96 bits for short-term protection and a maximum key length of 128 bits for long-term protection. Anyway, to be flexible and user friendly, LBC accepts a variable-length user key, and thus the user can use a key length of his choice, as long as it is between 96 and 128 bits (a key shorter than 96 bits may be used, which we do not recommend); for example, a 112-bit user key may be used for medium-term protection. Using a variable key length enables the user to have more flexibility to choose an appropriate key length according to the expected lifetime for the concerned security application, so as not to waste computing and hardware resources.

5.4. S-Box Layer. In reality, a general-purpose block cipher typically uses an 8×8-bit S-box, and a lightweight block cipher typically uses a 4×4-bit S-box, in order to meet the requirements of lightweight application environments on memory and space, since a 4×4-bit S-box is generally more compact in hardware than an 8×8-bit S-box.

The PRESENT block cipher uses a 4×4-bit S-box based on Leander and Poschmann's work [23]. However, the S-box has a weak security property in the sense of linear cryptanalysis, that is, there are a number of combinations of one-bit input mask and one-bit output mask [24]. In 2015, Zhang et al. [17] studied 4×4-bit optimal S-boxes with more security criteria and presented three classes of 4×4-bit optimal S-boxes. The number of valid combinations of one-bit input difference and one-bit output difference is x, and the number of valid combinations of one-bit input mask and one-bit output mask is $4 - x$, where $x = 0, 1, 2$.

LBC uses a key selected S-box $\mathcal{S}$ that is based on four ordinary 4×4-bit S-boxes $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ eight times to build the S-box layer $\mathbf{S}$ in its round function $\mathbf{F}$ and uses the same S-box layer in the 25 rounds.

From Zhang et al.'s $(2, 2)$-Num1-DL category of 4×4-bit optimal S-boxes, we further chose each 4×4-bit S-box $\mathcal{S}_i$ $(0 \leq i \leq 3)$ by the following additional security criterion:

(i) The two valid combinations of one-bit input difference and one-bit output difference do not use the same input/output difference; the two valid combinations of (one-bit input mask and one-bit output mask) do not use the same input/output mask. Here, one-bit input difference/mask means that the binary representation of the input difference/mask has one and only one bit position with a one, that is, it has zeros everywhere except for one bit position. The same statement applies subsequently throughout the rest of this paper, although we do not explicitly make it further.

That is, we use the following security criteria in total:

(1) The S-box is bijective, that is, $\mathcal{S}_i(x) \neq \mathcal{S}_i(y)$ if $x \neq y$.

(2) The S-box has no fixed point, that is, $\forall \alpha \in \{0,1\}^4$: $\mathcal{S}_i(x) \neq x$.

(3) $\forall \alpha \in \{0,1\}^4 - \{0^4\}$ and $\forall \beta \in \{0,1\}^4 - \{0^4\}$:

$$\#\{x \mid \mathcal{S}_i(x) \oplus \mathcal{S}_i(x \oplus \alpha) = \beta, x \in \{0,1\}^4\} \leq 4. \quad (6)$$

(4) $\forall \alpha \in \{0,1\}^4 - \{0^4\}$ and $\forall \beta \in \{0,1\}^4 - \{0^4\}$:

$$-4 \leq \#\{x \mid x \odot \alpha = \mathcal{S}_i(x) \odot \beta, x \in \{0,1\}^4\} - 8 \leq 4. \quad (7)$$

(5) The number of valid combinations of one-bit input difference and one-bit output difference is 2, and the number of valid combinations of one-bit input mask and one-bit output mask is 2, too.

(6) Either of the valid combinations of one-bit input difference and one-bit output difference has the smallest (valid) possibility, that is, 1/8 for a 4×4-bit S-box.

(7) Either of the valid combinations of one-bit input mask and one-bit output mask has the smallest (valid) bias, that is, $\pm 1/8$ for a 4×4-bit S-box.

(8) The two valid combinations of one-bit input difference and one-bit output difference do not use the same input/output difference.

(9) The two valid combinations of one-bit input mask and one-bit output mask do not use the same input/output mask.

The four ordinary 4×4-bit S-boxes $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ together meet the following security criterion.

(10) Ideally, any two 4×4-bit S-boxes do not involve a common valid combination of one-bit input difference and one-bit output difference or one-bit input mask and one-bit output mask.

(11) Ideally, any two 4×4-bit S-boxes do not concurrently have the largest (valid) probability (i.e., 1/4) under any (input difference and output difference) pair and do not concurrently have the largest (valid) bias (i.e., $\pm 1/4$) under any (input mask and output mask) pair.

5.5. Diffusion Layer. The diffusion layer **L** has a branch number [25] of 4, which provides a sufficiently large avalanche effect to make LBC secure against currently known cryptanalysis techniques such as differential and linear cryptanalysis, together with the S-box layer. **L** performs only simple operations (namely, rotation and XOR) and is very lightweight in hardware implementation and is suitable not only for hardware implementation but also for software implementation.

5.6. Key Schedule. To achieve a good performance, many lightweight or moderate block ciphers use a simple key schedule, for example, HIGHT [26] and PRESENT; in particular, the style of the key schedule of PRESENT was followed by many subsequent lightweight block ciphers such as LBlock [18] and RECTANGLE [17]. However, the full-round HIGHT was shown in 2011 to suffer from a related-key [27, 28] attack [29], and the full-round PRESENT was shown in 2015 to suffer from a known-key distinguisher [30, 31], mainly due to their simple key schedules. In reality, a block cipher may be used to build or abused sometimes as a hash function for data authenticity to save hardware space, where the unknown key parameter under the block cipher corresponds to the known message parameter under the hash function. Thus, HIGHT and PRESENT are not suitable for this case because of the related-key and known-key cryptanalysis results, and the known-key distinguisher on the full PRESENT puts a security concern on PRESENT-based hash functions.

We aim to design a strong key schedule for LBC so that LBC can resist key schedule attacks and can be used to build or abused as a hash function to provide data authenticity in some devices, considering that confidentiality without authenticity is usually not sufficient for a real-life application (note that a 64-bit digest size may be short for some applications, since the birthday bound is 2^{32}; however, it is practically okay for many real-life applications). The key schedule of LBC is based on the round function, so as to have a good nonlinearity and save some hardware area; it makes LBC secure against related-key cryptanalysis [27, 28, 32] as well as slide attacks [33, 34] (together with the encryption or decryption procedure of LBC). When the key length parameter k is determined by the user, the ordinary S-box used for the key selected S-box in the key schedule can be easily determined (in other words, the key selected S-box becomes an ordinary S-box), and the order of the ordinary S-boxes in the S-box layer can also be easily determined, which results in a determinate S-box layer and thus a simple hardware implementation.

The key schedule of LBC is very user friendly in several aspects. First, a variable-length user key enables the user to have more flexibility in choosing an appropriate key length according to the expected lifetime for the target application, so as not to waste computing and hardware resources. Second, the key schedule uses the same algorithm for different key lengths, which makes LBC different from most existing block ciphers that usually use different key schedule algorithms for different key lengths (if supported); this feature is user-friendly, for example, it enables the user to make a hardware implementation easily for different key length versions. Third, with computing power increasing as time goes on, it is often the case to upgrade to a larger key length when the current key length becomes insufficient after some usage time. A variable-length key enables the LBC user to upgrade to the exactly required key length, so as to efficiently utilize hardware resource by avoiding having to upgrade to a much larger prespecified key length than required. For example, published in 2007, PRESENT accepts only 80- and 128-bit user keys, but since an 80-bit key is considered to be too short nowadays, as mentioned in Section 5.3, PRESENT should be upgraded now if it had been deployed with an 80-bit key in reality, although it is not very long since its publication; however, a 128-bit key may be too long for many lightweight security applications and thus may be wasteful.

The idea of using a variable length key for LBC is motivated by the general block ciphers, Serpent [35] and SHACAL-2 [36], but LBC processes a variable length key in a manner different from that used by Serpent or SHACAL-2: the latter requires extending a shorter user key to the maximum key length by concatenating as many zeros as required or a one followed by as many zeros as required (and thus does not distinguish different key length versions much), while LBC does not require extending a shorter user key to the maximum key length and it distinguishes different key length versions by involving the key length parameter in the key schedule, to avoid potential key-schedule attacks.

6. Security Gain Evaluation

In this section, we briefly give our evaluation results on the security of LBC against a list of advanced cryptanalysis techniques (under the worst case assumption) and finally get the security gain of LBC over the LBC version with the ordinary S-box mechanism. Conservative frameworks are developed for analysing the security of the key selected S-box mechanism against differential and linear cryptanalysis. Recall that like most of block cipher designs, we only consider the black-box security of the algorithm and do not consider its gray-box security such as side-channel attacks [11], which usually assume a more powerful attacker and need additional resistance countermeasures. Note first that LBC uses a user key of at least 96 bits and can withstand elementary cryptanalysis methods. We start with two properties of LBC.

6.1. Properties of LBC. A simple analysis of the **L** operation reveals the following property.

Property 1. For the **L** operation, if the input X and the output $Y = \mathbf{L}(X)$ are represented each as eight 4-bit nibbles corresponding to the eight S-boxes, that is, $X = (X_7, \ldots, X_1, X_0)$ and $Y = (Y_7, \ldots, Y_1, Y_0)$, then

(1) The eight 4-bit nibbles of the output Y can be expressed with the eight 4-bit nibbles of the input X as follows:

$$\begin{aligned} Y_0 &= X_0 \oplus X_3 \oplus X_6, \\ Y_1 &= X_1 \oplus X_4 \oplus X_7, \\ Y_2 &= X_0 \oplus X_2 \oplus X_5, \\ Y_3 &= X_1 \oplus X_3 \oplus X_6, \\ Y_4 &= X_2 \oplus X_4 \oplus X_7, \\ Y_5 &= X_0 \oplus X_3 \oplus X_5, \\ Y_6 &= X_1 \oplus X_4 \oplus X_6, \\ Y_7 &= X_2 \oplus X_5 \oplus X_7. \end{aligned} \tag{8}$$

(2) The eight 4-bit nibbles of the input X can be expressed with the eight 4-bit nibbles of the output Y as follows:

$$\begin{aligned} X_0 &= Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7, \\ X_1 &= Y_0 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7, \\ X_2 &= Y_0 \oplus Y_1 \oplus Y_5 \oplus Y_6 \oplus Y_7, \\ X_3 &= Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_6 \oplus Y_7, \\ X_4 &= Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_7, \\ X_5 &= Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4, \\ X_6 &= Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5, \\ X_7 &= Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6. \end{aligned} \tag{9}$$

A simple detailed investigation reveals the following property.

Property 2. The propagation of a single bit:

(i) A single bit will get at least 62 subkey bits involved, after 3 rounds, depending on the bit position. Detailed numbers of involved subkey bits are given in Table 2, in comparison with the numbers of involved subkey bits under the ordinary mechanism.

(ii) A single bit will get at least 92 subkey bits and about 60 output bits involved, after 4 rounds, depending on the bit position. Detailed numbers of involved subkey bits are given in Table 2, in comparison with the numbers of involved subkey bits under the ordinary mechanism.

(iii) A single bit will get all 96 subkey bits and all 64 output bits involved, after 5 rounds.

The numbers of disjoint subkey bits involved in the propagation of a single nibble position through 3 and 4 rounds are summarised in Table 2 and are briefly illustrated in Figures 4–11.

6.2. Differential Cryptanalysis. As mentioned in Section 3.3, a key selected S-box makes it difficult for an attacker to apply differential cryptanalysis. Anyway, we develop a conservative framework for the differential cryptanalysis of block ciphers using a key selected S-box. We start the framework with introducing the concept of the combined difference distribution (CDD) table for a key selected S-box as follows.

Definition 2. The combined difference distribution (CDD) table for a key selected S-box: $\{0,1\}^m \times \{0,1\}^v \longrightarrow \{0,1\}^q$ (for specific values of m, v, and q) is a table with 2^m rows being the 2^m possible input differences, 2^q columns being the 2^q output differences, and the (i, j)th entry being the set of the possible combinations (the number of m-bit inputs satisfying the input difference and output difference pair (i, j) under an ordinary $m \times q$-bit S-box and the number of ordinary $m \times q$-bit S-boxes that have the number of m-bit inputs satisfying the input difference and output difference pair (i, j)), where $0 \le i \le 2^m - 1$ and $0 \le j \le 2^q - 1$.

As an example, we compute the CDD table for the key selected S-box $\mathcal{S}$ used in LBC, which is given as Table 3. Each

Table 2: The numbers of disjoint subkey bits involved in the propagation of a single nibble position through 3 and 4 rounds.

