Shadow: A Lightweight Block Cipher for IoT Nodes

Ying Guo, Lang Li, and Botao Liu

***Abstract*—The advancement of the Internet of Things (IoT) has promoted the rapid development of low-power and multifunctional sensors. However, it is seriously significant to ensure the security of data transmission of these nodes. Meanwhile, sensor nodes have the characteristics of converting analog signals into digital signals for operation processing in wireless sensor networks (WSNs). Given the particularity of Addition or AND, Rotation, and XOR (ARX) operations, its round function can only be based on the Feistel structure or generalized Feistel structure, otherwise, the process of decryption cannot be completed correctly. Furthermore, the existing ARX ciphers have the problems of only changing half of the plaintext block in one round and iterating for many rounds. In this article, a new logical combination method of generalized Feistel structure and ARX operations is proposed to improve the diffusion speed of ARX ciphers, called Shadow. Shadow overcomes the shortcomings of traditional ARX ciphers that only diffuse half of the block in one round. To ensure the efficiency of the encryption hardware circuit while ensuring the security of the physical-layer signal, we studied the round-based hardware architecture and the serial hardware architecture for Shadow cipher. Particularly, we conducted a series of performance tests on Shadow, including the avalanche effect, FPGA implementation, and ASIC implementation. Also, we conducted a security analysis of the Shadow. As shown by our experiments and comparisons, Shadow is compact in IoT nodes and is of high security against cryptanalysis.**

***Index Terms*—Addition or AND, Rotation, and XOR (ARX), generalized Feistel structure, Internet of Things (IoT) nodes, lightweight block cipher.**

I. Introduction

INFORMATION interaction based on identification technology, pervasive computing, and fog/edge computing [1], [2] is the core of the Internet of Things (IoT), and IoT nodes are the medium of information interaction. Generally, data collected by users through IoT nodes may be highly confidential. For example, Dias and Cunha [3] monitored human health indicators through wearable vital signs sensing technology. The lack of security and privacy protection measures will have a significant impact on the sustainable development of IoT technology. Therefore, there is an urgent need to integrate more security functions into hardware and software devices to enhance the security of private data transmission, processing, and storage. How to protect the privacy data obtained by IoT nodes has caused extensive research by scholars. Xiong *et al.* [4] proposed a privacy and availability data clustering (PADC) scheme based on the *k*-means algorithm and differential privacy, which is superior to other schemes in protecting power privacy data. Zhang *et al.* [5] proposed a lightweight and verifiable privacy-preserving data aggregation (PPDA) scheme, named LVPDA, for the edge-computing-enabled IoT system, where the Paillier homomorphic encryption method and an online/offline signature technique are combined to ensure the privacy preserving and integrity verification during the data aggregation process.

With the development of IoT technology, cloud storage of node data has gradually become a popular method. However, most privacy data protection methods are not designed for the IoT perception layer. At the same time, the IoT nodes must have the protection ability to sense data. IoT nodes generally consist of sensors, microprocessors, wireless communications, and power supply modules. Xie *et al.* [6] investigated and studied as many as 11 mainstream attacks that WSN may be subject. Additionally, Dalmasso *et al.* [7] pointed out that the use of low-resource, high-efficiency, and lightweight encryption technology to encrypt and protect any extremely restricted devices is an inevitable requirement. In recent years, many experts and scholars in the world have focused on the design of lightweight block algorithms, and many well-designed lightweight block cipher algorithms have emerged, such as Loong [8], QTL [9], SFN [10], PRESENT [11], LBlock [12], GIFT [13], Midori [14], and RECTANGLE [15]. Traditional lightweight block ciphers mostly use S-box as nonlinear components. At the same time, there are a few block ciphers that use addition operations as nonlinear components. In 2015, researchers at the University of Luxembourg developed an open-source framework "fair evaluation of lightweight cryptographic system (FELICS) [16]." The platform determines the best algorithm by testing different state-of-the-art lightweight block ciphers on IoT devices. According to the results given by FELICS, the Addition or AND, Rotation, and XOR (ARX)-based block cipher implementation is superior to other S-box-based ciphers. The cryptographic algorithm based on ARX is to realize the nonlinearity, obfuscation, and diffusivity of the algorithm through the combination of Addition,

Manuscript received October 29, 2020; revised December 9, 2020 and January 7, 2021; accepted February 28, 2021. Date of publication March 8, 2021; date of current version August 6, 2021. This work was supported in part by the Postgraduate Scientific Research Innovation Project of Hunan Province under Grant CX20190980; in part by the Scientific Research Fund of Hunan Provincial Education Department under Grant 19A072; in part by the Science and Technology Innovation Program of Hunan Province under Grant 2016TP1020; and in part by the Application-Oriented Special Disciplines, Double First-Class University Project of Hunan Province under Grant Xiangjiaotong [2018] 469. *(Corresponding author: Lang Li.)*

The authors are with the Hunan Provincial Key Laboratory of Intelligent Information Processing and Application and the College of Computer Science and Technology, Hengyang Normal University, Hengyang 421002, China (e-mail: 1352520550@qq.com; lilang911@126.com; 2946671234@qq.com).

Digital Object Identifier 10.1109/JIOT.2021.3064203

Rotation, and XOR. The HIGHT [17] block cipher based on ARX operations proposed in 2006 adopts an 8-branch generalized Feistel structure, which mainly involves three operating modes of Addition, Rotation, and XOR. In 2013, the SIMON and SPECK [18] have greatly improved the block ciphers based on ARX operations, and they are specifically developed for hardware and software platforms. The AND operation proposed in the algorithm replaces the traditional addition operation to improve hardware efficiency. In order to improve the applicability of the LEA [19] algorithm on 8-b and 16-b microcontrollers, the CHAM [20] algorithm was proposed.

By comparing and analyzing these classic block ciphers based on ARX operations, we can find that there are two common points. On the one hand, the overall encryption structure can only be based on the Feistel structure or the generalized Feistel structure, otherwise, the decryption work cannot be completed. On the other hand, under the same conditions, their iteration rounds are generally more than the block ciphers with the SPN structure. The reason is that the Feistel structure or generalized Feistel structure only diffuses half of the plaintext block in one round. Some algorithm designers gain a certain diffusion effect by increasing the number of iterations. However, the power consumption of devices with extremely limited resources is very sensitive to the number of iterations of the encryption round. In this article, our research focuses on the study of a new logical combination method of the generalized Feistel structure and ARX operations to improve the diffusion speed of the cipher based on ARX operations. The main contributions of this article are as follows.

1) To improve the diffusion effect of ciphers, we proposed a new logical combination method of the generalized Feistel structure and ARX operations. This combination method overcomes the shortcomings of traditional ARX ciphers, which is that the diffusion rate is slow, and half of the block has not changed in one round. The specific combination method is described in Algorithm 1. All branches perform Rotation, AND, and XOR. After a round, the effect of changing the whole block is achieved.
2) To optimize the operational efficiency of the encryption hardware circuit while ensuring the security of the physical-layer signal, we proposed a round-based hardware implementation architecture and a serial hardware implementation architecture for Shadow. The serial architecture consumes fewer resources than the round-based architecture. However, the round-based architecture has higher throughput and less energy consumption. Also, we conducted FPGA implementation and ASIC implementation on Shadow. The results showed that it is suitable for low-resource and low-power IoT nodes.
3) We introduced an indicator, called uncertainty, to characterize the range of fluctuations in the average number of bits of change in the avalanche effect test. The smaller the value range, the stronger the algorithm stability. Through the avalanche effect experiment, Shadow was compared with the current classic ARX ciphers HIGHT, SIMON, Simeck [21], and CHAM. It can be seen from experiments that the diffusion and stability of the Shadow is better than the SIMON, Simeck, and CHAM. Although the uncertainty value of HIGHT is slightly lower than that of Shadow, and the average number of bits slightly higher than Shadow, due to the particularity of addition mod 2^8 operation, the HIGHT cipher consumes a lot of resources in hardware implementation.
4) Security analysis indicates that Shadow can resist the impossible differential attack and the biclique cryptanalysis. The result of performance evaluation shows that the proposed logical combination method is both effective and efficient.

