Liu *et al. Cybersecurity* (2023) 6:52
https://doi.org/10.1186/s42400-023-00184-7

Cybersecurity

RESEARCH

Open Access

Full-round impossible differential attack on shadow block cipher

Yuting Liu[1,2], Yongqiang Li[1,2*], Huiqin Chen[1,2] and Mingsheng Wang[1,2]

Abstract

Lightweight block ciphers are the essential encryption algorithm for devices with limited resources. Its goal is to ensure the security of data transmission through resource-constrained devices. Impossible differential cryptanalysis is one of the most effective cryptanalysis on block ciphers, and assessing the ability of resisting this attack is a basic design criterion. Shadow is a lightweight block cipher proposed by Guo et al. (IEEE Internet Things J 8(16):13014–13023, 2021). It utilizes a combination of ARX operations and generalized Feistel structure to overcome the weakness of the traditional Feistel structure that only diffuses half in one round. In this paper, we focus on the differential property of Shadow and its security against impossible differential cryptanalysis. First, we use the SAT method to automatically search for a full-round impossible differential distinguisher of Shadow-32. Then, based on the experimental results, we prove that Shadow has a differential property with probability 1 based on the propagation of the state. Further, we can obtain an impossible differential distinguisher for an arbitrary number of rounds of Shadow. Finally, we perform a full key recovery attack on the full-round Shadow-32 and Shadow-64. Both experimentally and theoretically, our results indicate that Shadow is critically flawed, and regardless of the security strength of the internal components and the number of rounds applied, the overall cipher remains vulnerable to impossible differential cryptanalysis.

Keywords Lightweight block cipher, Shadow, Impossible differential cryptanalysis, SAT

Introduction

Along the accelerated development of information technology, the Internet of Things (IoT) technologies such as RFID and wireless sensors are increasingly applied in daily life, and they are often integrated into devices with limited storage and computing resources. However, traditional block ciphers are not suitable for these devices, as their high software and hardware implementation requirements cannot guarantee the security of data transmission. Thus, there is a demand for lightweight block ciphers that can provide high performance and security in resource-constrained environments.

Driven by protecting private data from resource-constrained devices, lightweight block ciphers aim to achieve low resource utilization, low power consumption, high computational efficiency, and maintain the security of block ciphers. In line with this objective, many well-designed lightweight block ciphers have been proposed, such as SEA (Standaert et al. 2006), HIGHT (Hong et al. 2006), PRESENT (Bogdanov et al. 2007), LBlock (Wu and Zhang 2011), SIMON and SPECK (Beaulieu et al. 2015), Midori (Banik et al. 2015) and Shadow (Guo et al. 2021) et al. Moreover, security evaluation for lightweight block ciphers is essential, and a new proposed lightweight block cipher needs to be assessed for its security against traditional cryptanalysis attacks, i.e. differential cryptanalysis (Biham and Shamir 1991), linear cryptanalysis (Matsui

*Correspondence:
Yongqiang Li
liyongqiang@iie.ac.cn
[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

1994), impossible differential cryptanalysis, and other cryptanalysis.

Impossible differential cryptanalysis was first proposed by Knudsen (1998) and Biham et al. (1999) respectively. It is one of the most effective cryptanalysis on block ciphers, and assessing the ability of resisting this cryptanalysis is a basic design criterion. Its basic idea is to exclude wrong keys that lead to zero-probability difference and then recover the correct key by exhausting the candidate keys. In general, impossible differential cryptanalysis contains two phases, i.e., the search for impossible differential distinguisher phase and the key recovery phase. The key to the impossible differential analysis is to search for the longest-round impossible differential distinguisher.

Research on the automated search method has been an important issue for the last 20 years. The first critical tool for automated search is the Mixed Integer Linear Programming (MILP), which was first employed by Mouha et al. (2012) to find the minimum number of active S-boxes for word-oriented block ciphers. Later, Sun et al. (2014) extended the method from word-oriented to bit-oriented, and assessed the ability of bit-oriented block ciphers to resist the (related-key) differential attack. Since then, the MILP has been widely used for the cryptanalysis of block ciphers. Cui et al. (2016) and Sasaki and Todo (2017) applied the MILP to impossible differential automatic search, respectively. In 2017, Abdelkhalek et al. (2017) applied the MILP to block ciphers with 8-bit S-boxes. In recent years, the MILP has remained a popular tool for automated search for differential distinguishers (Zhu et al. 2019; Kumar and Yadav 2022; Kaur et al. 2023).

Another important tool for automated search is to rely on the Boolean Satisfiability Problem or satisfiability modulo theories (SAT/SMT). In 2012, Mouha et al. (2012) first used the SAT/SMT method to automatically seek optimal differential characteristics of Salsa20. Later in 2015, Kölbl et al. (2015) employed the SAT/SMT method to automatically search for optimal differential and linear characteristics of SIMON. In 2017, Sun et al. (2017) automatically search for bit-based integral distinguishers of ARX block ciphers based on the SAT/SMT method. In 2020, Hu et al. (2020) moved away from focusing on the propagation of the difference and proposed an SAT/SMT-aided search method for impossible differential that used the propagation of the state. Later in 2021, Sun et al. (2021) focused on the acceleration of using the SAT/SMT methods to seek differential and linear characteristics. In 2023, Sun and Wang (2023) developed SAT/SMT models to search for differential and linear characteristics of block ciphers with large S-boxes.

Shadow, a lightweight block cipher, is proposed by Guo et al. (2021) to protect private data transmission through IoT nodes. Shadow utilizes a combination of ARX operations and a generalized Feistel structure, which resolves the issue of the current lightweight block ciphers based on ARX operations that only diffuse half in one round. The security of Shadow was first evaluated by the designers. They performed impossible differential cryptanalysis and biclique cryptanalysis on Shadow, where the impossible differential attack mainly utilizes a 4-round impossible differential distinguisher to perform a 7-round key recovery attack and the biclique attack constructs an 8-round biclique structure. Consequently, the designers of Shadow asserted that Shadow exhibits a high level of resistance against cryptanalysis. In this paper, we show that Shadow can not resist impossible differential cryptanalysis, and we identify significant security weaknesses in the current design of Shadow.

Our contributions In this paper, we focus on the differential property of Shadow and its security against impossible differential cryptanalysis. For the first time, we perform an impossible differential attack on the full-round Shadow-32 and Shadow-64. Our results indicate that Shadow is critically flawed, and regardless of the security strength of the internal components and the number of rounds applied, the overall cipher remains vulnerable to impossible differential cryptanalysis. The specific results are displayed in Table 1. Our contributions can be concluded as follows.

- We use the SAT method to find a full-round impossible differential distinguisher of Shadow-32.
- We prove that Shadow has a differential property with probability 1 based on the propagation of state proposed by Hu et al. (2020), and then present an impossible differential distinguisher for an arbitrary number of rounds.
- We perform full key recovery on full-round Shadow-32 with 2^{30} data complexity, 2^{48} 16-round encryption time complexity and 2^{43} 32-bit block memory complexity.
- We perform full key recovery on full-round Shadow-64 with 2^{61} data complexity, 2^{96} 32-round encryption time complexity and 2^{90} 64-bit block memory complexity.

Table 1 Analytical results of Shadow-32/64

Cipher	Attacked rounds (full round)	Data complexity	Time complexity	Memory complexity
Shadow-32	16	2^{30}	2^{48}	2^{43}
Shadow-64	32	2^{61}	2^{96}	2^{90}

Organization The subsequent sections of this paper are arranged as follows. Section "Preliminaries" describes the background knowledge used in this paper. Section "Automatic search for impossible differential distinguisher" shows how to automatically search for impossible differential distinguishers using the SAT method. Section "A proof of impossible differential distinguishers for an arbitrary number of rounds" proves the differential property with probability 1 of Shadow-32/64 based on the propagation of state. Full key recovery attack on the full-round Shadow-32 and Shadow-64 are mounted in section "Key recovery attack on full-round Shadow-32/64". Finally, section "Summary" summarizes the paper.

Preliminaries

Notation

In this subsection, we first present the following notations that are utilized throughout the paper.

- L_0^{i-1}: The input state for the first branch on the left of ith round;
- L_1^{i-1}: The input state for the second branch on the left of ith round;
- R_0^{i-1}: The input state for the first branch on the right of ith round;
- R_1^{i-1}: The input state for the second branch on the right of ith round;
- $\triangle L_0^{i-1}$:The input difference for the first branch on the left of ith round;
- $\triangle L_1^{i-1}$: The input difference for the second branch on the left of ith round;
- $\triangle R_0^{i-1}$: The input difference for the first branch on the right of ith round;
- $\triangle R_1^{i-1}$: The input difference for the second branch on the right of ith round;
- x^{i-1}: The input state of ith round;
- $\hat{x}^{i-1}$: The another input state of ith round;
- $\triangle x^{i-1}$: The input difference of ith round;
- $\mathbb{F}_2$: The binary field;
- key^i: The ith round subkey;
- RN: full round;
- r: iterative rounds;
- m: block size of the cipher;
- &: bitwise AND;
- $\oplus$: XOR;
- $\lll n$: rotation to the left by n bits;

Description of Shadow

Shadow utilizes a combination of ARX operations and a generalized Feistel structure, which includes two versions: Shadow-32 and Shadow-64. The block sizes of Shadow-32 and Shadow-64 are 32 and 64 bits, respectively, with key sizes of 64 and 128 bits and round numbers of 16 and 32, respectively.

