Shining Light on the Shadow: Full-round Practical Distinguisher for Lightweight Block Cipher Shadow

Sunyeop Kim[1], Myoungsu Shin[1], Seonkyu Kim[1], Hanbeom Shin[1], Insung Kim[1], Donggeun Kwon[1], Dongjae Lee[1], Seonggyeom Kim[4], Deukjo Hong[3], Jaechul Sung[2], and Seokhie Hong[1]

[1] Institute of Cyber Security & Privacy (ICSP), Korea University, South Korea
kin3548@gmail.com, houma757@gmail.com, kimsg125@korea.ac.kr, shb115@naver.com, cmcom35@korea.ac.kr, donggeun.kwon@gmail.com, ldj0676@korea.ac.kr, shhong@korea.ac.kr

[2] Department of Mathematics, University of Seoul, South Korea jcsung@uos.ac.kr

[3] Department of Information Technology & Engineering, Jeonbuk National University, South Korea deukjo.hong@jbnu.ac.kr

[4] System LSI Business, Samsung Electronics, South Korea jeffgyeom@gmail.com

Abstract. Shadow is a lightweight block cipher proposed at IEEE IoT journal 2021. Shadow's main design principle is adopting a variant 4-branch Feistel structure in order to provide a fast diffusion rate. We define such a structure as Shadow structure and prove that it is almost identical to the Generalized Feistel Network, which invalidates the design principle. Moreover, we give a structural distinguisher that can distinguish Shadow structure from random permutation with only two plaintext/ciphertext pairs. By exploiting the key schedule, the distinguisher can be extended to key recovery attack with only one plaintext/ciphertext pair. Furthermore, by considering Shadow's round function, only certain forms of monomials can appear in the ciphertext, resulting in an integral distinguisher of four plaintext/ciphertext pairs. Even more, the algebraic degree does not increase more than 12 for Shadow-32 and 20 for Shadow-64 regardless of rounds used. Our results show that Shadow is highly vulnerable to algebraic attacks, and that algebraic attacks should be carefully considered when designing ciphers with AND, rotation, and XOR operations.

Keywords: Block cipher, algebraic attack, cube attack

1 Introduction

The rapid advancement of digital technology and the internet has ushered in the era of the Internet of Things (IoT), where various devices are interconnected and interact with each other. In this IoT environment, security issues are becoming increasingly important as wireless communication and data exchange become prevalent. Additionally, traditional encryption techniques are facing challenges due to their heavy processing tasks and high costs, making them less efficient.

In response to this situation, numerous lightweight block ciphers are gaining attention. Lightweight block ciphers offers high security with minimal resources, ensuring efficient processing speed and low energy consumption compared to conventional encryption techniques. As a result, it has become even more critical in the IoT environment to maintain security while efficiently managing resources. A variety of lightweight block ciphers, such as Midori[1], PRESENT[5], HIGHT[14], GIFT[2], SIMON and SPECK[3], LEA[13], SKINNY[4], PIPO[16], PRINCE[6], LED[11], have been introduced in the literature and Shadow[12] is one of them.

Shadow is a lightweight block cipher whose round function consists of AND, Rotation, and XOR (AND-RX) operations. Shadow uses a variant of the generalized Feistel structure and the designers claim that their new logical combination method of AND-RX operations improves the diffusion rate of AND-RX ciphers.

A well-designed symmetric-key cipher should appear in a large number of monomials and have a high algebraic degree. Failure to do so makes it vulnerable to algebraic attacks using Gröbner basis[9], XL algorithms[8], as well as higher-order differential attacks [17], interpolation attacks [15], cube attacks [10]. In general, algebraic degree of block size increases with the number of rounds, reaching a maximum degree of (block size -1) in a given round. The complexity of an algebraic attack is usually calculated with the assumption that all possible monomials can appear, so the complexity of the attack increases exponentially as the degree increases, making the attacks impossible. An upper bound of the algebraic degree of a block cipher can be computed based on [7].

1.1 Our Contribution

1. **Structural Distinguisher for Shadow Structure.** We define a Shadow structure (Figure 1 in Appendix A) which is a generalization of Shadow by replacing the AND-RX update function (Figure 2 in Appendix A) to an arbitrary function F. We then show that the Shadow structure is almost equivalent to the 4-branch Generalized Feistel Network by transforming it, and shows that Shadow's main design principle for fast diffusion does not work and even allows a full-round[5] distinguisher. The structural distinguisher can distinguish Shadow structure from random permutation with only two plaintext/ciphertext pairs. The distinguisher can be extended to low data key recovery attack that only needs one plaintext/ciphertext pair by exploiting the key schedule.