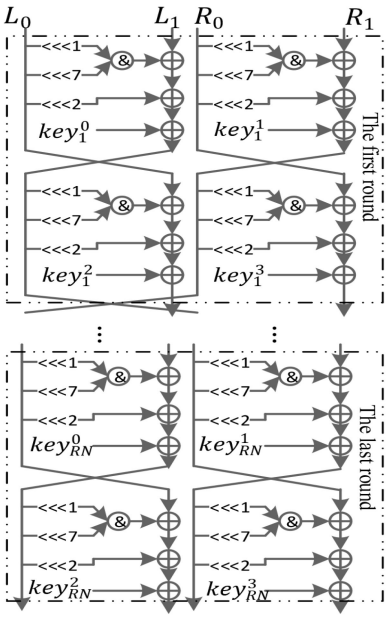


Fig. 1. Round function of Shadow.

The remainder of this article is organized as follows. We introduce the specific of Shadow in Section II and provide an elaborate description of hardware performance evaluation and avalanche effect in Section III. Our security analysis of Shadow is presented in Section IV. We conclude this article in Section V.

II. SPECIFICATION OF SHADOW

Shadow is based on a generalized Feistel structure with 4-branch and has 32-b block with 64-b key and 64-b block with 128-b key. Round number (RN) of 16 and 32 rounds, respectively. According to the size of the two blocks, algorithms are denoted as Shadow-32 and Shadow-64.

A. Encryption Algorithm

Because of the special structure of Shadow designed in this article, the decryption process is consistent with the encryption process, and the round keys are used in the reverse order. Shadow mainly involves three operations: 1) AND; 2) Rotation; and 3) XOR. The round function of Shadow is shown in Fig. 1.

In the process of Shadow encryption, the plaintext block is divided into several blocks of the same size. With the help of the primary key, the ciphertext is generated after multiple iterations. Correspondingly, as described in Algorithm 1.

Algorithm 1 Shadow Encryption Routine

Input: Plaintext, key;
Output: Ciphertext;
1: $(L_0, L_1, R_0, R_1) \leftarrow$ Plaintext;
2: for $r = 1$ to RN do
3: $state0 = (L_{0(\lll 1)} \& L_{0(\lll 7)}) \oplus L_1 \oplus L_{0(\lll 2)} \oplus key_r^0$;
4: $state1 = (R_{0(\lll 1)} \& R_{0(\lll 7)}) \oplus R_1 \oplus R_{0(\lll 2)} \oplus key_r^1$;
5: $L_0' = state1$;
6: $L_1' = (state_{0(\lll 1)} \& state_{0(\lll 7)}) \oplus L_0 \oplus state_{0(\lll 2)} \oplus key_r^2$;
7: $R_0' = state0$;
8: $R_1' = (state_{1(\lll 1)} \& state_{1(\lll 7)}) \oplus R_0 \oplus state_{1(\lll 2)} \oplus key_r^3$;
9: end for
10: Ciphertext $\leftarrow (L_0', L_1', R_0', R_1')$;
11: **Return** Ciphertext;

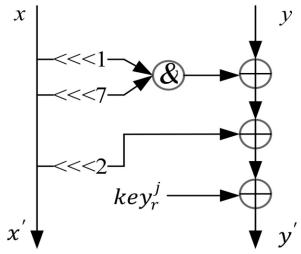


Fig. 2. Round function of 2-branch.

From Fig. 1, we can see that the round function of Shadow can be simplified into a 2-branch structure. In this case, it is called four times during each round of encryption to reduce logic hardware resource consumption, as shown in Fig. 2.

The round function of 2-branch with respect to ARX is satisfied

$$\begin{cases} \{0,1\}^{n/4} \times \{0,1\}^{n/4} \rightarrow \{0,1\}^{n/4} \\ x' = x \\ y' = (x_{\lll 1} \text{and} x_{\lll 7}) \bigoplus x_{\lll 2} \bigoplus \text{key}_r^j \end{cases} \tag{1}$$

where $1 \le r \le$ RN, $0 \le j \le 3$ and $n = 32$ or $n = 64$, RN $= 32$ or RN $= 64$.

B. Subkey Generator

According to the size of the plaintext block, the Shadow cipher subkey generator has two types: 1) generator1 and 2) generator2. The primary key K is described as $k_0 \| k_1 \| k_2 \cdots \| k_{62} \| k_{63}$ or $k_0 \| k_1 \| k_2 \cdots \| k_{126} \| k_{127}$. The round constant r is described as $c_0 \| c_1 \| c_2 \| c_3 \| c_4$ or $c_0 \| c_1 \| c_2 \| c_3 \| c_4 \| c_5$. When the plaintext block size is 32-b, the 64-b primary key K enters generator1. In generator1, 5-b $k_3 \| k_4 \| k_5 \| k_6 \| k_7$ executes AddRoundconstant operation, 8-b $k_{56} \| k_{57} \| \cdots \| k_{62} \| k_{63}$ executes NX operation, and after the above operations, the 64-b key performs Permutation. Then, the first round subkey K' generated. The upper 32-b of K' is divided into four parts to participate in the round key addition operation. At the same time, the K' continues the above process, providing different subkeys for each round of function operation until the end of encryption. The specific operation process is shown in Fig. 3.

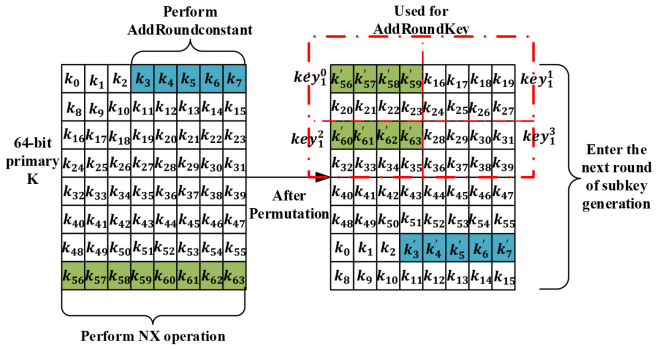


Fig. 3. Operation process of generator1.

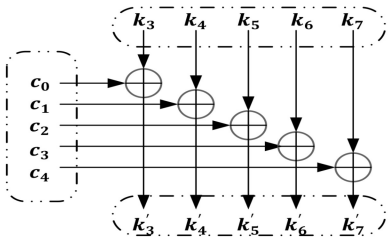


Fig. 4. Process of the AddRoundconstant module.

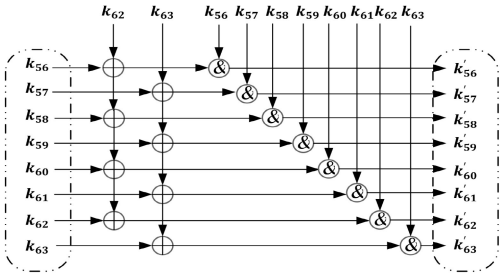


Fig. 5. Process of the NX module.

