Front. Comput. Sci., 2023, 17(3): 173805
https://doi.org/10.1007/s11704-022-1677-5

RESEARCH ARTICLE

DBST: a lightweight block cipher based on dynamic S-box

Liuyan YAN[1,2], Lang LI (✉)[1,2], Ying GUO[1,2]

1 College of Computer Science and Technology, Hengyang Normal University, Hengyang 421002, China
2 Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang 421002, China

Abstract IoT devices have been widely used with the advent of 5G. These devices contain a large amount of private data during transmission. It is primely important for ensuring their security. Therefore, we proposed a lightweight block cipher based on dynamic S-box named DBST. It is introduced for devices with limited hardware resources and high throughput requirements. DBST is a 128-bit block cipher supporting 64-bit key, which is based on a new generalized Feistel variant structure. It retains the consistency and significantly boosts the diffusion of the traditional Feistel structure. The SubColumns of round function is implemented by combining bit-slice technology with subkeys. The S-box is dynamically associated with the key. It has been demonstrated that DBST has a good avalanche effect, low hardware area, and high throughput. Our S-box has been proven to have fewer differential features than RECTANGLE S-box. The security analysis of DBST reveals that it can against impossible differential attack, differential attack, linear attack, and other types of attacks.

Keywords internet of things, 5G, dynamic S-box, bit-slice technology, lightweight block cipher

1 Introduction

5G communication technology is widespread and very popular. 5G is significantly faster than 4G, which ensures real-time data transmission. We are at a critical juncture of the fourth industrial revolution that still revolves around the Internet of Things (IoT) technology. 5G communication technology must be promoted and developed on this basis. When 5G wireless networks transfer, the massive data stream contains much private and sensitive information. It is critical to prevent any information leaks. Therefore, IoT confidential communication has been proposed under 5G. In this case, the encryption algorithms must adapt to the high throughput communication environment and the limited hardware resources.

Recently, several papers on lightweight block ciphers have been published in the international academic community such as PRESENT [1], SCENERY [2], WARP [3], SIMON and SPECK [4], Shadow [5], VH [6], BORON [7], CHAM [8], LVPDA [9], GIFT [10], QTL [11], FPL [12], SFN [13], and RECTANGLE [14]. Bogdanov et al. [1] proposed the lightweight block cipher PRESENT in CHES 2007. The hardware implementation of PRESENT requires 1570GE, which makes it appropriate for severely constrained environments such as sensor networks and RFID tags. In this article, the proposed DBST is based on dynamic S-box for 5G IoT confidential communication devices, which requires high throughput and low hardware implementation. The main design goal of DBST is the dynamic S-box based on bit-slice technology. The bit-slice technology was introduced by Elie Biham [15] to accelerate the implementation of DES. For DBST, we allow the data and the subkey to perform block-to-block logical operations to complete the SubColumns. The approach can obtain different replacing results according to different subkeys in the SubColumns process.

The non-linear S-box is an important component of lightweight block ciphers, which directly affects the security of entire algorithms. An S-box can be classified as either a static or dynamic S-box according to whether the state value is fixed or not. Previous S-boxes are generally static, which must have the smallest differential propagation probability and the lowest linear correlation. Besides, the mixing layer associated with the S-box should propagate the nonlinearity to the entire algorithm. The advantage of static S-boxes is that they can directly demonstrate resistance to linear and differential attacks. Nonetheless, they require more rounds to ensure security and the fixed structure allows for potential attacks [16]. S-box analysis requires analysis of specific S-boxes whether it is differential cryptanalysis or linear cryptanalysis. Hence, the dynamic S-box of DBST improves security and compensates for the shortcomings of static S-boxes.

DBST adopted a new 4-branch generalized Feistel variant structure to balance among security, cost and efficiency. A 2-branch XOR operation was used in the middle part of the structure, which made all branches related to each other. The algorithm passes the avalanche effect requirement and the strict avalanche effect criterion in the second round. The round function has a good avalanche effect and diffusion. It has been demonstrated that DBST has 1697.13GE of hardware resources and 1446.552 Mbps of throughput, so DBST is

Received November 23, 2021; accepted March 21, 2022

E-mail: lilang911@126.com

suitable for high throughput and low hardware resource devices under 5G IoT. DBST primarily includes two aspects: (1) novel generalized Feistel variant structure, and (2) dynamic S-box. The algorithm uses a combination of the bit-slice technique and subkeys to construct a novel dynamic S-box structure.

The layout of the article is as follows: Section 2 discusses the entire design and implementation process of DBST. The design principles for each level are presented in Section 3. Section 4 employs a variety of analysis methods to assess the algorithm's security. Section 5 discusses its implementation in terms of hardware resources and the comparison of resource consumption between algorithms, and Section 6 summarizes it.

2 The block cipher of DBST

We illustrate the basic process of DBST.

2.1 Symbol definition

We give the definition of all symbols in Table 1.

2.2 Encryption process

The block size of DBST is 128bit with 32 rounds, which supports 64-bit key. The encryption is shown in Fig. 1 and the encryption procedure of DBST is presented in Table 2.

SubColumns: A 32-bit intermediate calculation data and a 32-bit subkey data are involved in the SubColumns.

Let $w_{31}\|\cdots\|w_1\|w_0$ represent an intermediate calculation data. The first 8 bits $w_7\|\cdots\|w_1\|w_0$ are treated as W_0, the next 8 bits $w_{15}\|\cdots\|w_9\|w_8$ are treated as W_1, so analogy. A 32-bit intermediate calculation data can be depicted as a 4×8 matrix, as shown in Fig. 2. Let $r_{31}\|\cdots\|r_1\|r_0$ represent a subkey data. A 32-bit subkey data can be depicted as a 4×8 matrix in the above way, as illustrated in Fig. 3. For convenience, the matrices are presented in two-dimensions, as shown in Fig. 4.

Let $W_i = w_{i,7}\|\cdots\|w_{i,1}\|w_{i,0}$ and $R_i = r_{i,7}\|\cdots\|r_{i,1}\|r_{i,0}$ represent the ith rows $(i = 0,1,2,3)$. Let W_0, W_1, W_2, W_3 and R_0, R_1, R_2, R_3 be a 64-bit input and B_0, B_1, B_2, B_3 be the 32-bit output of the SubColumns. Let $T_j (j = 0,\ldots,25,26)$ represent 8-bit temporary variables. The SubColumns can be achieved by the following logical operations of Table 3.