2. **Algebraic Weakness on Shadow.** Considering Shadow's AND-RX structure update function, we prove that only certain forms of monomials can appear in ciphertext, leading to a very weak algebraic property. This allows us to find a low data integral distinguisher that only needs four ciphertexts. Also we theoretically computed upper bounds of the algebraic degree for

[5] In this distinguisher, the number of rounds is irrelevant. Therefore, the distiguisher works for any number of rounds.

full-round Shadow-32 and Shadow-64; 12 (≤ 31) and 20 (≤ 63), respectively. This means that the full-round ciphers have the algebraic degree much less than (block size $-$ 1) giving algebraic weakness. All of the above results are experimentally verified and especially, the evaluation result that the algebraic degree of full-round Shadow-32 can reach up to 12 shows that the upper bound on the algebraic degree is tight.

1.2 Organization

Section 2 introduces block cipher Shadow, Shadow structure, and notations needed. In Section 3, we give a structural distinguisher and extend it to key recovery attacks by exploiting the key schedule. In Section 4, we show the algebraic weakness of Shadow, present integral distinguisher, and provide upper bound of algebraic degree of shadow. We conclude the paper in Section 5

2 Preliminary

This chapter introduces the block cipher Shadow and defines the Shadow structure.

2.1 Boolean Functions and Vectorial Boolean Functions

A Boolean function f on n variables is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. f can be represented by a polynomial on n variables over $\mathbb{F}_2$, called algebraic normal form(ANF). The algebraic degree of f ,denoted by $deg(f)$, is defined as the degree of its ANF. f is called linear if $deg(f) \leq 1$. A (m, n)-vectorial Boolean function F is a function from $\mathbb{F}_2^m$ to $\mathbb{F}_2^n$ and algebraic degree of F is defined as the highest degree of its coordinates. For vectorial Boolean function $F = (f_0, f_1, ..., f_{n-1}), G = (g_0, g_1, ..., g_{n-1})$, we write $F|G$ if there exist a Boolean function h_i s.t $h_i f_i = g_i$ for all $0 \leq i < n$

2.2 Cube Attack

let $k = (k_1, ..., k_n)$ and $v = (v_1, ..., v_m)$ be a n secret variable and m public variables, respectively. Then each bit of symmetric-key cryptosystem can be represented as Boolean function $f(k, v)$. In our case, k will denote masterkey of block cipher and v will denote plaintext and $f(k, v)$ will denote each bit of ciphertext.

Let a set of public variables $I = \{v_{i_1}, v_{i_1}, ..., v_{i_d}\}$ be a set of cube variables. Then $f(k, v)$ can be rewritten as

$$f(k, v) = t_I \cdot p_I(k, v) \oplus q_I(k, v)$$

where $t_I = \prod_{v \in I} v$ and p_I does not contain any variable in I, and each term in q_I is not divisible by t_I.

Let C_I, which is referred as a cube, be a set of $2^{|I|}$ values where variables in I are taking all possible combinations of values, and all remaining variables are fixed to some arbitrary vales. Then following equation holds.

$$\bigoplus_{(v_{i_1}, v_{i_1}, \ldots, v_{i_d}) \in \{0,1\}^d} f(k, v) = p_I(k, v)$$

We will mainly use the fact that if $f(x, v)$ does not contain a multiple of t_I, then $p(x, v) = 0$ which is usually called integral distinguisher.

2.3 Notations

$$\begin{aligned} \& &: \text{bitwise } AND \text{ (can be omitted)} \\ \oplus &: \text{bitwise } XOR \\ S^j(x) &: j\text{-bit left rotation of } x \\ S^I(x) &:= \prod_{j \in I} S^j(x) \\ x[n] &: n\text{-th bit of } x \text{ (counted from 0)} \end{aligned}$$

2.4 Specification of Block Cipher Shadow

Shadow is a block cipher suggested in [12] which is based on variant of 4-branch generalized Feistel structure. Shadow-32 uses 32-bit block size and 64-bit key, and Shadow-64 uses 64-bit block size and 128-bit key. Round number (RN) is 16 and 32, respectively.

Shadow Structure Before defining block cipher Shadow, we first define the Shadow structure (Figure 1 in Appendix A) that restricts the Shadow's update function to an arbitrary function F. For round key $K_0, K_1, K_2, K_3 \in \{0,1\}^n$ and update function $F : \{0,1\}^n \to \{0,1\}^n$, Round function for Shadow structure is $R^0 \circ R^1$ where

$$R^0, R^1 : \{0,1\}^{n \times 4} \to \{0,1\}^{n \times 4}$$

$$R^0(A, B, C, D) = (B \oplus F(A) \oplus K_0, B, D \oplus F(C) \oplus K_1, C)$$

$$R^1(A, B, C, D) = (C, B \oplus F(A) \oplus K_2, A, D \oplus F(C) \oplus K_3)$$

We define the input of the i-round R^0, R^1 as $L_0^{i-1} \| L_1^{i-1} \| R_0^{i-1} \| R_1^{i-1}$ and $M_0^{i-1} \| M_1^{i-1} \| N_0^{i-1} \| N_1^{i-1}$, divided into four n-bit branches, respectively. We also define i-round roundkey for R^0,R^1 as K_L^{i-1}, K_R^{i-1} and K_M^{i-1}, K_N^{i-1} respectively.

