DABC: A dynamic ARX-based lightweight block cipher with high diffusion

Wen Chen[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
[e-mail: chenwen129@foxmail.com; lilang911@126.com; guoying129@foxmail.com]
[*]Corresponding author: Lang Li

Received November 17, 2022; accepted January 20, 2023;
published January 31, 2023

Abstract

The ARX-based lightweight block cipher is widely used in resource-constrained IoT devices due to fast and simple operation of software and hardware platforms. However, there are three weaknesses to ARX-based lightweight block ciphers. Firstly, only half of the data can be changed in one round. Secondly, traditional ARX-based lightweight block ciphers are static structures, which provide limited security. Thirdly, it has poor diffusion when the initial plaintext and key are all 0 or all 1. This paper proposes a new dynamic ARX-based lightweight block cipher to overcome these weaknesses, called DABC. DABC can change all data in one round, which overcomes the first weakness. This paper combines the key and the generalized two-dimensional cat map to construct a dynamic permutation layer P1, which improves the uncertainty between different rounds of DABC. The non-linear component of the round function alternately uses NAND gate and AND gate to increase the complexity of the attack, which overcomes the third weakness. Meanwhile, this paper proposes the round-based architecture of DABC and conducted ASIC and FPGA implementation. The hardware results show that DABC has less hardware resource and high throughput. Finally, the safety evaluation results show that DABC has a good avalanche effect and security.

Keywords: ARX-based lightweight block cipher; Dynamic encryption; Generalized two-dimensional cat map; Hardware implementation, High diffusion.

1. Introduction

With the development of technologies such as wireless communication and the Internet of Things (IoT), many sensor devices were widely used in our daily life [1]. However, IoT devices usually work in a wireless sensor network environment where sensor nodes can be easily captured by attackers [2]. It leads to some information leakage. Li et al. [3] considered the importance of providing security services in IoT, which makes people begin to pay attention to security in resource-constrained environments. At the same time, traditional block cipher (such as AES) cannot adapt to the resource-constrained environment due to high hardware resource consumption. Cipher designers have begun studying lightweight block ciphers to address this challenge. Different from traditional block ciphers, lightweight block ciphers with low resource and energy consumption are suitable for resource-constrained environments. The Feistel and the substitution-permutation networks (SPN) structure are its two mainstream cipher structures, respectively. Encryption and decryption of the Feistel structure are consistent, but the diffusion effect is poor, such as Piccolo [4], TWINE [5], QTL [6], etc. The SPN structure has a good diffusion speed but generally consumes additional resources when decrypting, such as PRESENT [7], Midori [8], RECTANGLE [9], SKINNY [10], etc.

Most of these mainstream ciphers use S-box to enhance confusion. There are ARX-based lightweight block ciphers without S-boxes in addition to these mainstream ciphers. ARX-based lightweight block ciphers are generally designed with simple operations, such as Addition, AND, Rotation, and XOR. It is superior to other ciphers with S-box according to the test results of the "Fair Evaluation of Lightweight Cryptography System (FELICS) [11]" platform designed by researchers at the University of Luxembourg. Some classical ARX-based lightweight block ciphers have been proposed in recent years. D. Hong et al. [12] proposed HIGHT by ARX operations, which is adapted to low-resource environments. R. Beaulieu et al. [13] proposed SIMON and SPECK with good performance in hardware implementation and software implementation, respectively. However, proof of their theoretical security has always been a difficult task. G. Yang et al. [14] combined the advantages of SIMON and SPECK to propose the Simeck, which is less than the SMION in terms of hardware resources and power consumption. Y. Guo and L. Li et al. [15] proposed Shadow to protect the security of sensory layer nodes in IoT, which overcomes the problem of poor diffusion of ARX-based lightweight block cipher. S. Chen and M. Wang et al. [16] proposed SAND. The authors used equivalent S-box-based representation to evaluate the security of SAND, which provides a new way to analyze the security of ARX-based lightweight block ciphers. In general, these ARX-based lightweight block ciphers are competitive in terms of hardware and software implementations. However, except for Shadow, these ARX-based lightweight block ciphers generally have the weakness of slow diffusion (only half the data can be changed in one round). For Shadow, it solves the weakness by combining four ARX operations and branch swapping. Therefore, Shadow is not as impressive in terms of latency. Furthermore, ARX-based lightweight block cipher generally suffers from poor diffusion when the initial plaintext and key are all 0 or all 1.

The lightweight ARX-based block ciphers currently designed are all static structures. Some security analysis of them has been carried out by cryptographers. B. Koo et al. [17] and J. Song et al. [18] proposed a correlation key attack and a biclique attack for the HIGHT full-round, respectively. B. Ryabko and A. Soskov [19] proposed an integral attack on Simeck

48/96 full-round. J. Lu et al. [20] proposed Simeck64/128 to construct 17-round neural discriminators (NDs) and 21-round correlated key neural discriminators (RKNDs). K. Fu et al. [21] proposed a mixed integer linear program (MILP)-based method to automatically search for differential features and linear approximations of SPECK. They also use this feature to improve the differential analysis of SPECK, which improves the analysis effect. It can be seen that the security provided by static structures is limited. Therefore, this paper attempts to improve the resistance of ARX-based lightweight block ciphers to potential attacks by adding less dynamic control logic and dynamic calling of some components. There are also some current designs for dynamic ciphers. J. Yang and L. Li et al. [22] proposed an SPN-based dynamic lightweight block cipher DULBC, which is a dynamic variation of the overall structure of the cipher. A. D. Dwivedi [23] proposed BRISK, which uses two different cryptographic components to encrypt data. It is worth noting that dynamic changes to the overall structure of the cipher will require more hardware resources. In contrast to these cipher design ideas, this paper is a dynamic variation on the linear component of the cipher. It does not consume excessive additional hardware resources in hardware implementation. In each round of encryption, the non-linear operations are still performed first and then the diffusion of the cipher is enhanced by different linear permutation operations.

For ARX-based lightweight block ciphers, this paper summarizes three of their currently prevalent weaknesses. 1) Only half of the data is changed in one round, which is poorly diffused. It also makes them require a large number of iterative rounds to improve the security of the cipher. 2) Traditional ARX-based lightweight block ciphers are usually a static structure. The security of their encryption is usually not particularly high. 3) ARX-based lightweight block ciphers have poor diffusion when the initial plaintext and key are all 0 or all 1. It is mainly due to the nature of the AND operation. To overcome the above three weaknesses, this paper proposes a new dynamic ARX-based lightweight block cipher, called DABC. Our contributions mainly include the following aspects:

- This paper combines ARX operation and Feistel structure to design a new variant of the generalized Feistel network (GFN) structure. The structure can change all data in one round, which overcomes the weakness of traditional ARX-based lightweight block cipher that only changes half of the data in one round. Meanwhile, its encryption and decryption are highly consistent.
- The dynamic permutation layer is proposed by combining the generalized two-dimensional cat map with the key. The first 2 bits of the key are used as the parameters of the generalized two-dimensional cat map. The dynamic permutation layer with four different permutation effects is constructed by circular shift and iterative operations on the generalized two-dimensional cat map, which improves the uncertainty between different rounds of the cipher. It can implement by wiring and does not consume excessive resources in hardware implementation.
- The odd and even rounds use round functions F_1 and F_0 for encryption, respectively. F_0 uses AND operation as a non-linear component, while F_1 uses the NAND operation. The approach increases the complexity of the attack without consuming too many hardware implementation resources. It solves the weakness of poor diffusion well when the initial plaintext and the key are all 0 or all 1.
- This paper conducts ASIC implementation and FPGA implementation on DABC. Avalanche effect tests and safety analysis are also performed in this paper. The results show that DABC is safe, efficient, and adaptable to resource-constrained environments.

The rest of the paper is organized as follows. This paper introduces the cipher specifications of DABC in Section 2 and the design principles of DABC in Section 3. The

hardware implementation of DABC and the avalanche effect test are arranged in Section 4. Section 5 is the security analysis of the DABC. Finally, Section 6 concludes this paper.