Function F_1: 32-bit subkey XOR first, then Each row is cyclically shifted to the left. Let $<<< x$ represent a circular left shift of x bit(s). Let $B_i = t_{i,7}\|\cdots t_{i,1}\|t_{i,0} (i = 0,1,2,3)$ represent the ith row. Except for row 0 without shifting, the

Table 1 Notations of DBST

Notations	Descriptions
P:	The 128-bit plaintext
C:	The 128-bit ciphertext
K:	The 64-bit master key
$rkey_i$:	The 32-bit round subkey in i round
F_1,F_2:	The function F_1, the function F_2
Nr:	The round number
$X[j]^i$:	The 32-bit data in i round $(j = 0,1,2,3)$
Sub:	SubColumns
$\|$:	Concatenation of two binary strings
$\|$:	Bitwise OR operation
$\sim$:	Bitwise negation operation
&:	Bitwise AND operation
$\oplus$:	Bitwise exclusive-OR operation

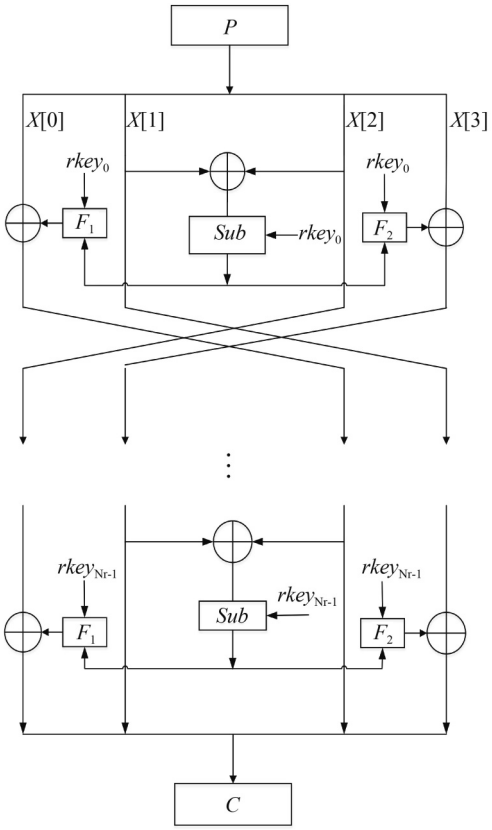

Fig. 1 The encryption of DBST

Table 2 The encryption routine of DBST

Alogrithm 1 DBST encryption foutine

Input: $P_{(128)}, K_{(64)}$

Output: $C_{(128)}$

1: $P_{(128)} \rightarrow X[0]^0_{(32)}\|X[1]^0_{(32)}\|X[2]^0_{(32)}\|X[3]^0_{(32)}$

2: $GetrateKey(K_{(64)}, rkey_0)$

3: $for\ i = 0\ to\ Nr-1\ do\ the\ following$

4: $Mid_i \leftarrow Sub((X[1]^i_{(32)} \oplus X[2]^i_{(32)}), rkey_i)$

5: $X[0]^{i+1}_{(32)} \leftarrow X[2]^i_{(32)}$

$X[1]^{i+1}_{(32)} \leftarrow X[3]^i_{(32)} \oplus F_1(Mid_i, rkey_i)$

$X[2]^{i+1}_{(32)} \leftarrow X[0]^i_{(32)} \oplus F_2(Mid_i, rkey_i)$

$X[3]^{i+1}_{(32)} \leftarrow X[1]^i_{(32)}$

6: end for

7: $C_{(128)} \leftarrow X[2]^i_{(32)}\|X[3]^i_{(32)}\|X[0]^i_{(32)}\|X[1]^i_{(32)}$

$$\begin{vmatrix} W_7 & W_6 & \ldots & W_1 & W_0 \\ W_{15} & W_{14} & \ldots & W_9 & W_8 \\ W_{23} & W_{22} & \ldots & W_{17} & W_{16} \\ W_{31} & W_{30} & \ldots & W_{25} & W_{24} \end{vmatrix}$$

Fig. 2 A 32-bit intermediate calculation data

$$\begin{vmatrix} r_7 & r_6 & \cdots & r_1 & r_0 \\ r_{15} & r_{14} & \cdots & r_9 & r_8 \\ r_{23} & r_{22} & \cdots & r_{17} & r_{16} \\ r_{31} & r_{30} & \cdots & r_{25} & r_{24} \end{vmatrix}$$

Fig. 3 A 32-bit subkey state

$$\begin{vmatrix} W_{0,7} & W_{0,6} & \cdots & W_{0,1} & W_{0,0} \\ W_{1,7} & W_{1,6} & \cdots & W_{1,1} & W_{1,0} \\ W_{2,7} & W_{2,6} & \cdots & W_{2,1} & W_{2,0} \\ W_{3,7} & W_{3,6} & \cdots & W_{3,1} & W_{3,0} \end{vmatrix} \begin{vmatrix} r_{0,7} & r_{0,6} & \cdots & r_{0,1} & r_{0,0} \\ r_{1,7} & r_{1,6} & \cdots & r_{1,1} & r_{1,0} \\ r_{2,7} & r_{2,6} & \cdots & r_{2,1} & r_{2,0} \\ r_{3,7} & r_{3,6} & \cdots & r_{3,1} & r_{3,0} \end{vmatrix}$$

Fig. 4 Two-dimensional ways

Table 3 Logical expressions of the SubColumns

$T_0 = \sim W_1$	$T_1 = W_2 \oplus W_3$
$B_1 = T_1 \oplus (W_0 \& T_0)$	$T_2 = W_0 \oplus (W_3 \mid T_0)$
$B_0 = W_2 \oplus T_2$	$B_2 = (W_1 \oplus W_2) \oplus (T_1 \& T2)$
$T_3 = \sim W_3$	$T_4 = \sim (W_2 \mid W_1)$
$T_5 = \sim (W_1 \mid W_0) \& W_3$	$T_6 = \sim (W_3 \mid W_1) \& W_0$
$T_7 = T_3 \& W_2 \& W_1$	$T_8 = R_1 \& R_0$
$T_9 = T_8 \& (T_4 \mid T_5 \mid T_6 \mid T_7)$	$T_{10} = \sim (W_2 \mid W_0)$
$T_{11} = T_{10} \& W_1$	$T_{12} = W_3 \& W_1 \& W_0$
$T_{13} = \sim W_2 \& W_3$	$T_{14} = R_3 \oplus R_2$
$T_{15} = \sim R_0$	$T_{16} = T_{14} \& T_{15}$
$T_{17} = T_{16} \& (T_6 \mid T_{11} \mid T_{12} \mid T_{13})$	$T_{18} = \sim (W_3 \mid W_1 \mid W_0)$
$T_{19} = W_3 \& T_4$	$T_{20} = T_3 \& W_1 \& W_0$
$T_{21} = \sim T_{14} \& T_{15}$	$T_{22} = T_{21} \& (T_{10} \mid T_{18} \mid T_{19} \mid T_{20})$
$T_{23} = \sim W_0 \& W_1 \& W_2$	$T_{24} = W_3 \& W_2$
$T_{25} = \sim R_1 \& R_0$	$T_{26} = T_{25} \& (T_6 \mid T_{12} \mid T_{23} \mid T_{24})$
$B_3 = T_9 \mid T_{17} \mid T_{22} \mid T_{26}$	

remaining rows 1, 2 and 3 are cyclically shifted left by 1, 4 and 5 bits respectively. The steps of function F_1 are as follows:

$$B_0 = t_{0,7} \| \cdots t_{0,1} \| t_{0,0} \overset{<<<0}{\leftarrow} B_0 = t_{0,7} \| \cdots t_{0,1} \| t_{0,0},$$

$$B_1 = t_{1,6} \| \cdots t_{1,0} \| t_{1,7} \overset{<<<1}{\leftarrow} B_1 = t_{1,7} \| \cdots t_{1,1} \| t_{1,0},$$

$$B_2 = t_{2,3} \| \cdots t_{2,5} \| t_{2,4} \overset{<<<4}{\leftarrow} B_2 = t_{2,7} \| \cdots t_{2,1} \| t_{2,0},$$

$$B_3 = t_{3,2} \| \cdots t_{3,4} \| t_{3,3} \overset{<<<5}{\leftarrow} B_3 = t_{3,7} \| \cdots t_{3,1} \| t_{3,0}.$$