Block Cipher Shadow Shadow uses the AND-RX update function $F(x) = S^a(x)\&S^b(x) \oplus S^c(x)$ where $a = 1, b = 7, c = 2$ and $n = 8$ for Shadow-32 and $n = 16$ for Shadow-64 (Figure 2 in Appendix A). Shadow-32 and Shadow-64 use **generator 1** and **generator 2** as their key schedules respectively and consist of **Addroundconstant**, **NX**, and **permutation**. Masterkey K is represented by $k_0\|k_1\|k_2\|\cdots\|k_{62}\|k_{63}$ or $k_0\|k_1\|\cdots\|k_{126}\|k_{127}$ and the round number r is used as the round constant and are represented by $c_0\|c_1\|c_2\|c_3\|c_4$ or $c_0\|c_1\|c_2\|c_3\|c_4\|c_5$.

1. **generator 1**
   Round function of generator 1 consist of the following operations and uses the upper 32 bits after the round function as the round key.

   (a) **Addroundconstant**
   Addroundconstant XORs 5-bit $k_3\|k_4\|k_5\|k_6\|k_7$ with round constant $c_0\|c_1\|c_2\|c_3\|c_4$.

   (b) **NX**
   NX is an operation applied to 8-bit $k_{56}\|k_{57}\|\cdots\|k_{62}\|k_{63}$ and is represented as follows.

$$
\begin{aligned}
k'_{56} &= k_{56}\&(k_{56} \oplus k_{62})\\
k'_{57} &= k_{57}\&(k_{57} \oplus k_{63})\\
k'_{58} &= k_{58}\&(k_{58} \oplus k_{56} \oplus k_{62})\\
k'_{59} &= k_{59}\&(k_{59} \oplus k_{57} \oplus k_{63})\\
k'_{60} &= k_{60}\&(k_{60} \oplus k_{58} \oplus k_{56} \oplus k_{62})\\
k'_{61} &= k_{61}\&(k_{61} \oplus k_{59} \oplus k_{57} \oplus k_{63})\\
k'_{62} &= k_{62}\&(k_{62} \oplus k_{60} \oplus k_{58} \oplus k_{56} \oplus k_{62})\\
k'_{63} &= k_{63}\&(k_{63} \oplus k_{61} \oplus k_{59} \oplus k_{57} \oplus k_{63})
\end{aligned}
$$

   (c) **permutation**
   This operation is the permutation according to Table 4 in Appendix A.

2. **generator 2** Round function of generator 2 consists of the following operations and uses the upper 64 bits after the round function as the round key.

   (a) **Addroundconstant**
   Addroundconstant XORs 6-bit $k_2\|k_3\|k_4\|k_5\|k_6\|k_7$ with round constant $c_0\|c_1\|c_2\|c_3\|c_4\|c_5$.

   (b) **NX**
   NX is an operation applied to 24-bit $k_{104}\|k_{105}\|\cdots\|k_{126}\|k_{127}$ and is

represented as follows.

$$\begin{aligned}
k'_{104} &= k_{104} \& (k_{104} \oplus k_{126}) \\
k'_{105} &= k_{105} \& (k_{105} \oplus k_{127}) \\
k'_{106} &= k_{106} \& (k_{106} \oplus k_{104} \oplus k_{126}) \\
k'_{107} &= k_{107} \& (k_{107} \oplus k_{105} \oplus k_{127}) \\
k'_{108} &= k_{108} \& (k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\
k'_{109} &= k_{109} \& (k_{109} \oplus k_{107} \oplus k_{105} \oplus k_{127}) \\
k'_{110} &= k_{110} \& (k_{110} \oplus k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\
&\vdots \\
k'_{127} &= k_{127} \& (k_{127} \oplus k_{125} \oplus k_{123} \oplus \cdots \oplus k_{105} \oplus k_{127})
\end{aligned}$$

(c) **permutation**
This operation is the permutation according to Table 5 in Appendix A.