AddRoundconstant: As shown in Fig. 4, the 5-b $k_3 \| k_4 \| k_5 \| k_6 \| k_7$ performs XOR operation with the 5-b round constant $c_0 \| c_1 \| c_2 \| c_3 \| c_4$.

NX Module: The NX module mainly plays a nonlinear role in the subkey generator. For Shadow-32, the lower 8-b $k_{56} \| k_{57} \| k_{58} \cdots \| k_{62} \| k_{63}$ fulfills the NX module. The specific operation process is shown in Fig. 5.

The principle of the NX module can be expressed by

$$\begin{cases} k_{56}' = k_{56} \& \left(k_{56} \bigoplus k_{62}\right) \\ k_{57}' = k_{57} \& \left(k_{57} \bigoplus k_{63}\right) \\ k_{58}' = k_{58} \& \left(k_{58} \bigoplus k_{56} \bigoplus k_{62}\right) \\ k_{59}' = k_{59} \& \left(k_{59} \bigoplus k_{57} \bigoplus k_{63}\right) \\ k_{60}' = k_{60} \& \left(k_{60} \bigoplus k_{58} \bigoplus k_{56} \bigoplus k_{62}\right) \\ k_{61}' = k_{61} \& \left(k_{61} \bigoplus k_{59} \bigoplus k_{57} \bigoplus k_{63}\right) \\ k_{62}' = k_{62} \& \left(k_{62} \bigoplus k_{60} \bigoplus k_{58} \bigoplus k_{56} \bigoplus k_{62}\right) \\ k_{63}' = k_{63} \& \left(k_{63} \bigoplus k_{61} \bigoplus k_{59} \bigoplus k_{57} \bigoplus k_{63}\right). \end{cases} \tag{2}$$

Permutation: For Shadow-32, the 64-b key that has executed the AddRoundconstant module and NX module performs Permutation. The process of Permutation is as shown in Fig. 6. As shown in Table I, k_i represents the position order before Permutation, and k_i' represents the position order after Permutation.

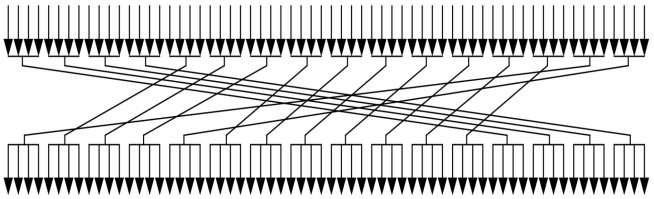


Fig. 6. Process of Permutation.

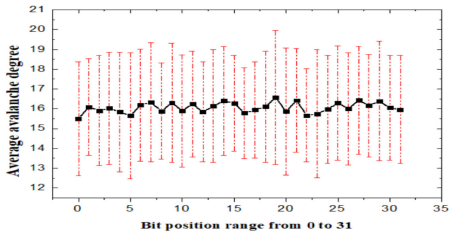


Fig. 7. Average avalanche degree of Shadow-32.

When the plaintext block size is 64-b, the 128-b primary key K enters generator2. In generator2, 6-b $k_2\|k_3\|k_4\|k_5\|k_6\|k_7$ and 6-b $c_0\|c_1\|c_2\|c_3\|c_4\|c_5$ perform AddRoundconstant operation. The lower 24-b $k_{104}\|k_{105}\|k_{106}\|\cdots\|k_{125}\|k_{126}\|k_{127}$ executes NX operation as shown in (3). After the above operations, the 64-b key performs Permutation. Then, the first round subkey K' generated. The upper 64-b of K' is divided into four parts to participate in the round key addition operation. At the same time, the K' continues the above process, providing different subkeys for each round of function operation until the end of encryption. Besides, the principles of the AddRoundconstant module, NX module, and Permutation module in generator2 are the same as those in generator1, but the number of bits is different. The arrangement position of Permutation is as shown in Table II

$$\begin{cases} k'_{104} = k_{104}\&\left(k_{104} \bigoplus k_{126}\right) \\ k'_{105} = k_{105}\&\left(k_{105} \bigoplus k_{127}\right) \\ k'_{106} = k_{106}\&\left(k_{106} \bigoplus k_{104} \bigoplus k_{126}\right) \\ k'_{107} = k_{107}\&\left(k_{107} \bigoplus k_{105} \bigoplus k_{127}\right) \\ k'_{108} = k_{108}\&\left(k_{108} \bigoplus k_{106} \bigoplus k_{104} \bigoplus k_{126}\right) \\ k'_{109} = k_{109}\&\left(k_{109} \bigoplus k_{107} \bigoplus k_{105} \bigoplus k_{127}\right) \\ k'_{110} = k_{110}\&\left(k_{110} \bigoplus k_{108} \bigoplus k_{106} \bigoplus k_{104} \bigoplus k_{126}\right) \\ \quad \vdots \\ k'_{127} = k_{127}\&\left(k_{127} \bigoplus k_{125} \bigoplus k_{123} \bigoplus \cdots \bigoplus k_{105} \bigoplus k_{127}\right). \end{cases} \tag{3}$$

C. Decryption Algorithm

IoT nodes have a built-in encryption circuit to transmit the sensed data in the form of ciphertext. Adopt the client decryption method to solve the security of data collection, transmission, and storage in the simplest "end-to-end" mode. Because of the special structure of the Shadow encryption algorithm designed in this article, the decryption process is consistent with the encryption process, and the round keys are used in reverse.

TABLE I
VALUES OF PERMUTATION OF SHADOW-32

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

III. PERFORMANCE EVALUATION

A. Avalanche Effect

For researchers, one of the most important design goals is to construct a cipher with a good avalanche effect. The traditional avalanche effect test method is to calculate the avalanche degrees of all bits and add the average value as the avalanche degree of the entire algorithm. In this article, we not only calculate the average avalanche of each bit but also calculate the uncertainty to reflect the range of data fluctuations. The larger the range of data fluctuations and the greater the uncertainty, the worse the diffusion stability of the algorithm. Conversely, the smaller the range of data fluctuations and the smaller the uncertainty, the better the diffusion stability of the algorithm. In mathematical statistics, the variance is generally used to express uncertainty. We measured 100 sets of data for each location in the plaintext and then calculated the average value, furthermore, calculated the uncertainty.

Take Shadow-32 as an example to illustrate the avalanche effect test steps. A set of 64-b keys was fixed randomly, then a set of 32-b plaintext was selected randomly. The key and plaintext are input into the algorithm together. After encryption, the initial ciphertext C is obtained. Next, the first bit of the plaintext is reversed, the remaining bits remain unchanged and input the algorithm together with the key. After encryption, ciphertext C_0 is obtained. The Hamming weight of $C \bigoplus C_0$ represents the avalanche degree of the first bit of the plaintext. The remaining 31-b plaintext avalanche degree test method is the same as the first bit. On this basis, 99 groups of 32-b plaintext are randomly selected to repeat the above calculation, and a total of 100 groups of 32-D avalanche data are obtained. Accumulate the average and variance as the average avalanche degree and uncertainty of each bit of Shadow-32, as shown in Fig. 7. The horizontal axis represents the Shadow plaintext from high to low, and the vertical axis represents the average avalanche degree. The avalanche effect of Shadow-64 is shown in Fig. 8. Similarly, use the same method to test the avalanche effects of CHAM, HIGH, SIMON, and Simeck, as shown in Figs. 9–12.