Function F_2: 32-bit subkey XOR first, then Each row is cyclically shifted to the left. Let $<<< x$ represent a circular left shift of x bit(s). Let $B_i = t_{i,7} \| \cdots t_{i,1} \| t_{i,0} (i = 0,1,2,3)$ represent the ith row. Except for row 0 without shifting, the remaining rows 1, 2, and 3 are cyclically shifted left by 2, 3, and 6 bits respectively. The steps of function F_2 are as follows:

$$B_0 = t_{0,7} \| \cdots t_{0,1} \| t_{0,0} \overset{<<<0}{\leftarrow} B_0 = t_{0,7} \| \cdots t_{0,1} \| t_{0,0},$$

$$B_1 = t_{1,5} \| \cdots t_{1,7} \| t_{1,6} \overset{<<<2}{\leftarrow} B_1 = t_{1,7} \| \cdots t_{1,1} \| t_{1,0},$$

$$B_2 = t_{2,4} \| \cdots t_{2,6} \| t_{2,5} \overset{<<<3}{\leftarrow} B_2 = t_{2,7} \| \cdots t_{2,1} \| t_{2,0},$$

$$B_3 = t_{3,1} \| \cdots t_{3,3} \| t_{3,2} \overset{<<<6}{\leftarrow} B_3 = t_{3,7} \| \cdots t_{3,1} \| t_{3,0}.$$

2.3 Key schedule

DBST accepts 64-bit key. The initial key $K = k_{63} \| \cdots k_1 \| k_0$ is firstly saved in a key register of the same bit size and formed as a 4×16 matrix, as shown in Fig. 5.

$$\begin{vmatrix} k_{15} & k_{14} & \cdots & k_1 & k_0 \\ k_{31} & k_{30} & \cdots & k_{17} & k_{16} \\ k_{47} & k_{46} & \cdots & k_{33} & k_{32} \\ k_{63} & k_{62} & \cdots & k_{49} & k_{48} \end{vmatrix} \begin{vmatrix} k_{0,15} & k_{0,14} & \cdots & k_{0,1} & k_{0,0} \\ k_{1,15} & k_{1,14} & \cdots & k_{1,1} & k_{1,0} \\ k_{2,15} & k_{2,14} & \cdots & k_{2,1} & k_{2,0} \\ k_{3,15} & k_{3,14} & \cdots & k_{3,1} & k_{3,0} \end{vmatrix}$$

Fig. 5 A 64-bit state key and its two-dimensional representation

The jth row is represented by $Y_j = k_{j,15} \| \cdots k_{j,1} \| k_{j,0} (0 \leqslant j \leqslant 3)$. At round $i (i = 0,1,\cdots,31)$, the 32-bit subkey is composed of the last 2 rows of the key register in the current state, i.e., $rkey_i = Y_3 \| Y_2$. The key register is refreshed in the following way after $rkey_i$ has been extracted:

1) The $rkey_i$ and the rightmost 8 columns of the matrix are involved in dynamic SubColunmns. Let $B_j = b_{j,7} \| \cdots b_{j,1} \| b_{j,0}$ $(j = 0,1,2,3)$ be outputs. The permutation result is shown in Fig. 6.

$$\begin{vmatrix} k_{0,15} & \cdots & k_{0,8} & b_{0,7} & \cdots & b_{0,0} \\ k_{1,15} & \cdots & k_{1,8} & b_{1,7} & \cdots & b_{1,0} \\ k_{2,15} & \cdots & k_{2,8} & b_{2,7} & \cdots & b_{2,0} \\ k_{3,15} & \cdots & k_{3,8} & b_{3,7} & \cdots & b_{3,0} \end{vmatrix}$$

Fig. 6 The result after the SubColumns

2) Completing a linear transformation.

$$Y_0' := (Y_0 <<< 7) \oplus Y_1$$

$$Y_1' := Y_2$$

$$Y_2' := (Y_2 <<< 13) \oplus Y_3$$

$$Y_3' := Y_0$$

3) Part of key state $(k_{3,15} \| \cdots \| k_{3,10})$ is XORed with $Rcon[t] = t + 1 (t = 0,1,\cdots 30,31)$.

$$k_{3,15} \| \cdots \| k_{3,10} = (k_{3,15} \| \cdots \| k_{3,10}) \oplus Rcon[t]. \quad (1)$$

2.4 Decryption process

Due to the high symmetry of the round function, the decryption process of DBST is identical to the encryption process, which simply uses the encryption key in reverse.

3 Design rationale

3.1 DBST structure

The most used structures of lightweight block ciphers are Feistel and SPN. DBST uses a novel generalized Feistel variant structure with 4 high-symmetry branches, which improves the diffusivity of traditional Feistel structures. The Subcolumns is implemented based on bit-slice technology.

Due to the structural characteristics, there is no need to consider the inverse process of the SubColumns and the entire decryption algorithm can also recycle the encryption process.

3.2 Dynamic S-box structure based on bit-slice technology

3.2.1 Design criteria

For a $n \times m$ S-box, an n-bit binary input will produce an m-bit binary output. From a security performance aspect, the larger the n and m values are, the higher the degree of non-linearity and obfuscation is. From the perspective of implementation efficiency, the larger the S-box used in block ciphers, the more resources it consumes and the lower the cryptographic algorithm implementation efficiency. DBST decided to use the 4×4 S-box to achieve high-efficiency hardware performance and meet lightweight block cipher standards after considerable thought on choosing the size of the S-box.

Definition 1 Permutation. Let $S: F_2^n \to F_2^n$. For any $x, x' \in F_2^n$ and $x \neq x'$, there is $S(x) \neq S(x')$, then S is said to be a permutation.

Definition 2 [17] Differential Uniformity. Let f be a function defined on $F_2^n \to F_2^n$. For any difference corresponding to $(a,b) \in F_2^n \times F_2^n$, define the set $D_f(a \to b) = \{x \in F_2^n | f(x \oplus a) \oplus f(x) = b\}$and the number of elements in the set $D_f(a \to b)$ as $\delta_f(a,b)$, then the difference uniformity of the function f is $\delta(f) = \max_{a \neq 0, b} \delta_f(a,b)$ and the function f is said to be $\delta(f)$-uniform.