3 Structural Attack on Shadow Structure

3.1 Practical Distinguisher of Shadow Structure with Probability 1

Shadow's encryption function can be re-expressed as $(R^0 \circ R^1)^n = R^0 \circ (R^1 \circ R^0)^{n-1} \circ R^1$. If we expand $R^1 \circ R^0$, we get

$$\begin{aligned}
R^1 \circ R^0(M_0^i, M_1^i, N_0^i, N_1^i) = (&M_1^i \oplus F(M_0^i) \oplus F(N_0^i) \oplus K_L^{i+1} \oplus \\
&K_M^i, N_0^i, N_1^i \oplus F(M_0^i) \oplus f(N_0^i) \oplus K_R^{i+1} \oplus K_N^i, M_0^i)
\end{aligned} \tag{1}$$

Thus, contrary to the author's claim that using two F functions per round in both directions makes fast diffusion, Shadow structure is just a branch permutation of the GFN type 2 except for the first and last rounds. We can also see from the equality (1) that

$$\begin{aligned}
M_0^{i+1} \oplus N_0^{i+1} &= M_1^i \oplus N_1^i \oplus K_M^i \oplus K_N^i \oplus K_L^{i+1} \oplus K_R^{i+1} \\
M_1^{i+1} \oplus N_1^{i+1} &= M_0^i \oplus N_0^i
\end{aligned} \tag{2}$$

Applying the equality (2) iteratively, we get

$$\begin{aligned}
&M_{i-1 \bmod 2}^i \oplus N_{i-1 \bmod 2}^i = M_1^0 \oplus N_1^0 \oplus A^i \\
&M_{i \bmod 2}^i \oplus N_{i \bmod 2}^i = M_0^0 \oplus N_0^0 \oplus B^i \\
&\text{where } A^i = \sum_{j=0}^{\lfloor (i-1)/2 \rfloor} K_M^{2j} \oplus K_N^{2j} \oplus K_L^{2j+1} \oplus K_R^{2j+1} \\
&\text{and } B^i = \sum_{j=1}^{\lfloor i/2 \rfloor} K_M^{2j-1} \oplus K_N^{2j-1} \oplus K_L^{2j} \oplus K_R^{2j}
\end{aligned} \tag{3}$$

Substituting the equality 3 for L, R, we get following equalities.

$$\begin{aligned} &L_0^i \oplus R_0^i \oplus L_0^0 \oplus R_0^0 = A^i && \text{if } i \text{ is } \textit{even}, \\ &L_0^i \oplus R_0^i \oplus F(L_0^0) \oplus F(R_0^0) \oplus L_1^0 \oplus R_1^0 = B^i \oplus K_L^0 \oplus K_R^0 && (4) \end{aligned}$$

$$\begin{aligned} &L_1^i \oplus R_1^i \oplus F(L_0^i) \oplus F(R_0^i) \oplus L_0^0 \oplus R_0^0 = A^i \oplus K_M^{i-1} \oplus K_N^{i-1} && \text{if } i \text{ is } \textit{odd}, \\ &L_1^i \oplus R_1^i \oplus L_1^0 \oplus R_1^0 \oplus F(L_0^i) \oplus F(R_0^i) \oplus F(L_0^0) \oplus F(R_0^0) \\ &= B^i \oplus K_L^0 \oplus K_R^0 \oplus K_M^{i-1} \oplus K_N^{i-1} && (5) \end{aligned}$$

Thus, given only 2 plaintext and ciphertext pairs, we can distinguish a random permutation from a Shadow structure with advantage $1-2^{-2n}$ by checking that the left-hand side of the equation (4) or (5) matches according to the parity of i. However, even if we guess the round keys of the first or last rounds, right-side of the equation (4) or (5) only gives us the information of sum of the round keys. So it is not trivial to extend the above distinguisher to key recovery attacks.

3.2 Key recovery Attack on Shadow Structure

If the keyschedule is known, the master key can be recovered with only single ciphertext/plaintext pair by first filtering out possible keys based on the keyschedule, and then exhaustively searching the remaining keys. The attack Procedure and its complexity is as follows(k denotes the length of masterkey).

1. **checking keyschedule** Compute the right-hand side of the equation (4) or (5) through the given ciphertext/plaintext pair. Then filter out only the master keys that can generate it. There are two ways to filter out the master key depending on the (non-)linearity of the keyschedule.

   (a) **nonlinear keyschedule**
   Just bruteforce the masterkey on the keyschedule. If we define the complexity of computing a keyschedule once as C_{ks}, the complexity is $C_{ks} \times 2^k$.

   (b) **linear keyschedule**
   Represent the right-side of the equation (4) or (5) as an equation of the master key, and solve the equation. Since the complexity of solving linear equations is relatively negligible, we ignore this complexity.

2. **Bruteforce remaining masterkeys** Since the expected number of remaining masterkey is 2^{k-2n}, the complexity is $C_{enc} \times 2^{k-2n}$ if we define the complexity of one encryption as C_{enc}

If $C_{enc} \gg C_{ks}$, which is the majority case, we can recover masterkey with lower complexity than exhaustive search. In particular, if the keyschedule is linear, we can recover the master key with much low complexity $C_{enc} \times 2^{k-2n}$.