Position	Nibble			
	3 rounds		4 rounds	
	Ordinary mechanism	Key selected mechanism	Ordinary mechanism	Key selected mechanism
0	58	66	87	95
1	59	63	88	92
2	57	64	86	93
3	58	71	88	101
4	57	68	87	97
5	57	62	87	92
6	57	65	86	94
7	57	69	86	98

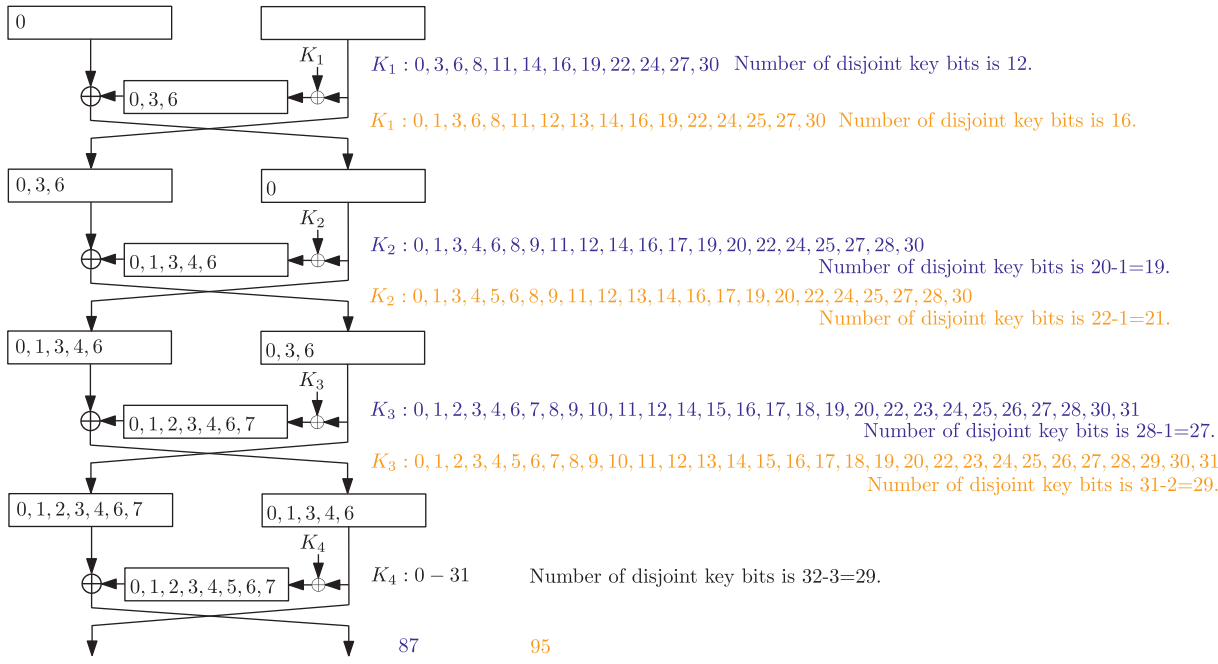


Figure 4: Disjoint subkey bits involved in the propagation of the first nibble through four rounds.

entry except $(0, 0)$ has at most three combinations, which follow the difference distribution tables of the four ordinary S-boxes (see Table 4). Note that one may enhance the combined difference distribution table by associating every combination with its corresponding probability/probabilities.

Now by treating the eight key selected S-boxes in the S-box layer of LBC as eight identical ordinary S-boxes with the difference distribution table being the CDD table, we can check the minimum number of active S-boxes for a differential characteristic of a certain number of rounds, in a manner similar to that Matsui did for DES (under the general assumption for differential cryptanalysis) in [37]. As each ordinary S-box has a maximum (valid) probability of 2^{-2}, we can obtain an upper bound for a differential characteristic of a certain number of rounds and get its security against differential cryptanalysis. Clearly, the upper bound is overestimated, since it is based on the CDD table and, each of the four ordinary difference distribution tables is only a subset of the CDD table. By this way, we can bound the security against differential cryptanalysis in the worst case from the point of the user of the cipher.

We made a computer program to compute the minimum numbers of active S-boxes of i-round differential characteristics under the CDD table ($1 \le i \le 18$), and the results are given in Table 5.

From Table 5, we see that the number of active S-boxes is larger than 32 for 18 or more rounds. In particular, for an 18-round differential characteristic, the number of active S-boxes is at least 33. 33 active S-boxes require a total of 66 selecting key bits, which means that there are only 128 − 66 = 62 key bits left for the key recovery phase. Table 2 shows that a single nibble/bit will get at least 62 subkey bits involved after propagating through 3 rounds. A total of five rounds appended at both ends of an 18-round differential characteristic will indicate at least 3 rounds in an end, which would require an attacker to guess all the remaining 62 key bits in the key recovery phase. As a result, we can assume at most an 18-round differential characteristic and assume appending at most a total of five rounds at both ends.

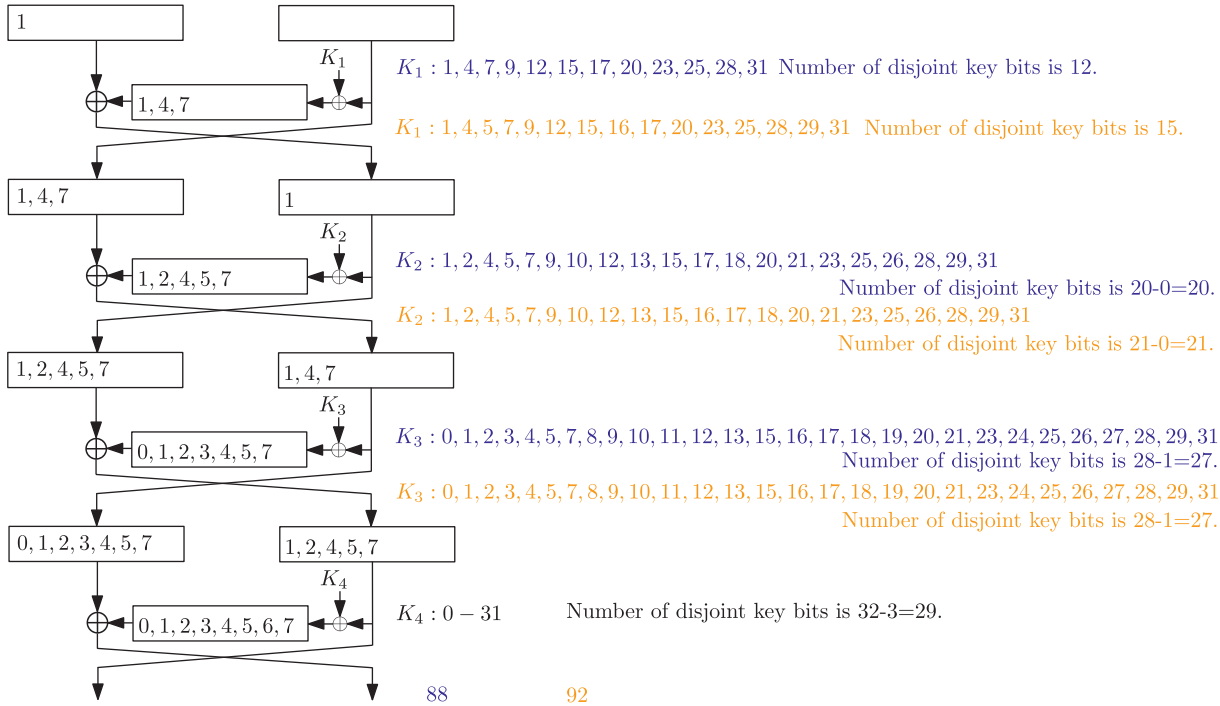


Figure 5: Disjoint subkey bits involved in the propagation of the second nibble through four rounds.

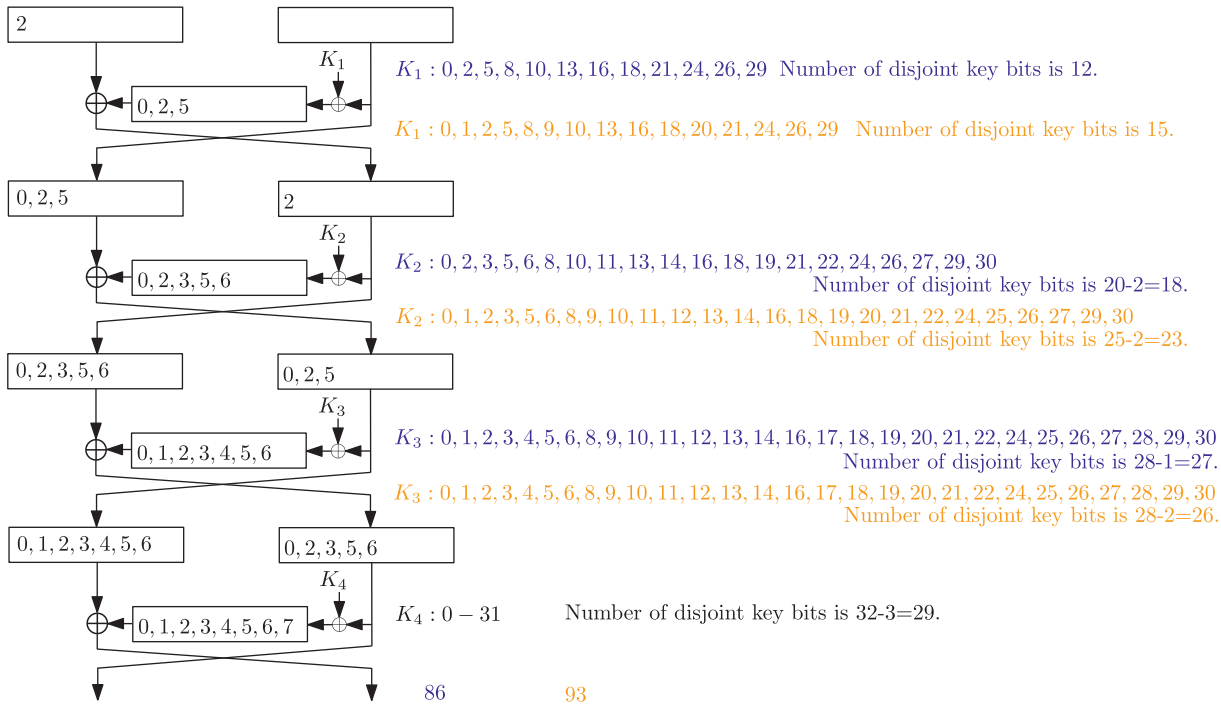


Figure 6: Disjoint subkey bits involved in the propagation of the third nibble through four rounds.

Remind that multiple differential cryptanalysis does not work well in the key selected S-box mechanism because a different differential characteristic will require a different set of selecting key bits, which would further shrink the space of the key bits that can be guessed in the key recovery phase. Therefore, 25-round LBC should be secure against differential cryptanalysis.

6.3. Linear Cryptanalysis. To analyse the security of LBC against linear cryptanalysis, we first have the following property of the **L** operation.

Property 3. For the **L** operation, if the input mask ΓX and the output mask ΓY are represented each as eight 4-bit nibbles corresponding to the eight S-boxes, that is, $\Gamma X =$

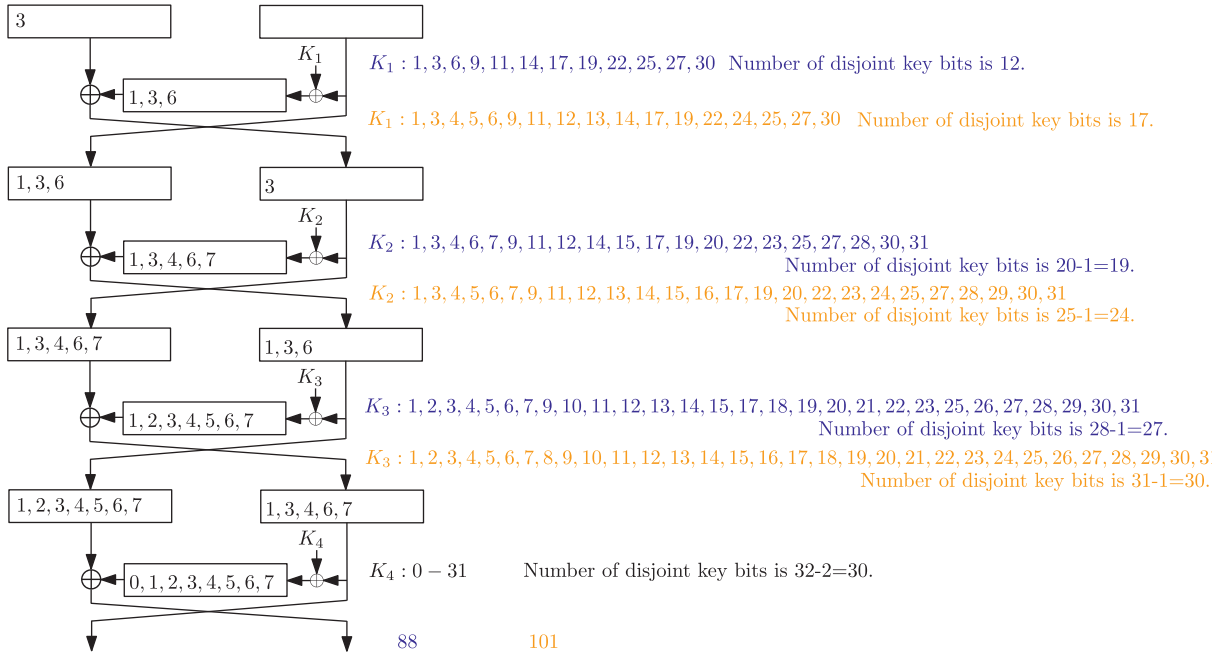


Figure 7: Disjoint subkey bits involved in the propagation of the fourth nibble through four rounds.