Definition 3 [17] Linearity. Let f be a function defined on $F_2^n \to F_2^n$. For any linear combination, define the *Walsh* transform as:

$$F_2^n \times F_2^n \to Z(a,b) \to \lambda_f(a,b) = \sum_{x \in F_2^n} (-1)^{<b,f(x)> \oplus <a,x>}. \quad (2)$$

Then the linearity of the function f is $\lambda(f) = \max_{a,b \in F_2^n, b \neq 0} |\lambda_f(a,b)|$ where $<b, f(x)>$ and $<a, x>$ denote the inner product operation.

Definition 4 The difference distribution table of an S-box. Let $m, n \in N$, the non-linear mapping from F_2^n to F_2^n (also known as the S-box) be denoted as: $S: F_2^n \to F_2^n$. Given $\alpha \in F_2^n, \beta \in F_2^n$, construct $2^n \times 2^n$ tables with α as the input differential of the S-box and β as the output differential of the S-box. The intersection of rows and columns takes the value $N_s(\alpha,\beta)$.

$$\begin{aligned} IN_s(\alpha,\beta) &= \{x \in F_2^n : S(x \oplus \alpha) \oplus S(x) = \beta\}, \\ N_s(\alpha,\beta) &= \#IN_s(\alpha,\beta). \end{aligned} \quad (3)$$

Definition 5 Linear approximation table for an S-box. Let $m, n \in N$, the non-linear mapping from F_2^n to F_2^n (also known as the S-box) be denoted as: $S: F_2^n \to F_2^n$. Given $\alpha \in F_2^n, \beta \in F_2^n$, construct $2^n \times 2^n$ tables with α as the input mask of the S-box and β as the output mask of the S-box. The intersection of rows and columns takes the value $N_s(\alpha,\beta) - 2^{n-1}$.

$$\begin{aligned} IN_s(\alpha,\beta) &= \{x \in F_2^n : \alpha \cdot x = \beta \cdot S(x)\}, \\ N_s(\alpha,\beta) &= \#IN_s(\alpha,\beta). \end{aligned} \quad (4)$$

The design standards of the S-box of DBST are as follows:

1) The S-box satisfies the substitution criterion of Definition 1.

2) The S-box is a 4×4 S-box with differential uniformity $\delta(s) = \max_{a \neq 0, b} \delta_s(a,b) \leqslant 4$.

3) The S-box is a 4×4 S-box with linearity $\lambda(S) = \max_{a,b \in F_2^4, b \neq 0} |\lambda_s(a,b)| \leqslant 8$.

3.2.2 Design of dynamic S-box

The Subcolumns implemented by lookup tables is public in most algorithms. An S-box can be classified as either a static or dynamic S-box whether the state value in the table is fixed or not. Previous algorithm S-boxes are generally static, which must have the smallest differential propagation probability and the lowest linear correlation [16]. In the algorithm, bit-slice technology is introduced into the S-box, so that the SubColumns can be completed not only by the traditional lookup table method but also by logical operations. Moreover, subkeys are added into the logical operations, which makes the S-box present a dynamic form. It compensates for the shortcomings of static S-boxes and lookup tables. We allow the data and the subkey to perform block-to-block logical operations to complete the S-box operation. The S-box is a dynamic related to the key. The dynamic S-box not only has high randomness but also improves algorithm security. In addition, it can effectively resist cipher attacks based on specific S-box analysis. We finally selected four S-boxes to construct the dynamic S-box of DBST after the continuous screening. The experimental results show that DBST S-box has fewer differential features than RECTANGLE S-box, which in turn has higher security.

3.3 F_1 and F_2 functions structure

In the DBST encryption process, the data is diffused by F_1 and F_2 functions after the SubColumns. Actually, the SubColumns is equivalent to each column substitution independently. The F_1 and F_2 functions make each output column depend on other columns, which provide the necessary spread. They use different flows to make the 4-branches of the algorithm round function different. And beyond that, the round key XOR operation also provides good security.

4 Security evaluation

4.1 Avalanche effect

The avalanche effect is an important metric for cryptographic algorithms. Kam, Davida [18] and H. Feistel [19] first proposed the concepts of dependency and the avalanche effect. Afterward, Webster and Tavares [20] introduced the strict avalanche criterion. For DBST, we tested it for completeness d_c, avalanche effect d_a, and strict avalanche effect d_{sa}.

We selected 1000 specific plaintext blocks and randomly changed 1 bit to be some test cases. Tables 4 and 5 show the test results of DBST and PRESENT [21], respectively.

From Tables 4 and 5, we can get that DBST fully meets the dependency requirement and the strict avalanche effect from the 2nd round. The PRESENT fully meets the dependency requirement from the 9th round. The number of rounds for PRESENT and DBST is 31 and 32. So we fully believe that the 32-round DBST meets the avalanche effect criterion.

4.2 Differential cryptanalysis

Differential cryptanalysis [22] is a selective plaintext attack method. The essence of differential cryptanalysis is to study

Table 4 Avalanche effect test results of DBST

Round	2	3	4	32
d_a	0.999023	0.999450	0.999859	0.999093
d_{sa}	0.999878	0.999946	0.999995	0.999083
d_c	1.000000	1.000000	1.000000	1.000000

Table 5 Avalanche effect test results of PRESENT

Round	7	8	9	31
d_a	0.998103	0.998881	0.999166	0.999037
d_{sa}	0.991738	0.992044	0.992029	0.992019
d_c	1.000000	1.000000	1.000000	1.000000

the propagation characteristics of differences in the encryption (decryption) process. The section mainly introduces the differential cryptanalysis of DBST.

We assume that the input difference is a half byte activity. The output differentials are obtained by the S-box differential distribution table. We track the iterative process of 32-round DBST and traverse 360 differential trails, while each round takes the maximum difference probability through the S-box. Finally, The different probability values are shown in Table 6.

When we use differential attack methods to attack n-bit block ciphers, there must be very few rounds of differential features greater than 2^{1-n} in the entire encryption process. For DBST, we take the largest differential probability feature. The upper limit of the 32-round difference is $2^{-170} < 2^{-127}$. This differential probability is very small, so the 32-round DBST has enough ability to resist differential attack.

4.3 Linear cryptanalysis

In this paper, we obtain the S-box linear approximation table through linear analysis [23] and assume S-box input mask value. Subsequently, The output mask values are determined according to the linear approximation table. We track the iterative process of the DBST round function, traverse 360 input masks, and take the maximum linear approximation probability of the S-box in each round. The results of linear distribution are shown in Table 7. The upper limit of linear feature probability is 2^{-165}. It is very small, so the 32-round DBST has sufficient resistance ability of linear analysis.

4.4 Impossible differential cryptanalysis

Impossible differential cryptanalysis [24] uses a zero difference probability to eliminate a large quantity of candidate keys. The most effective method of finding impossible differences is the intermediate encounter.