Encryption algorithm

Shadow comprises three main operations: AND, Rotation, and XOR. Let $(L_0^{i-1}, L_1^{i-1}, R_0^{i-1}, R_1^{i-1})$ be the input state of the ith round function, $(L_0^i, L_1^i, R_0^i, R_1^i)$ be the corresponding output state, and $key_j^i (0 \le j \le 3)$ is selected from round subkey key^i. The round function of Shadow is depicted in Fig. 1.

From Fig. 1, the round function of Shadow calls f four times, where f is

$$f(x) = ((x_{\lll 1}) \& (x_{\lll 7})) \oplus (x_{\lll 1}),$$

and this operation reduces logic hardware and software consumption.

For the RN-round encryption process of Shadow, the plaintext $P = (L_0^0, L_1^0, R_0^0, R_1^0)$ is divided into four equal-sized blocks. First, the first branch on the left L_0^0 calls the f function and then performs the XOR operation with the second branch on the left L_1^0 and the subkey key_0^1 to get $P0$, i.e. $P0 = f(L_0^0) \oplus L_1^0 \oplus key_0^1$. Similarly, the first branch on the right R_0^0 performs the same operation with the second branch on the right R_1^0 to get $P1$, i.e. $P1 = f(R_0^0) \oplus R_1^0 \oplus key_1^1$. The half-round output $(P0, L_0^0, P1, R_0^0)$ is obtained by swapping the left and right branches separately. Next, the $P0$ calls the f function and then performs the XOR operation with the L_0^0 and the subkey key_2^1 to get the L_1^1, i.e. $L_1^1 = f(P0) \oplus L_0^0 \oplus key_2^1$.

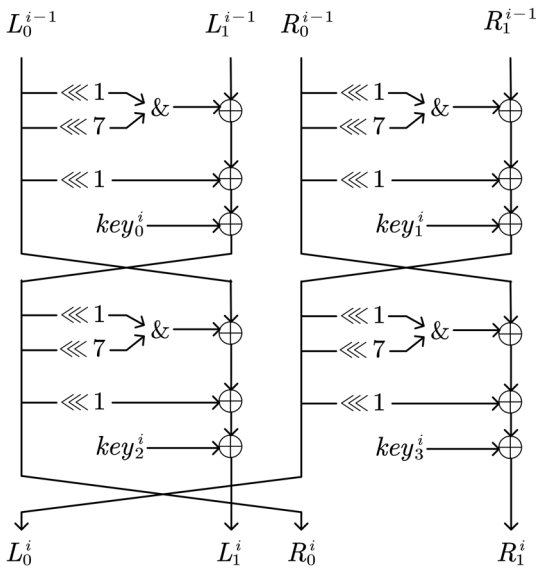

Fig. 1 The ith round function of Shadow

Similarly, the $P1$ performs the same operation with the R_0^0 to get the R_1^1, i.e. $R_1^1 = f(P1) \oplus R_0^0 \oplus key_3^1$. After the data exchange, the first round output is $(P1, L_1^1, P0, R_1^1)$. Repeat the above operation for RN rounds to generate the ciphertext $C = (L_0^{RN}, L_1^{RN}, R_0^{RN}, R_1^{RN})$. Notice that there is no data exchange in the last round. The corresponding encryption algorithm is exhibited in Algorithm 1. Since Shadow uses a generalized Feistel structure, the decryption algorithm only needs to use the round subkey in reverse order compared to the encryption algorithm.

Algorithm 1 Encryption Algorithm of Shadow

Input: Plaintext, key;

Output: Ciphertext;

$(L_0^0, L_1^0, R_0^0, R_1^0) \leftarrow$ Plaintext;

for $i = 1$ to RN **do**

$P0 = ((L_{0(\ll 1)}^{i-1} \& L_{0(\ll 7)}^{i-1})) \oplus L_1^{i-1} \oplus L_{0(\ll 2)}^{i-1} \oplus key_0^i$;

$P1 = ((R_{0(\ll 1)}^{i-1} \& R_{0(\ll 7)}^{i-1})) \oplus R_1^{i-1} \oplus R_{0(\ll 2)}^{i-1} \oplus key_1^i$;

$L_0^i = P1$;

$L_1^i = ((P0_{(\ll 1)} \& P0_{(\ll 7)})) \oplus L_0^{i-1} \oplus P0_{(\ll 2)} \oplus key_2^i$;

$R_0^i = P0$;

$R_1^i = ((P1_{(\ll 1)} \& P1_{(\ll 7)})) \oplus R_0^{i-1} \oplus P1_{(\ll 2)} \oplus key_3^i$;

end for

Ciphertext $\leftarrow (L_0^{RN}, L_1^{RN}, R_0^{RN}, R_1^{RN})$;

return Ciphertext;

Key schedule

Depending on the block size of Shadow, there are two kinds of round subkey generators, i.e. Generator1 and Generator2. For Shadow-32, the 64-bit primary key K is described as $k_0||k_1||k_2||\ldots||k_{62}||k_{63}$ and enters the Generator1. The Generator1 contains three operations, i.e. AddRoundConstant, NX Module, and Permutation. Firstly, the AddRoundConstant operation is performed on the 5-bit key $k_3||k_4||k_5||k_6||k_7$, followed by the NX Module on the 8-bit key $k_{56}||k_{57}||k_{58}||\ldots||k_{62}||k_{63}$, and finally the Permutation on the 64-bit key. Subsequently, the subkey K' of the first round is obtained, where the front 32-bits of K' are partitioned into four equal-sized segments for the round key XOR operation. The K' is then input to Generator1 to generate the subkeys for each round until the encryption is completed. The specific operation procedure of Generator1 is depicted in Fig. 2.

AddRoundConstant The round constant r is first expanded into its binary representation $c_0||c_1||c_2||c_3||c_4$, after which the 5-bit key $k_3||k_4||k_5||k_6||k_7$ is XORed with the 5-bit value $c_0||c_1||c_2||c_3||c_4$.

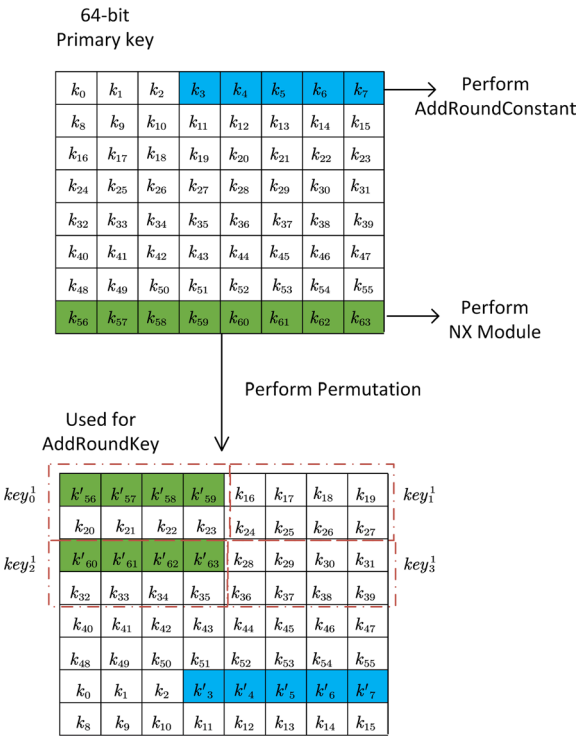

Fig. 2 The detailed operation procedure of Generator1

NX Module The only non-linear operation in Generator1 is the NX Module. For Shadow-32, the 8-bit key $k_{56}||k_{57}||\ldots||k_{62}||k_{63}$ executes NX Module. The NX Module operates based on the following principle:

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

Permutation After the AddRoundConstant and NX Module operations are executed in Generator1, the Permutation is implemented for the 64-bit key. As shown in Table 2, p_i denotes the position index before the Permutation, while p'_i denotes the position index after the Permutation.