2. Specification of DABC

DABC is based on a variant of the GFN structure with four branches, which has a 64-bit block with a 96-bit or 128-bit key block. The number of encryption rounds (R) for DABC64/96 and DABC64/128 is 27 and 33, respectively.

2.1 Encryption algorithm

The plaintext is divided into four blocks of the same size in the encryption process of DABC. Then the operations such as the round function, AddRoundKey, and P1 permutation are performed sequentially. DABC mainly uses NAND and AND operations as non-linear components, while the P1 permutation layer is used as a linear component. In odd rounds, the encryption process calls the round function F_1 and the key update calls the round function F_0. The opposite is true in even rounds. **Fig. 1** shows the encryption process of DABC. **Algorithm 1** shows the detailed process of DABC.

Algorithm 1 DABC Encryption Routine.
Input: Plaintext, Key;
Output: Ciphertext;
1: $(X_0, X_1, X_2, X_3) \leftarrow$ Plaintext;
2: **for** $r = 1$ **to** R **do**
3: $X_0' = X_0 \oplus X_2 \oplus k_i^r$;
4: **if** $(r \,\%\, 2 == 0)$ **then**
5: $X_1' = X_1 \oplus F_0(X_0) \oplus k_i^l$;
6: $X_3' = X_3 \oplus F_0(X_0) \oplus X_1$;
7: $X_2' = X_2 \oplus F_0(X_3') \oplus k_i^r$;
8: **else**
9: $X_1' = X_1 \oplus F_1(X_0) \oplus k_i^l$;
10: $X_3' = X_3 \oplus F_1(X_0) \oplus X_1$;
11: $X_2' = X_2 \oplus F_1(X_3') \oplus k_i^r$;
12: **end if**
13: **if** $(r \leq (R-1))$ **then**
14: $state = \mathrm{P1}(X_0' \| X_1' \| X_2' \| X_3')$;
15: **else**
16: $state = X_0' \| X_1' \| X_2' \| X_3'$;
17: **end if**
18: **end for**
19: Ciphertext $\leftarrow state$;
20: **Return** Ciphertext;

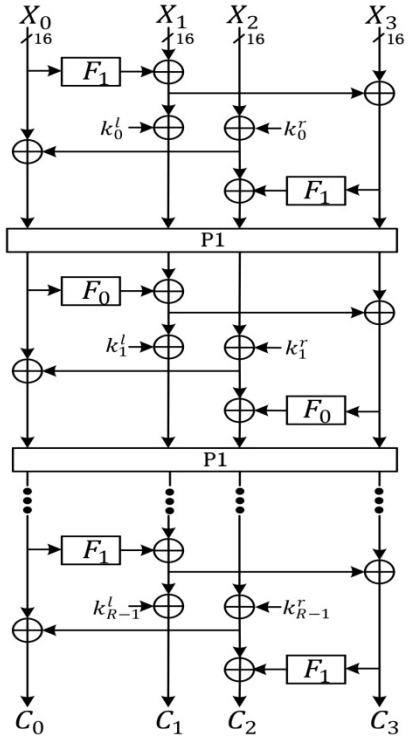

Fig. 1. The encryption process of DABC.

2.2 Round function

The main difference between the round functions F_0 and F_1 is that F_1 uses the NAND operation (!&) as a nonlinear component, while F_0 uses the AND operation (&). **Fig. 2** shows the round functions F_0 and F_1. They together include three operations: left circular shift by 3 bits ($\lll 3$), left circular shift by 1 bit ($\lll 1$), and an XOR operation ($\oplus$).

The F_0 as described in formula (1), where $n = 64$:

$$\begin{cases} \{0,1\}^{n/4} \times \{0,1\}^{n/4} \longrightarrow \{0,1\}^{n/4} \\ X' = X \\ Y' = F_0(X) \oplus Y = ((X \,\&\, (X^{\lll 3})) \oplus X^{\lll 1}) \oplus Y. \end{cases} \tag{1}$$

The F_1 as described in formula (2):

$$\begin{cases} \{0,1\}^{n/4} \times \{0,1\}^{n/4} \longrightarrow \{0,1\}^{n/4} \\ X' = X \\ Y' = F_1(X) \oplus Y = ((X \,!\& \, (X^{\lll 3})) \oplus X^{\lll 1}) \oplus Y. \end{cases} \tag{2}$$

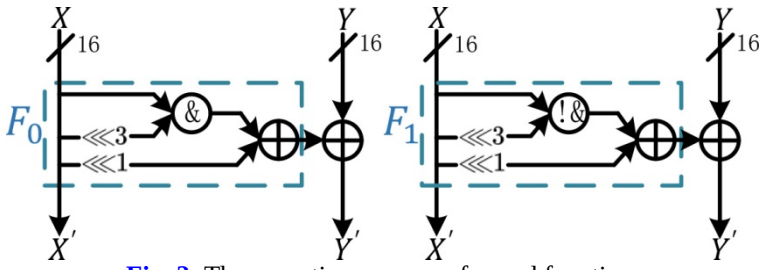

Fig. 2. The operation process of round function.

2.3 P1 permutation layer and AddRoundKey

P1 permutation layer: The P1 permutation layer of DABC is dynamic. It performs different bit-level permutation operations on the input data according to the value of the key. More details are described in Section 3.3.

AddRoundKey: The AddRoundKey means that two 16-bit block data are XORed with the round keys of this encryption round.

2.4 Key generator and AddRoundconstant

Key generator: The key generator of DABC supports two different versions of keys, 96-bit and 128-bit. The 96-bit initial key $k_0||k_1||\ldots||k_{95}$ is divided into six 16-bit blocks $K_0||K_1||K_2||K_3||K_4||K_5$. Then proceed as follows, where n takes 0 in odd rounds and 1 in even rounds, $0 \le i \le R-1$, $0 \le counter \le R-1$:

- $K_0||K_1||K_2||K_3||K_4||K_5 = (k_0||k_1||\cdots||k_{94}||k_{95})^{<<<37}$;
- $k_{60}||k_{61}||k_{62}||k_{63}||k_{64} = k_{60}||k_{61}||k_{62}||k_{63}||k_{64} \oplus counter$;
- $k_i^l = F_n(K_0) \oplus K_2, k_i^r = F_n(K_1) \oplus K_3$.

The 128-bit initial key $k_0||k_1||\cdots||k_{127}$ is divided into eight 16-bit blocks $K_0||K_1||K_2||K_3||K_4||K_5||K_6||K_7$. The following operations are performed in turn:

- $K_0||K_1||K_2||K_3||K_4||K_5||K_6||K_7 = (k_0||k_1||\cdots||k_{127})^{<<<38}$;
- $k_{60}||k_{61}||k_{62}||k_{63}||k_{64}||k_{65} = k_{60}||k_{61}||k_{62}||k_{63}||k_{64}||k_{65} \oplus counter$;
- $k_i^l = F_0(K_0) \oplus K_2 \oplus F_1(K_4), k_i^r = F_0(K_1) \oplus K_3 \oplus F_1(K_5)$.

$Key' = K_0||K_1||k_i^l||k_i^r||K_4||K_5$ (for 128-bit initial key, $Key' = K_0||K_1||k_i^l||k_i^r||K_4||K_5||K_6||K_7$) as the input value for next round of key updates. The generated k_i^l and k_i^r are round keys used for the AddRoundKey operation. **Fig. 3** shows the core process of the DABC64/96 key generator.

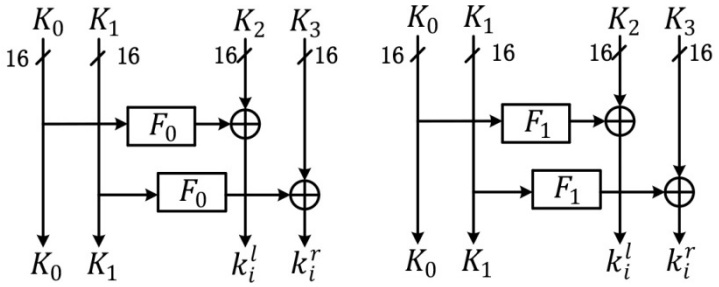

Fig. 3. The core operation process of key generator (left: odd rounds, right: even rounds).