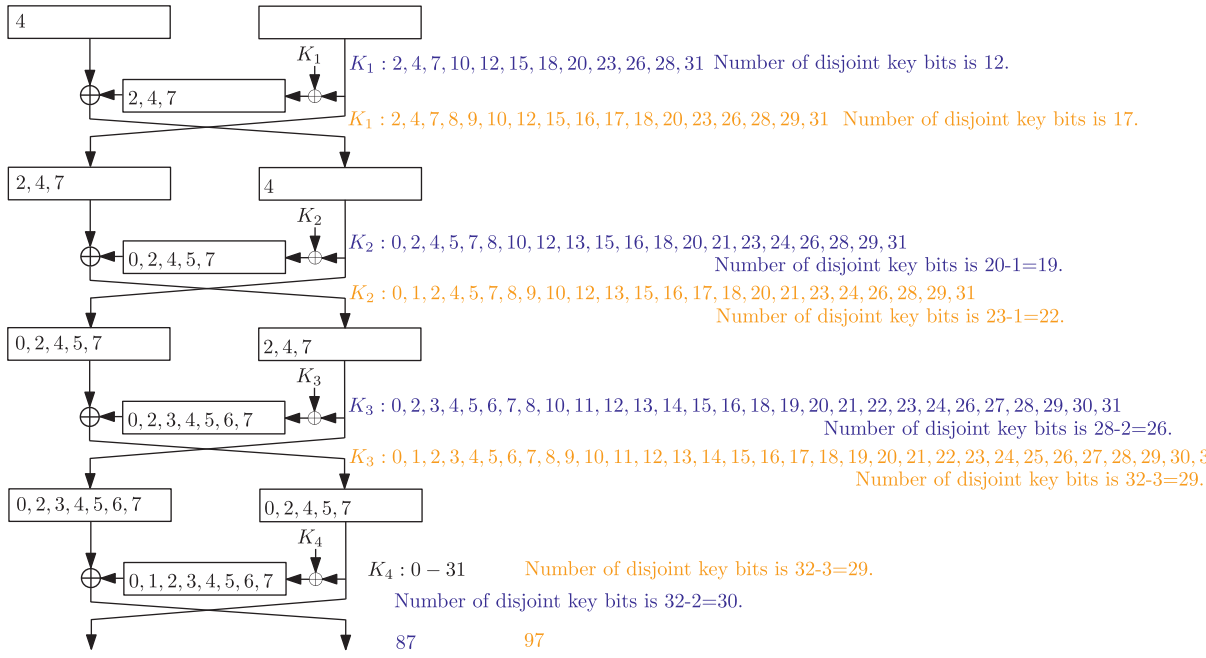


Figure 8: Disjoint subkey bits involved in the propagation of the fifth nibble through four rounds.

$(\Gamma X_7, \ldots, \Gamma X_1, \Gamma X_0)$ and $\Gamma Y = (\Gamma Y_7, \ldots, \Gamma Y_1, \Gamma Y_0)$, then the following is obtained.

(1) The eight 4-bit nibbles of the output mask ΓY can be expressed with the eight 4-bit nibbles of the input mask ΓX as follows:

$$\begin{aligned}
&\Gamma X_0 = \Gamma Y_0 \oplus \Gamma Y_2 \oplus \Gamma Y_5, \Gamma X_1 = \Gamma Y_1 \oplus \Gamma Y_3 \oplus \Gamma Y_6,\\
&\Gamma X_2 = \Gamma Y_2 \oplus \Gamma Y_4 \oplus \Gamma Y_7, \Gamma X_3 = \Gamma Y_0 \oplus \Gamma Y_3 \oplus \Gamma Y_5,\\
&\Gamma X_4 = \Gamma Y_1 \oplus \Gamma Y_4 \oplus \Gamma Y_6, \Gamma X_5 = \Gamma Y_2 \oplus \Gamma Y_5 \oplus \Gamma Y_7,\\
&\Gamma X_6 = \Gamma Y_0 \oplus \Gamma Y_3 \oplus \Gamma Y_6, \Gamma X_7 = \Gamma Y_1 \oplus \Gamma Y_4 \oplus \Gamma Y_7.
\end{aligned} \tag{10}$$

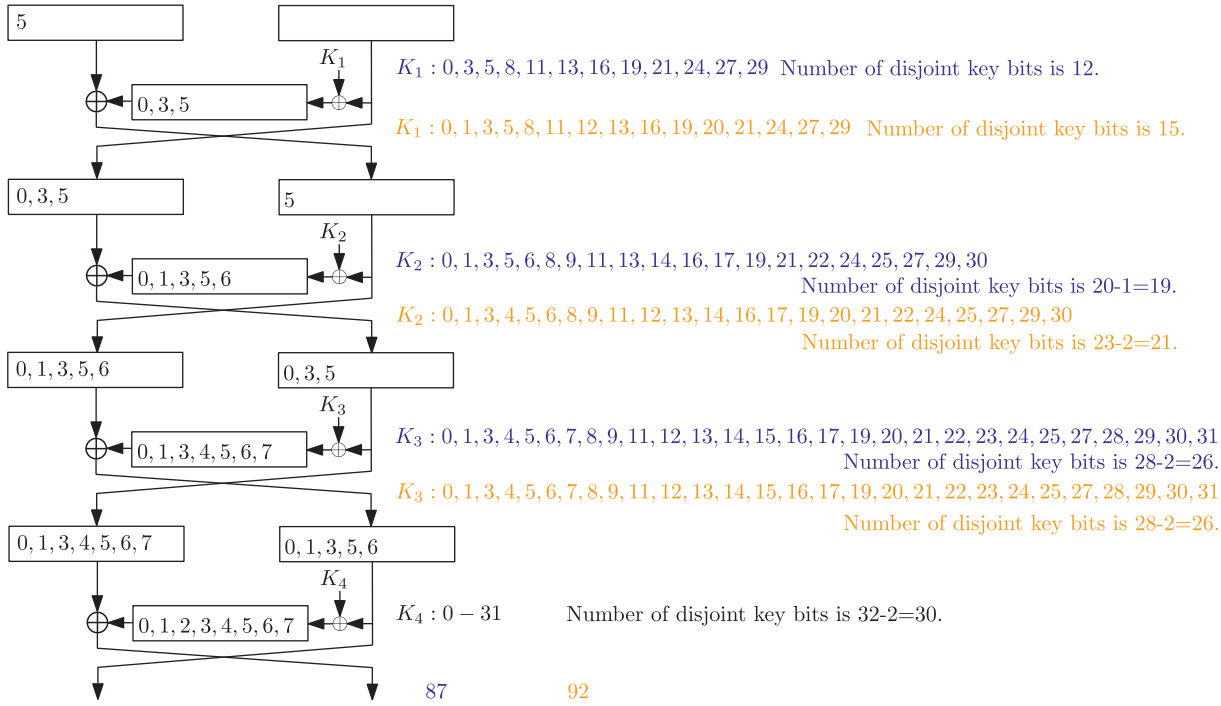


Figure 9: Disjoint subkey bits involved in the propagation of the sixth nibble through four rounds.

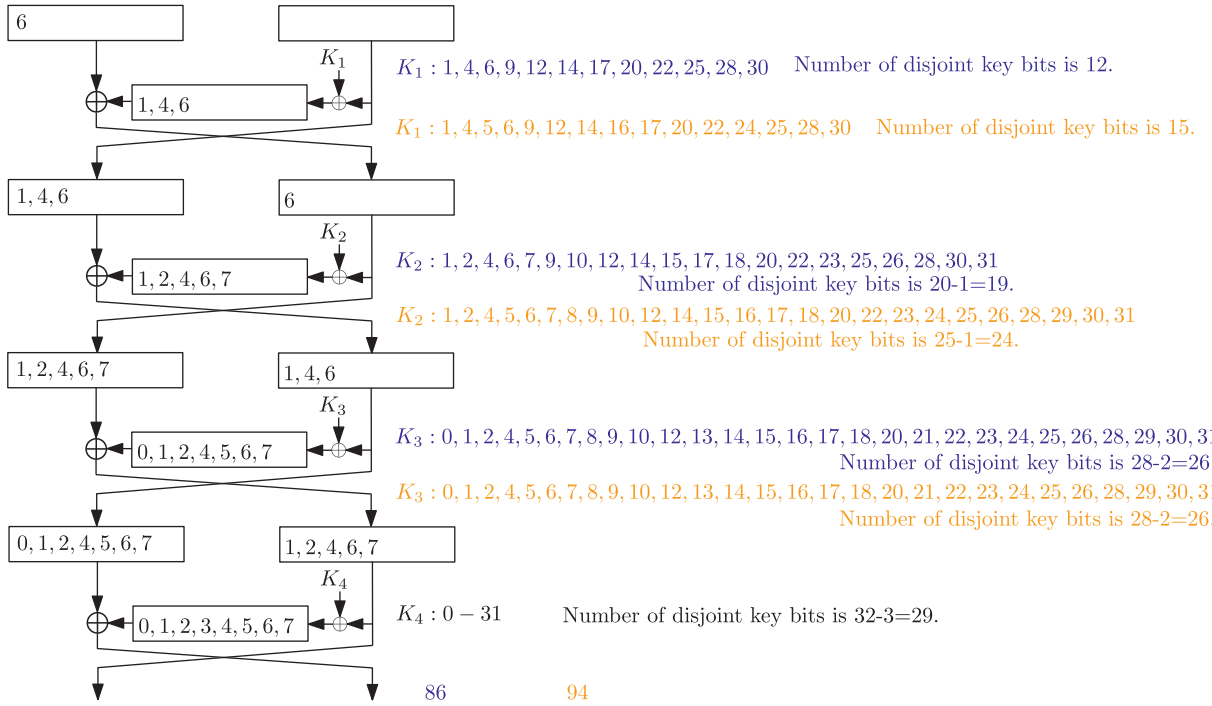


Figure 10: Disjoint subkey bits involved in the propagation of the seventh nibble through four rounds.

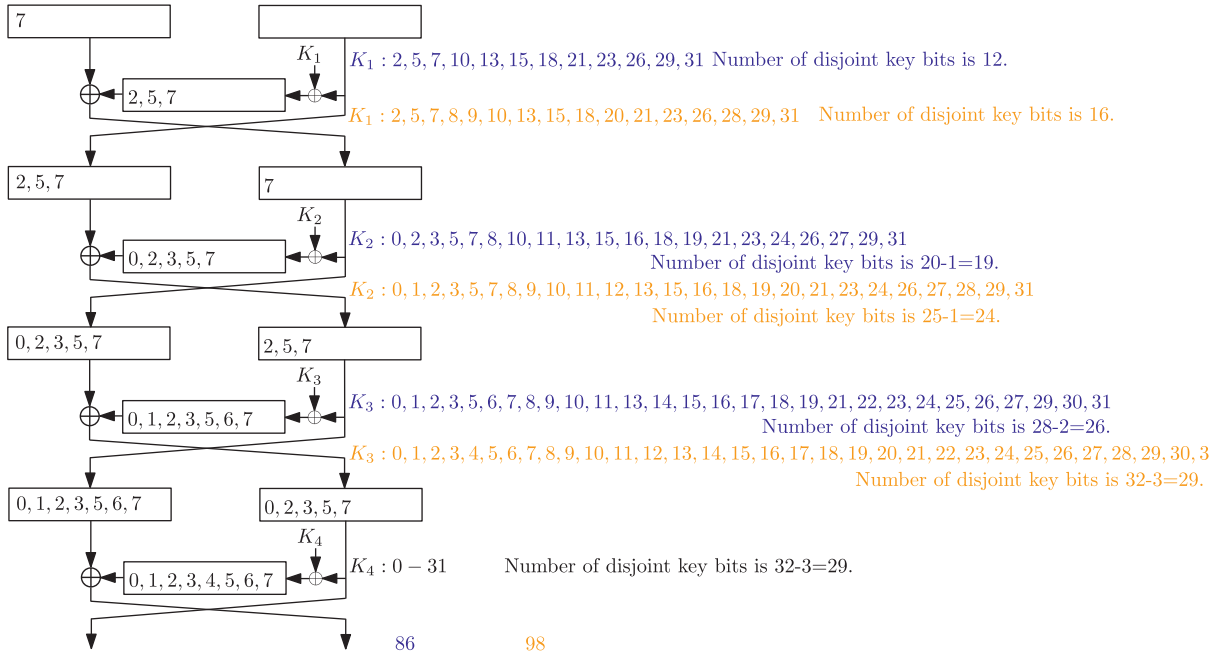


FIGURE 11: Disjoint subkey bits involved in the propagation of the eighth nibble through four rounds.