For Shadow-64, the 128-bit primary key K is described as $k_0||k_1||k_2||\ldots||k_{126}||k_{127}$ and enters the Generator2. The Generator2 also contains three operations, i.e. AddRoundConstant, NX Module, and Permutation. In Generator2, the round constant r is first expanded into its binary representation $c_0||c_1||c_2||c_3||\ c_4||c_5$, which is XORed with the 6-bit key $k_2||k_3||k_4||k_5||\ k_6||k_7$. Subsequently, the NX Module is applied to the 24-bit key

Table 2 Permutation of Shadow-32

p_i	p'_i	p_i	p'_i	p_i	p'_i	p_i	p'_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

$k_{104}||k_{105}||\ldots||k_{126}||k_{127}$, followed by the Permutation operation on the 128-bit key. Finally, the subkey K' of the first round is obtained, where the front 64-bits of K' are partitioned into four equal-sized segments for the round key XOR operation. The K' is then input to Generator2 to derive the round keys until the encryption is completed. Additionally, the principle of Generator2 is similar to that of Generator1, but with different numbers of bits. The detailed operation procedure of Permutation is depicted in Table 3 and the NX Module operates based on the following principle:

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

Table 3 Permutation of Shadow-64

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

Impossible differential cryptanalysis

Impossible differential cryptanalysis is a variant of differential cryptanalysis, which was proposed by Knudsen (1998) and Biham et al. (1999) respectively. Impossible differential cryptanalysis, as opposed to classical differential cryptanalysis which utilizes a high probability differential characteristic, utilizes a zero-probability differential characteristic to recover keys. Its basic idea is to exclude wrong keys that lead to zero-probability difference and then recover the correct key by exhausting the candidate keys. Impossible differential cryptanalysis comprises two phases: the phase of searching for impossible differential distinguishers, and the phase of key recovery. The key to the impossible differential analysis is to search for the longest-round impossible differential distinguisher, as a higher number of rounds indicates a weaker resistance against the impossible differential attacks. The traditional search impossible differential distinguisher is to describe the propagation of difference in block ciphers, but the propagation of difference through non-linear components is uncertain, making it impossible to consider the details of non-linear components and the key schedule.

Definition 1 (*Block Cipher*) Let $\mathbb{F}_2$ be the binary field, and $\mathbb{F}_2^m$ and $\mathbb{F}_2^t$ be m-dimensional and t-dimensional vector space over the finite field $\mathbb{F}_2$, respectively. If the plaintext $P \in \mathbb{F}_2^m$, the ciphertext $C \in \mathbb{F}_2^m$, and the master key $K \in \mathbb{F}_2^t$, then the iterative block cipher E_K^m with $\mathbb{F}_2^m$ as the plaintext space (ciphertext space) and $\mathbb{F}_2^t$ as the key space is

$$E_K^m : \mathbb{F}_2^m \times \mathbb{F}_2^t \mapsto \mathbb{F}_2^m.$$

Definition 2 (*Impossible Differential Distinguisher*) For an iterative block cipher E_K^m, let $\alpha \in \mathbb{F}_2^m$ be the input difference and $\beta \in \mathbb{F}_2^m$ be the r-round output difference, if differential propagation probability $Pr(\alpha \to \beta) = 0$, then $\alpha \nrightarrow \beta$ is a r-round impossible differential distinguisher.

Since the input difference can be obtained by XOR of two input states, Hu et al. (2020) characterize the propagation of difference by describing the propagation of two sets of initial states. That is, given two input states (x_0^0, x_1^0), perform r-round encryption and obtain two groups of state propagation traces, i.e. $(x_0^0, x_0^1, x_0^2, \ldots, x_0^r)$ and $(x_1^0, x_1^1, x_1^2, \ldots, x_1^r)$, then by $x_0^i \oplus x_1^i (0 \le i \le r)$ we can get the input difference and the output difference for each round, i.e. differential characteristic $(\triangle x^0, \triangle x^1, \ldots, \triangle x^r)$. Compared to the traditional impossible differential analysis, the impossible differential analysis based on the propagation of state not only takes into account the details of non-linear components but also allows to consider the impact of the key schedule.

Definition 3 (*Impossible Differential Distinguisher Based on the Propagation of State*) For an iterative block cipher E_K^m, if $\forall (x_0, x_1) \in \{(a_0, a_1) \in \mathbb{F}_2^m \times \mathbb{F}_2^m | a_0 \oplus a_1 = \alpha\}$ and α is the input difference, $\forall (y_0, y_1) \in \{(b_0, b_1) \in \mathbb{F}_2^m \times \mathbb{F}_2^m | b_0 \oplus b_1 = \beta\}$ and β is the output difference, and $E_K^r(x_0) \oplus E_K^r(x_1) \ne y_0 \oplus y_1$, i.e. differential propagation probability $Pr(\alpha \to \beta) = 0$, then $\alpha \nrightarrow \beta$ is a r-round impossible differential distinguisher based on the propagation of state.

The following is to use an obtained $(r-1)$-round impossible differential distinguisher to recover the r-round key.

- Find a $(r-1)$-round impossible differential distinguisher $\alpha \to \beta$;
- Select plaintext pairs $(P, \hat{P})$ with $P \oplus \hat{P} = \alpha$, then perform r-round encryption and get the ciphertext pairs $(C, \hat{C})$;
- Guess possible values of the r-round key k_r. For each possible value of k_r, decrypt ciphertext C and $\hat{C}$ one round forward and obtain $(D, \hat{D})$. Judge if $D \oplus \hat{D} = \beta$ holds, if holds, then the guessed key is wrong;
- Repeat the above steps until the only correct key remains.

Assuming that $|K|$ bit keys can be obtained by the above attack, each plaintext pair can eliminate 2^{-t} of the key information. To ensure that the correct key is uniquely determined, the required plaintext pairs N must satisfy

$$(2^{|K|} - 1) \times (1 - 2^{-t})^N < 1,$$

When t is relatively large, it gives

$$N > 2^t \times \ln 2 \times |K| \approx 2^{t-0.53}|K|.$$

The above equation shows that, when performing the impossible differential attack, the data complexity is almost independent of the amount of guessed key bits, and the main effect is the key information that can be eliminated for each plaintext pair.

SAT problem

The Boolean Satisfiability Problem (SAT) is a foundational computational problem in the fields of computer science and mathematical logic. It involves determining

whether a given boolean formula, composed of boolean variables and logical operators such as AND, OR, and NOT, can be assigned truth values that satisfy the formula. STP is the publicly accessible solver for the SAT problem. Its input is expected to be a file with the ".stp" extension, adhering to the CVC language format.

When solving an SAT problem, the first step is to construct a model using the CVC language and save it as a file with the ".stp" extension. Subsequently, the STP solver is invoked for this file. If the STP returns "Valid.", it indicates that the target problem has no solution. Otherwise, it returns a solution of the target problem and "Invalid.". For more details about the STP solver and the CVC language, please refer to https://stp.github.io/.

The following are the CVC terms used in this paper:

1 *ASSERT*(): The command statement.
2 $BITVECTOR(n)$: Declare variables as n bits.
3 $t_1@t_2@\ldots@t_m$: The connection operation.
4 $t_1\,\&t_2\,\&\cdots\&t_m$: The bitwise AND operation.
5 $BVXOR(t_1, t_2)$: The bitwise XOR operation.

Automatic search for impossible differential distinguisher

In this section, we use the SAT method to automatically search for impossible differential distinguishers based on the propagation of the state, and find a full-round impossible differential distinguisher of Shadow-32.

Bit-oriented SAT model based on the propagation of the state

In this subsection, we will demonstrate the process of constructing the SAT model for searching impossible differential distinguishers based on the propagation of the state.

According to Definition 3, the modeling process is composed of two steps, the first step is to describe the propagation of the two sets of states under r rounds of iterations, and the second step is to obtain the input difference and the $r-$round output difference by XORing the two sets of states and assign the given values. For the first step, the core is to model the propagation of the state under basic operations. Since Shadow utilizes ARX operations and is bit-oriented, we will use the CVC language to generate statements for the propagation of the state under the operations bit-oriented COPY, bit-oriented AND, bit-oriented Rotation, and bit-oriented XOR. For the second step, we will use the CVC language to generate statements for the computation of the difference and the constraints on the difference.

Model 1 (*COPY*) Let F be a COPY function, where the input state is $x \in \mathbb{F}_2^q$ and the output $y^0, y^1, \ldots, y^{t-1} \in \mathbb{F}_2^q$ is caculated as $(y^0, y^1, \ldots, y^{t-1}) = (x, x, \ldots, x)$. The bit vector format is $x = (x_0, \ldots, x_{q-1}), y^i = (y_0^i, y_1^i, \ldots, y_{q-1}^i)$, where $x_j, y_j^i \in \mathbb{F}_2$, $0 \le j \le q-1$ and $0 \le i \le t-1$. Then, the modeling of the propagation of the state under the COPY operation is described in CVC format as

$$\begin{cases} ASSERT(y_0^0@\ldots@y_{q-1}^0 = x_0@\ldots@x_{q-1}); \\ ASSERT(y_0^1@\ldots@y_{q-1}^1 = x_0@\ldots@x_{q-1}); \\ \quad\vdots \\ ASSERT(y_0^{t-1}@\ldots@y_{q-1}^{t-1} = x_0@\ldots@x_{q-1}); \end{cases}$$

The COPY operation is usually omitted in practical modeling because the value of the state remains unchanged after the COPY operation.