TABLE II
VALUES OF PERMUTATION OF SHADOW-64

k_i	$k_i^{'}$	k_i	$k_i^{'}$	k_i	$k_i^{'}$	k_i	$k_i^{'}$	k_i	$k_i^{'}$	k_i	$k_i^{'}$	k_i	$k_i^{'}$	k_i	$k_i^{'}$
0	104	16	108	32	112	48	116	64	120	80	124	96	0	112	16
1	105	17	109	33	113	49	117	65	121	81	125	97	1	113	17
2	106	18	110	34	114	50	118	66	122	82	126	98	2	114	18
3	107	19	111	35	115	51	119	67	123	83	127	99	3	115	19
4	32	20	44	36	48	52	60	68	80	84	92	100	4	116	20
5	33	21	45	37	49	53	61	69	81	85	93	101	5	117	21
6	34	22	46	38	50	54	62	70	82	86	94	102	6	118	22
7	35	23	47	39	51	55	63	71	83	87	95	103	7	119	23
8	36	24	48	40	52	56	64	72	84	88	96	104	8	120	24
9	37	25	49	41	53	57	65	73	85	89	97	105	9	121	25
10	38	26	50	42	54	58	66	74	86	90	98	106	10	122	26
11	39	27	51	43	55	59	67	75	87	91	99	107	11	123	27
12	40	28	52	44	56	60	68	76	88	92	100	108	12	124	28
13	41	29	53	45	57	61	69	77	89	93	101	109	13	125	29
14	42	30	54	46	58	62	70	78	90	94	102	110	14	126	30
15	43	31	55	47	59	63	71	79	91	95	103	111	15	127	31

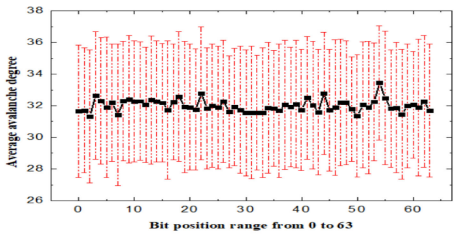


Fig. 8. Average avalanche degree of Shadow-64.

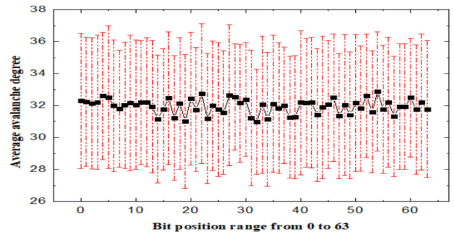


Fig. 9. Average avalanche degree of CHAM.

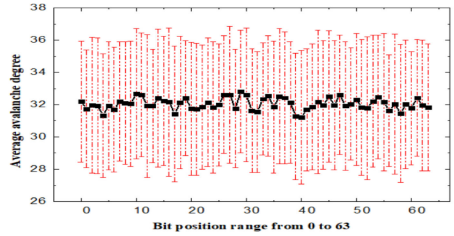


Fig. 10. Average avalanche degree of HIGHT.

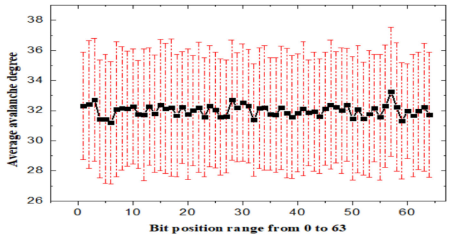


Fig. 11. Average avalanche degree of SIMON.

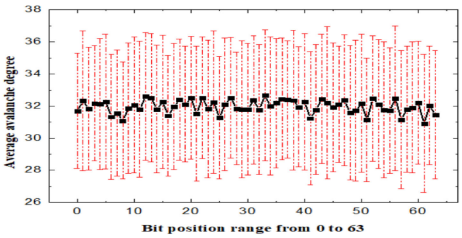


Fig. 12. Average avalanche degree of Simeck.

TABLE III
COMPARISON OF AVALANCHE EFFECT DATA OF VARIOUS CIPHERS

Algorithm	Block size	Key size	average avalanche	Average uncertainty
Shadow	32	64	16.05781	2.778694
Shadow	64	128	31.99828	3.955348
CHAM	64	128	31.94531	4.02618
HIGHT	64	128	32.04291	3.95454
Simeck	64	128	31.96750	4.025697
SIMON	64	128	31.99781	3.987345

In the test of the avalanche effect of the above ciphers, we used an error bar to indicate the uncertainty of the measured data. The error bar is a line segment drawn in the direction indicating the magnitude of the measured value with the measured arithmetic mean as the midpoint, and half of the line segment length is equal to the uncertainty.

As shown in Table III, the average change bit of the CHAM, Simeck, and SIMON after bit inversion is all less than 32, and the uncertainty is all greater than 4. However, the HIGHT is greater than half, and the uncertainty is less than 4. In this case, we can think that the avalanche effect of the HIGHT is better than the CHAM, Simeck, and SIMON. The Shadow-32 cipher designed in this article is greater than half the size of the plaintext block, and the uncertainty reaches 2.778694. At the same time, the average change bits and uncertainty of Shadow-64 are very close to HIGHT. Thus, Shadow can be considered to have a good avalanche effect.

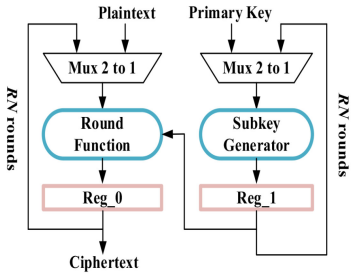


Fig. 13. Round-based architecture for Shadow.

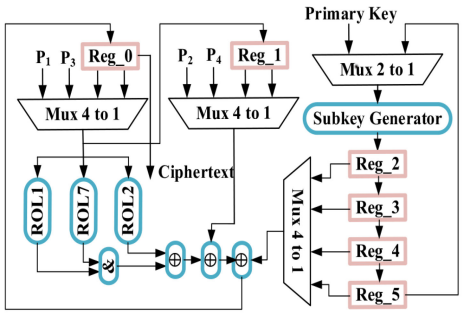


Fig. 14. Serial architecture for Shadow.

TABLE IV
COMPARISON OF ROUND-BASED ARCHITECTURE AND SERIAL ARCHITECTURE

Structure	Block size	Slice	LUT	FlipFlop	Energy (μJ/bit)	Throughput (Mbps)
Round-based	32	102	111	79	0.61	915.118
Serial	32	85	93	70	2.43	231.192
Round-based	64	199	227	156	0.76	743.038
Serial	64	182	209	148	2.48	226.809

TABLE V
COMPARISON BETWEEN SHADOW AND OTHER CIPHERS

Algorithm	Slice	LUT	FlipFlop	Throughput (Mbps)	Device	Ref.
LEA128/128	392	249	-	205.450	Virtex5	[19]
LEA128/128	122	360	382	-	Virtex5	[22]
LEA128/128	272	735	832	1206.300	Virtex5	[23]
^{R}PRESENT64/128	88	283	200	316.120	Virtex5	[24]
^{S}PRESENT64/128	72	237	203	61.190	Virtex5	[24]
PRESENT64/128	73	239	201	203.190	Virtex5	[25]
Shadow64/128	199	227	156	743.038	Virtex5	*Tw
Shadow64/128	182	209	148	226.809	Virtex5	*Tw

* : This work. ^{S} : Serial architecture. ^{R} : Round-based architecture.

B. Hardware Implementation

In the case of sufficient hardware resources, we can choose speed optimization as the target, while resource-limited devices such as sensor nodes generally choose area optimization as the comprehensive target. In particular, not only the area limitation is considered but the hardware implementation must also consider issues, such as energy and throughput.