(2) The eight 4-bit nibbles of the input mask ΓX can be expressed with the eight 4-bit nibbles of the output mask ΓY as follows:

$$\begin{aligned}
\Gamma Y_0 &= \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_4 \oplus \Gamma X_5,\\
\Gamma Y_1 &= \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6,\\
\Gamma Y_2 &= \Gamma X_3 \oplus \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7,\\
\Gamma Y_3 &= \Gamma X_0 \oplus \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7,\\
\Gamma Y_4 &= \Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7,\\
\Gamma Y_5 &= \Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_6 \oplus \Gamma X_7,\\
\Gamma Y_6 &= \Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_7,\\
\Gamma Y_7 &= \Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_4.
\end{aligned} \tag{11}$$

Proof. By Property 1 (1), we have

$$\begin{aligned}
&\sum_{i=0}^{7} (\Gamma Y_i \odot Y_i)\\
&= \Gamma Y_0 \odot (X_0 \oplus X_3 \oplus X_6) \oplus \Gamma Y_1 \odot (X_1 \oplus X_4 \oplus X_7) \oplus\\
&\quad \Gamma Y_2 \odot (X_0 \oplus X_2 \oplus X_5) \oplus \Gamma Y_3 \odot (X_1 \oplus X_3 \oplus X_6) \oplus\\
&\quad \Gamma Y_4 \odot (X_2 \oplus X_4 \oplus X_7) \oplus \Gamma Y_5 \odot (X_0 \oplus X_3 \oplus X_5) \oplus\\
&\quad \Gamma Y_6 \odot (X_1 \oplus X_4 \oplus X_6) \oplus \Gamma Y_7 \odot (X_2 \oplus X_5 \oplus X_7) \oplus\\
&= X_0 \odot (\Gamma Y_0 \oplus \Gamma Y_2 \oplus \Gamma Y_5) \oplus X_0 \odot (\Gamma Y_1 \oplus \Gamma Y_3 \oplus \Gamma Y_6) \oplus\\
&\quad X_2 \odot (\Gamma Y_2 \oplus \Gamma Y_4 \oplus \Gamma Y_7) \oplus X_0 \odot (\Gamma Y_0 \oplus \Gamma Y_3 \oplus \Gamma Y_5) \oplus\\
&\quad X_4 \odot (\Gamma Y_1 \oplus \Gamma Y_4 \oplus \Gamma Y_6) \oplus X_0 \odot (\Gamma Y_2 \oplus \Gamma Y_5 \oplus \Gamma Y_7) \oplus\\
&\quad X_6 \odot (\Gamma Y_0 \oplus \Gamma Y_3 \oplus \Gamma Y_6) \oplus X_0 \odot (\Gamma Y_1 \oplus \Gamma Y_4 \oplus \Gamma Y_7) \oplus.
\end{aligned} \tag{12}$$

By Property 1 (2), we have

$$\begin{aligned}
&\sum_{i=0}^{7} (\Gamma X_i \odot X_i) = \Gamma X_0 \odot (Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7) \oplus\\
&\quad \Gamma X_1 \odot (Y_0 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7) \oplus \Gamma X_2 \odot (Y_0 \oplus Y_1 \oplus Y_5 \oplus Y_6 \oplus Y_7) \oplus\\
&\quad \Gamma X_3 \odot (Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_6 \oplus Y_7) \oplus \Gamma X_4 \odot (Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_7) \oplus\\
&\quad \Gamma X_5 \odot (Y_0 \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4) \oplus \Gamma X_6 \odot (Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5) \oplus\\
&\quad \Gamma X_7 \odot (Y_1 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6) \oplus\\
&= Y_0 \odot (\Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_4 \oplus \Gamma X_5) \oplus Y_1 \odot (\Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6) \oplus\\
&\quad Y_2 \odot (\Gamma X_3 \oplus \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7) \oplus Y_3 \odot (\Gamma X_0 \oplus \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7) \oplus\\
&\quad Y_4 \odot (\Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7) \oplus Y_5 \odot (\Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_6 \oplus \Gamma X_7) \oplus\\
&\quad Y_6 \odot (\Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_7) \oplus Y_7 \odot (\Gamma X_0 \oplus \Gamma X_1 \oplus \Gamma X_2 \oplus \Gamma X_3 \oplus \Gamma X_4) \oplus.
\end{aligned} \tag{13}$$

Table 3: The combined difference distribution (CDD) table of the key selected S-box.$\mathcal{S}$

Input difference	Output difference															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0	(16, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)
0x1	(0, 4)	(0, 1) (2, 2) (4, 1)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 1) (2, 3)	(0, 3) (4, 1)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 3) (2, 1)
0x2	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 1) (4, 1)	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 1) (2, 2) (4, 1)	(0, 2) (2, 2)	(0, 3) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 1) (2, 3)	(0, 3) (2, 1)
0x3	(0, 4)	(0, 2) (2, 2)	(2, 3) (4, 1)	(0, 1) (2, 2) (4, 1)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 1) (2, 2) (4, 1)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 3) (2, 1)
0x4	(0, 4)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 1) (2, 3)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)
0x5	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 4)	(2, 4)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 1) (2, 3)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 1) (2, 3)
0x6	(0, 4)	(0, 4)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 1) (2, 2) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 3) (4, 1)	(0, 1) (2, 3)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 1) (2, 2) (4, 1)	(0, 3) (4, 1)
0x7	(0, 4)	(0, 1) (2, 3)	(0, 3) (4, 1)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 1) (2, 3)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 3) (2, 1)
0x8	(0, 4)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 3) (4, 1)	(0, 3) (4, 1)	(0, 2) (2, 2)	(0, 1) (2, 2) (4, 1)	(0, 1) (2, 3)	(0, 4)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 1) (2, 3)	(0, 1) (2, 2) (4, 1)	(0, 3) (2, 1)
0x9	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 1) (2, 2) (4, 1)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)
0xA	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 1) (2, 2) (4, 1)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 3) (2, 1)	(0, 1) (2, 3)	(0, 1) (2, 3)	(0, 2) (2, 1) (4, 1)
0xB	(0, 4)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 3) (4, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 1) (2, 3)	(0, 1) (2, 2) (4, 1)
0xC	(0, 4)	(0, 2) (2, 1) (4, 1)	(0, 4)	(0, 3) (4, 1)	(0, 3) (4, 1)	(0, 2) (2, 2)	(2, 4)	(0, 2) (2, 1) (4, 1)	(0, 4)	(0, 1) (2, 2) (4, 1)	(0, 3) (4, 1)	(0, 3) (2, 1)	(2, 4)	(0, 1) (2, 3)	(0, 4)	(0, 2) (2, 2)
0xD	(0, 4)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 3) (4, 1)	(0, 3) (2, 1)	(0, 1) (2, 3)	(0, 3) (2, 1)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 1) (4, 1)
0xE	(0, 4)	(0, 3) (2, 1)	(0, 3) (4, 1)	(0, 3) (2, 1)	(2, 3) (4, 1)	(0, 3) (2, 1)	(0, 4)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 1) (2, 3)	(0, 1) (2, 3)	(0, 2) (2, 2)	(2, 4)	(0, 2) (2, 2)	(0, 2) (2, 2)	(0, 2) (2, 2)
0xF	(0, 4)	(0, 2) (2, 1) (4, 1)	(0, 3) (4, 1)	(0, 3) (4, 1)	(0, 3) (2, 1)	(0, 1) (2, 3)	(0, 2) (2, 2)	(0, 3) (2, 1)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 2)	(0, 2) (2, 1) (4, 1)	(0, 2) (2, 1) (4, 1)	(0, 3) (4, 1)	(0, 3) (4, 1)	(0, 3) (2, 1)	(0, 3) (4, 1)

Thus, the results follow trivially.

As mentioned earlier, a key selected S-box also makes it difficult for an attacker to apply linear cryptanalysis. Here, we similarly develop a conservative framework for the linear cryptanalysis of block ciphers using a key selected S-box, which is based on the concept of the combined bias distribution (CBD) table for a key selected S-box as follows. □

Definition 3. The combined bias distribution (CBD) table for a key selected S-box: $\{0,1\}^m \times \{0,1\}^v \longrightarrow \{0,1\}^q$ (for specific values of m, v, and q) is a table with 2^m rows being the 2^m possible input masks, 2^q columns being the 2^q output masks, and the (i, j)th entry being the set of the possible combinations (the number of m-bit inputs satisfying the input mask and output mask pair (i, j) under an ordinary

Table 4: The difference distribution tables of $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ S-boxes.

Input difference	Output difference															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
Difference distribution table of $\mathcal{S}_0$																
0x0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	2	2	0	2	4	2	0	2	2	0
0x2	0	0	0	0	0	2	0	2	0	4	0	4	2	0	2	0
0x3	0	2	4	2	2	0	0	2	0	0	0	0	2	2	0	0
0x4	0	0	0	0	0	2	4	2	0	0	0	0	2	0	2	4
0x5	0	0	0	0	0	2	2	0	4	2	0	2	2	2	0	0
0x6	0	0	0	0	4	0	0	0	0	4	0	4	0	0	0	4
0x7	0	2	4	2	2	2	0	0	0	0	0	0	0	2	2	0
0x8	0	0	0	4	0	0	2	2	0	2	0	2	0	2	2	0
0x9	0	2	0	2	0	2	0	2	2	0	2	0	2	0	2	0
0xA	0	0	4	0	2	2	0	0	2	0	2	0	0	2	2	0
0xB	0	0	0	0	4	2	0	2	0	0	0	0	2	0	2	4
0xC	0	4	0	0	0	2	2	0	0	2	0	2	2	2	0	0
0xD	0	2	0	2	0	0	4	0	2	0	2	0	0	0	0	4
0xE	0	0	4	0	2	0	0	2	2	0	2	0	2	2	0	0
0xF	0	4	0	4	0	0	0	0	4	0	4	0	0	0	0	0
Difference distribution table of $\mathcal{S}_1$																
0x0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	4	0	2	2	4	0	0	2	0	0	0	0	0	0	2
0x2	0	2	4	0	0	0	0	2	2	0	0	0	0	4	2	0
0x3	0	0	2	4	0	2	0	0	2	2	0	2	0	2	0	0
0x4	0	0	2	4	4	0	2	0	0	0	2	0	0	0	2	0
0x5	0	0	0	0	0	2	2	4	0	0	0	4	2	0	0	2
0x6	0	0	0	2	2	2	2	0	2	0	0	0	2	0	4	0
0x7	0	2	0	0	0	2	2	2	0	2	2	2	0	2	0	0
0x8	0	0	2	0	0	2	2	2	0	2	2	2	0	0	2	0
0x9	0	2	2	2	2	0	0	0	0	0	4	2	0	0	0	2
0xA	0	0	0	0	0	0	0	4	4	2	2	0	2	0	0	2
0xB	0	0	2	0	0	0	2	0	2	2	0	2	2	2	0	2
0xC	0	2	0	0	4	0	2	0	0	2	0	0	2	2	0	2
0xD	0	4	2	0	0	0	0	2	2	0	0	0	0	2	4	0
0xE	0	0	0	2	2	0	0	0	0	2	2	2	2	2	2	0
0xF	0	0	0	0	0	2	2	0	0	2	2	0	4	0	0	4
Difference distribution table of $\mathcal{S}_2$																
0x0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	2	2	2	2	0	0	0	0	0	2	0	4	0	2	0
0x2	0	2	2	2	0	0	2	0	2	0	2	2	0	0	0	2
0x3	0	2	2	2	2	0	2	2	2	0	0	0	0	2	0	0
0x4	0	4	0	0	0	0	0	0	2	2	0	4	2	2	0	0
0x5	0	2	2	0	0	2	0	2	0	2	2	0	0	2	0	2
0x6	0	0	0	2	0	0	2	4	2	2	2	0	0	0	2	0
0x7	0	0	0	0	0	2	2	0	0	2	4	2	2	2	0	0
0x8	0	0	2	0	0	0	0	2	0	0	0	2	2	2	4	2
0x9	0	0	4	0	0	2	2	0	2	2	0	0	2	0	2	0
0xA	0	2	0	0	4	0	2	0	0	2	0	0	0	0	2	4
0xB	0	0	0	2	0	2	0	0	2	0	4	0	0	4	2	0
0xC	0	0	0	4	0	2	2	4	0	0	0	0	2	0	0	2
0xD	0	0	2	2	2	2	0	0	0	2	0	2	0	2	0	2
0xE	0	0	0	0	4	2	0	2	2	2	0	0	2	0	0	2
0xF	0	2	0	0	2	2	2	0	2	0	0	4	0	0	2	0
Difference distribution table of $\mathcal{S}_3$																
0x0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	2	2	2	2	0	0	0	2	4	0	0	2	0	0	0
0x2	0	0	0	4	0	2	2	4	0	0	0	0	0	2	2	0
0x3	0	0	2	0	2	0	0	4	0	0	0	2	2	0	2	2
0x4	0	2	0	2	0	0	0	0	0	0	2	2	2	4	0	2
0x5	0	2	2	2	0	2	0	0	2	0	0	0	2	0	2	2
0x6	0	0	2	0	2	0	0	0	2	0	2	2	2	2	2	0

Table 4: Continued.