AddRoundconstant: The current number of encryption rounds is used as the round constant ($counter$) of DABC, which is described as $c_0||c_1||c_2||c_3||c_4$ (for 128-bit initial key, $counter = c_0||c_1||c_2||c_3||c_4||c_5$). The round constant is XORed with the corresponding bits of the key.

2.5 Decryption algorithm

It is necessary to perform linear rP1 and NP1 permutations on the encryption data to make the encryption and decryption structures highly reusable. See section 3.3 for the principle of NP1. In addition, the round keys are used in reverse order during decryption. The two round keys k_i^l and k_i^r of the same round also need to be swapped in the AddRoundKey operation. rP1 permutation means that the branches are output in reverse order, which is performed before and after the round function operation. In the hardware implementation, the rP1 and NP1 permutations can be implemented by wiring operations. Therefore the decryption process does not consume many additional resources.

3. Design rationale of DABC

DABC is designed mainly to solve the poor diffusion of traditional ARX-based lightweight block ciphers and increases security through a dynamic encryption structure. In order to avoid continuous linear operations, this paper chooses dynamic changes in some components instead of dynamic changes in the overall structure of the cipher. It also reduces the resources consumed in hardware implementation. The specific design details of DABC are described next.

3.1 Cipher structure

The DABC proposed in this paper has the following characteristics. Firstly, it can change all data in one round of encryption, which overcomes the slow diffusion of the traditional ARX-based lightweight block cipher. Secondly, the round functions F_1 and F_0 are used alternately in DABC encryption, which is safer than a single round function. At the same time, a dynamic permutation layer P1 is designed based on the generalized two-dimensional cat map, which increases the randomness of the cipher structure. Finally, DABC retains the advantage that the decryption process of the Feistel structure can highly reuse the encryption process. Therefore, the decryption process does not consume many additional resources.

3.2 Round function F_0 and F_1

The ARX-based lightweight block cipher suffers from poor diffusion when the plaintext and the key are all 0 or all 1. For example, the first 4 rounds of SIMON, SPECK, and SAND ciphertext change very little when plaintext and key are all 0 or all 1. It is not until rounds 5 or 6 that the ciphertext changes significantly, which is caused by the nature of the AND operation. This paper introduces the NAND operation to break the effect of AND operation. The purpose of using round functions alternatively is also to solve this problem. DABC changed significantly in the second round of ciphertext when the plaintext and the key are all 0 or all 1, so it can be considered that DABC has high diffusion. AND and NAND operations are used as non-linear components for the round functions F_0 and F_1, respectively. NAND gate resources in most standard libraries are less than AND gate resources. Compared to traditional ARX-based lightweight block ciphers, the alternate use of round functions can increase the complexity of the attack without increasing too many hardware resources. The round function also involves left circular shift and XOR operations. Overall, it is beneficial to hardware implementation.

3.3 Key generator

DABC supports two key sizes to accommodate resource-constrained environments with different security strengths. For high-security environments, the 128-bit version of the key meets the security key length requirements in the NIST standard. The order of round function calls in the key generator is the opposite of the encryption process. It effectively improves the problem of poor diffusion when the plaintext and the key are all 0 or all 1.

3.4 P1 and NP1 permutation layer

The two-dimensional cat map is a pixel shuffling scheme that goes through multiple iterations to upset the pixel positions in the image, which happens to be similar to the effect of a linear permutation layer in a grouping cipher. However, the first position coordinate does not change when position coordinates are transformed through the two-dimensional cat map. To avoid this situation, a left-loop operation is added after each iteration in this paper.

This paper designs a key-related dynamic permutation layer P1 based on the generalized two-dimensional cat map. P1 increases the randomness of the cipher structure more than the traditional static permutation layer. It can be implemented with wiring operations, which do not consume excessive hardware resources. The generalized two-dimensional cat map used by DABC is shown in formula (3):

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} g+2 & g+1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \bmod N \tag{3}$$

where g is the control parameter of the generalized two-dimensional cat map. N is the number of digits of input data, which is 8. Let x_0 and y_0 be the original position coordinates of the data, and let x_1 and y_1 be the position coordinates after being scrambled by the generalized two-dimensional cat map.

The input data of the P1 is arranged in the form of 8×8 two-dimensional matrix, as shown in formula (4):

$$\begin{bmatrix} Y_{0,0} & Y_{0,1} & \cdots & Y_{0,y_0} \\ Y_{1,0} & Y_{1,1} & \cdots & Y_{1,y_0} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{x_0,0} & Y_{x_0,1} & \cdots & Y_{x_0,y_0} \end{bmatrix} \tag{4}$$

The position coordinates of data in the matrix are used as the input values of generalized two-dimensional cat map. The results of map are used as the new position coordinates of data. The above operation is iterated 16 times. In the first 15 iterations, the result is subjected to a 3-bit left circular shift operation after each iteration is completed. The permutation relationship between the final permutation result and the position coordinates of initial data is used as the P1 permutation. It should be noted that the first 2 bits of the key are used as the value of map control parameter g. Four permutation tables are generated by the above method based on different values of the control parameter g. Select one of the tables to replace the data based on the first 2 bits of the key. **Tables 1 to 4** show the different elemental values of P1, respectively.

Table 1. P1 permutation table when $g = 0$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	27	26	29	33	3	2	23	52	36	37	7	44	24	6	11	8
1	13	20	63	0	59	4	61	53	62	46	1	51	55	21	32	42
2	15	30	60	22	31	34	35	54	10	38	39	40	41	58	25	12
3	45	48	57	19	49	50	9	18	14	5	43	17	16	28	56	47

Table 2. P1 permutation table when $g = 1$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	30	26	7	53	20	34	38	33	1	39	35	8	23	6	18	12
1	21	3	55	58	24	50	28	9	49	37	19	56	4	54	27	61
2	45	52	46	36	29	59	63	22	60	14	10	40	41	31	43	44
3	16	42	25	15	13	11	47	48	5	62	32	17	57	0	2	51

Table 3. P1 permutation table when $g = 2$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	55	19	35	17	20	11	3	4	5	6	13	44	9	10	38	8
1	0	14	15	61	25	21	2	43	39	22	53	31	16	46	27	28
2	29	30	56	32	33	34	23	41	37	7	59	40	51	24	1	12
3	45	54	47	48	49	18	36	52	50	42	62	26	57	58	63	60

Table 4. P1 permutation table when $g = 3$.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	46	9	37	23	35	13	21	34	17	59	56	8	28	29	60	12
1	55	36	4	50	32	25	49	39	10	7	38	47	2	11	27	43
2	51	48	16	54	45	0	61	52	18	53	1	40	22	42	58	44
3	26	31	62	33	15	20	3	6	19	24	63	30	57	5	14	41

The NP1 permutation layer is constructed from the inverse map of the generalized two-dimensional cat map. Formula (5) shows its inverse map.

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 1 & -g-1 \\ -1 & g+2 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \bmod N \tag{5}$$

The NP1 layer is constructed in a similar way to the P1 layer. The difference is that in the first 15 of the 16 iterations, a 3-bit right circular shift operation is performed after each completion.

4. Performance evaluation

4.1 Avalanche effect

The avalanche effect is an indicator of the diffusion effect, which is related to the randomness of the cipher. If the avalanche effect is too poor, the cipher can be easily cracked by an attacker. Therefore, the designed cipher needs to meet the avalanche effect. Half of the output binary bits change when one of the binary bits of the cipher input changes. It can be considered the cipher with a good avalanche effect. The strict avalanche criterion means that every bit in the output has a 50% probability of changing when any input bit is inverted.