Table 6 Differential feature probability distribution table

Probability interval	Probability number	Probability interval	Probability number
$(2^{-175},2^{-170}]$	48	$(2^{-180},2^{-175}]$	48
$(2^{-190},2^{-185}]$	24	$(2^{-205},2^{-200}]$	24
$(2^{-210},2^{-205}]$	8	$(2^{-220},2^{-215}]$	8
$(2^{-245},2^{-240}]$	16	$(2^{-255},2^{-250}]$	16
$(2^{-265},2^{-260}]$	8	$(2^{-270},2^{-265}]$	8
$(2^{-280},2^{-275}]$	12	$(2^{-285},2^{-280}]$	44
$(2^{-290},2^{-285}]$	4	$(2^{-300},2^{-295}]$	4
$(2^{-510},2^{-300}]$	88		

Table 7 Linear feature probability distribution table

Probability interval	Probability number	Probability interval	Probability number
$(2^{-170},2^{-165}]$	4	$(2^{-175},2^{-170}]$	4
$(2^{-180},2^{-175}]$	32	$(2^{-185},2^{-180}]$	12
$(2^{-190},2^{-185}]$	36	$(2^{-205},2^{-200}]$	20
$(2^{-210},2^{-205}]$	28	$(2^{-215},2^{-210}]$	16
$(2^{-220},2^{-215}]$	20	$(2^{-225},2^{-220}]$	12
$(2^{-230},2^{-225}]$	16	$(2^{-235},2^{-230}]$	32
$(2^{-250},2^{-245}]$	12	$(2^{-255},2^{-250}]$	4
$(2^{-310},2^{-255}]$	112		

Here, we give out the DBST algorithm for 6 rounds of impossible difference. Cross transforms were considered in the final round. 3 rounds of the different trail with the probability of 1 were taken along the encryption direction and 3 rounds of different trail with the probability of 1 were taken along the decryption direction. The result is shown in Table 8, where "*" represents an uncertain bit. It can be seen that there is a contradiction in round 3. So, once the 3 rounds of DBST reach full dependence, then 32-round DBST will be safe enough to resist impossible differential attack.

4.5 Algebraic cryptanalysis

Algebraic Attack is proposed by Courtois and Pieprzyk in the analysis of AES [25]. It is to transform the attack problem on

Table 8 DBST 6-round impossible differential tail

Round 0	0000	0000	0000	0000	0000	0000	0000	0000	0000
	0000	0100	0000	0000	0000	0000	0000	0000	0000
	0000	0000	0000	0000	0000	0000	0000	0000	0000
			0000	0000	0000	0000	0000		
Round 1	0000	0000	0000	0000	0000	0000	0000	0000	000*
	0000	0000	00*0	0000	0001	0100	0000	0000	*000
	0000	0*00	1000	0000	0100	0000	0000	0000	0100
			0000	0000	0000	0000	0000		
Round 2	0000	*000	0000	0*00	1000	0000	0100	0000	***0
	0***	****	**00	0***	***0	*00*	****	****	00**
	****	*00*	00**	****	*00*	****	000*	0000	0000
			00*0	0000	0001	0100	0000		
Round 3	****	00**	****	*00*	00**	****	***0***	****	****
	****	****	****	****	****	****	****	****	****
	****	****	****	****	****	****	***0	0***	****
			**00	0***	***0	***0*	****		
Round 3	0000	*000	0000	0*00	*000	0000	0**1**00	0000	***0
	0***	****	**00	00**	****	*00*	****	****	00**
	****	*00*	00**	****	*10*	****	000*	0000	0000
			00*0	0000	000*	0**1**00	0000		
Round 4	0000	0000	0000	0000	0000	0000	0100	0000	000*
	0000	0000	00*0	0000	000*	0100	0000	0000	*000
	0000	0*00	*000	0000	0100	0000	0000	0000	0000
			0000	0000	0000	0000	0000		
Round 5	0000	0000	0000	0000	0000	0000	0000	0000	0000
	0000	0000	0000	0000	0000	0000	0000	0000	0000
	0000	0000	0000	0000	0100	0000	0000	0000	0000
			0000	0000	0000	0000	0000		
Round 6	0000	0000	0000	0000	0000	0000	0100	0000	0000
	0000	0000	0000	0000	0000	0000	0000	0000	0000
	0000	0000	0000	0000	0000	0000	0000	0000	0000
			0000	0000	0000	0000	0000		

a cryptographic system into solving corresponding algebraic equations. The S-box is the only nonlinear part of algorithms, which determines the security of DBST. A 4-bit S-box can be indicated by more than 21 quadratic equations. DBST has 4 dynamic S-boxes, so the entire cipher block has $m\times 21$ quadratic equations in $m\times 4$ variables (m is the number of S-box usage in DBST). DBST has a total of $m = 24\times 8 + 23\times 8 = 376$ 4-bit S-boxes, which is 7896 equations in 1504 variables. Literature [25] uses the working factor to estimate the XSL attack complexity on a block cipher. For DBST, this paper estimates as follows:

$$\Gamma = ((t-r)/s)^{\lceil((t-r)/s)\rceil},$$
$$WF \approx \Gamma^{w}(Block\ size)^{w\frac{t-r}{s}}(Number\ of\ rounds)^{2w\frac{t-r}{s}} > 2^{625}. \tag{5}$$

This far exceeds 2^{128} operations required for exhaustive search, so the DBST is able to resist algebra attack.

5 Hardware performance

We mainly carry out ASIC and FPGA of DBST to verify its performance. DBST is implemented by Verilog-HDL. In the FPGA hardware implementation, the performance analysis of DBST comprehensive download is performed by ISE14.6 with the FPGA model Xilinx Virtex-5VLX50T. DBST consumes FF, LUTS, and Slices of 199,320, and 102, respectively. Through the test, we found that the clock period of DBST was 2.765 ns. The clock frequency was 361.638 MHz and the throughput was 1446.552 Mbps. In the ASIC hardware implementation, DBST is simulated by ModelSim SE-64 10.4 evaluation, and synthesized into a standard library based on SMIC 0.18 μm CMOS technology. Figure 7 illustrates the datapath of DBST.

During the encryption process, the 128-bit plaintext of DBST requires 576GE for the register. 32-bit data XOR requires 85.44GE and the SubColumns requires 90.57GE. The F_1 and F_2 functions require the same area resources, respectively 85.44GE. Branch XOR requires 170.88GE. The Hardware implementation of permutation and row shifting is through connection operation, so the hardware resources are

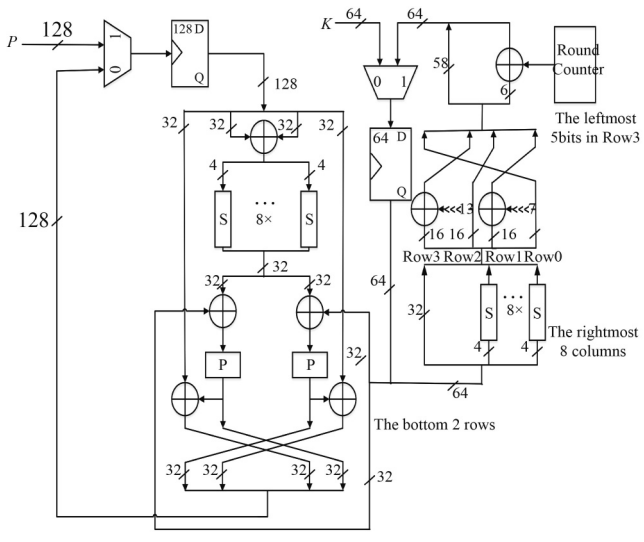

Fig. 7 The datapath of the round-based DBST

0GE.