Input difference	Output difference															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x7	0	2	0	2	2	0	2	0	2	0	0	2	2	0	0	2
0x8	0	2	0	0	4	2	4	0	0	2	0	0	0	2	0	0
0x9	0	0	0	0	0	4	2	2	2	2	0	0	0	0	0	4
0xA	0	2	2	2	0	0	0	2	2	0	2	2	0	0	2	0
0xB	0	2	2	0	0	0	4	0	2	0	2	0	0	0	2	2
0xC	0	0	0	0	0	0	2	2	0	4	4	0	2	2	0	0
0xD	0	0	0	2	2	4	0	0	2	0	2	2	0	0	2	0
0xE	0	2	0	0	2	0	0	0	0	2	2	2	2	0	2	2
0xF	0	0	4	0	0	2	0	2	0	2	0	2	0	4	0	0

Table 5: The minimum number of active S-boxes of i-round differential characteristics under the CDD table, where $(1 \le i \le 18)$.

#Rounds	#S-boxes
1	0
2	1
3	2
4	5
5	7
6	10
7	12
8	14
9	15
10	16
11	19
12	21
13	24
14	26
15	28
16	29
17	30
18	33
19	35
20	38

$m \times q$-bit S-box and the number of ordinary $m \times q$-bit S-boxes that have the number of m-bit inputs satisfying the input mask and output mask pair (i, j)), where $0 \le i \le 2^m - 1$ and $0 \le j \le 2^q - 1$.

Likewise, we can compute the CBD table for the key selected S-box $\mathcal{S}$ used in LBC, which is given as Table 6. Each entry except $(0, 0)$ has at most five combinations, namely, 0, ± 2, and ± 4, which follow the bias distribution tables of the four ordinary S-boxes (see Table 7). Note that one may enhance the combined difference distribution table by associating every combination with its corresponding probability/probabilities.

Now by treating the eight key selected S-boxes in the S-box layer of LBC as eight identical ordinary S-boxes with the bias distribution table being the CBD table, we can check the minimum number of active S-boxes for a linear approximation of a certain number of rounds, in a manner similar to that Matsui used for DES (under the general assumption for linear cryptanalysis) in [37]. As each ordinary S-box has a maximum (valid) bias probability of 2^{-2}, we can obtain an upper bound for a linear approximation of a certain number of rounds and get its security against linear cryptanalysis. Clearly, the upper bound is overestimated, since it is based on the CBD table, and each of the four ordinary bias distribution tables is only a subset of the combined bias distribution table. By this way, we can bound the security against linear cryptanalysis in the worst case from the point of the user of the cipher.

We made a computer program to compute the minimum numbers of active S-boxes of i-round linear approximations under the CBD table $(1 \le i \le 5)$, and the results for 1, 2, 3, 4, and 5 rounds are 0, 1, 2, 5, and 8, respectively (it is rather time consuming for 6 or more rounds). Thus, a 20-round linear approximation has a minimum of $4 \times 8 = 32$ active S-boxes, and 32 active S-boxes have at most a bias of $2^{32-1} \times (2^{-2})^{32} = 2^{-33}$, which is not valid for a linear cryptanalysis attack because $2^{-33} < 2^{-32}$. As a result, we can assume at most a 20-round linear approximation and can assume appending at most a total of five rounds at both ends, since a total of five rounds appended at both ends will indicate at least 3 rounds in an end, which would require an attacker to guess all the remaining 62 key bits. Remind that multiple linear cryptanalysis does not work well in the key selected S-box mechanism because a different linear approximation requires a different set of selecting key bits, which would further shrink the space of remaining key bits that can be guessed in the key recovery phase. Therefore, 25-round LBC should be secure against linear cryptanalysis.

6.4. Impossible Differential Cryptanalysis. Impossible differential cryptanalysis [38, 39] is a special case of differential cryptanalysis, which is based on a differential with a zero probability. Here, we analyse the security of LBC against impossible differential cryptanalysis.

Consider a plaintext pair with difference $(L_0, R_0) = (\Delta x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where x is a nonzero 4-bit value.

The output difference of Round 1 is of the form $(\Delta L_1, \Delta R_1) = (0, 0, 0, 0, 0, 0, 0, 0, \Delta x, 0, 0, 0, 0, 0, 0, 0)$. The output difference of Round 2 is of the form $(\Delta L_2, \Delta R_2) = (\Delta x, 0, 0, 0, 0, 0, 0, 0, \Delta y, 0, 0, \Delta y, 0, 0, \Delta y, 0)$, where $\Delta y = \mathcal{S}(\Delta x)$. The output difference of Round 3 is of the form $(\Delta L_3, \Delta R_3) = (\Delta y, 0, 0, \Delta y, 0, 0, \Delta y, 0, ?, ?, 0, ?, \star, 0, ?, 0)$, where the question mark '?' denotes a 4-bit indeterminate value that can be zero or nonzero, and $\star$ denotes a nonzero

Table 6: The combined bias distribution (CBD) table of the key selected S-box.$\mathcal{S}$

Input mask	Output mask															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0	(8, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)	(0, 4)
0x1	(0,4)	(−4,1) (−2,1) (0,2)	(0, 1) (+2,3)	(−2,2) (0,1) (+4,1)	(−2,1) (0,3)	(0,4)	(−2,2) (0,1) (+4,1)	(−2,1) (0,1) (+2,2)	(−2,1) (0,2) (+2,1)	(−2,3) (+4,1)	(−2,3) (0,1)	(−4,1) (−2,1) (+2,1)	(−2,2) (0,1) (+2,1)	(−2,1) (0,1) (+2,1)	(−2,1) (0,1) (+2,1)	(−2,1) (0,1) (+2,2)
0x2	(0,4)	(−2,1) (0,3)	(−4,1) (−2,1) (0,2)	(−2,1 (0,2) (+2,1)	(−2,2) (0,1) (+2,1)	(−2,1) (0,2) (+2,1)	(−2,1) (0,1) (+2,1) (+4,1)	(0,2) (+2,1) (+4,1)	(−2,1) (0,1) (+2,1) (+4,1)	(−2,1) (0,1) (+2,2)	(−2,1) (0,2) (+4,1)	(−2,1) (0,2) (+2,1)	(−2,1) (0,2) (+4,1)	(−4,1) (0,3)	(−4,1) (−2,1) (0,1) (+2,1)	(−4,1) (−2,1) (0,1) (+4,1)
0x3	(0,4)	(−2,1) (0,2) (+2,1)	(−2,2) (+2,2)	(−2,1) (0,2) (+2,1)	(−2,2) (0,2)	(−2,1) (0,2) (+2,1)	(0,2) (+2,2)	(−2,2) (0,1) (+4,1)	(−2,1) (0,1) (+2,1) (+4,1)	(−4,1) (0,2) (+4,1)	(0,2) (+2,2)	(−2,2) (+2,2)	(−4,1) (−2,2) (0,1)	(−2,1) (0,2) (+2,1)	(−4,1) (−2,1) (+2,1) (+4,1)	(−4,1) (0,1) (+2,2)
0x4	(0,4)	(−2,2) (0,1) (+4,1)	(0,3) (+2,1)	(−2,2) (0,1) (+2,1)	(−2,1) (0,2) (+4,1)	(−4,1) (−2,2) (+2,1)	(−2,2) (0,1) (+4,1)	(−2,1) (+2,3)	(−2,1) (+2,2) (+4,1)	(−2,1) (0,1) (+2,2)	(−2,1) (0,2) (+2,1)	(−4,1) (0,1) (+2,2)	(−2,1) (0,1) (+2,1)	(−4,1) (0,1) (+2,2)	(0,3) (+2,1)	(0,3) (+2,1)
0x5	(0,4)	(−2,1) (0,3)	(−2,2) (0,2)	(−2,1) (0,3)	(−4,1) (−2,1) (0,1) (+2,1)	(−4,1) (−2,2) (+2,1)	(−2,2) (0,1) (+4,1)	(−4,1) (0,1) (+2,1) (+4,1)	(0,2) (+2,1) (+4,1)	(−4,1) (0,3)	(0,2) (+2,1) (+4,1)	(−2,1) (0,2) (+2,1)	(−2,2) (0,1) (+2,1)	(−4,1) (−2,1) (0,1) (+2,1)	(−4,1) (0,2) (+2,1)	(−2,1) (0,2) (+2,1)
0x6	(0,4)	(−4,1) (+2,3)	(−4,1) (−2,1) (0,1) (+2,1)	(−2,1) (0,1) (+2,2)	(−4,1) (−2,1) (0,1) (+2,1)	(0,3) (+2,1)	(−2,1) (0,2) (+2,1)	(−2,1) (0,1) (+2,2)	(−2,2) (0,1) (+2,1)	(−4,1) (0,3)	(−2,3) (+4,1)	(−4,1) (0,1) (+2,2)	(−2,1) (0,3)	(−2,2) (0,1) (+4,1)	(−2,2) (0,1) (+2,1)	(−4,1) (−2,1) (0,1) (+2,1)
0x7	(0,4)	(0,1) (+2,2) (+4,1)	(−2,2) (0,1) (+2,1)	(−4,1) (−2,2) (+2,1)	(−2,1) (0,3)	(−4,1) (−2,1) (0,2)	(−2,2) (+2,2)	(−4,1) (−2,2) (0,1)	(0,1) (+2,3)	(−2,1) (+2,2)	(−4,1) (0,3)	(−2,2) (0,1) (+4,1)	(−2,2) (+2,2)	(−4,1) (−2,1) (0,1)	(0,3) (+2,1)	(0,3) (+2,1)
0x8	(0,4)	(−2,1) (0,2) (+2,1)	(−2,2) (+2,2)	(−2,2) (0,1) (+4,1)	(0,1) (+2,2) (+4,1)	(0,2) (+2,2)	(−2,2) (0,2)	(−2,2) (0,1) (+4,1)	(0,1) (+2,2) (+4,1)	(0,3) (+4,1)	(−2,2) (0,2)	(−2,2) (+2,2)	(−2,1) (0,1) (+2,1)	(−4,1) (−2,1) (0,1)	(−4,1) (−2,1) (0,1)	(0,2) (+2,2)
0x9	(0,4)	(−4,1) (−2,1) (0,2)	(0,3) (+2,1)	(−4,1) (0,3)	(−2,1) (0,2) (+4,1)	(−4,1) (−2,2) (0,1)	(−2,1) (0,1) (+2,1) (+4,1)	(−4,1) (0,1) (+2,1) (+4,1)	(−2,2) (0,2)	(−2,2) (0,1) (+2,1)	(−4,1) (−2,1) (+2,2)	(−2,1) (0,2) (+2,1)	(−2,1) (0,2) (+4,1)	(−2,1) (0,2) (+2,1)	(0,2) (+2,2)	(−4,1) (−2,1) (0,2)
0xA	(0,4)	(−4,1) (−2,1) (+2,2)	(−4,1) (−2,2) (+2,1)	(−4,1) (0,1) (+2,2)	(0,2) (+2,1) (+4,1)	(−4,1) (−2,1) (0,1) (+2,1)	(−2,2) (+2,2)	(−2,1) (0,3)	(0,4)	(−2,3) (0,1)	(−4,1) (−2,2) (+2,1)	(−2,2) (0,2)	(−4,1) (−2,1) (0,2)	(0,2) (+2,2)	(−2,1) (+2,3)	(0,2) (+2,1) (+4,1)
0xB	(0,4)	(−2,2) (0,1) (+4,1)	(0,3) (+4,1)	(−2,1) (0,1) (+2,1) (+4,1)	(−2,2) (0,1) (+4,1)	(−4,1) (0,1) (+2,2)	(−2,1) (0,2) (+2,1)	(−2,1) (0,2) (+2,1)	(−4,1) (−2,1) (0,1) (+2,1)	(−4,1) (0,3)	(−2,1) (0,1) (+2,1) (+4,1)	(−4,1) (0,3)	(−4,1) (−2,2) (0,1)	(−2,1) (0,2) (+2,1)	(−2,2) (0,2)	(−2,2) (0,2)
0xC	(0,4)	(−2,1) (0,2) (+2,1)	(−4,1) (−2,2) (+2,1)	(0,2) (+2,1) (+4,1)	(−4,1) (+2,3)	(0,1) (+2,3)	(0,2) (+2,2)	(−2,2) (0,2)	(−2,1) (+2,2) (+4,1)	(−2,2) (0,1) (+2,1)	(0,2) (+2,2)	(0,1) (+2,2) (+4,1)	(0,1) (+2,2) (+4,1)	(−4,1) (0,3)	(0,2) (+2,1) (+4,1)	(−2,2) (0,1) (+2,1)
0xD	(0,4)	(−2,2) (0,1) (+2,1)	(−2,2) (0,1) (+4,1)	(−2,1) (0,1) (+2,2)	(−2,1) (0,2) (+2,1)	(−2,2) (+2,1) (+4,1)	(0,3) (+4,1)	(−2,1) (0,1) (+2,2)	(−2,1) (0,1) (+2,2)	(−4,1) (0,3)	(−2,3) (0,1)	(−2,1) (0,2) (+2,1)	(−2,1) (0,1) (+2,2)	(−4,1) (−2,2) (0,1)	(−2,1) (+2,2) (+4,1)	(−4,1) (0,2) (+4,1)
0xE	(0,4)	(−4,1) (−2,2) (+2,1)	(−2,1) (0,1) (+2,1) (+4,1)	(−2,1) (0,1) (+2,2)	(−4,1) (0,3)	(−2,2) (0,1) (+2,1)	(−2,1) (0,1) (+2,1) (+4,1)	(−2,3) (0,1)	(−2,1) (0,1) (+2,2)	(0,3) (+4,1)	(−2,3) (+4,1)	(−4,1) (−2,1) (0,1) (+2,1)	(0,1) (+2,3)	(0,4)	(−2,2) (0,1) (+2,1)	(−2,1) (0,1) (+2,1) (+4,1)
0xF	(0,4)	(−2,2) (+2,2)	(−4,1) (−2,1) (0,2)	(−4,1) (−2,1) (+2,2)	(−2,1) (0,2) (+4,1)	(−2,2) (+2,1) (+4,1)	(−4,1) (−2,1) (0,1) (+2,1)	(−2,1) (0,2) (+2,1)	(−4,1) (0,2) (+2,1)	(−2,1) (0,1) (+2,2)	(−2,2) (0,2)	(0,2) (+2,2)	(−2,1) (0,2) (+2,1)	(−2,1) (0,1) (+2,1) (+4,1)	(0,1) (+2,3)	(−4,1) (0,2) (+2,1)

Table 7: The bias distribution tables of $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ S-boxes.