This paper assesses average change digit (w_v), average change probability (Pr_v), avalanche effect degree (A_d), and strict avalanche criterion degree (A_{cd}) of the output change when the input 1-bit changes. In addition, $w_v \approx n/2$ (n is the bit width of the plaintext), $A_d \approx 1$, and $A_{cd} \approx 1$ indicate that the avalanche effect of the cipher is splendid. $Pr_v \approx 0.5$ indicates that the cipher satisfies the strict avalanche criterion.

In order to make the experimental data results more accurate and scientific, this paper has carried out 10,000 sample tests on DABC, Piccolo64/128, HIGHT64/128, Simeck64/128, SIMON64/96, and SPECK64/96 according to the same avalanche effect test method. **Table 5** shows that w_v, Pr_v, A_d, and A_{cd} of DABC are as close to ideal as the compared ciphers. In summary, DABC can be considered to have a good avalanche effect.

Table 5. Comparison of avalanche effect data of various ciphers.

Cipher	Block size	Key size	w_v	Pr_v	A_d	A_{cd}
Piccolo	64	128	31.9939	0.4999	0.9998	0.9998
HIGHT	64	128	31.9952	0.4999	0.9999	0.9998
Simeck	64	128	31.9890	0.4998	0.9998	0.9998
SIMON	64	96	32.0074	0.5001	0.9996	0.9996
SPECK	64	96	31.9954	0.4999	0.9997	0.9997
DABC64/96	64	96	32.0017	0.5000	0.9999	0.9997
DABC64/128	64	128	32.0076	0.5001	0.9998	0.9998

4.2 Hardware implementation

The throughput of cipher is important, which is related to the efficiency of data transfer in the network. At the same time, it is also necessary to consider the hardware resource and energy consumption of the cipher in a resource-constrained environment. This paper proposes a round-based architecture for DABC. **Fig. 4** shows the round-based architecture for DABC64/96. The architecture is conducive to the low-resource implementation of DABC for resource-constrained environments. The design idea of round-based architecture is repeatedly invoking the components of encryption. One round of encryption is one clock cycle. Twenty-seven and thirty-three clock cycles are required for DABC64/96 and DABC64/128, respectively.

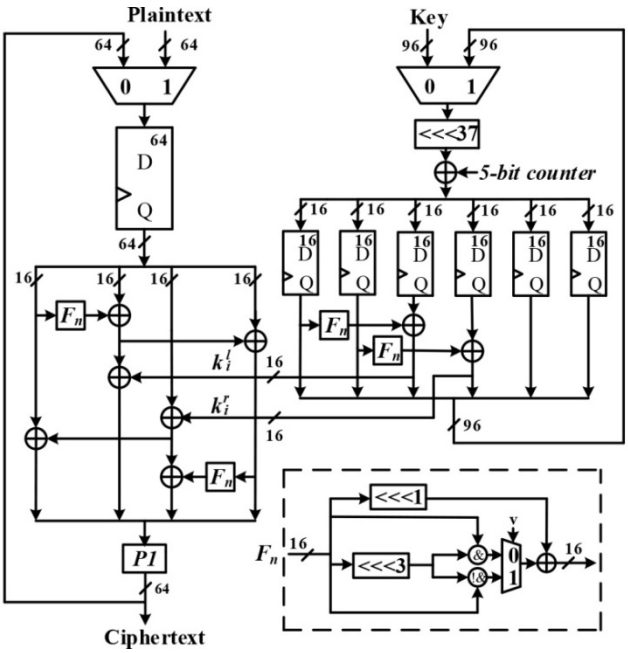

Fig. 4. Round-based architecture for DABC.

4.2.1 ASIC implementation

The ASIC implementation of DABC has been synthesized by the UMCL18G212 Synchronous Design Compiler A-2007.12-SP1T3 standard cell library. The logic process of the library is UMCL180 0.18μm 1P6M with a clock frequency of 100KHz and a standard voltage of 1.8V.

The round-based architecture of DABC mainly includes the following operations: round function, AddRoundKey, AddRoundconstant, and dynamic P1 permutation. Among them, the circular shift operation in the round function module can be realized by wiring operation without spending additional hardware resources. The dynamic effect of P1 permutation is achieved by adding a 4-to-1 multiplexer. Use the first 2 bits of each Key as a control signal to select the permutation table corresponding to g=0, 1, 2, or 3 in the order, when the 2 bits are "00", "01", "10", or "11". The specific hardware resource consumption of DABC is as follows (in 0.18μm standard process library): an AND gate consumes 1.33GE, a NAND gate consumes 1GE, and an XOR gate consumes 2.67GE. The round function module includes a 16-bit XOR gate, a 16-bit AND gate, and a 16-bit NAND gate, which require a total of 80GE. Two 16-bit AddRoundKey is equivalent to two 16-bit XOR gates, which require 85.44GE. The encryption process includes four 16-bit XOR operations and two round function modules, which require a total of 330.88GE. 5-bit AddRoundconstant is equivalent to 5-bit XOR operation, which requires 13.35GE (which requires 16.02GE for DABC64/128). The 96-bit and 128-bit key generators require 245.44GE and 405.44GE, respectively. The 64-bit plaintext and 96-bit key registers require a total of 832.32GE. The 64-bit plaintext and 128-bit key registers require a total of 981.76GE. The control logic requires 20GE. In total, DABC64/96 and DABC64/128 require 1527.43GE and 1839.54GE, respectively. For better performance comparison, this paper also implemented DABC64/96 and DABC64/128 under IBM 130μm standard process library (for 0.13μm, an AND gate consumes 1.25GE, a NAND gate consumes 0.75GE, and an XOR gate consumes 2GE). **Table 6** shows the resource comparison of DABC with other lightweight packet ciphers. In general, DABC has certain advantages in hardware implementation.

Table 6. Comparison of hardware resources between DABC and other ciphers.

Cipher	Structure	Data(bit)	Key(bit)	Area(GE)	Logic process	Ref.
PRINCE	SPN	64	128	3491	0.18μm	[24]
PRESENT	SPN	64	80	2018	0.18μm	[25]
PRESENT	SPN	64	128	2783	0.18μm	[26]
SIMON	ARX	64	128	1751	0.13μm	[27]
SPECK	ARX	64	128	2014	0.13μm	[27]
LEA	ARX	128	128	3826	0.18μm	[28]
Piccolo	GFN	64	80	1634	0.18μm	[25]
TWINE	GFN	64	80	1799	90nm	[25]
HIGHT	GFN	64	128	3048	0.25μm	[12]
DULBC64/80	SPN	64	80	1484	0.13μm	[22]
DULBC64/128	SPN	64	128	1765	0.13μm	[22]
DABC64/96	GFN	64	96	1363.76	0.13μm	This work
DABC64/128	GFN	64	128	1675.22	0.13μm	This work
DABC64/96	GFN	64	96	1527.43	0.18μm	This work
DABC64/128	GFN	64	128	1839.54	0.18μm	This work

4.2.2 FPGA implementation

The round-based RTL file of DABC was obtained through ISE Design Suite 14.6 and mapped to the Xilinx Virtex-5 FPGA development board to evaluate its hardware performance. **Table 7** shows the hardware performance of the DABC.

Table 7 Hardware performance of DABC on Virtex-5.

Structure	Cipher	Slice	LUT	Flip Flop	Energy (nJ/bit)	Throughput (Mbps)
Round-based	DABC64/96	138	370	167	1.09	719.668
	DABC64/128	192	485	200	1.3	573.506

This paper implements DABC on Virtex-5 FPGA and tries its best to find some lightweight block ciphers implemented on Virtex-5 FPGA. **Table 8** shows the FPGA implementation performance with different ciphers, where Thr is the throughput at the maximum frequency (F). This paper also counts the throughput Thr^* under a unified 13.56MHz to better compare the throughput. Although the Thr^* of LEA128/128 [29] and LEA128/128 [30] are higher than DABC64/128, the LUT of LEA128/128 [29] and LEA128/128 [30] are 47% and 52% higher than DABC64/128, respectively. The Flip Flop, Slice, and LUT of other ciphers are not much different from DABC, but Thr^* is lower than DABC. **Fig. 5** shows the comparison of Flip Flop, Slice, and LUT parameters implemented on Virtex-5 FPGA. **Fig. 6** shows the throughput comparison of each cipher at the 13.56MHz frequency on Virtex-5 FPGA. In **Fig. 5** and **Fig. 6**, a, b, c, d, e, f, g, h, i, j, k, and l represent the lightweight block ciphers ^{I}PRESENT, PRESENT64/80, ^{R}PRESENT64/128, PRESENT64/128, LEA128/128 [28], LEA128/128 [29], LEA 128/128 [30], ^{RAM}RECTANGLE, QTL64/64, XTEA, DABC64/96, and DABC64/128, respectively. In summary, it can be seen that DABC achieves a good balance between throughput and hardware implementation resources from the above comparison results.