The specific resources occupied in the key schedule are described as follows: the 64-bit key is kept in the register and it requires 384GE. The SubColumns requires 90.57GE. One round of Feistel requires 85.44GE. Round counter XOR requires 13.35GE. In the algorithm implementation, a total of 30GE is required for the control logic unit and the counter. The sum of hardware resources is 1697.13GE, as shown in Table 9. Table 10 shows the implementation comparison between DBST and other ciphers.

Table 9 Area requirement of DBST

Module (round function)	GE	Proportion/%
Plaintext register	576	33.94
Key register	384	22.63
The SubColumns	181.14	10.67
Xor	525.99	30.99
Control logic and other counters	30	1.77
Total	1697.13	100.00

Table 10 Comparison of lightweight block cipher implementations

Ciphers	Block size	Key size	Structure	GE
PRESENT	64	80	SPN	1570
RECTANGLE	64	80	SPN	1600
KLEIN	64	80	SPN	2202
SIMON	64	128	Feistel	1751
SFN	64	96	SPN	1877
SPECK	64	128	Feistel	2014
SKINNY	64	128	SPN	1696
DBST	**128**	**64**	**Feistel**	**1697.13**

In conclusion, the DBST algorithm has the characteristics of high throughput and low hardware consumption.

6 Conclusion

The proposed DBST is a dynamic S-box lightweight block cipher based on bit-slice technology. Our S-box is key-dependent, which increases the difficulty of differential and linear analysis. The round function consists of a novel generalized Feistel variant structure. It is more diffusible than the traditional Feistel structure. After analysis, we found that DBST met the avalanche effect in the second round. The hardware resources of DBST require 1697.13GE that conforms to low hardware implementation requirements (2000GE). Simultaneously, DBST has a throughput of 1446.552 Mbps, which is higher than most algorithms. It is suitable for 5G IoT devices. Eventually, DBST achieves not only high symmetry but also high security.

Acknowledgements This work was supported by the Scientific Research Fund of Hunan Provincial Education Department (19A072), the Science and Technology Innovation Program of Hunan Province (2016TP1020), Application-oriented Special Disciplines, Double First-Class University Project of Hunan Province (Xiangjiaotong [2018] 469), Hengyang Normal University Training Programs of Innovation and Entrepreneurship for Undergraduates (cxcy2021011), and Hunan Provincial Training Programs of Innovation and Entrepreneurship for Undergraduates (S202110546017).

Appendixes

Appendix A Test vector

We have given the test vectors of DBST as shown in Table A1.

Table A1 The test vectors of DBST

Plaintext	Key	Ciphertext
0000-0000-0000-0000-0000-0000-0000-0000	0000-0000-0000-0000	CE7D-A5B7-F11C-45F6-F96B-3764-12BC-8C53
0000-0000-0000-0000-0000-0000-0000-0000	FFFF-FFFF-FFFF-FFFF	FA3F-491C-C94F-EBB3-1833-25EB-43F7-3871
FFFF-FFFF-FFFF-FFFF-FFFF-FFFF-FFFF-FFFF	0000-0000-0000-0000	3182-5A48-0EE3-BA09-0694-C89B-BD43-73AC
0000-0000-0000-0000-FFFF-FFFF-FFFF-FFFF	FFFF-FFFF-FFFF-FFFF	05C0-B6E3-36B0-144C-E7CC-DA14-BC08-C78E
0123-4567-89AB-CDEF-0123-4567-89AB-CDEF	0123-4567-89AB-CDEF	8B25-8BF8-D8F1-D036-768C-F922-73CD-C6D1

The data in the table is expressed in hexadecimal.

Appendix B The SubColumns

Let $m_{31}\|\cdots m_1\|m_0$ represent the 32-bit data state. Let $r_{31}\|\cdots r_1\|r_0$ represent the 32-bit subkey state. The data and sub-key are respectively drawn into matrix, as shown in Figs. A1 and A2. They are depicted in two dimensions for convenience, as illustrated in Fig. A3.

The S-boxes of DBST are given by the Table A2. Let $x_i = m_{3,i}\|m_{2,i}\|m_{1,i}\|m_{0,i}(i=0,\ldots,6,7)$ as input for each S-box. Let $y_i = r_{3,i}\|r_{2,i}\|r_{1,i}\|r_{0,i}(i=0,\ldots,6,7)$ is used as an identifier to search corresponding S-box. Let $S(x_i) = b_{3,i}\|b_{2,i}\|b_{1,i}\|b_{0,i}$ $(i=0,\ldots,6,7)$ represent the output of the S-box. The SubColumns is shown in Fig. A4.

Appendix C Differential analysis and comparison of dynamic S-box

(1) DBST consists of 4 S-boxes, as shown in Table A3. We carry out differential analysis on 4 S-boxes respectively.

The differential distribution table of $S_1(x)$ and $S_2(x)$ is as Table A4. According to the differential distribution table of S-box, some characteristics of $S_1(x)$ and $S_2(x)$ can be obtained:

$$\begin{array}{lll} 0001-***1 & 0100-**11 & 0101-***0 \\ 1000-**1* & 1100-**0* & *01*-0100 \\ *1**-1000 & **1*-1001 & *1**-1100 \\ **0*-1101. & & \end{array} \tag{6}$$

$$\begin{vmatrix} m_7 & m_6 & \ldots & m_1 & m_0 \\ m_{15} & m_{14} & \ldots & m_9 & m_8 \\ m_{23} & m_{22} & \ldots & m_{17} & m_{16} \\ m_{31} & m_{30} & \ldots & m_{25} & m_{24} \end{vmatrix}$$

Fig. A1 32-bit data

$$\begin{vmatrix} r_7 & r_6 & \ldots & r_1 & r_0 \\ r_{15} & r_{14} & \ldots & r_9 & r_8 \\ r_{23} & r_{22} & \ldots & r_{17} & r_{16} \\ r_{31} & r_{30} & \ldots & r_{25} & r_{24} \end{vmatrix}$$

Fig. A2 32-bit subkey

$$\begin{vmatrix} m_{0,7} & m_{0,6} & \ldots & m_{0,1} & m_{0,0} \\ m_{1,7} & m_{1,6} & \ldots & m_{1,1} & m_{1,0} \\ m_{2,7} & m_{2,6} & \ldots & m_{2,1} & m_{2,0} \\ m_{3,7} & m_{3,6} & \ldots & m_{3,1} & m_{3,0} \end{vmatrix} \quad \begin{vmatrix} r_{0,7} & r_{0,6} & \ldots & r_{0,1} & r_{0,0} \\ r_{1,7} & r_{1,6} & \ldots & r_{1,1} & r_{1,0} \\ r_{2,7} & r_{2,6} & \ldots & r_{2,1} & r_{2,0} \\ r_{3,7} & r_{3,6} & \ldots & r_{3,1} & r_{3,0} \end{vmatrix}$$