Input mask	Output mask															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
Bias distribution table of $\mathcal{S}_0$																
0x0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	0	2	−2	0	0	−2	2	0	4	−2	−2	0	4	2	2
0x2	0	0	0	0	0	0	4	4	0	0	0	0	0	0	−4	4
0x3	0	0	2	−2	0	0	2	−2	4	0	2	2	−4	0	2	2
0x4	0	−2	0	−2	0	2	0	2	-2	0	2	−4	−2	−4	2	0
0x5	0	−2	−2	0	−4	−2	−2	4	2	0	0	2	−2	0	0	−2
0x6	0	2	0	2	−4	2	0	−2	2	0	−2	−4	−2	0	−2	0
0x7	0	2	−2	−4	0	−2	−2	0	2	−4	0	−2	2	0	0	2
0x8	0	0	−2	−2	0	0	−2	−2	0	4	−2	2	0	−4	−2	2
0x9	0	0	0	−4	0	0	4	0	0	0	−4	0	0	0	0	−4
0xA	0	−4	−2	2	0	−4	2	−2	0	0	−2	−2	0	0	2	2
0xB	0	4	0	0	0	−4	0	0	-4	0	0	0	−4	0	0	0
0xC	0	2	−2	0	−4	2	2	0	-2	0	0	2	2	0	4	2
0xD	0	2	4	2	0	−2	0	2	2	0	−2	0	2	−4	2	0
0xE	0	2	−2	0	0	−2	2	0	2	4	4	−2	2	0	0	−2
0xF	0	2	−4	2	4	2	0	2	2	0	−2	0	−2	0	2	0
Bias distribution table of $\mathcal{S}_1$																
0x0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	−4	2	−2	0	0	−2	−2	2	−2	0	4	−2	−2	0	0
0x2	0	−2	0	2	−2	0	2	0	4	2	4	−2	−2	0	2	0
0x3	0	2	2	0	−2	0	0	−2	2	−4	0	−2	0	2	−2	−4
0x4	0	0	2	2	4	−4	−2	−2	2	2	0	0	2	2	0	0
0x5	0	0	0	0	0	−4	4	0	0	0	0	0	0	−4	−4	0
0x6	0	2	2	0	2	0	0	2	-2	0	4	2	0	−2	2	−4
0x7	0	2	0	−2	−2	0	−2	−4	0	2	0	−2	2	−4	2	0
0x8	0	0	2	−2	2	2	0	4	4	0	−2	−2	2	−2	0	0
0x9	0	−4	0	0	−2	−2	−2	2	-2	−2	2	−2	4	0	0	0
0xA	0	2	2	−4	0	−2	2	0	0	−2	2	0	0	2	2	4
0xB	0	−2	0	−2	4	2	0	−2	-2	0	2	−4	−2	0	−2	0
0xC	0	0	−4	0	2	2	2	−2	2	−2	2	2	4	0	0	0
0xD	0	0	−2	2	2	−2	0	0	0	−4	−2	−2	−2	−2	4	0
0xE	0	−2	4	2	0	2	4	−2	-2	0	−2	0	2	0	2	0
0xF	0	−2	−2	−4	0	−2	2	0	0	2	−2	0	0	2	2	−4
Bias distribution table of $\mathcal{S}_2$																
0x0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	0	0	4	0	0	4	0	−2	−2	−2	2	2	2	−2	2
0x2	0	0	−2	−2	2	2	0	0	−2	−2	0	0	4	−4	−2	−2
0x3	0	0	−2	2	−2	−2	0	4	0	0	2	−2	−2	−2	−4	0
0x4	0	4	0	0	−2	−2	−2	2	2	2	−2	2	4	0	0	0
0x5	0	0	0	0	2	−2	−2	2	0	−4	4	0	2	2	2	2
0x6	0	−4	−2	−2	0	0	−2	2	0	0	−2	2	0	4	−2	−2
0x7	0	0	−2	−2	0	−4	2	−2	2	−2	0	4	−2	−2	0	0
0x8	0	2	2	0	4	2	−2	0	2	0	0	2	−2	0	−4	2
0x9	0	−2	2	0	4	−2	2	4	0	2	−2	0	0	−2	2	0
0xA	0	2	−4	2	2	0	−2	0	0	−2	−4	−2	−2	0	2	0
0xB	0	−2	4	2	−2	0	−2	0	2	−4	−2	0	0	−2	0	−2
0xC	0	−2	−2	4	2	0	0	−2	4	2	2	0	2	0	0	−2
0xD	0	−2	−2	0	−2	4	0	2	2	0	0	2	0	−2	2	4
0xE	0	−2	0	−2	0	−2	0	−2	2	0	−2	−4	2	0	−2	4
0xF	0	−2	0	2	0	−2	−4	−2	−4	2	0	2	0	−2	0	2
Bias distribution table of $\mathcal{S}_3$																
0x0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	0	−2	2	0	−2	0	0	2	0	−2	−2	−4	−2	0	4	−2
0x2	0	0	−4	0	−2	−2	−2	2	2	2	−2	2	0	0	0	−4
0x3	0	−2	−2	0	0	2	2	0	−2	4	0	2	−2	0	4	2
0x4	0	−2	0	−2	0	−2	4	2	4	−2	0	2	0	2	0	2
0x5	0	0	−2	−2	−2	2	0	−4	4	0	2	−2	−2	−2	0	0
0x6	0	2	−4	2	−2	0	2	0	−2	−4	−2	0	0	−2	0	2

Table 7: Continued.

Input mask	Output mask															
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x7	0	4	2	2	0	0	2	−2	2	2	−4	0	−2	2	0	0
0x8	0	−2	−2	4	2	0	0	−2	2	0	0	−2	4	2	2	0
0x9	0	0	0	0	0	−4	0	−4	−2	−2	2	2	−2	2	2	−2
0xA	0	−2	−2	0	4	2	−2	0	0	−2	−2	0	−4	2	−2	0
0xB	0	0	0	4	−2	2	2	2	0	0	4	0	−2	2	−2	−2
0xC	0	0	2	2	2	2	0	0	2	−2	0	4	0	−4	2	−2
0xD	0	−2	0	−2	0	2	4	−2	−2	0	−2	0	2	0	−2	−4
0xE	0	−4	2	2	−4	0	−2	−2	0	0	−2	2	0	0	−2	2
0xF	0	2	0	−2	−2	4	−2	0	0	−2	0	2	2	4	2	0

4-bit value. The output difference of Round 4 is of the form $(\Delta L_4, \Delta R_4) = (?, ?, 0, ?, \star, 0, ?, 0, ?, ?, \star, ?, ?, 0, ?, ?)$. The output difference of Round 5 is of the form $(\Delta L_5, \Delta R_5) = (?, ?, \star, ?, ?, 0, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)$.

On the contrary, given the output difference $(\Delta\widehat{L}_9, \Delta\widehat{R}_9) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \Delta z, 0, 0, 0, 0)$ after Round 9, we can similarly get that the input difference just before Round 5 is of the form $(\Delta\widehat{L}_5, \Delta\widehat{R}_5) = (0, ?, ?, ?, ?, \star, ?, ?, 0, ?, 0, ?, ?, 0, ?, \star)$, where z is a nonzero 4-bit value.

Observe that $L_2^5 = 0$ and $\widehat{L}_2^5 = \star$. Thus, $(\Delta x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \longrightarrow (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \Delta z, 0, 0, 0, 0)$ is an impossible differential, which we denote by $(\Delta x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \nrightarrow (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \Delta z, 0, 0, 0, 0)$.

This 9-round impossible differential of LBC is illustrated in Figure 12, and it can be used to attack at most 19 rounds of LBC, by assuming even 5 rounds at either end. As a result, 25-round LBC should be sufficiently secure against impossible differential cryptanalysis. Note that there also exist similar other 9-round impossible differentials.

6.5. Boomerang and Rectangle Attacks. Boomerang, amplified boomerang, and rectangle attacks [40–42] are variants of differential cryptanalysis, which treat a block cipher as two cascades and use two short differentials with larger probabilities instead of a long differential with a smaller probability. Here, we analyse the security of LBC against boomerang, amplified boomerang, and rectangle attacks.

Typically, differential cryptanalysis is based on the idea of using a long differential characteristic with a usually small probability. Different from the idea of differential cryptanalysis, boomerang attack [42] is based on the idea of using two short differential characteristics with relatively larger probabilities. Suppose two short differential characteristics with probability p and q, respectively; then, p and q should satisfy $p \times q < 2^{-(n/2)}$ to construct a valid boomerang distinguisher, where n is the block size of the concerned cipher. Amplified boomerang and rectangle attacks refine boomerang attack mainly by using more than two differential characteristics with the same input or output difference.

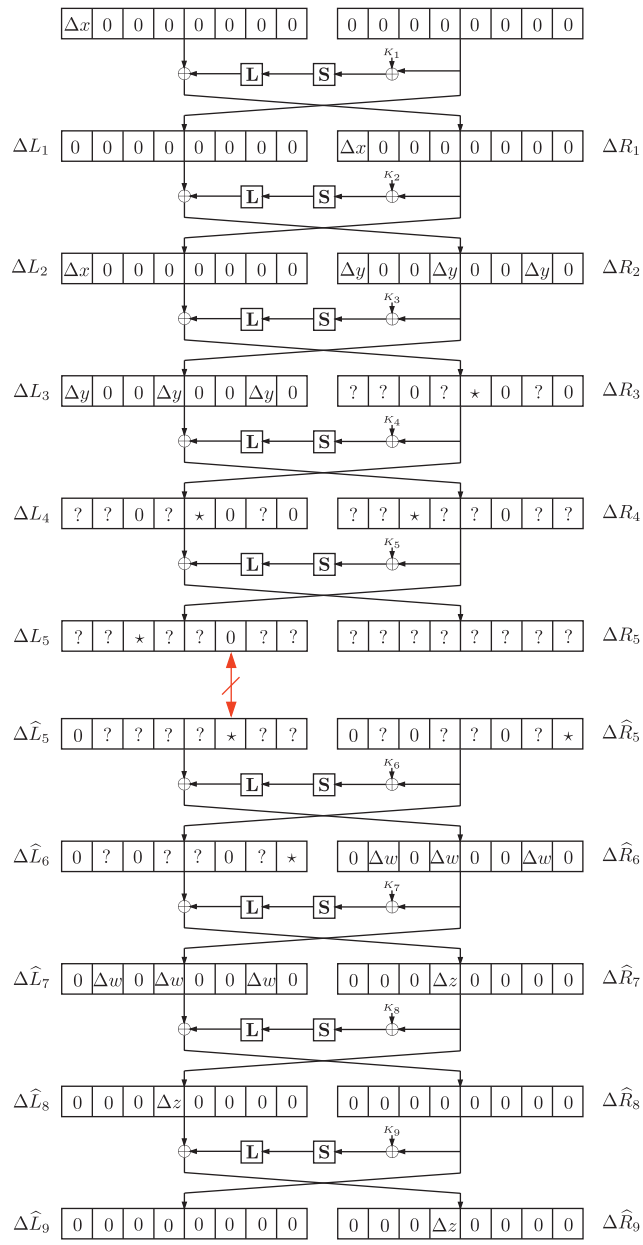


Figure 12: A 9-round impossible differential of LBC.

For LBC, from Table 5, we can learn that an 11-round boomerang distinguisher has a minimum of 16 active S-boxes, which means that the product of the probabilities of two differential characteristics operating on 11 rounds is at most 2^{-32}. Thus, 25-round LBC should be sufficiently secure against boomerang attack as well as amplified boomerang and rectangle attacks.