4 Algebraic Weakness on Shadow

Shadow ciphers have the additional weakness that only certain forms of monomials can appear on the ANF of ciphertext due to the AND-RX structure of F function. In this chapter, we present a theoretical proof and experimental results for this. Throughout this chapter, we assume that key is fixed and regard each branch as a vectorial Boolean function of plaintext only. Also, we assume that $n = 16$ and a, b are odd, which coincides to the parameter of Shadow-32. However, n can be easily generalized to any even number.

4.1 Possible Monomials on Shadow

Define X_0 and X_1 as follows

$$X_0 = M_0^0 \oplus N_0^0$$
$$X_1 = M_1^0 \oplus N_1^0$$

By modifying the equality (2), we can get following equation

$$M_0^i \oplus N_0^i = X_{i \mod 2} \oplus C^i \tag{6}$$

where $C^i = \begin{cases} B^i & \text{if } i \text{ is even} \\ A^i & \text{if } i \text{ is odd} \end{cases}$.

Using the equality (6), we can expand first element of equality (1) as follows.

$$\begin{aligned}
&M_0^{i+1} \oplus K_L^{i+1} \oplus K_M^i \\
&= f(M_0^i) \oplus f(N_0^i) \oplus M_1^i \\
&= S^a(M_0^i)S^b(M_0^i) \oplus S^a(N_0^i)S^b(N_0^i) \oplus S^c(M_0^i) \oplus S^c(N_0^i) \oplus M_1^i \\
&= S^a(M_0^i)S^b(M_0^i) \oplus S^a(X_{i \bmod 2} \oplus M_0^i \oplus C^i)S^b(X_{i \bmod 2} \oplus M_0^i \oplus C^i) \\
&\quad \oplus S^c(X_{i \bmod 2} \oplus C^i) \oplus N_0^{i-1} \\
&= S^a(M_0^i)S^b(X_{i \bmod 2}) \oplus S^a(X_{i \bmod 2})S^b(M_0^i) \oplus S^a(C^i)S^b(M_0^i) \\
&\quad \oplus S^a(M_0^i)S^a(C^i) \oplus M_0^{i-1} \oplus S^a(X_{i \bmod 2})S^b(X_{i \bmod 2}) \oplus S^a(C^i)S^b(X_{i \bmod 2}) \\
&\quad \oplus S^a(X_{i \bmod 2})S^a(C^i) \oplus S^c(X_{i \bmod 2}) \\
&\quad \oplus X_{i-1 \bmod 2} \oplus S^a(C^i)S^b(C^i) \oplus S^c(C^i) \oplus C^{i-1}
\end{aligned} \tag{7}$$

Note that we can make similar equality for M_0^i.

Theorem 1. *Let*

$$P_0 = \{l\&S^{\{0,2,4,6\}}(X_0)S^{\{1,3,5,7\}}(X_1) : l \text{ is linear over } M_0^0, N_0^0, M_1^0, N_1^0\}$$
$$P_1 = \{l\&S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1) : l \text{ is linear over } M_0^0, N_0^0, M_1^0, N_1^0\}$$

and define

$$span(S) := \{\sum_{i=1}^{k} \lambda_i \& v_i : k \in \mathbb{N}, v_i \in S, \lambda_i \in \mathbb{F}_2^n\},$$

where S is a set of $(m, n$-vectorial Boolean functions$)$, and

$$L(S) := span(\{p : \exists s \in S \ s.t \ p|s\}),$$

where S is a set of vectorial Boolean functions. Then $M_0^i, N_0^i \in L(P_{i \bmod 2})$ if a, b are odd.

Proof. without loss of generality, we will prove for M_0^i. First we define following two properties.

1. **interchangeability** : S^a, S^b will move the elements of P_i to $P_{i+1 \bmod 2}$
2. **closedness** : $L(P_i)$ is closed for the multiplication of $S^a(X_{i+1 \bmod 2})$ and $S^b(X_{i+1 \bmod 2})$ since a, b are both odd

Based on the above properties, our proof proceeds with mathematical induction.