Fig. A3 Two-dimensional way

Table A2 4-bit S-boxes associated with the key in hexadecimal form

	S(x)	x															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
y	0,2,C,F	9	2	C	D	A	5	3	E	F	8	B	6	4	7	0	1
	1,5,9,D	1	A	4	5	2	D	B	6	7	0	3	E	C	F	8	9
	3,7,B,F	9	A	4	5	2	D	B	E	F	8	3	6	C	7	0	1
	4,6,8,A	1	A	C	5	2	D	3	6	F	8	B	E	4	7	0	9

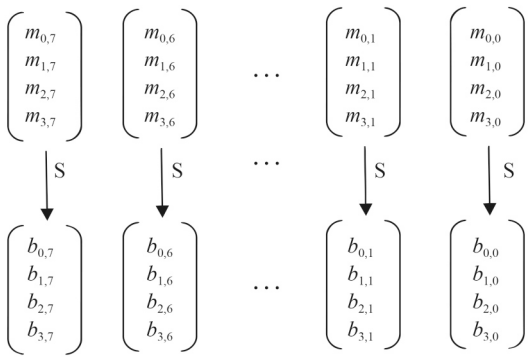

Fig. A4 The SubColumns operates on the columns of the state

Table A3 The 4 S-boxes of DBST

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	9	2	C	D	A	5	3	E	F	8	B	6	4	7	0	1
$S_2(x)$	1	A	4	5	2	D	B	6	7	0	3	E	C	F	8	9
$S_3(x)$	9	A	4	5	2	D	B	E	F	8	3	6	C	7	0	1
$S_4(x)$	1	A	C	5	2	D	3	6	F	8	B	E	4	7	0	9

Table A4 The differential distribution table of $S_1(x)$ and $S_2(x)$

Differential input	Differential output															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	0	2	0	0	0	2	0	0	0	2	0	4	0	2
2	0	0	0	0	4	2	2	0	0	2	0	2	0	0	2	2
3	0	0	0	2	4	2	2	2	0	2	0	0	0	0	2	0
4	0	0	0	4	0	0	0	4	0	0	0	4	0	0	0	4
5	0	0	2	0	0	0	2	0	4	0	2	0	4	0	2	0
6	0	2	0	0	0	0	2	0	2	2	2	0	2	0	0	4
7	0	2	2	0	0	0	0	4	2	2	0	0	2	0	2	0
8	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
9	0	4	2	0	0	0	2	0	0	0	2	0	0	4	2	0
A	0	0	2	2	4	2	0	2	0	2	2	0	0	0	0	0
B	0	0	2	0	4	2	0	0	0	2	2	2	0	0	0	2
C	0	0	0	0	0	4	0	0	4	0	0	0	4	4	0	0
D	0	0	2	0	0	4	2	0	0	0	2	0	0	4	2	0
E	0	2	0	4	0	0	2	0	2	2	2	0	2	0	0	0
F	0	2	2	0	0	0	0	0	2	2	0	4	2	0	2	0

The differential distribution table of $S_3(x)$ is as Table A5. Some characteristics of $S_3(x)$ can be obtained:

$$\begin{array}{lll} 0001-***1 & 0100-**11 & 1000-**1* \\ 1100-**0* & *1**-0100 & **0*-0101 \\ *1**-1000 & **1*-1001 & *01*-1100. \end{array} \quad (7)$$

The differential distribution table of $S_4(x)$ is as Table A6. Some characteristics of $S_4(x)$ can be obtained:

$$\begin{array}{lll} 0001-***1 & 0100-**11 & 0101-***0 \\ 1000-**1* & 1100-**0* & **1*-0001 \\ *01*-0100 & **0*-0101 & *1**-1000 \\ *1**-1100. & & \end{array} \quad (8)$$

Since the S-box of DBST is a group of dynamic S-box that depends on the subkeys, the differential characteristics of DBST S-box are same parts of $S_1(x)$, $S_2(x)$, $S_3(x)$, and $S_4(x)$, namely:

$$\begin{array}{lll} 0001-***1 & 0100-**11 & 1000-**1* \\ 1100-**0* & *1**-1000. & \end{array} \quad (9)$$

(2) RECTANGLE S-box is shown in Table A7. The differential distribution table of the S-box is Table A8.

Table A5 The differential distribution table of $S_3(x)$

Differential input	Differential output															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	0	2	0	4	0	2	0	0	0	2	0	0	0	2
2	0	0	0	2	0	0	2	0	0	2	0	0	4	2	2	2
3	0	0	0	0	0	0	2	2	0	2	0	2	4	2	2	0
4	0	0	0	4	0	0	0	4	0	0	0	4	0	0	0	4
5	0	0	2	0	4	0	2	0	4	0	2	0	0	0	2	0
6	0	2	2	0	2	0	2	0	2	2	0	0	0	0	0	4
7	0	2	0	0	2	0	0	4	2	2	2	0	0	0	2	0
8	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
9	0	4	2	0	0	4	2	0	0	0	2	0	0	0	2	0
A	0	0	2	0	0	0	0	2	0	2	2	2	4	2	0	0
B	0	0	2	2	0	0	0	0	0	2	2	0	4	2	0	2
C	0	0	0	0	4	4	0	0	4	0	0	0	0	4	0	0
D	0	0	2	0	0	4	2	0	0	0	2	0	0	4	2	0
E	0	2	2	0	2	0	2	0	2	2	0	4	0	0	0	0
F	0	2	0	4	2	0	0	0	2	2	2	0	0	0	2	0

Table A6 The differential distribution table of $S_4(x)$

Differential input	Differential output															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	4	0	2	0	4	0	2	0	0	0	2
2	0	2	0	0	4	0	2	0	0	0	0	2	0	2	2	2
3	0	2	0	2	4	0	2	2	0	0	0	0	0	2	2	0
4	0	0	0	4	0	0	0	4	0	0	0	4	0	0	0	4
5	0	0	2	0	0	0	2	0	4	0	2	0	4	0	2	0
6	0	2	2	0	0	0	0	0	2	2	0	0	2	0	2	4
7	0	2	0	0	0	0	2	4	2	2	2	0	2	0	0	0
8	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
9	0	0	2	0	0	4	2	0	0	4	2	0	0	0	2	0
A	0	2	2	2	4	0	0	2	0	0	2	0	0	2	0	0
B	0	2	2	0	4	0	0	0	0	0	2	2	0	2	0	2
C	0	0	0	0	0	4	0	0	4	0	0	0	4	4	0	0
D	0	0	2	0	0	4	2	0	0	0	2	0	0	4	2	0
E	0	2	2	4	0	0	0	0	2	2	0	0	2	0	2	0
F	0	2	0	0	0	0	2	0	2	2	2	4	2	0	0	0

Table A7 The RECTANGLE S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	6	5	C	A	1	E	7	9	B	0	3	D	8	F	4	2