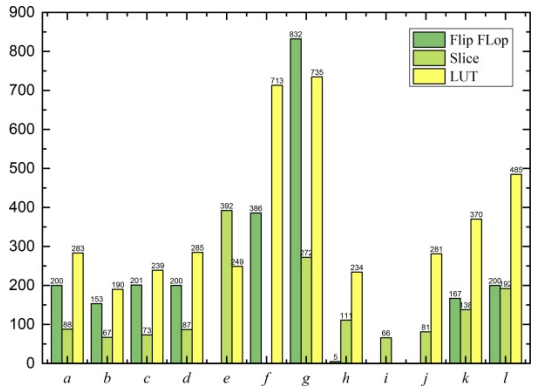

Fig. 5. Flip Flop, Slice and LUT comparison of various ciphers on Virtex-5.

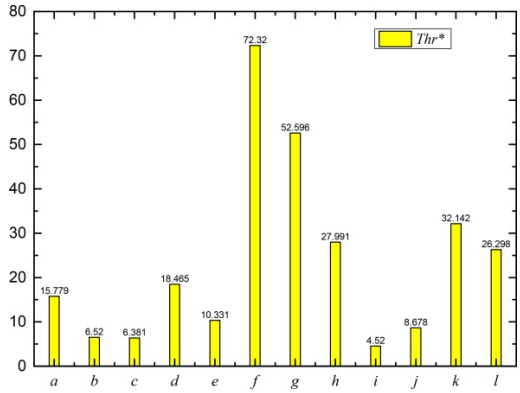

Fig. 6. Throughput comparison of various ciphers at 13.56MHz onVirtex-5.

Table 8. Hardware performance comparison of various ciphers on Virtex-5.

Cipher	Slice	LUT	Flip Flop	F(MHz)	Thr(Mbps)	Thr^*(Mbps)	Ref.
^{I}PRESENT	88	283	200	271.67	316.12	15.779	[31]
PRESENT64/80	67	190	153	542.3	260.96	6.52	[31]
^{R}PRESENT64/128	87	285	200	250.86	341.64	18.465	[32]
PRESENT64/128	73	239	201	431.78	203.19	6.381	[31]
LEA128/128	392	249	-	26.658	205.45	10.331	[28]
LEA128/128	-	713	382	217.806	1161.63	72.32	[29]
LEA128/128	272	735	832	311	1206.3	52.596	[30]
^{RAM}RECTANGLE	81	281	-	390.778	250.098	8.678	[33]
QTL64/64	111	235	5	111.63	230.46	27.991	[34]
XTEA	66	-	-	333	111	4.52	[35]
DABC64/96	138	370	167	303.61	719.668	32.142	This work
DABC64/128	192	485	200	295.714	573.506	26.298	This work

Thr^*: represents the parameter calculated at a fixed frequency of 13.56 MHz. F: represents maximum frequency. -: represents uncertainty. ^{I}: represents the Iterative architecture. ^{R}: represents the Round-based architecture. ^{RAM}: represents the RAM based architecture.

5. Security analysis

5.1 Differential analysis

Differential analysis is the analysis method introduced by E. Biham and A. Shamir [36] in the cryptanalysis of FEAL, which was the classical analysis method in cryptanalysis. The main idea of differential analysis is to take a pair of plaintexts and analyze the propagation of their differences by the iterative process.

This paper refers to the method of K. Fu et al. [21] to calculate the differential probability of DABC. Differential analysis modeling of DABC using MILP automatic analysis method with the help of Gurobi optimizer. The best differential characteristics were obtained, as shown in **Table 9**. It can be seen that the differential probability for the first 10 rounds of DABC is 2^{-146}, which is much smaller than 2^{-64}. Therefore, DABC64/96 for 27 rounds and DABC64/128 for 33 rounds can be considered to resist differential analysis.

Table 9. Differential characteristic for DABC.

Round	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\log_2 p_i$	2^{-2}	2^{-4}	2^{-6}	2^{-12}	2^{-19}	2^{-28}	2^{-34}	2^{-41}
$\sum_{i=1}^{10} \log_2 p_i$	2^{-146}							

5.2 Linear analysis

Linear analysis is a new attack method proposed by M. Matsui [37] for the DES. Similar to differential analysis, linear analysis is also a classical analysis method in cryptanalysis. It recovers the key by studying the linear relationship between plaintext and ciphertext. This paper assumes its input mask value and tracks the iterative process of the DABC round function to obtain the corresponding linear characteristic probabilities. Its value is sufficiently small by calculating its linear probability. Therefore, DABC is considered to resist linear analysis.

5.3 Impossible differential analysis

Impossible differential analysis [38] is a variant of differential analysis. In this section, impossible differential analysis is performed for DABC. This paper finds two differential paths E_0 and E_1 of DABC with a differential probability of 1. E_0: $\Delta\alpha \rightarrow \Delta\beta$ is the path of encryption direction. E_1: $\Delta\gamma \rightarrow \Delta\zeta$ is the path of decryption direction. "1" means that the bit difference value is 1, "0" means that the bit difference value is 0, and "*" means that the bit difference value is uncertain. The input difference $\Delta\alpha$ of E_0 is (0000, 0001, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000). Carrying out two rounds of encryption operation, the output difference $\Delta\beta$ is obtained as (0*0*, *0**, ****, ****, 0*00, 00**, 0*0*, ***0, 00**, 0***, ****, 0**0, ****, 0*1*, 000*, ****). The input difference $\Delta\gamma$ of E_1 is (1000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000). After two rounds of decryption operations, the output difference $\Delta\zeta$ is obtained as (00*1, 0*00, 0000, 0000, 0000, 0001, 0000, 0010, 0100, 0000, 0000, 0000, 0*00, *0*1, 0000, 0000). **Fig. 7** shows the impossible differential path for 4 rounds of DABC. It can be seen from **Fig. 7** that there is a contradiction when E_0 and E_1 meet, as shown in the red box in **Fig. 7**. The contradiction is the difference value for the 33rd position of $\Delta\beta$ is not equal to the 33rd position of $\Delta\zeta$. It constitutes a 4-round impossible difference path of DABC.

One round of forwarding decryption and two rounds of backward encryption based on the 4-round impossible difference path. The 7-round impossible difference path of DABC is obtained, as shown in **Fig. 8**.

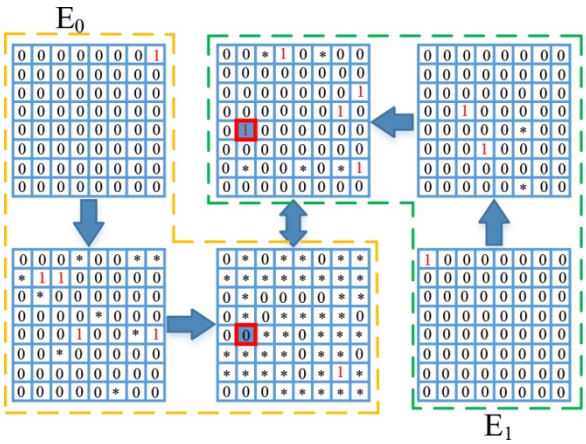

Fig. 7. Impossible differential path for 4 rounds of DABC.