Model 2 (*XOR*) Let F be an XOR function, where the two input states are $x, y \in \mathbb{F}_2^q$ and the output $z \in \mathbb{F}_2^q$ is calculated as $z = x \oplus y$. The bit vector format is $x = (x_0, x_1, \ldots, x_{q-1}), y = (y_0, y_1, \ldots, y_{q-1})$, and $z = (z_0, z_1, \ldots, z_{q-1})$, where $x_j, y_j, z_j \in \mathbb{F}_2$ and $0 \le j \le q-1$. Then, the modeling of the propagation of the state under the XOR operation is described in CVC format as

$$\begin{aligned} &ASSERT(z_0@\ldots@z_{q-1} = BVXOR \\ &\qquad (x_0@\ldots@x_{q-1}, y_0@\ldots@y_{q-1})); \end{aligned}$$

Model 3 (*AND*) Let F be a AND function, where the two input states are $x, y \in \mathbb{F}_2^q$ and the output $z \in \mathbb{F}_2^q$ is calculated as $z = x \ \&\ y$. The bit vector format is $x = (x_0, x_1, \ldots, x_{q-1}), y = (y_0, y_1, \ldots, y_{q-1})$, and $z = (z_0, z_1, \ldots, z_{q-1})$, where $x_j, y_j, z_j \in \mathbb{F}_2$ and $0 \le j \le q-1$. Then, the modeling of the propagation of the state under the AND operation is described in CVC format as

$$\begin{aligned} &ASSERT(z_0@\ldots@z_{q-1} = (x_0@\ldots@x_{q-1}) \\ &\qquad\qquad\qquad \&(y_0@\ldots@y_{q-1})); \end{aligned}$$

Model 4 (*Rotation*) Let F be a Rotation function, where the input state is $x \in \mathbb{F}_2^q$ and the output $y \in \mathbb{F}_2^q$ is calculated as $y = x \lll n$, where n is a constant. The bit vector format is $x = (x_0, x_1, \ldots, x_{q-1})$ and $y = (y_0, y_1, \ldots, y_{q-1})$, where $x_j, y_j \in \mathbb{F}_2$ and $0 \le j \le q-1$. Then, the modeling

of the propagation of the state under the Rotation operation is described in CVC format as

$$ASSERT(y_0@\ldots@y_{q-1} = (x_n@x_{n+1}@\ldots@x_{q-1} @x_0@\ldots@x_{n-1}));$$

Computation of difference Let $(x^0, \hat{x}^0) \in \mathbb{F}_2^q \times \mathbb{F}_2^q$ be the two sets of initial states, and after r rounds of propagation, the r-round states $(x^r, \hat{x}^r)$ is obtained. By $\triangle x^0 = x^0 \oplus \hat{x}^0$ and $\triangle x^r = x^r \oplus \hat{x}^r$ we can obtain the input difference $\triangle x^0$ and the $r-$round output difference $\triangle x^r$. The bit vector format is $A = (A_0, A_1, \ldots, A_{q-1})$, where $A = x^0, \hat{x}^0, x^r, \hat{x}^r, \triangle x^0, \triangle x^r$. Then, the computation of the input difference and the $r-$round output difference can be described in CVC format as

$$\begin{aligned} &ASSERT(\triangle x_0^0@\ldots@\triangle x_{q-1}^0 \\ &\quad = BVXOR(x_0^0@\ldots@x_{q-1}^0, \hat{x}_0^0@\ldots@\hat{x}_{q-1}^0)); \\ &ASSERT(\triangle x_0^r@\ldots@\triangle x_{q-1}^r \\ &\quad = BVXOR(x_0^r@\ldots@x_{q-1}^r, \hat{x}_0^r@\ldots@\hat{x}_{q-1}^r)); \end{aligned}$$

Constraints on difference Let the input difference $\triangle x^0 = \alpha$ and the $r-$round output difference $\triangle x^r = \beta$, where $\alpha, \beta \in \mathbb{F}_2^q$ are the given value. The bit vector format is $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{q-1})$ and $\beta = (\beta_0, \beta_1, \ldots, \beta_{q-1})$, where $\alpha_j, \beta_j \in \mathbb{F}_2$ and $0 \le j \le q-1$. Then, the constraints on the input difference and the output difference can be described in CVC format as

$$\begin{aligned} &ASSERT(\triangle x_0^0@\ldots@\triangle x_{q-1}^0 = \alpha_0@\ldots@\alpha_{q-1}); \\ &ASSERT(\triangle x_0^r@\ldots@\triangle x_{q-1}^r = \beta_0@\ldots@\beta_{q-1}); \end{aligned}$$

The search algorithm for impossible differential distinguisher of shadow

In this subsection, we will show how to automatically seek impossible differential distinguishers. The automated search method consists of two phases: statements generation phase and impossible differential distinguishers search phase. For the statements generation phase, Algorithm 2 automatically generates statements describing the input difference $\triangle x^0$ propagate to the $r-$round output difference $\triangle x^r$ with $\triangle x^0 = \alpha$ and $\triangle x^r = \beta$, and saves these statements as a file. For the impossible differential distinguishers search phase, Algorithm 3 invokes the STP to solve the file generated by Algorithm 2 to determine whether there is an impossible differential distinguisher by traversing sets of input differences and output differences satisfying certain conditions.

Algorithm 2 Generate statements in CVC language

Input: the block size of Shadow m, the number of rounds r, the input difference α and the output difference β;

Output: A file that has a system of statements in CVC format;

1: Declare the two sets of state variables x and $\hat{x}$;
2: Declare the intermediate variables;
3: Declare the input difference variables $\triangle x^0$ and the output difference variables $\triangle x^r$;
4: **for** $i = 0$ to r **do**
5: Model the i-round propagation of x^i to x^{i+1};
6: Model the i-round propagation of $\hat{x}^i$ to $\hat{x}^{i+1}$;
7: **end for**
8: Generate statements about the computation of the input difference $\triangle x^0$ and the output difference $\triangle x^r$;
9: Generate the constraints on the input difference $\triangle x^0 = \alpha$ and the output difference $\triangle x^r = \beta$;
10: Add the statement "QUERY(FALSE);" ;
11: Add the statement "COUNTEREXAMPLE;";

Algorithm 3 Search impossible differential distinguishers

Input: The input difference set Id and the output difference set Od;

Output: An impossible differential distinguisher or no found;

1: flag=false;
2: **for** α in Id **do**
3: **for** β in Od **do**
4: file=Generate(α,β);
5: res=STPSolver(file);
6: **if** res==Valid. **then**
7: **return** (α,β);
8: flag=true;
9: break;
10: **end if**
11: **end for**
12: **if** flag==true **then**
13: break;
14: **end if**
15: **end for**
16: **if** flag==false **then**
17: **return** no found;
18: **end if**

For Shadow, Algorithm 3 gives the overall framework for searching impossible differential distinguishers, i.e., the Main function; Algorithm 2 models the propagation of a given input difference to a given output difference, i.e., the Generate function. The inputs to Algorithm 3 are the input difference set Id and the output difference set Od, where $Id = \{\alpha \in \mathbb{F}_2^m | wt(\alpha) = 1\}$ and $Od = \{\beta \in \mathbb{F}_2^m | wt(\beta) = 1\}$, i.e., Id and Od are the sets of all input and output difference of weight 1. For each $\alpha \in Id$ and $\beta \in Od$, Algorithm 3 first invokes Algorithm 2 to generate the file describing the propagation of α to β, then invokes the STP to solve the file, if it returns "Valid.", then (α, β) is an impossible difference distinguisher and terminates the algorithm, otherwise, continues to traverse the Id and Od.

We present some specific explanations about Algorithm 2 as follows.

- Line 1–3. Here $x = (x^0, \ldots, x^r)$ and $\hat{x} = (\hat{x}^0, \ldots, \hat{x}^r)$, where $x^i, \hat{x}^i \in \mathbb{F}_2^m$. Declare the state variables x_j^i and $\hat{x}_j^i$ as 1 bit, where $0 \le i \le r$ and $0 \le j \le m-1$. Declare the intermediate variables and the difference variables as 1 bit.
- Line 4–7. Using the provided propagation rules for each operation, model the propagation of x^0 to x^r and $\hat{x}^0$ to $\hat{x}^r$ by incorporating intermediate variables.
- Line 8–9. Based on the modeling for the computation of difference and the constraints on difference, generate the corresponding statements.
- Line 10–11. The statement "QUERY(FALSE);" and the statement "COUNTEREXAMPLE;" need to be added at the ending of the file because these two statements are essential in solving an SAT problem using STP. By adding the two statements, if the STP returns "Valid.", it means the SAT problem has no solution, otherwise, it returns a solution and "Invalid.".