Table A8 The differential distribution table of RECTANGLE $S(x)$

Differential input	Differential output															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	4	2	0	0	0	2	0	0	4	2
2	0	0	0	0	0	0	2	2	2	0	2	0	2	4	0	2
3	0	0	0	2	0	0	2	0	2	4	2	2	2	0	0	0
4	0	0	0	4	0	0	0	4	0	0	0	4	0	0	0	4
5	0	2	0	0	4	2	0	0	4	2	0	0	0	2	0	0
6	0	2	4	0	2	0	0	0	0	0	0	2	2	2	0	2
7	0	0	4	0	2	2	0	0	0	2	0	2	2	0	0	2
8	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
9	0	2	0	0	0	2	4	0	0	2	0	0	0	2	4	0
A	0	0	0	0	0	4	2	2	2	0	2	0	2	0	0	2
B	0	4	0	2	0	0	2	0	2	0	2	2	2	0	0	0
C	0	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0
D	0	2	0	0	0	2	0	0	0	2	4	0	0	2	4	0
E	0	0	4	2	2	2	0	2	0	2	0	0	2	0	0	0
F	0	2	4	2	2	0	0	2	0	0	0	0	2	2	0	0

Some characteristics of the S-box can be obtained:

$$\begin{array}{lll} 0001-**1* & 0100-**11 & 0101-**0* \\ 1000-***1 & 1100-***0 & *11*-0010 \\ *1**-0100 & *0**-0110 & **1*-1100 \\ **0*-1110. & & \end{array} \quad (10)$$

After a comparison, it is possible to conclude that the key-related dynamic SubColumns has a certain degree of randomness and hides some of the characteristics of $S_1(x)$, $S_2(x)$, $S_3(x)$, and $S_4(x)$. Compared with the RECTANGLE S-box, it is safer for difference analysis.

References

1. Bogdanov A, Knudsen L R, Leander G, Paar C, Poschmann A, Robshaw M J B, Seurin Y, Vikkelsoe C. PRESENT: an ultra-lightweight block cipher. In: Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems. 2007, 450–466
2. Feng J, Li L. SCENERY: a lightweight block cipher based on Feistel structure. Frontiers of Computer Science, 2022, 16(3): 163813
3. Banik S, Bao Z, Isobe T, Kubo H, Liu F, Minematsu K, Sakamoto K, Shibata N, Shigeri M. WARP: revisiting GFN for lightweight 128-bit block cipher. In: Proceedings of the 27th International Conference on Selected Areas in Cryptography. 2020, 535–564
4. Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L. The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference. 2015, 175
5. Guo Y, Li L, Liu B. Shadow: a lightweight block cipher for IoT nodes. IEEE Internet of Things Journal, 2021, 8(16): 13014–13023
6. Dai X, Huang Y, Chen L, Lu T, Su F. VH: a lightweight block cipher based on dual pseudo-random transformation. In: Proceedings of the 1st International Conference on Cloud Computing and Security. 2015, 3–13
7. Bansod G, Pisharoty N, Patil A. BORON: an ultra-lightweight and low power encryption design for pervasive computing. Frontiers of Information Technology & Electronic Engineering, 2017, 18(3): 317–331

8. Koo B, Roh D, Kim H, Jung Y, Lee D G, Kwon D. CHAM: a family of lightweight block ciphers for resource-constrained devices. In: Proceedings of the 20th International Conference on Information Security and Cryptology. 2017, 3–25
9. Zhang J, Zhao Y, Wu J, Chen B. LVPDA: a lightweight and verifiable privacy-preserving data aggregation scheme for edge-enabled IoT. IEEE Internet of Things Journal, 2020, 7(5): 4016–4027
10. Banik S, Pandey S K, Peyrin T, Sasaki Y, Sim S M, Todo Y. GIFT: a small present: towards reaching the limit of lightweight encryption. In: Proceedings of the 19th International Conference on Cryptographic Hardware and Embedded Systems. 2017, 321–345
11. Li L, Liu B, Wang H. QTL: a new ultra-lightweight block cipher. Microprocessors and Microsystems, 2016, 45: 45–55
12. Kwon J, Lee B, Lee J, Moon D. FPL: white-box secure block cipher using parallel table look-ups. In: Proceedings of Cryptographers' Track at the RSA Conference. 2020, 106–128
13. Li L, Liu B, Zhou Y, Zou Y. SFN: a new lightweight block cipher. Microprocessors and Microsystems, 2018, 60: 138–150
14. Zhang W, Bao Z, Lin D, Rijmen V, Yang B, Verbauwhede I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences, 2015, 58(12): 1–15
15. Biham E. A fast new DES implementation in software. In: Proceedings of the 4th International Workshop on Fast Software Encryption. 1997, 260–272
16. Chen L K, Zhang R T. Novel software block cipher using dynamic s-box and p-box. Computer Science, 2009, 36(2): 78–81
17. Chabaud F, Vaudenay S. Links between differential and linear cryptanalysis. In: Proceedings of Workshop on the Theory and Application of Cryptographic Techniques. 1994, 356–365
18. Kam J B, Davida G I. Structured design of substitution-permutation encryption networks. IEEE Transactions on Computers, 1979, C-28(10): 747–753
19. Feistel H. Cryptography and computer privacy. Scientific American, 1973, 228(5): 15–23
20. Webster A F, Tavares S E. On the design of S-boxes. In: Williams H C, ed. Advances in Cryptology — CRYPTO '85 Proceedings. Berlin: Springer, 1985, 523–534
21. Huang Y H, Dai X J, Shi Y Y, Liu N Z, Zeng Q X, Su F. Ultra-lightweight block cipher algorithm (PFP) based on feistel structure. Computer Science, 2017, 44(3): 163–167
22. Tiwari V, Singh A, Tentu A N. Differential cryptanalysis on DES cryptosystem up to eight rounds. International Journal of Information Privacy, Security and Integrity, 2019, 4(1): 1–29
23. Ashur T, Dunkelman O, Masalha N. Linear cryptanalysis reduced round of piccolo-80. In: Proceedings of the 3rd International Symposium on Cyber Security Cryptography and Machine Learning. 2019, 16–32
24. Tolba M, Abdelkhalek A, Youssef A M. Impossible differential cryptanalysis of reduced-round SKINNY. In: Proceedings of the 9th International Conference on Cryptology in Africa. 2017, 117–134
25. Courtois N T, Pieprzyk J. Cryptanalysis of block ciphers with overdefined systems of equations. In: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security. 2002, 267–287

Liuyan Yan was admitted to Hengyang Normal University, China in 2019 and is currently studying for a bachelor's degree at Hengyang Normal University, China. Since 2020, her research interests includes embedded systems and information security.

Lang Li received his PhD and Master's degrees in computer science from Hunan University, China in 2010 and 2006, respectively, and earned his BS degree in circuits and systems from Hunan Normal University, China in 1996. Since 2011, he has been working as a professor in the College of Computer Science and Technology at the Hengyang Normal University, China. His research interests include embedded computing and information security.

Ying Guo received the BS degree from Hengyang Normal University, China in 2019 and she is currently working toward a Master's degree in Hengyang Normal University, China. Since 2019, her current research interests include embedded systems and information security.