1. **Base condition($i = 0$)** Trivially holds since M_0^0 itself is linear.
2. **Base condition($i = 1$)** Holds if we put $i = 0$ on the equality (7)
3. **Inductive Step** Assume that hypothesis holds for $i \leq k$. Then for $i = k+1$, it is enough to show that every term in (7) are in $L(P_{k+1 \bmod 2})$.
   (a) $S^a(M_0^k)S^b(X_{k \bmod 2})$ and $S^a(X_{k \bmod 2})S^b(M_0^k)$
   By induction hypothesis $M_0^k \in L(P_{k \bmod 2})$ and thus both term are in $L(P_{k+1 \bmod 2})$ by the closedness and interchangeability.
   (b) $S^a(C^k)S^a(M_0^k)$ and $S^a(M_0^k)S^a(C^k)$
   By induction hypothesis $M_0^k \in L(P_{k \bmod 2})$. Thus both terms are in $L(P_{k+1 \bmod 2})$ since C^k is constant and interchangeability.
   (c) M_0^{k-1}
   $M_0^{k-1} \in L(P_{k+1 \bmod 2})$ by the induction hypothesis
   (d) $S^a(X_{k \bmod 2})S^b(X_{k \bmod 2})$
   $S^a(X_{k \bmod 2})S^b(X_{k \bmod 2}) \in L(P_{k+1 \bmod 2})$ since $S^a(X_{k \bmod 2})$ is a linear term.

(e) extra terms

They are all in $L(P_{k+1 \bmod 2})$ since they are all linear or constant term.

■

From theorem 1, we can see the vulnerability that only elements of $L(P_0)$ or $L(P_1)$ can appear on the ciphertext. This vulnerability allows us to derive the following low data integral distinguisher.

Corollary 1. *For even r, e_0, e_1 and odd o, $L_0^r[o]$ does not contain $L_1^0[e_0]L_1^0[e_1]$ or it's multiples. Therefore for a cube variables $I = \{L_1^0[e_0], L_1^0[e_1]\}$,*

$$\bigoplus_{C_I} L_0^r[o] = 0$$

i.e, I forms a integral distinguisher

Proof. By Theorem 1, $L_0^r = N_0^{r-1} \in L(P_1)$ and remember that

$$P_1 = \{l \& S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1) : l \text{ is linear over } M_0^0, N_0^0, M_1^0, N_1^0\}$$
$$X_0 = N_0^0 \oplus M_0^0 = L_1^0 \oplus R_1^0 \oplus F(L_0^0) \oplus F(R_0^0) \oplus K_L^0 \oplus K_R^0$$
$$X_1 = N_1^0 \oplus M_1^0 = L_0^0 \oplus R_0^0$$

Therefore except the arbitrary linear term l, L_1^0 can only lies on $S^1(X_0)S^3(X_0)S^5(X_0)S^7(X_0)$. Since a, b are both odd, the odd bits of L_1^0 cannot affect the odd bits of $S^1(X_0)S^3(X_0)S^5(X_0)S^7(X_0)$. Thus, even if we consider l, L_r^0 does not include $L_1^0[e_0]L_1^0[e_1]$ and its multiples. Other cases when the parity of the round and cube indices is changed are listed in Table 1.

■

Table 1. Cube variables with integral bits

Round	Cube variables for L_1^0, R_1^0	Integral bits for L_0^r, R_0^r
even	any two even bits	odd
even	any two odd bits	even
odd	any two even bits	even
odd	any two odd bits	odd

From theorem 1, we can also derive the non-trivial upper bound of algebraic degree for each branch of Shadow.

Corollary 2. *For arbitrary round r, upper bound of algebraic degree of each branch can computed as follows if a, b are odd.*

$$deg(L_0^r), deg(R_0^r) \leq 2 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1))$$
$$deg(L_1^r), deg(R_1^r) \leq 4 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1))$$

Proof. We will assume that r is even and give proofs for L_0^r, L_1^r. Other cases can also be proved similarly.

1. **proof for** L_0^r
   By Theorem 1, $L_0^r = N_0^{r-1} \in L(P_{r-1 \bmod 2}) = P_1$. Thus

$$\begin{aligned} deg(L_0^r) &\leq deg(l \& S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \\ &\leq deg(l) + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \\ &\leq 2 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \end{aligned}$$

   Note that $deg(l) = 2$ since we consider it as a variable of $L_0^0, L_1^0, R_0^0, R_1^0$.
2. **proof for** L_1^r
   since

$$\begin{aligned} L_1^r &= M_1^{r-1} \oplus F(M_0^{r-1}) \oplus K_L^{r-1} \\ &= N_0^{r-2} \oplus S^a(M_0^{r-1})S^b(M_0^{r-1}) \oplus S^c(M_0^{r-1}) \oplus K_L^{r-1} \end{aligned}$$

   We will compute the upper bound of degree part by part.
   (a) $deg(N_0^{r-2})$
   since $N_0^{r-2} \in L(P_{r \bmod 2}) = P_0$,

$$\begin{aligned} deg(N_0^{r-2}) &\leq 2 + deg(S^{\{0,2,4,6\}}(X_0)S^{\{1,3,5,7\}}(X_1)) \\ &= 2 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \end{aligned}$$

   (b) $deg(S^a(M_0^{r-1})S^b(M_0^{r-1}))$
   since $M_0^{r-1} \in L(P_1)$,

   $S^a(M_0^{r-1})S^b(M_0^{r-1}) \in L(\{l_1 l_2 \& S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1) : l_1, l_2$
   are linear over $M_0^0, N_0^0, M_1^0, N_1^0\})$

$$\begin{aligned} \rightarrow deg(S^a(M_0^{r-1})S^b(M_0^{r-1}) &\leq deg(l_1 l_2) + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \\ &\leq 4 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \end{aligned}$$

Table 2. upper bound of algebraic degree for each branch of Shadow-32,64.