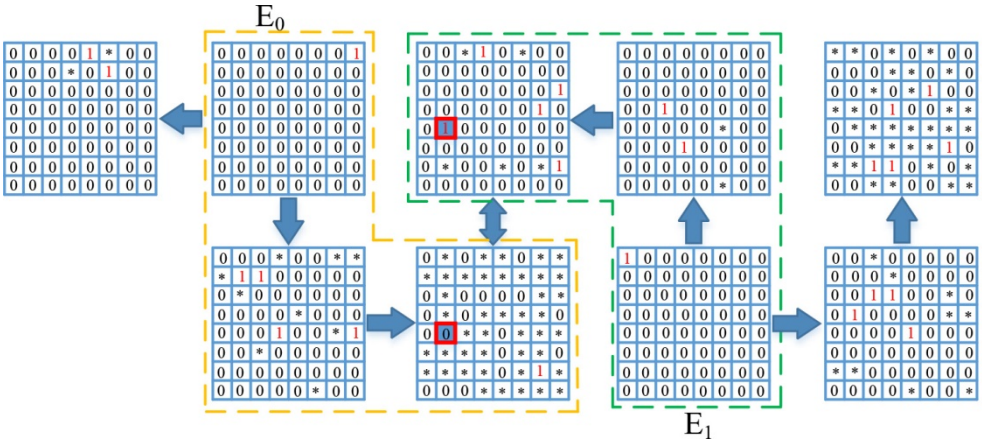

Fig. 8. Impossible differential path for 7 rounds of DABC.

According to the 7-round impossible differential path, the plaintext input differential is set as:

$$\begin{aligned} X_0 &= (0000,1*00,000*,0100);\\ X_1 &= (0000,0000,0000,0000);\\ X_2 &= (0000,0000,0000,0000);\\ X_3 &= (0000,0000,0000,0000). \end{aligned} \tag{6}$$

Setting the plaintext structure as follows:

$$\begin{aligned} W_{X_0}^0 &= (\alpha_1\alpha_2\alpha_3\alpha_4, \alpha_5\beta_1\alpha_6\alpha_7, \alpha_8\alpha_9\alpha_{10}\beta_2, \alpha_{11}\alpha_{12}\alpha_{13}\alpha_{14});\\ W_{X_1}^0 &= (\alpha_{15}\alpha_{16}\alpha_{17}\alpha_{18}, \alpha_{19}\alpha_{20}\alpha_{21}\alpha_{22}, \alpha_{23}\alpha_{24}\alpha_{25}\alpha_{26}, \alpha_{27}\alpha_{28}\alpha_{29}\alpha_{30});\\ W_{X_2}^0 &= (\alpha_{31}\alpha_{32}\alpha_{33}\alpha_{34}, \alpha_{35}\alpha_{36}\alpha_{37}\alpha_{38}, \alpha_{39}\alpha_{40}\alpha_{41}\alpha_{42}, \alpha_{43}\alpha_{44}\alpha_{45}\alpha_{46});\\ W_{X_3}^0 &= (\alpha_{47}\alpha_{48}\alpha_{49}\alpha_{50}, \alpha_{51}\alpha_{52}\alpha_{53}\alpha_{54}, \alpha_{55}\alpha_{56}\alpha_{57}\alpha_{58}, \alpha_{59}\alpha_{60}\alpha_{61}\alpha_{62}), \end{aligned} \tag{7}$$

where $\alpha_i (1 \le i \le 62)$ is a constant, $\beta_j (1 \le j \le 2)$ is an arbitrary value.

Then, the ciphertext pairs satisfying the following differential forms are selected after 7 rounds of encryption:

$$\begin{aligned} \Delta C_{X_0}^7 &= (\tau_1\tau_2 0\tau_3, 0\tau_4 00, 000\tau_5, \tau_6 0\tau_7 0);\\ \Delta C_{X_1}^7 &= (00\delta_1 0, \delta_2 100, \delta_3\delta_4 01, 00\delta_5\delta_6);\\ \Delta C_{X_2}^7 &= (0\zeta_1\zeta_2\zeta_3, \zeta_4\zeta_5\zeta_6\zeta_7, 00\zeta_8\zeta_9, \zeta_{10}\zeta_{11}10);\\ \Delta C_{X_2}^7 &= (\eta_1\eta_2 11, 0\eta_3 0\eta_4, 00\eta_5\eta_6, 00\eta_7\eta_8). \end{aligned} \tag{8}$$

It can be seen from formula (8) that the plaintext structure contains 2^{62} plaintexts, forming a total of 2^{123} plaintext pairs. If 2^N structures are selected, there are 2^{62} plaintexts and 2^{N+62} data pairs. Finally, the data complexity of the attack is $2^N \times 2^{62} = 2^{N+62}$ selected plaintext.

5.4 Biclique analysis

Biclique analysis [39] is a variant of the meet-in-the-middle (MITM) attacks, which is widely used in cryptanalysis for key recovery attacks based on a two-party matrix. Reducing the complexity of MITM attacks by constructing biclique structures is the basic idea behind biclique analysis.

$$K\begin{cases} K[0] &= k_0||k_1||k_2||k_3||k_4||k_5||k_6||k_7 \\ K[1] &= k_8||k_9||k_{10}||k_{11}||k_{12}||k_{13}||k_{14}||k_{15} \\ & \vdots \\ K[10] &= k_{80}||k_{81}||k_{82}||k_{83}||k_{84}||k_{85}||k_{86}||k_{87} \\ K[11] &= k_{88}||k_{89}||k_{90}||k_{91}||k_{92}||k_{93}||k_{94}||k_{95} \end{cases} \tag{9}$$

The initial key of DABC is divided into the form shown in formula (9). **Table 10** shows the initial keys involved in each round of round keys. The 4-round Δ_i differential path is obtained by backward encrypting 4 rounds using the 18th round key $K[6]$. The 4-round ∇_j differential path is obtained by forward decrypting 4 rounds using the 19th round key $K[1]$. The active bytes in the Δ_i differential path are inactive bytes in the ∇_jdifferential path and vice versa (active bytes refer to bytes with a difference other than 0, and inactive bytes refer to bytes with a difference of 0). The two differential paths are combined to form an 8-round biclique structure from rounds 15 to 22. Formula (10) shows parting the key. It is represented by a 4×2 matrix to simplify the key space, where $K[\bullet]$ represents an 8-bit block of data.

$$\begin{bmatrix} K[1], K[2] & K[3], K[4] \\ K[5], \boldsymbol{K[6]} & K[7], K[8] \\ K[10], K[11] & K[0], \boldsymbol{K[1]} \\ K[2], K[3] & K[4], K[5] \end{bmatrix} \tag{10}$$

Table 10. The initial keys involved in each round of round keys.

r	The value of [·]	*r*	The value of [·]
0	{8,9},{10,11}	14	{11,0},{1,2}
1	{1,2},{3,4}	15	{4,5},{6,7}
2	{5,6},{7,8}	16	{8,9},{10,11}
3	{10,11},{0,1}	17	{1,2},{3,4}
4	{2,3},{4,5}	18	{5,6},{7,8}
5	{7,8},{9,10}	19	{10,11},{0,1}
6	{11,0},{1,2}	20	{2,3},{4,5}
7	{4,5},{6,7}	21	{7,8},{9,10}
8	{8,9},{10,11}	22	{11,0},{1,2}
9	{1,2},{3,4}	23	{4,5},{6,7}
10	{5,6},{7,8}	24	{8,9},{10,11}
11	{10,11},{0,1}	25	{1,2},{3,4}
12	{2,3},{4,5}	26	{5,6},{7,8}
13	{7,8},{9,10}		

The 8-round biclique structure consists of 2^d ciphertexts $C_n(0 \le n \le 2^d)$, 2^d intermediate states $S_n(0 \le n \le 2^d)$, and keys $K(K_{(i,j)})$. The difference Δ_i and the difference ∇_j are derived from Δ_i^K and ∇_j^K respectively. In this paper, the difference

corresponding to $K[6]$ is i and the difference corresponding to $K[1]$ is j. The remaining 80-bit will iteratively take all possible values.