First, we proposed a round-based hardware architecture for Shadow, as is shown in Fig. 13. One round of Shadow encryption is 1 clock cycle. Due to the particularity of the generalized Feistel structure, the decryption structure is the same as the encryption structure, and there is no need to consume excessive resources in hardware implementation. Second, we proposed a serial architecture for Shadow. At the same time, the 4-branch round function component of the Shadow cipher can be simplified to a 2-branch function when implementing the serial architecture. One round of Shadow encryption is 4 clock cycles. As shown in Fig. 14, P_1, P_2, P_3, and P_4 represent the four branches of the initial data to be encrypted. Two hardware architectures for Shadow only implement the encryption module.

1) FPGA Implementation: We used the ISE Design Suite 14.6 to synthesize the RTL generated by the two architectures of Shadow and map them to the Xilinx Virtex-5 FPGA development board based on XC5VLX50T-2ff1136 to evaluate the indicators, such as the Slice, LUT, FlipFlop, Energy, and Throughput. As shown in Table IV, the serial architecture consumes slightly fewer resources than the round-based architecture, the round-based architecture has higher throughput and less energy consumption than the serial architecture. The Shadow round function component involves three operations: 1) AND; 2) XOR; and 3) Rotation. Among them, the hardware implementation of Rotation is through the connection operation, there is no need to spend hardware resources. In the serial architecture, we reduced the hardware resources consumed by AND operation and XOR operation by increasing the reuse rate of components. At the same time, we have added some control logic resources to achieve the original encryption effect. The increased control logic resources offset part of the reduced AND operation and XOR operation resources, so the serial architecture resources are slightly lower than the round-based architecture. Also, we tried our best to find some classic lightweight block cipher algorithms that were synthesized on Virtex-5 XC5VLX50T-2ff1136, as shown in Table V. When comparing the FlipFlop, Slice, LUT, and Throughput of each cipher in Table V, the Shadow we designed has certain advantages.

2) ASIC Implementation: Shadow with serial structure has been synthesized using Synopsys Design Compiler version A-2007.12-SP1 with UMCL18G212T3 standard cell library, which is based on the UMC L180 0.18 μm 1P6M logic process with a typical voltage of 1.8 V and a clock frequency of 100 kHz.

An AND gate costs 1.33 GE, and an XOR gate costs 2.67 GE. In the hardware implementation, the shift operation and permutation do not consume hardware resources. The plaintext and the key are stored in the registers. To store the 32-b plaintext register in Shadow-32 requires 277.12 GE, and the 64-b key register requires 384 GE. The AddRoundkey contains a 32-b XOR operation and the 32-b XOR requires 85.44 GE. The key schedule process includes the round constant 5-b XOR, NX operation module, and bit permutation. The round constant 5-b XOR requires 17 GE. The NX operation uses four AND gates and 13 XOR gates, which requires 40.03 GE. The round function uses one AND gate and three XOR gates, which costs 9.34 GE. The logic and other counters of Shadow-32 require 23 GE. Eventually, the area of Shadow-32 is 835.93 GE, and the detailed list is given in

TABLE VI
AREA REQUIREMENT OF THE SHADOW-32

Modules	GE	%
Data Register	277.12	33.15
Key Register	384	45.94
Key XOR	85.44	10.22
Constants	17	2.03
NX operation	40.03	4.79
Round function	9.34	1.12
Control logic and other counters	23	2.75
Total	835.93	100

TABLE VII
COMPARISON BETWEEN THE HARDWARE RESOURCES OF THE SHADOW AND OTHER CIPHERS

Algorithm	Structure	Data(bit)	Key(bit)	Area(GE)	Logic process	Ref.
PRESENT	SPN	64	80	2018	$0.18\mu m$	[26]
PRESENT	SPN	64	128	2783	$0.18\mu m$	[27]
PRINCE	SPN	64	128	3491	$0.18\mu m$	[28]
Piccolo	GFN	64	80	1634	$0.18\mu m$	[26]
HIGHT	ARX	64	128	3048	$0.25\mu m$	[17]
SIMON	ARX	64	128	1751	$0.13\mu m$	[29]
SPECK	ARX	64	128	2014	$0.13\mu m$	[29]
LEA	ARX	128	128	3826	$0.18\mu m$	[19]
LEA	ARX	128	128	4295.5	$0.13\mu m$	[19]
Shadow	ARX	32	64	835.93	$0.18\mu m$	*Tw
Shadow	ARX	64	128	1687.86	$0.18\mu m$	*Tw

* : This work.

Table VI. The area of Shadow-128 is 1687.86 GE. Table VII shows the comparison between the hardware resources of the Shadow and other ciphers.

IV. SECURITY ANALYSIS

A. Impossible Differential Analysis

Impossible differential analysis is an extension of differential attack [30]. In this article, we conducted impossible differential analysis on Shadow-32.

First, two differential paths with transfer probability of 1 are given. The differential path E_0 of the first n_1 round: $\triangle\alpha \rightarrow \triangle\beta$ corresponds to the encryption direction, the second n_2 round differential path $E_1 : \triangle\lambda \rightarrow \triangle\gamma$ corresponds to the direction of decryption. If $\triangle\beta \neq \triangle\gamma$, $\triangle\alpha \rightarrow \triangle\gamma$ is an impossible differential path for round $n_1 + n_2$.

A differential path E_0 with probability of 1 is found for the Shadow-32 algorithm, and the differential input of the E_0 is $\Delta\alpha$:(0000, 0001, 0000, 0000, 0000, 0000, 0000, 0000). After 2-round encryption, the differential output is $\Delta\beta$:(0 ∗ 01, ∗∗∗∗, ∗1∗∗, ∗∗∗∗, 0∗∗1, ∗∗∗∗, ∗1∗∗, ∗∗∗∗). Another differential path E_1 for the Shadow-32 algorithm with probability of 1 is $\Delta\lambda$:(0000, 0000, 0000, 0000, 0100, 0000, 0000, 0000). After 2-round decryption, the differential input is $\Delta\gamma$:(∗ ∗ ∗0, ∗01∗, ∗ ∗ ∗∗, 0 ∗ ∗∗, ∗ ∗ ∗0, ∗01∗, ∗ ∗ ∗∗, 1 ∗ ∗∗).

As shown in Table VIII, there is a contradiction between the $\Delta\beta$ and $\Delta\gamma$. Thus, the differential paths E_0 and E_1 can form the 4-round impossible differential path of the shadow-32. Based on the 4-round impossible differential path, decrypt one round forward, and encrypt two rounds backward to build 7-round impossible differential paths. The differential state is as shown in Table IX.

Let the difference of plaintext be

$$\begin{aligned} \Delta L_0 &= (0,0),\ \ \Delta L_1 = (0,0) \\ \Delta R_0 &= (*000, 01*0) \\ \Delta R_1 &= (0*01, ****). \end{aligned} \tag{4}$$

Define the following plaintext structure:

$$\begin{aligned} W^0_{L_0} &= (\alpha_1\alpha_2\alpha_3\alpha_4, \alpha_5\alpha_6\alpha_7\alpha_8) \\ W^0_{L_1} &= (\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}, \alpha_{13}\alpha_{14}\alpha_{15}\alpha_{16}) \\ W^0_{R_0} &= (\beta_1\alpha_{17}\alpha_{18}\alpha_{19}, \alpha_{20}\alpha_{21}\beta_2\alpha_{22}) \\ W^0_{R_1} &= (\alpha_{23}\beta_3\alpha_{24}\alpha_{25}, \beta_4\beta_5\beta_6\beta_7) \end{aligned} \tag{5}$$