6.6. Integral Cryptanalysis. Here, we analyse the security of LBC against integral cryptanalysis [43]. Let A denote a 4-bit nibble position which takes all the possible 16 values, B denote a 4-bit nibble position which is balanced (in other words, its XOR sum is zero), C denote a constant 4-bit nibble, and "?" denote a constant 4-bit nibble whose status is unclear about whether it is any of the above three statuses.

Consider a set of 16 plaintexts which takes all the possible 16 values on a certain 4-bit nibble position, say $(L_0, R_0) = (A, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C)$.

The output of Round 1 is of the form $(L_1, R_1) = (C, C, C, C, C, C, C, C, A, C, C, C, C, C, C, C)$. The output of Round 2 is of the form $(L_2, R_2) = (A, C, C, C, C, C, C, C, A, C, C, A, C, C, A, C)$. The output of Round 3 is of the form $(L_3, R_3) = (A, C, C, A, C, C, A, C, B, B, C, B, A, C, B, C)$. The output of Round 4 is of the form $(L_4, R_4) = (B, B, C, B, A, C, B, C, ?, ?, A, ?, ?, C, ?, ?)$. The output of Round 5 is of the form $(L_5, R_5) = (?, ?, A, ?, ?, C, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)$. Now, there is a 4-bit nibble position with symbol "C" and a 4-bit nibble position with symbol "A" in the output of Round 5. If we continue with one more round, all the 4-bit nibble positions of the output of the resulting round will have an unclear status. Thus, we get a 5-round integral distinguisher of one dimension, which is illustrated in Figure 13(a), here "one dimension" means there is only one active nibble position in the set of inputs.

If we would like to obtain a longer integral distinguisher by adding more rounds from the beginning, we can only add at most 4 rounds before reaching the full plaintext space, as illustrated in Figure 13(b). As a result, 25-round LBC should be sufficiently secure against integral cryptanalysis.

6.7. Slide Attack. The key schedule involves the round numbers (i.e., i) to avoid slide attacks [33, 34], and the rotation number "29" in Step 3 (a) guarantees that the three least significant bits of i get involved in the generation of the subkey of the next round (if any), and the two most significant bits of i get involved in the generation of another round subkey (if any).

6.8. Related-Key Cryptanalysis. The key schedule is based on the LBC round function to have a high level of nonlinearity and involves the key length parameter (i.e., k) to distinguish the different key versions, so as to avoid (potential) related-key attacks [27, 28, 32] under different key lengths. The use of the key selected S-box in the encryption/decryption algorithm makes it more difficult to apply related-key cryptanalysis, since the order of the ordinary S-boxes involved in the S-box layer of a fixed round is indeterminate if a key is unknown, and is very likely to vary when a key is changed.

6.9. Summary. We also analysed the security of LBC against other cryptanalysis methods. In summary, the potential 20-round linear approximation of Section 6.3 is the longest cryptanalysis distinguisher we have obtained, and thus 25-round LBC should be sufficiently secure.

Note that differential cryptanalysis and linear cryptanalysis require a different framework in the key selected S-box mechanism, while impossible differential cryptanalysis and integral cryptanalysis work similarly as in the ordinary mechanism, and boomerang, amplified boomerang, and rectangle attacks follow from differential cryptanalysis.

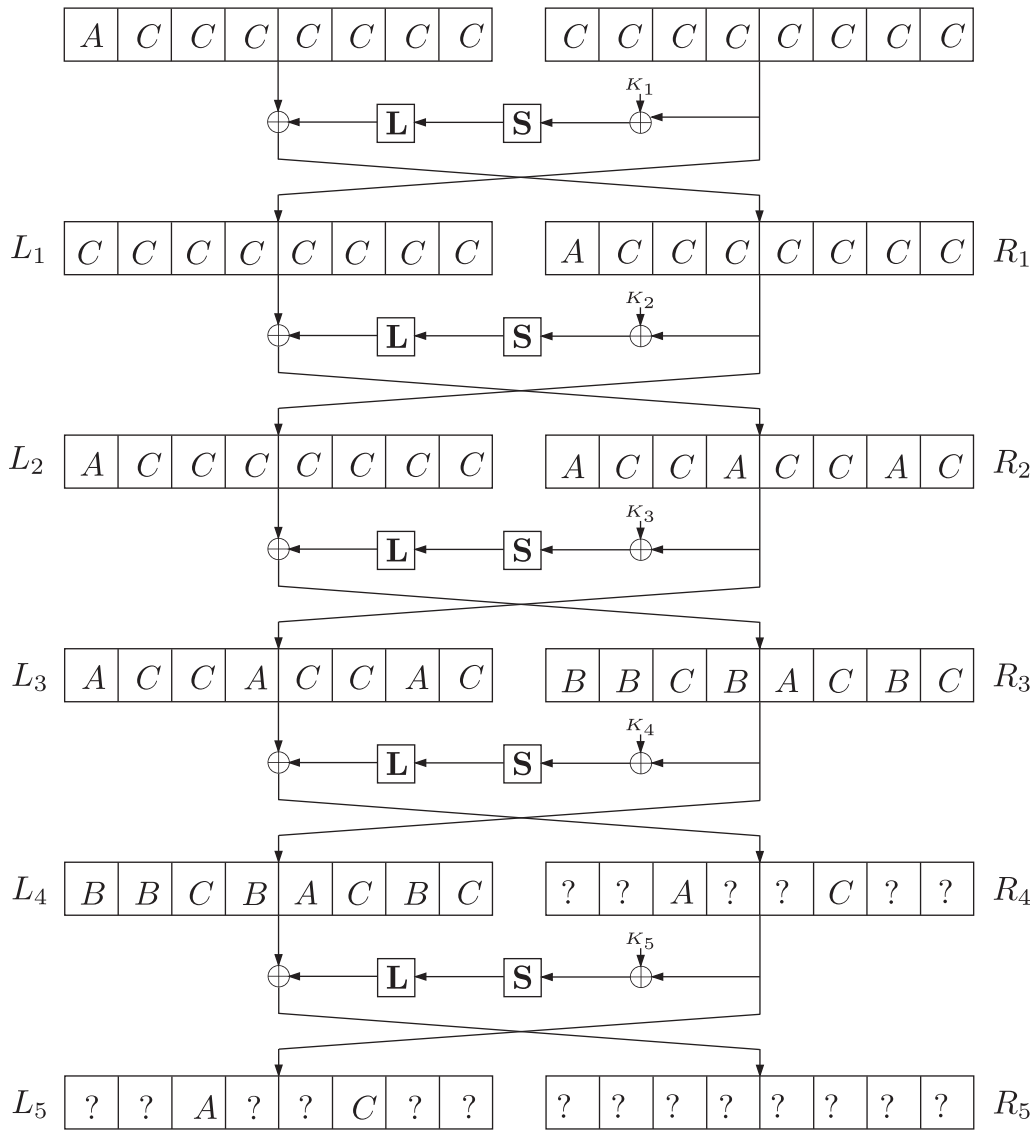
6.10. Security of LBC with the Ordinary S-Box Mechanism. In comparison, for LBC with the ordinary S-box mechanism rather than the key selected S-box mechanism (i.e., using an ordinary S-box $\mathbf{S}(x)$, say $\mathcal{S}_0(x)$, rather than a key selected S-box $\mathbf{S}_{K^{(i)}}(x)$), as shown in Figure 14, a single nibble/bit will get all the 128 subkey bits involved after propagating through at least 6 rounds. A total of 11 rounds appended at both ends of a linear approximation will indicate at least 6 rounds in an end, which would ensure that an attacker guesses all the 128 key bits in the key recovery phase; multiple linear cryptanalysis works well in the ordinary mechanism and thus we should take its effect into consideration. As a result, LBC with the ordinary S-box mechanism would require 32 rounds to be secure, assuming a 20-round linear approximation with a total of 11 rounds appended at both ends, plus one additional round for preventing the potential effect of multiple linear cryptanalysis.

7. Performance Gain Evaluation

In this section, we briefly give our performance gain evaluation of LBC over LBC with the ordinary S-box mechanism. Recall that as discussed in Section 6 from a design perspective, LBC requires 25 rounds to be secure, while LBC with the ordinary S-box mechanism requires 32 rounds to be secure.

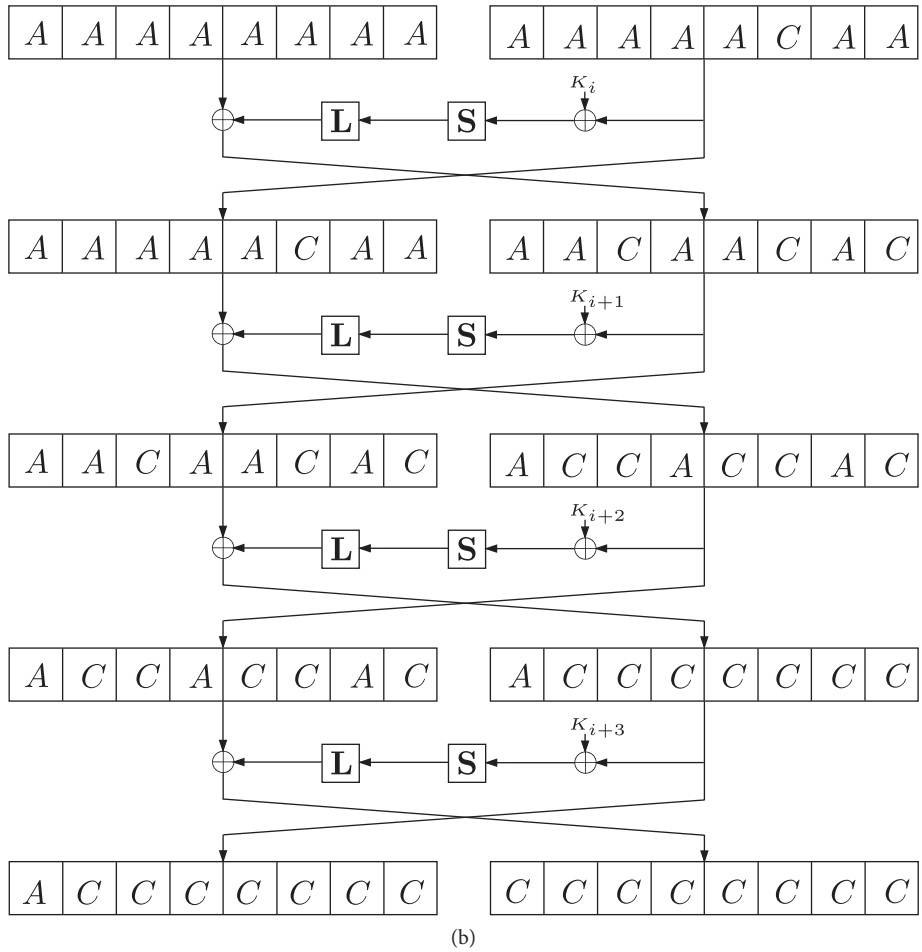
We test software performances on two types of processors, one type has enough storage and computing resources for general purposes such as servers, and the other type has low or moderate storage and computing resources for resource-constrained devices such as smartphones. Note that there are various software and hardware implementation optimizations and trade-offs among such metrics as memory, cost, area, and throughput, faster or slower than the presented performance results.

7.1. Software Performance on Intel i3. The last second subcolumn of Table 8 shows the encryption-only performances of the two LBC versions under the same Single Instruction Multiple Data (SIMD) implementation method on a popular Intel i3 CPU i5-4200U @ 1.6GHz processor (x64 architecture) with enough storage and computing resources for general purposes such as servers, where the results are only for the encryption parts, and the round keys are stored for use after being generated, which is the usual case for a server.



(a)

Figure 13: Continued.



(b)

Figure 13: A 5-round integral distinguisher of LBC and a 4-round extension.

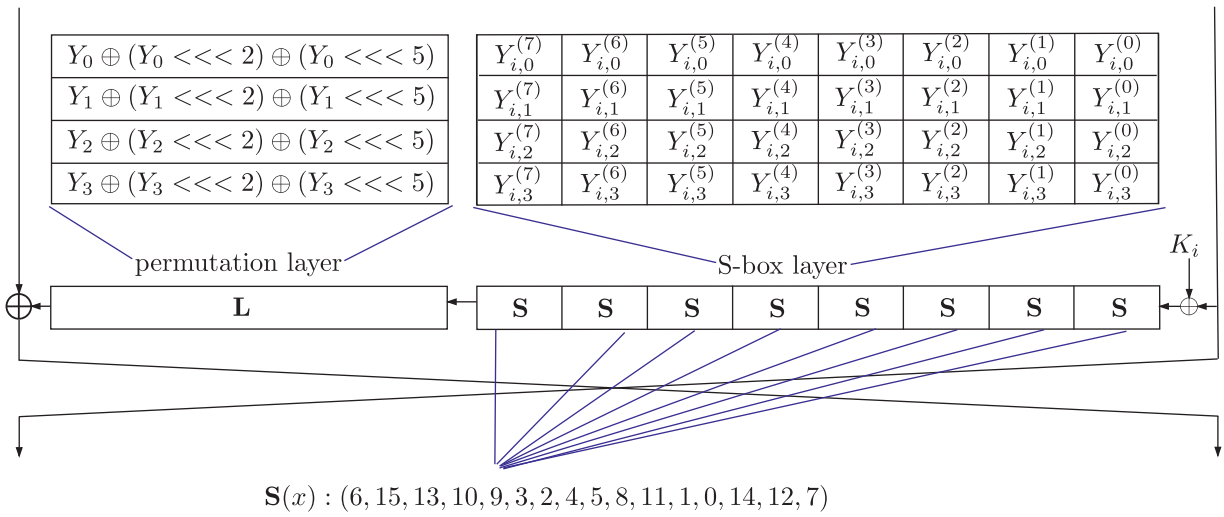


Figure 14: An encryption round of LBC with the ordinary S-box mechanism.