The non-linear component in this paper refers to the AND operation and the NAND operation, but these differential components do not share active non-linear components. This paper uses the formula (11) to express the difference relation between $(S_0, \Delta_i^K, \nabla_j^K, C_0)$.

$$S_0 \oplus \Delta_i^K \xrightarrow[f]{K_{[0,0]} \oplus \Delta_i^K \oplus \nabla_j^K} C_0 \oplus \nabla_j^K \tag{11}$$

This paper has successfully constructed an 8-dimensional double factorial structure for DABC. Corresponding plaintext pair P_i is obtained when the ciphertext pair C_n is decrypted with the key K_{hide}. Formula (12) holds when $K_{[i,j]} = K_{hide}$. Meanwhile, $\vec{v}$ and $\overleftarrow{v}$ are called matches, which means $K_{[i,j]}$ is a candidate key for K_{hide}. Then the full round of DABC keys is recovered.

$$P_i \xrightarrow[\varepsilon_1]{K_{[i,j]}} \vec{v} \stackrel{?}{=} \overleftarrow{v} \xleftarrow[\varepsilon_2]{K_{[i,j]}} S_j \tag{12}$$

5.4.1 Complexity of recovering the key

The time complexity is usually used to measure the effectiveness of a biclique attack. Formula (13) shows the time complexity calculation process of key recovery:

$$C_{full} = \quad 2^{K-2d}(C_{Biclique} + C_{precomp} + C_{recomp} + C_{falsepas}), \tag{13}$$

where C_{full} represents the time complexity of recovering the key. K is the bit size of the key. d is the dimension of the constructed biclique structure, where $d = 8$. $C_{Biclique}$ represents the complexity of constructing the biclique structure. It is at most 8-rounds of calculation after our analysis, $C_{Biclique} = 2^{d+1} \times (8/27) = 2^{7.25}$. $C_{precomp}$ represents the pre-calculated complexity of matching detection. Pre-calculating the remaining rounds of DABC, the obtained complexity result is $2^8 \times (19/27) \approx 2^{7.49}$. C_{recomp} represents the complexity of the recalculation process. Recalculation requires sixteen 16-bit AND operations and sixteen 16-bit NAND operations in our analysis. DABC requires a total of fifty-two 16-bit AND operations and fifty-four 16-bit NAND operations, $C_{recomp} = 2^{16} \times \frac{16 \times 16}{52 \times 54} \approx 2^{12.54}$. $C_{falsepas}$ refers to the complexity of matching v at the wrong position. The value is 2^8 when matching a single byte. The matching in formula (12) is performed on a single byte. Finally, $C_{full} = 2^{96-2\times8} \times (2^{7.25} + 2^{7.49} + 2^{12.54} + 2^8) \approx 2^{92.68}$.

5.5 Side-channel analysis

Side-channel analysis is a current attack against hardware implementations of cryptographic algorithms. It takes advantage of the correlation between the physical leakage generated by the encryption device at runtime and the intermediate state operations. The correlation is analyzed by mathematical-statistical methods. The key is recovered according to the results of the analysis. A variety of attack methods have been proposed, such as power analysis, time analysis, electromagnetic analysis, and so on. The defense against such attacks is generally by adding mask countermeasures to the algorithm. For example, Y. Ou and L. Li [40] resist side-channel analysis by adding the first-order mask to AES. The n-order mask can prevent n-order side-channel analysis. Its attack cost increases exponentially for each additional order of attack. Considering that lightweight block ciphers are generally applied in resource-constrained environments, they are generally protected by adding first-order masks. Since mask technology is mature, this paper will not discuss it too much here.

6. Conclusion

This paper proposes a new dynamic ARX-based lightweight block cipher to address the slow diffusion and low security of ARX-based lightweight block ciphers, called DABC. Firstly, DABC adopted a new variant of the GFN structure, which has high diffusion and can change all data in one round. It overcomes the weakness that the traditional ARX-based lightweight block cipher only changes half of the data in one round. Second, the generalized two-dimensional cat map and the key are combined to construct a dynamic linear permutation layer. The data is subjected to different permutation operations according to different values of the key in each round. Compared with a single linear permutation layer, it increases the randomness of the cipher structure and improves the poor security of the traditional ARX-based lightweight block cipher. In the round function part, NAND and AND operations are used as nonlinear components for odd and even rounds, respectively, which further increases the complexity of the attack. It also overcomes the weakness of poor diffusion when the initial plaintext and key are all 0 or all 1. This paper proposes a round-based architecture of DABC and conducted ASIC implementation and FPGA implementation. The results show that DABC consumes fewer hardware resources and has a high throughput, which is suitable for a resource-constrained environment. This paper performs avalanche tests on DABC, Piccolo, HIGHT, Simeck, SIMON, and SPECK by the same test methods. DABC has a good avalanche effect while satisfying the strict avalanche criterion after comparison. Finally, this paper conducts a security analysis of DABC. The analysis results show that DABC has good security and can resist differential analysis, linear analysis, impossible differential analysis, biclique analysis, and side-channel analysis.

Acknowledgement

This research is supported by the Hunan Provincial Natural Science Foundation of China(2022JJ30103, 2022JJ50016), the Science and Technology Innovation Program of Hunan Province (2016TP1020), Open fund project of Hunan Provincial Key Laboratory of Intelligent Information Processing and Application for Hengyang Normal University (2022HSKFJJ011), “the 14th Five-Year Plan” Key Disciplines and Application-oriented Special Disciplines of Hunan Province (Xiangjiaotong [2022] 351).

References

[1] J. Cheng, S. Guo, J. He, “An extended type-1 generalized feistel networks: Lightweight block cipher for iot,” *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11408-11421, 2022. Article (CrossRef Link).

[2] X. Li, M. H. Ibrahim, S. Kumari, et al., “Anonymous mutual authentication and key agreement scheme for wearable sensors in wireless body area networks,” *Computer Networks*, vol. 129, pp. 429-443, 2017. Article (CrossRef Link).

[3] H. Shin, H. K. Lee, H.-Y. Cha, et al., “Iot security issues and light weight block cipher,” in *Proc. of 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, IEEE, Okinawa, Japan, pp. 381-384, 2019. Article (CrossRef Link).

[4] K. Shibutani, T. Isobe, H. Hiwatari, et al., “Piccolo: an ultra-lightweight block cipher,” in *Proc. of International workshop on cryptographic hardware and embedded systems*, Springer, Berlin, Heidelberg, pp. 342-357, 2011. Article (CrossRef Link).