$\alpha_i (1 \leq i \leq 26)$ is a constant, and $\beta_i (1 \leq i \leq 6)$ takes an arbitrary value. The plaintext structure contains 2^{25} plaintexts, which can form 2^{49} plaintext pairs. In this article, 2^N structures are selected, and there are $2^{(N+25)}$ plaintext and $2^{(N+49)}$ data pairs. After seven rounds of encryption, select the ciphertext pair that satisfies the following form:

$$\begin{aligned} \Delta C^7_{L_0} &= (\gamma_1\gamma_2\gamma_30, \gamma_401\gamma_5) \\ \Delta C^7_{L_1} &= (\delta_1\delta_2\delta_3\delta_4, 1\delta_5\delta_6\delta_7) \\ \Delta C^7_{R_0} &= (\varepsilon_1\varepsilon_2\varepsilon_30, \varepsilon_401\varepsilon_5) \\ \Delta C^7_{R_1} &= (\epsilon_1\epsilon_2\epsilon_3\epsilon_4, 1\epsilon_5\epsilon_6\epsilon_7). \end{aligned} \tag{6}$$

There are $2^{(N+42)}$ ciphertext pairs in total that satisfy (6). According to the differential property of the AND operation and $\Delta r_2 = (0000, 0000, 0 * 01, * * **, *000, 01 * 0, 0000, 0000)$, we suspect 10-b key in second round, after screening, the plaintext pair is about $2^{(N+42)} \times 2^{(-10)} = 2^{(N+32)}$. According to $\Delta r_3 = (0*01, ****, *1**, ****, 0* *1, ****, *1**, ****)$, we suspect 4-b key in third round, after screening, the plaintext pair is about $2^{(N+28)}$. According to $\Delta r_7 = (***0, *01*, ****, 1***, ***0, *01*, ****, 1***)$, we suspect 9-b key in seventh round, after screening, the plaintext pair is about $2^{(N+19)}$. The number of remaining error keys is $2^{23} \times (1 - 2^{-1})^{2^{N+19}} < 1$. In summary, the data complexity of the attack is $2^{(N+25)}$ selected plaintext.

B. Biclique Analysis

1) Process of Biclique Analysis: The biclique attack [31] is a variant of the meet-in-the-middle (MITM) method of cryptanalysis, which is known for both full AES and full IDEA. Since then, biclique-based key recovery attacks have been widely used in the analysis of block ciphers. The basic idea of the biclique attack is to reduce the complexity of the middle encounter attack by constructing a biclique structure. For the convenience of expression, we represent the 128-b primary key K as a concatenation of $K[\cdot]$, which is

$$K = \begin{cases} K[0] = k_0 \| k_1 \| k_2 \| k_3 \| k_4 \| k_5 \| k_6 \| k_7 \\ K[1] = k_8 \| k_9 \| k_{10} \| k_{11} \| k_{12} \| k_{13} \| k_{14} \| k_{15} \\ \quad \vdots \\ K[14] = k_{112} \| k_{113} \| \cdots k_{116} \| k_{117} \| k_{118} \| k_{119} \\ K[15] = k_{120} \| k_{121} \| \cdots k_{124} \| k_{125} \| k_{126} \| k_{127}. \end{cases} \tag{7}$$

Then, the primary key involved for each round of AddRoundkey is shown in Table X.

TABLE VIII
4-ROUND IMPOSSIBLE DIFFERENTIAL PATH OF SHADOW-32

	ΔL_0	ΔL_1	ΔR_0	ΔR_1
Δr_4	(0000,0001)	(0000,0000)	(0000,0000)	(0000,0000)
Δr_5	(0000,0000)	(0*01,****)	(*000,01*0)	(0000,0000)
Δr_6	(0*0**1**,****)	(*1**,****)	(0***1**,****)	(*1**,****)
Δr_6	(***0**,*01*)	(****,0***)	(***0**,*01*)	(****,1***)
Δr_7	(1*0*,0000)	(***0,*01*)	(0000,0000)	(0000,0000)
Δr_8	(0000,0000)	(0000,0000)	(0010,0000)	(0000,0000)

TABLE IX
7-ROUND IMPOSSIBLE DIFFERENTIAL PATH OF SHADOW-32

	ΔL_0	ΔL_1	ΔR_0	ΔR_1
Δr_0	(0000,0000)	(0000,0000)	(*000,01*0)	(0*01,****)
Δr_1	(0000,0001)	(0000,0000)	(0000,0000)	(0000,0000)
Δr_2	(0000,0000)	(0*01,****)	(*000,01*0)	(0000,0000)
Δr_3	(0*01,****)	(*1**,****)	(0***1**,****)	(*1**,****)
Δr_3	(***0,*01*)	(****,0***)	(***0,*01*)	(****,1***)
Δr_4	(1*0*,0000)	(***0,*01*)	(0000,0000)	(0000,0000)
Δr_5	(0000,0000)	(0000,0000)	(0010,0000)	(0000,0000)
Δr_6	(1*0*,0000)	(0000,0000)	(0000,0000)	(***0,*01*)
Δr_7	(***0,*01*)	(****,1***)	(***0,*01*)	(****,1***)

TABLE X
$K[\cdot]$ USED FOR ADDROUNDKEY OF SHADOW-64

r	The value of $[\cdot]$	r	The value of $[\cdot]$
0	$13, 4, 5, 6, 14, 7, 8, 9$	16	$2, 7, 11, 6, 9, 12, 8, 0$
1	$1, 14, 7, 8, 2, 9, 15, 10$	17	$5, 9, 12, 8, 10, 0, 15, 13$
2	$4, 2, 9, 15, 5, 10, 3, 11$	18	$7, 10, 0, 15, 11, 13, 3, 1$
3	$14, 5, 10, 3, 7, 11, 6, 12$	19	$9, 11, 13, 3, 12, 1, 6, 4$
4	$2, 7, 11, 6, 9, 12, 8, 0$	20	$10, 12, 1, 6, 0, 4, 8, 14$
5	$5, 9, 12, 8, 10, 0, 15, 13$	21	$11, 0, 4, 8, 13, 14, 15, 2$
6	$7, 10, 0, 15, 11, 13, 3, 1$	22	$12, 13, 14, 15, 1, 2, 3, 5$
7	$9, 11, 13, 3, 12, 1, 6, 4$	23	$0, 1, 2, 3, 4, 5, 6, 7$
8	$10, 12, 1, 6, 0, 4, 8, 14$	24	$13, 4, 5, 6, 14, 7, 8, 9$
9	$11, 0, 4, 8, 13, 14, 15, 2$	25	$1, 14, 7, 8, 2, 9, 15, 10$
10	$12, 13, 14, 15, 1, 2, 3, 5$	26	$4, 2, 9, 15, 5, 10, 3, 11$
11	$0, 1, 2, 3, 4, 5, 6, 7$	27	$14, 5, 10, 3, 7, 11, 6, 12$
12	$13, 4, 5, 6, 14, 7, 8, 9$	28	$2, 7, 11, 6, 9, 12, 8, 0$
13	$1, 14, 7, 8, 2, 9, 15, 10$	29	$5, 9, 12, 8, 10, 0, 15, 13$
14	$4, 2, 9, 15, 5, 10, 3, 11$	30	$7, 10, 0, 15, 11, 13, 3, 1$
15	$14, 5, 10, 3, 7, 11, 6, 12$	31	$9, 11, 13, 3, 12, 1, 6, 4$