Table 8: Speeds of the two LBC versions under the same SIMD software implementation method.

Number of bits per operation	LBC version	Cycles per byte for 16 plaintext blocks	
		Intel i3	ARM NEON
64	Key selected S-box mechanism	24.6	80.3
	Ordinary S-box mechanism	29.4	91.7

As a result, the key selected S-box mechanism offers (29.4 – 24.6/29.4) ≈ 16% speedup in the example LBC cipher.

Note that if the key schedule part was included, the speedup would be greater, since the two versions use the same process for round keys, but LBC with the key selected mechanism has only 25 rounds, while LBC with the ordinary mechanism has 32 rounds. Note also that since the round keys are stored after being generated, the results also hold for the case that the server processes a larger number of plaintexts at a time.

7.2. Software Performance on ARM NEON. The last subcolumn of Table 8 shows the performance of the two LBC versions under the same SIMD implementation method on a popular ARM Cortex-A9@1.4GHz processor (×64 architecture) for cost-sensitive devices such as smartphones, where the results are for both encryption and key schedule parts, and the round keys are generated on the fly, which is the usual case for a resource-constrained device. As a result, the key selected S-box mechanism offers (91.7 – 80.3/91.7) ≈ 12% speedup in the example LBC cipher.

7.3. Hardware Performance. When implemented in a parallel hardware implementation with one cycle per round, to process a (64-bit) plaintext block, LBC with the key selected mechanism takes 25 cycles, and LBC with the ordinary mechanism takes 32 cycles to process. Thus, the key selected S-box mechanism offers about (32 – 25/32) ≈ 22% speed improvement under this implementation approach in the example LBC cipher. In this case, the key selected S-box mechanism requires slightly more hardware area or GEs than the ordinary S-box mechanism, which may make it not suitable for extremely resource-constrained environments, but nevertheless it is okay in moderately resource-constrained environments.

8. Concluding Remarks

We have presented and investigated a generalised version of Feistel's key selected S-box mechanism in modern block cipher design and have designed the LBC example cipher to demonstrate that the generalised key selected S-box mechanism can be advantageous over the ordinary S-box mechanism for improving security and/or performance without intensifying computational effort and storage space in some application environments. Especially, we have defined the combined difference distribution table and the combined bias distribution table for the generalised key selected S-box mechanism to analyse the security of a block cipher with a generalised key selected S-box against differential and linear cryptanalysis [44, 45].

As the first attempt, LBC is designed mainly as an example for the primary purpose of investigating relative security and performance gain of the generalised key selected S-box mechanism over the popular ordinary S-box mechanism in modern block cipher design. To us, the main overhead of the key selected S-box mechanism is that it requires slightly more hardware area or GEs than the ordinary S-box mechanism, which may make it not suitable for extremely lightweight application environments, depending on specific designs, but nevertheless it can gain better security and/or performance at least in general or moderately lightweight application environments. No single cipher design can be optimal in all application environments, this is the first detailed investigation on the key selected S-box mechanism, and we would like to see more investigations and better cipher designs on it.

Data Availability

The data are available from the corresponding author upon request.

Disclosure

An extended abstract version of this work was published in Proceedings of the 2017 IEEE Region Ten Conference (TENCON 2017, Penang, Malaysia, 5–0038 November, 2017) [46]. As the full version of the work, this paper gives more design rationale and security analysis of the LBC example cipher. The authors were with Institute for Infocomm Research (Singapore) when this work was completed.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award no. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. The authors are grateful to Matt Henricksen for his conversations, to Zhen Li for a preliminary software performance evaluation of LBC, and to Huaqun Guo and Jia Xu for verifying a software implementation of LBC.

References

[1] National Bureau of Standards (NBS) (of USA), *Data Encryption Standard (DES)*, FIPS-46, Gaithersburg, MD, USA, 1977.

[2] National Institute of Standards and Technology (NIST) (of USA), *Advanced Encryption Standard (AES)*, FIPS-197, Gaithersburg, MD, USA, 2001.

[3] H. Feistel, "Cryptography and computer privacy," *Scientific American*, vol. 228, no. 5, pp. 15–23, 1973.

[4] H. Feistel, W. A. Notz, and J. L. Smith, "Some cryptographic techniques for machine-to-machine data communications," *Proceedings of the IEEE*, vol. 63, no. 11, pp. 1545–1554, 1975.

[5] X. C. Yin, J. Yang, and L. Xie, "Key-controlled Rijndael algorithm with multiple S-boxes," *Journal of Communications*, vol. 28, no. 9, pp. 125–132, 2007.

[6] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.
[7] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Proceedings of EUROCRYPT 1993, LNCS 765*, pp. 386–397, Lofthus, Norway, May 1994.
[8] C. Blondeau and B. Gérard, "Multiple differential cryptanalysis: theory and practice," in *Proceedings of FSE 2011, LNCS 6733*, pp. 35–54, Lyngby, Denmark, February 2011.
[9] A. Biryukov, C. De Cannière, and M. Quisquater, "On multiple linear approximations," in *Proceedings of CRYPTO 2004, LNCS, 3152*, pp. 1–22, Santa Barbara, CA, USA, August 2004.
[10] M. Hermelin, J. Y. Cho, and K. Nyberg, "Multidimensional extension of matsui's algorithm 2," in *Proceedings of FSE 2009, LNCS 5665*, pp. 209–227, Leuven, Belgium, February 2009.
[11] I. M. R. Verbauwhede, *Secure Integrated Circuits and Systems*, Springer, Berlin, Germany, 2010.
[12] R. Merkle, "Fast software encryption functions," in *Proceedings of the CRYPTO 1990, LNCS 537*, pp. 476–501, Santa Barbara, CA, USA, August 1991.
[13] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *Proceedings of the FSE 1993, LNCS 809*, pp. 191–204, Cambridge, UK, December 1994.
[14] E. Biham and A. Biryukov, "How to strengthen DES using existing hardware," in *Proceedings of the ASIACRYPT 1994, LNCS 917*, pp. 398–412, Wollongong, Australia, November 1994.
[15] A. Sorkin, "Lucifer, a cryptographic algorithm," *Cryptologia*, vol. 8, no. 1, pp. 22–42, 1984.
[16] S. Harris and C. Adams, "Key-dependent S-box manipulations," in *Proceedings of the SAC 1998, LNCS 1556*, pp. 15–26, Ontario, Canada, August 1999.
[17] W. Zhang, Z. Bao, V. Rijmen, and M. Liu, "A new classification of 4-bit optimal S-boxes and its application to PRESENT, RECTANGLE and SPONGENT," in *Proceedings of the FSE 2015, LNCS 9054*, pp. 494–515, Istanbul, Turkey, March 2015.
[18] W. Wu and L. Zhang, "LBlock: a lightweight block cipher," in *Proceedings of the ACNS 2011, LNCS 6715*, pp. 327–344, Nerja, Spain, June 2011.
[19] A. Bogdanov, L. R. Knudsen, G. Leander et al., "PRESENT: an ultra-lightweight block cipher," in *Proceedings of the CHES 2007, LNCS 4727*, pp. 450–466, Vienna, Austria, September 2007.
[20] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology*, vol. 14, no. 4, pp. 255–293, 2001.
[21] I. Dinur, "Cryptanalytic time-memory-data tradeoffs for FX-constructions with applications to PRINCE and PRIDE," in *Proceedings of the EUROCRYPT 2015, LNCS 9056*, pp. 231–253, Sofia, Bulgaria, April 2015.
[22] Bitcoin network graphs, http://bitcoin.sipa.be/.
[23] G. Leander and A. Poschmann, "On the classification of 4 bit S-boxes," in *Proceedings of the WAIFI 2007, LNCS 4547*, pp. 159–176, Madrid, Spain, June 2007.
[24] J. Y. Cho, "Linear cryptanalysis of reduced-round PRESENT," in *Proceedings of the CT-RSA 2010, LNCS 5985*, pp. 302–317, San Francisco, CA, USA, March 2010.
[25] J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher Square," in *Proceedings of the FSE 1997, LNCS 1267*, pp. 149–165, Haifa, Israel, January 1997.
[26] D. Hong, J. Sung, S. Hong et al., "HIGHT: a new block cipher suitable for low-resource device," in *Proceedings of the CHES 2006, LNCS 2006*, pp. 46–59, Yokohama, Japan, October 2006.
[27] E. Biham, "New types of cryptanalytic attacks using related keys," in *Proceedings of the EUROCRYPT 1993, LNCS 765*, pp. 398–409, Lofthus, Norway, May 1993.
[28] L. R. Knudsen, "Cryptanalysis of LOKI 91," in *Proceedings of the ASIACRYPT 1992, LNCS 718*, pp. 196–208, Queensland, Australia, December 1993.
[29] B. Koo, D. Hong, and D. Kwon, "Related-key attack on the full HIGHT," in *Proceedings of the ICISC 2010, LNCS 6829*, pp. 49–67, Seoul, Korea, December 2011.
[30] L. R. Knudsen and V. Rijmen, "Known-key distinguishers for some block ciphers," in *Proceedings of the ASIACRYPT 2007, LNCS 4833*, pp. 315–324, Kuching, Malaysia, December 2007.
[31] C. Blondeau, T. Peyrin, and L. Wang, "Known-key distinguisher on full PRESENT," in *Proceedings of the EUROCRYPT 2015, LNCS 9215*, pp. 455–474, Santa Barbara, CA, USA, August 2015.
[32] J. Kelsey, B. Schneier, and D. Wagner, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES," in *Proceedings of CRYPTO 1996, LNCS 1109*, pp. 237–251, Santa Barbara, CA, USA, August 1996.
[33] A. Biryukov and D. Wagner, "Slide attacks," in *Proceedings of FSE 1999, LNCS 1636*, pp. 245–259, Rome, Italy, March 1999.
[34] A. Biryukov and D. Wagner, "Advanced slide attacks," in *Proceedings of EUROCRYPT 2000, LNCS 1807*, pp. 589–606, Bruges, Belgium, May 2000.
[35] E. Biham, R. Anderson, and L. Knudsen, "Serpent: a new block cipher proposal," in *Proceedings of FSE 1998, LNCS 1372*, pp. 222–238, Paris, France, March 1998.
[36] H. Handschuh and D. Naccache, "SHACAL," in *Proceedings of Second NESSIE workshop*, Egham, UK, September 2001.
[37] M. Matsui, "On correlation between the order of S-boxes and the strength of DES," in *Proceedings of EUROCRYPT 1994, LNCS 950*, pp. 366–375, Perugia, Italy, May 1995.
[38] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials," in *Proceedings of EUROCRYPT 1999, LNCS 1592*, pp. 12–23, Prague, Czech Republic, May 1999.
[39] L. R. Knudsen, "DEAL—a 128-bit block cipher," Technical report, Department of Informatics, University of Bergen, Bergen, Norway, 1998.
[40] E. Biham, O. Dunkelman, and N. Keller, "The rectangle Attack - rectangling the serpent," in *Proceedings of EUROCRYPT 2001, LNCS 2045*, pp. 340–357, Innsbruck, Austria, May 2001.
[41] J. Kelsey, T. Kohno, and B. Schneier, "Amplified boomerang attacks against reduced-round MARS and Serpent," in *Proceedings of FSE 2000, LNCS 1978*, pp. 75–93, Waterloo, Canada, August 2001.
[42] D. Wagner, "The boomerang attack," in *Proceedings of FSE 1999, LNCS 1636*, pp. 156–170, Rome, Italy, March 1999.
[43] L. Knudsen and D. Wagner, "Integral cryptanalysis," in *Proceedings of FSE 2002, LNCS 2365*, pp. 112–127, Leuven, Belgium, February 2002.
[44] National Institute of Standards and Technology (NIST) (of USA), *Recommendation for the Transitioning of Cryptographic Algorithms and Key Sizes*, pp. 800–131, NIST Special Publication, Gaithersburg, MD, USA, 2010.
[45] National Institute of Standards and Technology (NIST) (of USA), *Recommendation for Key Management - Part 1: General (Revision 3)*, pp. 800–857, NIST Special Publication, Gaithersburg, MD, USA, 2012.
[46] J. Lu and H. Seo, "An investigation of an S-box mechanism in modern block cipher design," in *Proceedings of TENCON 2017*, pp. 145–152, Penang, Malaysia, November 2017.