Experimental results In practice, we implemented Algorithms 3 and 2 using Python 3.8. and Cryptominisat. Finally, it took us approximately 41 hours to find a full-round impossible differential distinguisher of Shadow-32, i.e. $(0x80000000) \nrightarrow (0x40000000)$. The impossible differential distinguisher from the 1th to the 16th round and the time consumption is shown in Table 4. All the experiments are implemented on this platform: Intel(R) Xeon(R) CPU E5-2650 v4 @2.20GHz×48, 503.8G RAM, 64-bit Ubuntu 20.04.6 LTS with 4 threads. Conveniently, all the source codes are accessible at https://github.com/VanyaW/myproject.

Table 4 The impossible differential distinguisher of Shadow-32 from the 1th to the 16th round and the time consumption

Rounds	The input difference	The output difference	Time (s)
1	0x80000000	0x80000000	0.054
2	0x80000000	0x80000000	0.081
3	0x80000000	0x80000000	0.217
4	0x80000000	0x40000000	1.009
5	0x80000000	0x80000000	1.134
6	0x80000000	0x40000000	6.556
7	0x80000000	0x80000000	8.988
8	0x80000000	0x40000000	52.288
9	0x80000000	0x80000000	73.417
10	0x80000000	0x40000000	362.931
11	0x80000000	0x80000000	579.507
12	0x80000000	0x40000000	2836.981
13	0x80000000	0x80000000	4084.869
14	0x80000000	0x40000000	22812.789
15	0x80000000	0x80000000	28649.212
16	0x80000000	0x40000000	150813.247

From the experimental results, we find that Shadow may have an impossible differential distinguisher for an arbitrary number of rounds, which will be proved theoretically in the next section. Since the method would be limited by the block size and the number of rounds, we have not conducted experiments on Shadow-64 under the limited time and resources, but the next section proves theoretically the existence of an impossible differential distinguisher for an arbitrary number of rounds of Shadow-64.

A proof of impossible differential distinguishers for an arbitrary number of rounds

In this section, we will prove that Shadow has a differential property with probability 1 based on the propagation of state, then we can get an impossible differential distinguisher for an arbitrary number of rounds of Shadow.

Theorem 1 *For r-round Shadow, if for any input state* $(L_0^0, L_1^0, R_0^0, R_1^0) \in F_2^m$ *and* $(\hat{L}_0^0, \hat{L}_1^0, \hat{R}_0^0, \hat{R}_1^0) \in F_2^m$ *with the input difference* $(\triangle L_0^0, \triangle L_1^0, \triangle R_0^0, \triangle R_1^0)$*, after encrypting r rounds for the two sets of states, the corresponding output difference is* $(\triangle L_0^r, \triangle L_1^r, \triangle R_0^r, \triangle R_1^r)$*, then we have*

$$\triangle L_0^r \oplus \triangle R_0^r = \begin{cases} \triangle L_0^0 \oplus \triangle R_0^0 & r = 2n (n \in \mathbb{N}^*), \\ \triangle L_0^1 \oplus \triangle R_0^1 & r = 2n+1 (n \in \mathbb{N}^*). \end{cases}$$

Proof To analyze the overall structure of Shadow more intuitively, we simplify to Fig. 3, which depicts any two consecutive rounds in the *RN*-round Shadow encryption process. The red line of Fig. 3 represents a differential relationship as shown in Eq. (3), and the green line of Fig. 3 represents another differential relationship as shown in Eq. (4). Let the two input state for the $(i-1)$th round be

$$(L_0^{i-2}, L_1^{i-2}, R_0^{i-2}, R_1^{i-2}),$$
$$(\hat{L}_0^{i-2}, \hat{L}_1^{i-2}, \hat{R}_0^{i-2}, \hat{R}_1^{i-2}).$$

Then we have the $(i-1)$th round input difference

$$\begin{aligned}&(\triangle L_0^{i-2}, \triangle L_1^{i-2}, \triangle R_0^{i-2}, \triangle R_1^{i-2})\\ &= (L_0^{i-2} \oplus \hat{L}_0^{i-2}, L_1^{i-2} \oplus \hat{L}_1^{i-2},\\ &\quad R_0^{i-2} \oplus \hat{R}_0^{i-2}, R_1^{i-2} \oplus \hat{R}_1^{i-2}).\end{aligned}$$

Correspondingly, the output difference of the $(i-1)$th round is

$$\begin{aligned}&(\triangle L_0^{i-1}, \triangle L_1^{i-1}, \triangle R_0^{i-1}, \triangle R_1^{i-1})\\ &= (L_0^{i-1} \oplus \hat{L}_0^{i-1}, L_1^{i-1} \oplus \hat{L}_1^{i-1},\\ &\quad R_0^{i-1} \oplus \hat{R}_0^{i-1}, R_1^{i-1} \oplus \hat{R}_1^{i-1}),\end{aligned}$$

and the output difference of the ith round is

$$\begin{aligned}&(\triangle L_0^{i}, \triangle L_1^{i}, \triangle R_0^{i}, \triangle R_1^{i})\\ &= (L_0^{i} \oplus \hat{L}_0^{i}, L_1^{i} \oplus \hat{L}_1^{i}, R_0^{i} \oplus \hat{R}_0^{i}, R_1^{i} \oplus \hat{R}_1^{i}).\end{aligned}$$

From the red line of Fig. 3, we have

$$L_0^i = R_1^{i-1} \oplus F(R_0^{i-1}) \oplus key_1^i, \tag{1}$$

and

$$F(R_0^{i-1}) = L_1^{i-1} \oplus L_0^{i-2} \oplus key_2^{i-1}. \tag{2}$$

Then combine (1) and (2), we get

$$L_0^i = R_1^{i-1} \oplus L_1^{i-1} \oplus L_0^{i-2} \oplus key_2^{i-1} \oplus key_1^i.$$

Similarly, another input state holds

$$\hat{L}_0^i = \hat{R}_1^{i-1} \oplus \hat{L}_1^{i-1} \oplus \hat{L}_0^{i-2} \oplus key_2^{i-1} \oplus key_1^i.$$

Thus, the difference $\triangle L_0^i$ satisfies

$$\triangle L_0^i = L_0^i \oplus \hat{L}_0^i = \triangle R_1^{i-1} \oplus \triangle L_1^{i-1} \oplus \triangle L_0^{i-2}. \tag{3}$$

Enlightenedly, from the green line of Fig. 3, we find

$$\begin{aligned}&R_0^i = L_1^{i-1} \oplus F(L_0^{i-1}) \oplus key_0^i,\\ &F(L_0^{i-1}) = R_1^{i-1} \oplus R_0^{i-2} \oplus key_3^{i-1},\\ &R_0^i = L_1^{i-1} \oplus R_1^{i-1} \oplus R_0^{i-2} \oplus key_3^{i-1} \oplus key_0^i.\end{aligned}$$

Similarly, another input state holds

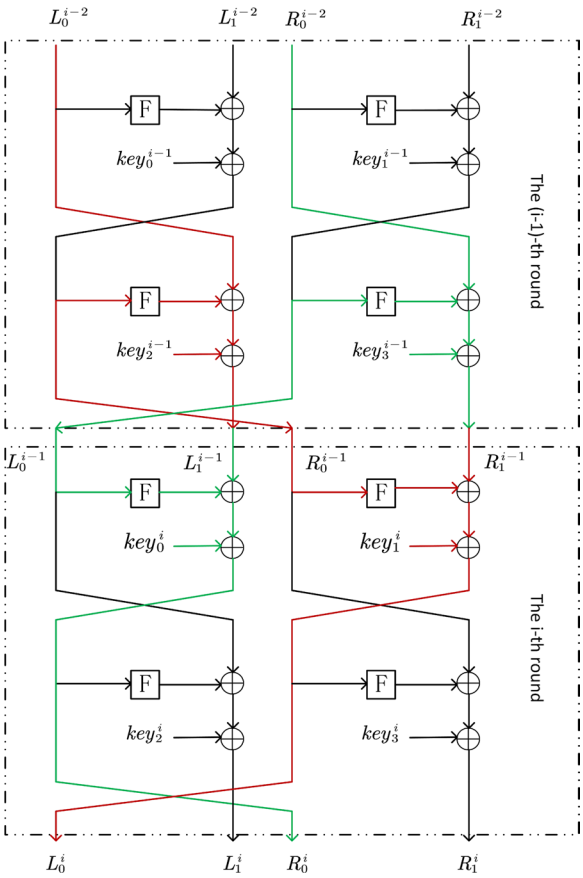

Fig. 3 Any two consecutive rounds of Shadow

$$\hat{R}_0^i = \hat{L}_1^{i-1} \oplus \hat{R}_1^{i-1} \oplus \hat{R}_0^{i-2} \oplus key_3^{i-1} \oplus key_0^i.$$