Block size	Upper bound of algebraic degree			
	L_0^r	L_1^r	R_0^r	R_1^r
32	10^*	12^*	10^*	12^*
64	18	20	18	20

* means bound is tight

(c) $deg(S^c(M_0^{r-1}))$
since $S^c(M_0^{r-1}) \in L(P_{c+1 \bmod 2})$,
$deg(S^c(M_0^{r-1})) \leq 2 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1))$
in summary,

$$\begin{aligned} deg(L_1^r) &\leq max(deg(N_0^{r-2}), deg(S^a(M_0^{r-1})S^b(M_0^{r-1})), deg(S^c(M_0^{r-1})) \\ &\leq 4 + deg(S^{\{1,3,5,7\}}(X_0)S^{\{0,2,4,6\}}(X_1)) \end{aligned}$$

■

4.2 Results and Experimental Verification

Shadow-32 We computed each cubesum corresponding to Corollary 1 and checked the validity. Also, We computed the upper bound of algebraic degree according to Corollary 1. The results are given in table 2. Table 3 shows the maximum algebraic degree each round of Shadow-32 computed experimentally. From round 8, the degree stops growing at 10,12, showing that the upper bound of Corollary 2 is tight.

Shadow-64 We also computed the upper bound algebraic degree for Shadow-64. Since $n = 16$ for Shadow-64, Corollary 2 should be modified as follows.

$$deg(L_0^r), deg(R_0^r) \leq 2 + deg(S^{\{1,3,5,7,9,11,13,15\}}(X_0)S^{\{0,2,4,6,8,10,12,14\}}(X_1))$$
$$deg(L_1^r), deg(R_1^r) \leq 4 + deg(S^{\{1,3,5,7,9,11,13,15\}}(X_0)S^{\{0,2,4,6,8,10,12,14\}}(X_1))$$

The result is also given in table 2.

4.3 On the Key Recovery Attack

Through the above analysis, we confirmed that Shadow has an algebraic vulnerability and attempted a key recovery attack using the integral distinguisher of corollary 1 and guessing the first or last round key. However, due to the existence of distinguishers in all rounds, the equality of the corollary holds for all guessed keys, making the key recovery attack impossible. We leave the key recovery attack as a future work.

Table 3. Experimental result of algebraic degree of Shadow-32 for each branch

Round(r)	Maximum algebraic degree			
	L_0^r	L_1^r	R_0^r	R_1^r
1	2	3	2	3
2	3	6	3	6
3	4	8	4	8
4	6	11	6	11
5	7	11	7	11
6	8	12	8	12
7	9	12	9	12
8	10	12	10	12
9	10	12	10	12
10	10	12	10	12
11	10	12	10	12
12	10	12	10	12
13	10	12	10	12

5 Conclusion

In this paper, we define the Shadow structure, a generalization of Shadow, and show that the structure is almost equivalent to the 4-branch Gerenalized Feistel Network. Moreover, we give a low data structural distinguisher that can distinguish Shadow structure from random permutation with only two plaintext/ciphertext pairs. The distinguisher can be extended to key recovery attack with only one plaintext/ciphertext pair by exploiting the key schedule. We also prove that Shadow's AND-RX update function leads to an algebraic weakness that only certain forms of monomials appear. Based on that, we show that there is an integral distinguisher of cube size 2 and that algebraic degree cannot increase beyond 12 for Shadow-32 and 20 for Shadow-64, regardless of the number of rounds. These strong algebraic vulnerabilities are thought to be due to the cancellation of higher-order monomials, and must be considered carefully when designing AND-RX structured ciphers. Since these properties are not easily identified at first glance, we would like to highlight that they might be negatively used as a backdoor for a block cipher.