[5] T. Suzaki, K. Minematsu, S. Morioka, et al., "Twine: A lightweight block cipher for multiple platforms," in *Proc. of International Conference on Selected Areas in Cryptography*, Springer, Berlin, Heidelberg, pp. 339-354, 2012. Article (CrossRef Link).
[6] L. Li, B. Liu, H. Wang, "Qtl: a new ultra-lightweight block cipher," *Microprocessors and Microsystems*, vol. 45, pp. 45-55, 2016. Article (CrossRef Link).
[7] A. Bogdanov, L. R. Knudsen, G. Leander, et al., "Present: An ultra-lightweight block cipher," in *Proc. of International workshop on cryptographic hardware and embedded systems*, Springer, Berlin, Heidelberg, pp. 450-466, 2007. Article (CrossRef Link).
[8] S. Banik, A. Bogdanov, T. Isobe, et al., "Midori: A block cipher for low energy," in *Proc. of International Conference on the Theory and Application of Cryptology and Information Security*, Springer, Berlin, Heidelberg, pp. 411-436, 2015. Article (CrossRef Link).
[9] W. Zhang, Z. Bao, D. Lin, et al., "Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms," *Science China Information Sciences*, vol. 58, no. 12, pp. 1-15, 2015. Article (CrossRef Link).
[10] C. Beierle, J. Jean, S. K¨olbl, et al., "The skinny family of block ciphers and its low-latency variant mantis," in *Proc. of Annual International Cryptology Conference*, Springer, Berlin, Heidelberg, pp. 123-153, 2016. Article (CrossRef Link).
[11] D. Dinu, A. Biryukov, J. Großsch¨adl, et al., "Felics–fair evaluation of lightweight cryptographic systems," in *Proc. of NIST Workshop on Lightweight Cryptography*, Vol. 128, 2015. Article (CrossRef Link).
[12] D. Hong, J. Sung, S. Hong, et al., "Hight: A new block cipher suitable for low-resource device," in *Proc. of International workshop on cryptographic hardware and embedded systems*, Springer, Berlin, Heidelberg, pp. 46-59, 2006. Article (CrossRef Link).
[13] R. Beaulieu, D. Shors, J. Smith, et al., "The simon and speck lightweight block ciphers," in *Proc. of the 52nd annual design automation conference, Association for Computing Machiner*, New York, NY, USA, pp. 1-6, 2015. Article (CrossRef Link).
[14] G. Yang, B. Zhu, V. Suder, et al., "The simeck family of lightweight block ciphers," in *Proc. of International Workshop on Cryptographic Hardware and Embeddedm Systems*, Springer, Berlin, Heidelberg, pp. 307-329, 2015. Article (CrossRef Link).
[15] Y. Guo, L. Li, B. Liu, "Shadow: A lightweight block cipher for iot nodes," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 13014-13023, 2021. Article (CrossRef Link).
[16] S. Chen, Y. Fan, L. Sun, et al., "Sand: an and-rx feistel lightweight block cipher supporting s-box-based security evaluations," *Designs, Codes and Cryptography*, vol. 90, no. 1, pp. 155-198, 2022. Article (CrossRef Link).
[17] B. Koo, D. Hong, D. Kwon, "Related-key attack on the full hight," in *Proc. of International Conference on Information Security and Cryptology*, Springer, Berlin, Heidelberg, pp. 49-67, 2010. Article (CrossRef Link).
[18] J. Song, K. Lee, H. Lee, "Biclique cryptanalysis on lightweight block cipher: Hight and piccolo," *International Journal of Computer Mathematics*, vol. 90, no. 12, 2013, pp. 2564-2580, 2013. Article (CrossRef Link).
[19] B. Ryabko, A. Soskov, "The distinguishing attack on speck, simon, simeck, hight and lea," *IACR Cryptol. ePrint Arch*, vol. 47, pp. 1–9, 2018. Article (CrossRef Link).
[20] J. Lu, G. Liu, Y. Liu, et al., "Improved neural distinguishers with (related-key) differentials: Applications in simon and simeck," *Cryptography and Security*, 2022. Article (CrossRef Link).
[21] K. Fu, M. Wang, Y. Guo, et al., "Milp-based automatic search algorithms for differential and linear trails for speck," in *Proc. of International Conference on Fast Software Encryption*, Springer, Berlin, Heidelberg, pp. 268-288, 2016. Article (CrossRef Link).
[22] J. Yang, L. Li, Y. Guo, et al., "Dulbc: A dynamic ultra-lightweight block cipher with high-throughput," *Integration*, vol. 87, pp. 221-230, 2022. Article (CrossRef Link).
[23] A. D. Dwivedi, "Brisk: Dynamic encryption based cipher for long term security," *Sensors*, vol. 21, no. 17, pp. 5744, 2021. Article (CrossRef Link).

[24] J. Borghoff, A. Canteaut, T. G¨uneysu, et al., "Prince–a low-latency block cipher for pervasive computing applications," in *Proc. of International conference on the theory and application of cryptology and information security*, Springer, Berlin, Heidelberg, pp. 208-225, 2012. Article (CrossRef Link).
[25] T. P. Berger, J. Francq, M. Minier, et al., "Extended generalized feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput," *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 2074-2089, 2016. Article (CrossRef Link).
[26] M. R. Zaba, N. Jamil, M. E. Rusli, et al., "I-present tm: An involutive lightweight block cipher," *Journal of Information Security*, vol. 5, no, 3, pp. 114-122, 2014. Article (CrossRef Link).
[27] R. Beaulieu, D. Shors, J. Smith, et al., "Simon and speck: Block ciphers for the internet of things," *Cryptology ePrint Archive*, 2015. Article (CrossRef Link).
[28] D. Lee, D.-C. Kim, D. Kwon, et al., "Efficient hardware implementation of the lightweight block encryption algorithm lea," *Sensors*, vol. 14, no. 1, pp. 975-994, 2014. Article (CrossRef Link).
[29] Z. Mishra, G. Ramu, B. Acharya, "Hight speed low area vlsi architecture for lea encryption algorithm," in *Proc. of the third international conference on microelectronics, computing and communication systems*, Springer, Singapore, pp. 155-160, 2019. Article (CrossRef Link).
[30] Z. Mishra, P. K. Nath, B. Acharya, "High throughput unified architecture of lea algorithm for image encryption," *Microprocessors and Microsystems*, vol. 78, pp. 103214, 2020. Article (CrossRef Link).
[31] C. A. Lara-Nino, A. Diaz-Perez, M. Morales-Sandoval, "Lightweight hardware architectures for the present cipher in fpga," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2544-2555, 2017. Article (CrossRef Link).
[32] N. Hanley, M. ONeill, "Hardware comparison of the iso/iec 29192-2 block ciphers," in *Proc. of 2012 IEEE computer society annual symposium on VLSI*, IEEE, Amherst, MA, USA, pp. 57-62, 2012. Article (CrossRef Link).
[33] L. Li, J. Feng, B. Liu, et al., "Implementation of prince with resource-efficient structures based on fpgas," *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 11, pp. 1505-1516, 2021. Article (CrossRef Link).
[34] N. Shrivastava, B. Acharya, A. S. Raghuvanshi, "Vlsi implementation of esf and qtl lightweight ciphers," in *Proc. of the Fourth International Conference on Microelectronics, Computing and Communication Systems*, Springer, Singapore, pp. 513-525, 2021. Article (CrossRef Link).
[35] Z. Mishra, S. Mishra, B. Acharya, "Lea: 128 high frequency architecture with image analysis," in *Proc. of 2020 International Conference for Emerging Technology (INCET)*, IEEE, Belgaum, India, pp. 1-5, 2020. Article (CrossRef Link).
[36] E. Biham, A. Shamir, "Differential cryptanalysis of des-like cryptosystems," *Journal of CRYPTOLOGY*, vol. 4, no. 1, pp. 3-72, 1991. Article (CrossRef Link).
[37] M. Matsui, "Linear cryptanalysis method for des cipher," in *Proc. of Workshop on the Theory and Application of Cryptographic Techniques*, Springer, Berlin, Heidelberg, pp. 386-397, 1993. Article (CrossRef Link).
[38] D. Yang, W.-F. Qi, H.-J. Chen, "Impossible differential attacks on the skinny family of block ciphers," *IET Information Security*, vol. 11, no. 6, pp. 377-385, 2017. Article (CrossRef Link).
[39] Z. Ahmadian, M. Salmasizadeh, M. R. Aref, "Biclique cryptanalysis of the full-round klein block cipher," *IET Information Security*, vol. 9, no. 5, pp. 294-301, 2015. Article (CrossRef Link).
[40] Y. Ou, L. Li, "Research on a high-order aes mask anti-power attack," *IET Information Security*, vol. 14, no. 5, pp. 580-586, 2020. Article (CrossRef Link).

Wen Chen received the B.Eng. degree from Haibin College of Beijing Jiaotong University, Huanghua, China, in 2020 and he is currently working toward a Master's degree in Hengyang Normal University, Hengyang, China. Since 2020, his current research interests include embedded systems and information security.

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

Ying Guo received her master's and B.S. degree from Hengyang Normal University, Hengyang, China, in 2022 and 2019, respectively. She is currently pursuing a Ph.D. degree at the School of Computer and Information Security, Guilin University Of Electronic Technology, Guilin, China. Her current research interests include embedded systems and information security.