Thus, the difference $\triangle R_0^i$ satisfies

$$\triangle R_0^i = R_0^i \oplus \hat{R}_0^i = \triangle L_1^{i-1} \oplus \triangle R_1^{i-1} \oplus \triangle R_0^{i-2}. \tag{4}$$

Finally, let (3) xor (4), we obtain

$$\triangle L_0^i \oplus \triangle R_0^i = \triangle L_0^{i-2} \oplus \triangle R_0^{i-2}.$$

As the number of rounds i is arbitrary and $i \geq 2$, if i is even, then

$$\triangle L_0^i \oplus \triangle R_0^i = \triangle L_0^{i-2} \oplus \triangle R_0^{i-2} = \ldots = \triangle L_0^0 \oplus \triangle R_0^0,$$

if i is odd, then

$$\triangle L_0^i \oplus \triangle R_0^i = \triangle L_0^{i-2} \oplus \triangle R_0^{i-2} = \ldots = \triangle L_0^1 \oplus \triangle R_0^1,$$

and complete the proof. □

Based on Theorem 1 and Definition 3, we can obtain Corollary 1.

Corollary 1 *For an arbitrary r-round Shadow, the input difference is* $(\triangle L_0^0, \triangle L_1^0, \triangle R_0^0, \triangle R_1^0)$*, correspondingly, the output difference is* $(\triangle L_0^r, \triangle L_1^r, \triangle R_0^r, \triangle R_1^r)$*. if* $\triangle L_0^0 \oplus \triangle R_0^0 \neq \triangle L_0^r \oplus \triangle R_0^r \quad (r = 2n)$ *or* $\triangle L_0^1 \oplus \triangle R_0^1 \neq \triangle L_0^r \oplus \triangle R_0^r \quad (r = 2n+1)$*, then* $(\triangle L_0^0, \triangle L_1^0, \triangle R_0^0, \triangle R_1^0) \nrightarrow (\triangle L_0^r, \triangle L_1^r, \triangle R_0^r, \triangle R_1^r)$ *is an impossible differential distinguisher of Shadow.*

Key recovery attack on full-round Shadow-32/64

In this section, we will use a concrete arbitrary N-round impossible differential distinguisher to perform key recovery for $(N+1)$-round Shadow-32 and $(N+1)$-round Shadow-64 respectively.

Key recovery attack on full-round Shadow-32

Theorem 2 (N-round Impossible Differential Distinguisher of Shadow-32) *In the single-key model, Shadow-32 exists an arbitrary N-round impossible differential distinguisher, i.e.*

$$(10000000, 00000000, 00000000, 00000000) \nrightarrow (01000000, 00000000, 00000000, 00000000),$$

where the Nth round includes data exchange.

Proof Firstly, we have $\triangle L_0^0 = 10000000$, $\triangle R_0^0 = 00000000$, $\triangle L_0^N = 01000000$, $\triangle R_0^N = 00000000$.

(1) When $N = 2n(n > 0)$, according to Theorem 1 and Corollary 1, since $\triangle L_0^0 \oplus \triangle R_0^0 \neq \triangle L_0^N \oplus \triangle R_0^N$, thus finding the contradiction.
(2) When $N = 2n+1(n > 0)$, after the propagation of difference for the first round, the output difference $(\triangle L_0^1, \triangle L_1^1, \triangle R_0^1, \triangle R_1^1)$ is $(00000000, *0*01**, 0*00001*, 00000000)$, according to Theorem 1 and Corollary 1, since $\triangle L_0^1 \oplus \triangle R_0^1 \neq \triangle L_0^N \oplus \triangle R_0^N$, thus finding the contradiction.

□

Next based on the N-round impossible differential distinguisher, encrypt one round backward to perform key recovery for $(N+1)$-round Shadow-32. The propagation of difference during the key recovery process is depicted in Fig. 4. The specific key recovery process is as follows.

- *Step 1* Let the difference of plaintext be

$$\begin{aligned}
\triangle x^0 &= (\triangle L_0^0, \triangle L_1^0, \triangle R_0^0, \triangle R_1^0)\\
\triangle L_0^0 &= (10000000)\\
\triangle L_1^0 &= (00000000)\\
\triangle R_0^0 &= (00000000)\\
\triangle R_1^0 &= (00000000).
\end{aligned}$$

Define the following plaintext structure

$$\begin{aligned}
x^0 &= (L_0^0, L_1^0, R_0^0, R_1^0)\\
L_0^0 &= (\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6\alpha_7\alpha_8)\\
L_1^0 &= (\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\alpha_{13}\alpha_{14}\alpha_{15}\alpha_{16})\\
R_0^0 &= (\alpha_{17}\alpha_{18}\alpha_{19}\alpha_{20}\alpha_{21}\alpha_{22}\alpha_{23}\alpha_{24})\\
R_1^0 &= (\alpha_{25}\alpha_{26}\alpha_{27}\alpha_{28}\alpha_{29}\alpha_{30}\alpha_{31}\alpha_{32}),
\end{aligned}$$

where $\alpha_i (1 \leq i \leq 32)$ is a constant. The plaintext can form 2 plaintext pairs. Select 2^n plaintext structures, and there are 2^{n+1} plaintext pairs $(x^0, \hat{x}^0)$. After $N+1$ rounds of encryption, obtain the corresponding ciphertext pairs $(x^{N+1}, \hat{x}^{N+1})$.

- *Step 2* Select the ciphertext pairs that satisfy the following form:

$$\begin{aligned}
\triangle x^{N+1} &= (\triangle L_0^{N+1}, \triangle L_1^{N+1}, \triangle R_0^{N+1}, \triangle R_1^{N+1})\\
\triangle L_0^{N+1} &= (*0*00001)\\
\triangle L_1^{N+1} &= (**0*01**)\\
\triangle R_0^{N+1} &= (00000000)\\
\triangle R_1^{N+1} &= (00000000),
\end{aligned}$$

where $* \in F_2$. Since the ciphertexts that satisfy the above form are 2^7, the probability is $2^7 \times 2^{-32} = 2^{-25}$. After screening, the ciphertext pair remains $2^{n+1} \times 2^{-25} = 2^{n-24}$.

- *Step 3* Guess 16-bit key in the $(N+1)$th round, i.e. key_2^{N+1} and key_0^{N+1}. Then decrypt each ciphertext pair from Step 2 one round forward, and get $(\triangle L_0^N, \triangle L_1^N)$. Judge if $\triangle L_0^N = 01000000$ and $\triangle L_1^N = 00000000$ hold, if hold, then the guessed key is wrong and is excluded. Repeat the above steps until the only correct key remains.

Complexity analysis After step 3, the error value of the key is approximately $(2^{16}-1) \times (1-2^{-1})^{2^{n-24}}$. When $n = 28$, $(2^{16}-1) \times (1-2^{-1})^{2^{n-24}} < 1$, therefore the wrong keys can all be excluded. Boura et al. (2014) presented that the data complexity is $2^{n+\triangle in+1}$, where the $\triangle in$ is the number of active bits for the difference of plaintext. So the data complexity is $2^{28+1+1} = 2^{30}$. Step 3 requires $2^{n-24} \times 2^{16} \times 2 = 2^{21}$ one round of encryption, in addition, the remaining 48 bits of the master key need to be searched exhaustively, so the time complexity