In this article, to establish an 8-round biclique structure from round 24 to 31 is constructed with $K[11]$ in round 26 and $K[3]$ in round 27, the keyspace is divided as follows. The primary keyspace is expressed as round keys from round 25 to 28. To simplify the representation for keyspace, round keys from round 25 to 28 are represented by a 4×4 matrix. Each component of the row represents four bytes of round keys. Thus, the keyspace is represented as follows:

$$\begin{bmatrix} K[1], K[14] & K[7], K[8] & K[2], K[9] & K[15], K[10] \\ K[4], K[2] & K[9], K[15] & K[5], K[10] & K[3], \mathbf{K[11]} \\ K[14], K[5] & K[10], \mathbf{K[3]} & K[7], K[11] & K[6], K[12] \\ K[2], K[7] & K[11], K[6] & K[9], K[12] & K[8], K[10] \end{bmatrix}.$$

A structure of a biclique includes 2^d ciphertexts $\{C_0, \ldots, C_{2^d-1}\}$, 2^d intermediate states $\{S_0, \ldots, S_{2^d-1}\}$, and $[K_{((i,j))}]$. First, the $\triangle_i$-differentials are derived from the difference $\triangle_i^K$ where the key difference of $K[11]$ is i. Second, the $\triangledown_j$-differentials are derived from the difference $\triangledown_j^K$, where the key difference of $K[3]$ is j. The remaining 112-b will iteratively take all possible values. Since these differentials do not have common active nonlinear component, which refers to AND operation in this article. The difference between $(\triangle_i^K, \triangledown_j^K, S_0, C_0)$ can be expressed by the following equation:

$$\begin{cases} \chi = K_{(0,0)} \bigoplus \triangle_i^K \bigoplus \triangledown_j^K \\ S_0 \bigoplus \triangledown_j \xrightarrow[f]{\chi} C_0 \bigoplus \triangle_i. \end{cases} \tag{8}$$

We succeed in establishing 8-round biclique structure with dimension 8. Assume that g_1 is the forward subcipher from rounds 0 to 11, and g_2 is the backward subcipher from rounds 12 to 23. At the same time, the plaintext P_i corresponding to each ciphertext C_i in an 8-round biclique is obtained by decryption oracle. Then, the full-round key recovery process

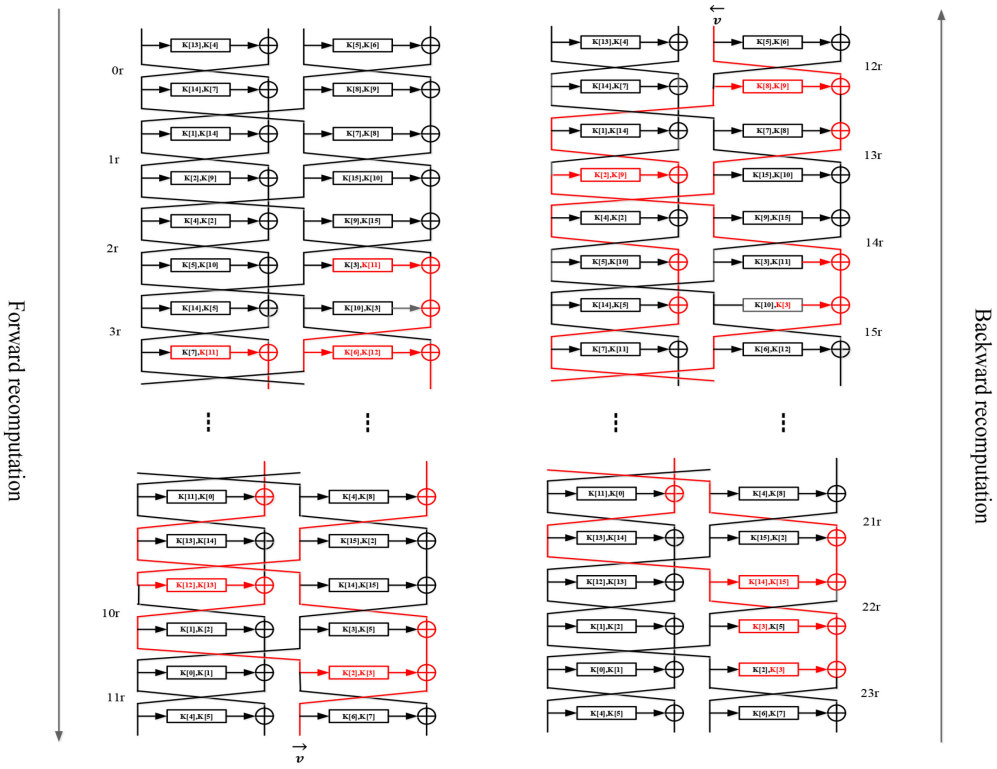


Fig. 15. 8-round biclique structure of Shadow.

of the Shadow-64 can be expressed by the following equation:

$$P_i \xrightarrow[g_1]{K_{(i,j)}} \rightarrow \overset{?}{=} \leftarrow \xleftarrow[g_2]{K(i,j)} S_i. \tag{9}$$

2) Complexity: The biclique attack is an optimized exhaustive algorithm. The effectiveness of an attack on a cryptographic algorithm can be measured by the time complexity. The time complexity of the biclique attack key recovery can be calculated by the following equation:

$$C_{\text{full}} = 2^{K-2d}\left[C_{\text{biclique}} + C_{\text{precomp}} + C_{\text{recomp}} + C_{\text{falsepos}}\right] \tag{10}$$

where K is the size of the key block, and d is the dimension to construct the 8-round biclique structure. C_{biclique} represents the complexity of constructing a single biclique structure. In our cryptanalysis, it is at most $2^{d+1} \times (8/32) = 2^7$ 8-round computations. C_{precomp} represents the precomputation complexity of matching detection. According to the above formulas, the complexity result obtained is $2^8 \times (24/31) \approx 2^{7.61}$. C_{recomp} represents the complexity of the recomputation process. According to Fig. 15, forward recomputation requires 13 16-b AND operations, and backward recomputation requires 5.5 16-b AND operations, which is approximately equal to 4.625 rounds. The complexity of the recomputation process is about $2^{(2\times 8)} \times (4.625/32) \approx 2^{13.21}$. C_{falsepos} indicates the complexity corresponding to the key detection process. Since the matching in (9) is performed on a single byte, C_{falsepos} is less than $2^{2d-8} = 2^8$ computations. Consequently, the total complexity is $2^{112} \times (2^7 + 2^{7.61} + 2^{13.21} + 2^8) = 2^{125.29}$.

V. Conclusion

The use of IoT nodes has spread to all areas of society, and the transmission of private data in sensor networks is extremely easy to intercept and tamper with. As living standards and the popularization of the Internet improve security and privacy protection have become particularly important. Therefore, this article proposed a new logical combination method of the generalized Feistel structure and ARX operation, named Shadow. Shadow overcomes the shortcomings of existing lightweight encryption algorithms based on ARX operations, which is that the diffusion rate is slow, and half of the data has not changed after one round. Second, we proposed a round-based hardware implementation architecture and a serial hardware implementation architecture for Shadow. Compared with the serial architecture, the round-based architecture has a higher throughput and consumes less energy. Besides, we conducted avalanche effect experimental analysis, hardware implementation, and security analysis on Shadow and current classic ciphers based on ARX operations. The results show that Shadow is suitable for low-resource and low-power IoT nodes. Finally, in actual deployment, IoT nodes will leak side-channel information, such as heat, power consumption, time, and electromagnetic radiation. Depending on the information used, a variety of attack methods have emerged, such as power attacks, electromagnetic radiation analysis attacks, time analysis attacks, and error injection attacks. In fact, in most cases, the attacker is restricted. More generally, the attacker has little access to the device. Under the assumption of restricted attacks, adding a mask to the algorithm for protection is currently a common choice. The mask protection technology is relatively mature and has strong portability. For example, Ou and Li [32] added high-order masks to the AES algorithm to resist power attacks. Taking into account the sensitivity of IoT nodes to the area, if necessary, a first-order mask can be added to the Shadow cipher for protection.