References

1. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. pp. 411–436. Springer (2015)

2. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: Gift: A small present: Towards reaching the limit of lightweight encryption. In: Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. pp. 321–345. Springer (2017)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck families of lightweight block ciphers. cryptology eprint archive (2013)
4. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The skinny family of block ciphers and its low-latency variant mantis. In: Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. pp. 123–153. Springer (2016)
5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9. pp. 450–466. Springer (2007)
6. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al.: Prince–a low-latency block cipher for pervasive computing applications. In: Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18. pp. 208–225. Springer (2012)
7. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of keccak and luffa. In: Fast Software Encryption: 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers 18. pp. 252–269. Springer (2011)
8. Buchberger, B.: Bruno buchberger's phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. Journal of symbolic computation **41**(3-4), 475–511 (2006)
9. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 392–407. Springer (2000)
10. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28. pp. 278–299. Springer (2009)
11. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The led block cipher. In: Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13. pp. 326–341. Springer (2011)
12. Guo, Y., Li, L., Liu, B.: Shadow: A lightweight block cipher for iot nodes. IEEE Internet of Things Journal **8**(16), 13014–13023 (2021)
13. Hong, D., Lee, J.K., Kim, D.C., Kwon, D., Ryu, K.H., Lee, D.G.: Lea: A 128-bit block cipher for fast encryption on common processors. In: Information Security Applications: 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers 14. pp. 3–27. Springer (2014)
14. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.S., Lee, C., Chang, D., Lee, J., Jeong, K., et al.: Hight: A new block cipher suitable for low-resource

device. In: Cryptographic Hardware and Embedded Systems-CHES 2006: 8th International Workshop, Yokohama, Japan, October 10-13, 2006. Proceedings 8. pp. 46–59. Springer (2006)
15. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings 4. pp. 28–40. Springer (1997)
16. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B.Y., Han, D.G., Seo, H., Kim, S., Hong, S., Sung, J., et al.: Pipo: A lightweight block cipher with efficient higher-order masking software implementations. In: Information Security and Cryptology–ICISC 2020: 23rd International Conference, Seoul, South Korea, December 2–4, 2020, Proceedings 23. pp. 99–122. Springer (2021)
17. Knudsen, L.R.: Truncated and higher order differentials. In: Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings 2. pp. 196–211. Springer (1995)

A Appendix

Table 4. Permutation for generator 1

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

Table 5. Permutation for generator 2

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

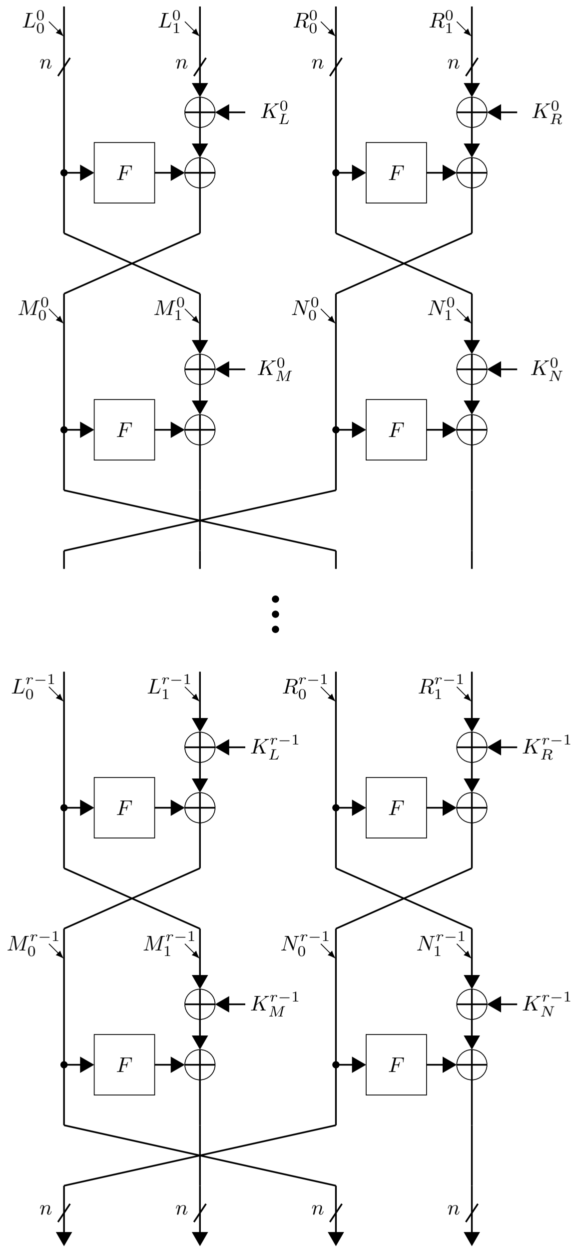

Fig. 1. Shadow Structure

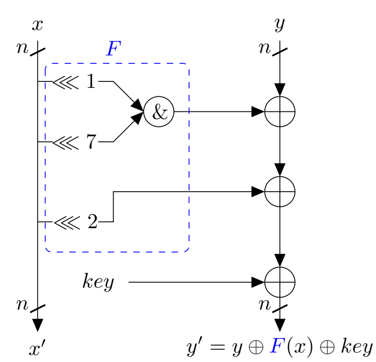

Fig. 2. Shadow Update Function F of Figure 1