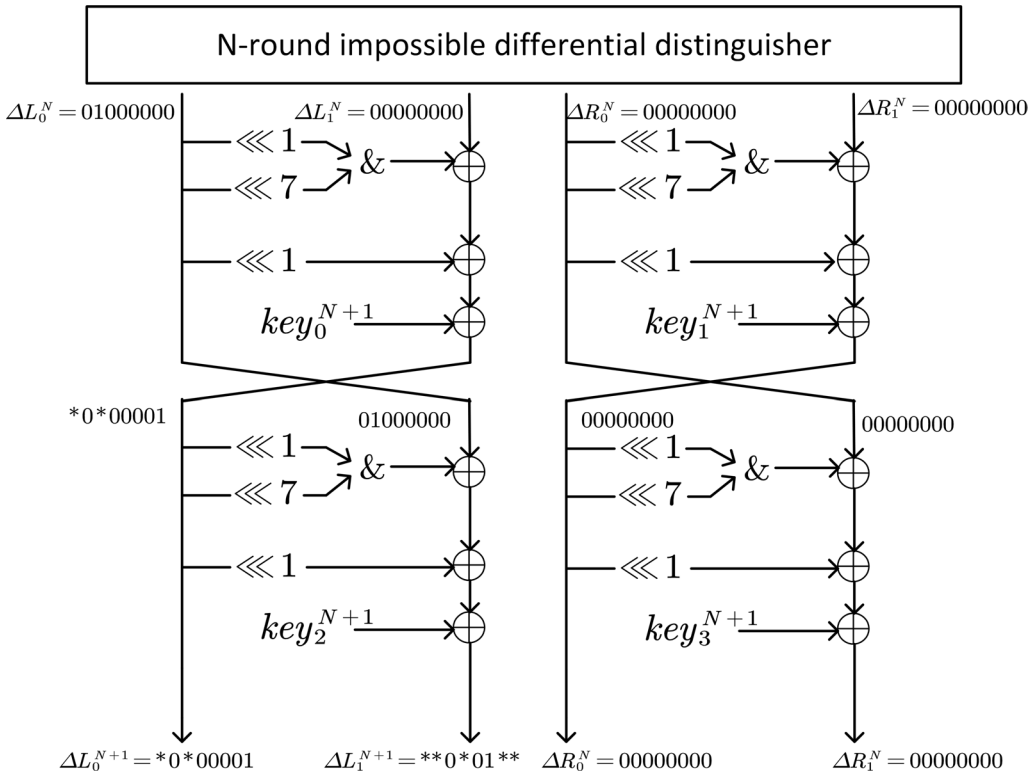

Fig. 4 $(N+1)$-round impossible differential cryptanalysis on Shadow-32

required to recover full key is $2^{21}/(N+1) + 2^{48} \approx 2^{48}$ (N+1)-round encryption. Since step 2 requires storing $2^{n-24} = 2^4$ ciphertext pairs and 2^{16} candidate keys, and an exhaustive search of 48 bits requires the storage of 2^{48}, the memory complexity required to recover full key is $(2^4 + 2^{16} + 2^{48})/32 \approx 2^{43}$ 32-bit block.

In summary, for Shadow-32, the round number is 16 and N is 15. A full-round impossible differential attack on Shadow-32 requires 2^{30} data complexity, 2^{48} 16-round encryption time complexity and 2^{43} 32-bit block memory complexity.

Key recovery attack on full-round Shadow-64

Theorem 3 (N-round Impossible Differential Distinguisher of Shadow-64) *In the single-key model, Shadow-64 exists an arbitrary N-round impossible differential distinguisher, i.e.*

$$(1000000000000000, 0000000000000000, 0000000000000000, 0000000000000000) \nrightarrow (0100000000000000, 0000000000000000, 0000000000000000, 0000000000000000),$$

where the Nth round includes data exchange.

Proof The process of proving Theorem 3 is similar to that of Theorem 2. Firstly, we have $\triangle L_0^0 = 1000000000000000$, $\triangle R_0^0 = 0000000000000000$, $\triangle L_0^N = 0100000000000000$, $\triangle R_0^N = 0000000000000000$.

(1) When $N = 2n(n > 0)$, according to Theorem 1 and Corollary 1, since $\triangle L_0^0 \oplus \triangle R_0^0 \neq \triangle L_0^N \oplus \triangle R_0^N$, thus finding the contradiction.
(2) When $N = 2n+1(n > 0)$, after the first difference propagation, the output difference $(\triangle L_0^1, \triangle L_1^1, \triangle R_0^1, \triangle R_1^1)$ is $(0000000000000000, 10 * 0000 * *000 * **0, 000000000 * 0000 * *, 0000000000000000)$, according to Theorem 1 and Corollary 1, since $\triangle L_0^1 \oplus \triangle R_0^1 \neq \triangle L_0^N \oplus \triangle R_0^N$, thus finding the contradiction.

□

Next based on the N-round impossible differential distinguisher, encrypt one round backward to perform key recovery for $(N+1)$-round Shadow-64. The propagation

of difference during key recovery is shown in Fig. 5. The specific key recovery process is as follows.

- *Step 1* Let the difference of plaintext be

$$\begin{aligned}\triangle x^0 &= (\triangle L_0^0, \triangle L_1^0, \triangle R_0^0, \triangle R_1^0)\\ \triangle L_0^0 &= (1000000000000000)\\ \triangle L_1^0 &= (0000000000000000)\\ \triangle R_0^0 &= (0000000000000000)\\ \triangle R_1^0 &= (0000000000000000).\end{aligned}$$

Define the following plaintext structure

$$\begin{aligned}x^0 &= (L_0^0, L_1^0, R_0^0, R_1^0)\\ L_0^0 &= (\alpha_1\alpha_2\ldots\alpha_{15}\alpha_{16})\\ L_1^0 &= (\alpha_{17}\alpha_{18}\ldots\alpha_{31}\alpha_{32})\\ R_0^0 &= (\alpha_{33}\alpha_{34}\ldots\alpha_{47}\alpha_{48})\\ R_1^0 &= (\alpha_{49}\alpha_{50}\ldots\alpha_{63}\alpha_{64}),\end{aligned}$$

where $\alpha_i (1 \le i \le 64)$ is a constant. The plaintext can form 2 plaintext pairs. Select 2^n plaintext structures, and there are 2^{n+1} plaintext pairs. After $N+1$ rounds of encryption, obtain the corresponding ciphertext pairs $(x^{N+1}, \hat{x}^{N+1})$.

- *Step 2* Select the ciphertext pairs that satisfy the following form:

$$\begin{aligned}\triangle x^{N+1} &= (\triangle L_0^{N+1}, \triangle L_1^{N+1}, \triangle R_0^{N+1}, \triangle R_1^{N+1})\\ \triangle L_0^{N+1} &= (*000000000*0000*)\\ \triangle L_1^{N+1} &= (010*0000**000***)\\ \triangle R_0^{N+1} &= (0000000000000000)\\ \triangle R_1^{N+1} &= (0000000000000000),\end{aligned}$$

where $* \in F_2$. Since the ciphertexts that satisfy the above form are 2^9, the probability is $2^9 \times 2^{-64} = 2^{-55}$. After screening, the ciphertext pair is $2^{n+1} \times 2^{-55} = 2^{n-54}$.

- *Step 3* Guess 32-bit key in the $(N+1)$th round, i.e. key_2^{N+1} and key_0^{N+1}. Then decrypt each ciphertext pair from Step 2 one round forward, and get $(\triangle L_0^N, \triangle L_1^N)$. Judge if $\triangle L_0^N = 0100000000000000$ and $\triangle L_1^N = 0000000000000000$ hold, if hold, then the guessed key is wrong and is excluded. Repeat the above steps until the only correct key remains.

Complexity analysis After step 3, the error value of the key is approximate $(2^{32}-1) \times (1-2^{-1})^{2^{n-54}}$. When $n = 59$, $(2^{32}-1) \times (1-2^{-1})^{2^{n-54}} < 1$, therefore the wrong keys can all be excluded. Boura et al. (2014) presented that the data complexity is $2^{n+\triangle in+1}$, where the $\triangle in$ is the number of active bits for the difference of plaintext. So the data complexity is $2^{59+1+1} = 2^{61}$. Step 3 requires

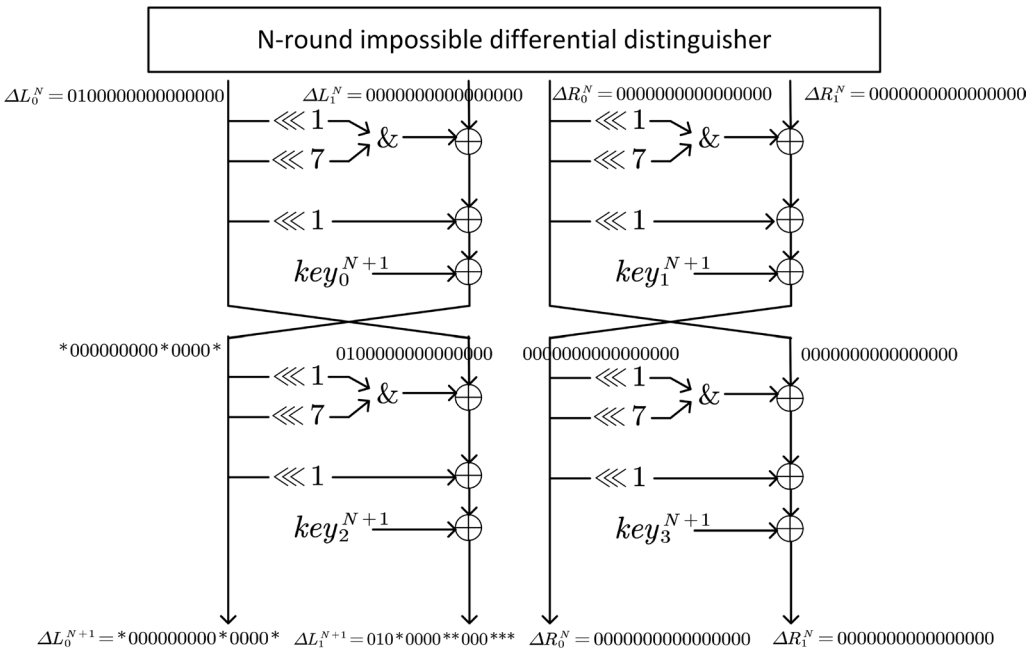

Fig. 5 $(N+1)$-round impossible differential cryptanalysis on Shadow-64

$2^{n-54} \times 2^{32} \times 2 = 2^{38}$ one round of encryption, in addition, the remaining 96 bits of the master key need to be searched exhaustively, so the time complexity required to recover the full key is $2^{38}/(N+1) + 2^{96} \approx 2^{96}$ $(N+1)$-round encryption. Since step 2 requires to store $2^{n-54} = 2^5$ ciphertext pairs and 2^{32} candidate keys, and an exhaustive search of 96 bits requires the storage of 2^{96}, the memory complexity required to recover the full key is $(2^5 + 2^{32} + 2^{96})/64 \approx 2^{90}$ 64-bit block.