REFERENCES

[1] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially aware energy efficient mobile edge collaboration for video distribution," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2197–2209, Oct. 2017.

[2] R. Wang, J. Yan, D. Wu, H. Wang, and Q. Yang, "Knowledge-centric edge computing based on virtualized D2D communication systems," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 32–38, May 2018.

[3] D. Dias and J. P. S. Cunha, "Wearable health devices—Vital sign monitoring, systems and technologies," *Sensors*, vol. 18, no. 8, pp. 2414–2433, Aug. 2018.

[4] J. Xiong *et al.*, "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1530–1540, Apr. 2019.

[5] J. Zhang, Y. Zhao, J. Wu, and B. Chen, "LVPDA: A lightweight and verifiable privacy-preserving data aggregation scheme for edge-enabled IoT," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4016–4027, May 2020.

[6] H. Xie, Z. Yan, Z. Yao, and M. Atiquzzaman, "Data collection for security measurement in wireless sensor networks: A survey," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2205–2224, Apr. 2019.

[7] L. Dalmasso, F. Bruguier, P. Benoit, and L. Torres, "Evaluation of SPN-based lightweight crypto-ciphers," *IEEE Access*, vol.7, pp. 10559–10567, 2019.

[8] B. Liu, L. Li, R.-X. Wu, M.-M. Xie, and Q. P. Li, "Loong: A family of involutional lightweight block cipher based on SPN structure," *IEEE Access*, vol. 7, pp. 136023–136035, 2019.

[9] L. Li, B. T. Liu, and H. Wang, "QTL: A new ultra-lightweight block cipher," *Microprocess. Microsyst.*, vol. 45, pp. 45–55, Aug. 2016.

[10] L. Li, B. T. Liu, Y. M. Zhou, and Y. Zou, "SFN: A new lightweight block cipher," *Microprocess. Microsyst.*, vol. 60, pp. 138–150, Jul. 2018.

[11] A. Bogdanov *et al.*, "PRESENT: An ultralightweight block cipher," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 4727. Berlin, Germany: Springer, Sep. 2007, pp. 450–466.

[12] W. L. Wu and L. Zhang, "LBlock: A lightweight block cipher," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 6715. Berlin, Germany: Springer, Jun. 2011, pp. 327–344.

[13] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A small present," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 10529. Cham, Switzerland: Springer, Aug. 2017, pp. 321–345.

[14] S. Banik *et al.*, "Midori: A block cipher for low energy," in *Advances in Cryptology* (Lecture Notes in Computer Science) vol. 9453. Berlin, Germany: Springer, Dec. 2015, pp. 411–436.

[15] W. T. Zhang, Z. Z. Bao, D. D Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "RECTANGLE: A bit-slice lightweight block cipher suitable For multiple platforms," *Sci. China Inf. Sci.*, vol. 58, no. 12, pp. 1–15, Dec. 2015.

[16] D. Dinu, A. Biryukov, J. Groszschaedl, D. Khovratovich, Y. Corre, and L. Perrin, "FELICS-fair evaluation of lightweight cryptographic systems," *Proc. NIST Workshop Lightweight Cryptogr.*, Gaithersburg, MD, USA, 2015, pp. 1–14.

[17] D. Hong *et al.*, "HIGHT: A new block cipher suitable for low-resource device," *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 4249. Berlin, Germany: Springer, Oct. 2006, pp. 46–59.

[18] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2015, pp. 1–6.

[19] D. Lee, D. C. Kim, D. Kwon, and H. Kim, "Efficient hardware implementation of the lightweight block encryption algorithm LEA," *Sensors*, vol. 14, no. 1, pp. 975–994, Nov. 2014.

[20] B. Koo, D. Roh, H. Kim, Y. Jung, D.-G. Lee, and D. Kwon, "CHAM: A family of lightweight block ciphers for resource-constrained devices," in *Information Security and Cryptology* (Lecture Notes in Computer Science), vol. 10779. Cham, Switzerland: Springer, Mar. 2017, pp. 3–25.

[21] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong, "The simeck family of lightweight block ciphers," in *Cryptographic Hardware and Embedded Systems*, vol. 9293. Berlin, Germany: Springer, Sep. 2015, pp. 307–329.

[22] Z. Mishra, G. Ramu, and B. Acharya, "Hight speed low area VLSI architecture for LEA encryption algorithm," in *Proc. 3rd Int. Conf. Microelectron. Comput. Commun. Syst.*, Jan. 2019, pp. 155–160.

[23] Z. Mishra, P. K. Nath, and B. Acharya, "High throughput unified architecture of LEA algorithm for image encryption," *Microprocess. Microsyst.*, vol. 78, Oct. 2020, Art. no. 103214.

[24] N. Hanley and M. O'Neill, "Hardware comparison of the ISO/IEC 29192–2 block ciphers," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Aug. 2012, pp. 57–62.

[25] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, "Lightweight hardware architectures for the present cipher in FPGA," in *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2544–2555, Sep. 2017.

[26] T. P. Berger, J. Francq, M. Minier, and G. Thomas, "Extended generalized feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput," *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2074–2089, Jul. 2016.

[27] M. R. Z'aba, N. Jamil, M. E. Rusli, M. Z. Jamaludin, and A. A. M. Yasir, "I-PRESENT^{TM}: An involutive lightweight block cipher," *J. Inf. Security*, vol. 5, no. 3, pp. 114–122, Jul. 2014.

[28] J. Borghoff *et al.*, "PRINCE—A low-latency block cipher for pervasive computing applications," *Advances in Cryptology— ASIACRYPT* (Lecture Notes in Computer Science), vol. 7658. Berlin, Germany: Springer, Dec. 2012, pp. 208–225.

[29] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "SIMON and SPECK: Block ciphers for the Internet of things," *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 585, Jul. 2015.

[30] D. Yang, W.-F. Qi, and H.-J. Chen, "Impossible differential attacks on the SKINNY family of block ciphers," *IET Inf. Security*, vol. 11, no. 6, pp. 377–385, Nov. 2017.

[31] G. Han, H. Zhao, and C. Zhao, "Unbalanced biclique cryptanalysis of full-round GIFT," *IEEE Access*, vol. 7, pp. 144425–144432, 2019.

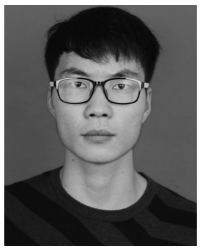
[32] Y. Ou and L. Li, "Research on a high-order AES mask anti-power attack," *IET Inf. Security*, vol. 14, no. 5, pp. 580–586, Sep. 2020.

Ying Guo received the B.S. degree from Hengyang Normal University, Hengyang, China, in 2019, where she is currently pursuing the master's degree.

Her current research interests include embedded systems and information security.

Lang Li received the B.S. degree in circuits and systems from Hunan Normal University, Changsha, China, in 1996, and the master's and Ph.D. degrees in computer science from Hunan University, Changsha, China, in 2006 and 2010, respectively.

Since 2011, he has been working as a Professor with the College of Computer Science and Technology, Hengyang Normal University, Hengyang. His research interests include embedded computing and information security.



Botao Liu received the B.S. degree from Hengyang Normal University, Hengyang, China, in 2016, and the master's degree in computer science from Guizhou University, Guizhou, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China.

His current research interests include cryptography and lightweight countermeasures against side-channel analysis.