For Shadow-64, the round number is 32 and N is 31. A full-round impossible differential attack requires 2^{61} data complexity, 2^{96} 32-round encryption time complexity and 2^{90} 64-bit block memory complexity. It is worth noting that simply increasing the number of iterative rounds of Shadow cannot resist the impossible differential attack.

Summary

In this paper, we focus on the differential property of Shadow and its security against the impossible differential attack. First, we use the SAT method to automatically search for a full-round impossible differential distinguisher of Shadow-32. Then, based on the experimental results, we prove that Shadow has a differential property with probability 1 based on the propagation of state. Further, we present an arbitrary number of rounds of impossible differential distinguisher for Shadow. Finally, we use a concrete arbitrary N-round impossible differential distinguisher to perform key recovery for $(N+1)$-round Shadow-32 and Shadow-64. For Shadow-32, a 16-round full key recovery attack requires 2^{30} data complexity, 2^{48} 16-round encryption time complexity and 2^{43} 32-bit block memory complexity. For Shadow-64, a 32-round full key recovery attack requires 2^{61} data complexity, 2^{96} 32-round encryption time complexity and 2^{90} 64-bit block memory complexity.

Both experimentally and theoretically, our results indicate that Shadow is critically flawed, and regardless of the security strength of the internal components and the number of rounds applied, the overall cipher remains vulnerable to the impossible differential attack.

Acknowledgements
I would like to express my sincere gratitude to my colleagues for their invaluable support, advice, and insightful discussions during the preparation of this thesis. I also wish to extend my appreciation to the anonymous reviewers for their constructive comments and feedback.

Author contributions
All the authors have equal contributions to this paper.

Funding
This work was supported by the National Natural Science Foundation of China (No. 12371525).

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 6 April 2023 Accepted: 22 August 2023
Published online: 07 December 2023

References

Abdelkhalek A, Sasaki Y, Todo Y, Tolba M, Youssef AM (2017) MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR Trans Symmetr Cryptol 99–129

Banik S, Bogdanov A, Isobe T, Shibutani K, Hiwatari H, Akishita T, Regazzoni F (2015) Midori: a block cipher for low energy. In: Proceedings of the 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, Part II. Springer, vol 21, pp 411–436

Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L (2015) The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd annual design automation conference, pp 1–6

Biham E, Shamir A (1991) Differential cryptanalysis of des-like cryptosystems. J Cryptol 4:3–72

Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Proceedings of the advances in cryptology-EUROCRYPT'99: international conference on the theory and application of cryptographic techniques Prague, Czech Republic, May 2–6, 1999. Springer, vol 18, pp 12–23

Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJ, Seurin Y, Vikkelsoe C (2007) Present: an ultra-lightweight block cipher. In: Proceedings of the cryptographic hardware and embedded systems-CHES 2007: 9th international workshop, Vienna, Austria, September 10–13, 2007. Springer, vol 9, pp 450–466

Boura C, Naya-Plasencia M, Suder V (2014) Scrutinizing and improving impossible differential attacks: applications to CLEFIA, Camellia, LBlock and Simon (full version). Ph.D. thesis, IACR cryptology ePrint archive

Cui T, Chen S, Jia K, Fu K, Wang M (2016) New automatic search tool for impossible differentials and zero-correlation linear approximations. Cryptology ePrint archive

Guo Y, Li L, Liu B (2021) Shadow: a lightweight block cipher for IoT nodes. IEEE Internet Things J 8(16):13014–13023

Hong D, Sung J, Hong S, Lim J, Lee S, Koo BS, Lee C, Chang D, Lee J, Jeong K et al (2006) Hight: a new block cipher suitable for low-resource device. In: Proceedings of the Cryptographic hardware and embedded systems-CHES 2006: 8th international workshop, Yokohama, Japan, October 10–13, 2006. Springer, vol 8, pp 46–59

Hu X, Li Y, Jiao L, Tian S, Wang M (2020) Mind the propagation of states: new automatic search tool for impossible differentials and impossible polytopic transitions. In: Proceedings of the advances in cryptology—ASIACRYPT 2020: 26th international conference on the theory and application of cryptology and information security, Daejeon, South Korea, December 7–11, 2020, Part I 26. Springer, pp 415–445

Kaur M, Yadav T, Kumar M, Dey D (2023) Full-round differential attack on ULC and LICID block ciphers designed for IoT. Cryptology ePrint archive

Knudsen L (1998) Deal-a 128-bit block cipher. Complexity 258(2):216

Kölbl S, Leander G, Tiessen T (2015) Observations on the SIMON block cipher family. In: Proceedings of the advances in cryptology—CRYPTO 2015: 35th annual cryptology conference, Santa Barbara, CA, USA, August 16–20, 2015, Part I. Springer, vol 35, pp 161–185

Kumar M, Yadav T (2022) MILP based differential attack on round reduced warp. In: Proceedings of the security, privacy, and applied cryptography engineering: 11th international conference, SPACE 2021, Kolkata, India, December 10–13, 2021. Springer, pp 42–59

Matsui M (1994) Linear cryptanalysis method for DES cipher. In: Proceedings of the advances in cryptology-EUROCRYPT'93: workshop on the theory and

application of cryptographic techniques Lofthus, Norway, May 23–27, 1993. Springer, vol 12, pp 386–397
Mouha N, Wang Q, Gu D, Preneel B (2012) Differential and linear cryptanalysis using mixed-integer linear programming. In: Information security and cryptology: 7th international conference, Inscrypt 2011, Beijing, China, November 30–December 3, 2011. Revised selected papers 7. Springer, pp 57–76
Sasaki Y, Todo Y (2017) New impossible differential search tool from design and cryptanalysis aspects: Revealing structural properties of several ciphers. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part III 36, pp. 185–215. Springer
Standaert FX, Piret G, Gershenfeld N, Quisquater JJ (2006) SEA: a scalable encryption algorithm for small embedded applications. In: Proceedings of the smart card research and advanced applications: 7th IFIP WG 8.8/11.2 international conference, CARDIS 2006, Tarragona, Spain, April 19–21, 2006. Springer, vol 7, pp 222–236
Sun S, Hu L, Wang P, Qiao K, Ma X, Song L (2014) Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES (l) and other bit-oriented block ciphers. In: Proceedings of the advances in cryptology—ASIACRYPT 2014: 20th international conference on the theory and application of cryptology and information security, Kaoshiung, Taiwan, ROC, December 7–11, 2014, Part I. Springer, vol 20, pp 158–178
Sun L, Wang M (2023) SoK: modeling for large s-boxes oriented to differential probabilities and linear correlations. IACR Trans Symmetric Cryptol 111–151
Sun L, Wang W, Wang M (2017) Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Proceedings of the advances in cryptology—ASIACRYPT 2017: 23rd international conference on the theory and applications of cryptology and information security, Hong Kong, China, December 3–7, 2017, Part I. Springer, vol 23, pp 128–157
Sun L, Wang W, Wang M (2021) Accelerating the search of differential and linear characteristics with the sat method. IACR Trans Symmetric Cryptol 269–315
Wu W, Zhang L (2011) LBlock: a lightweight block cipher. In: Proceedings of the applied cryptography and network security: 9th international conference, ACNS 2011, Nerja, Spain, June 7–10, 2011. Springer, vol 9, pp 327–344
Zhu B, Dong X, Yu H (2019) MILP-based differential attack on round-reduced gift. In: Proceedings of the topics in cryptology—CT-RSA 2019: the cryptographers' track at the RSA conference 2019, San Francisco, CA, USA, March 4–8, 2019. Springer, pp 372–390

Publisher's Note