Breaking the Shadow

Key Recovery Attack on Full-Round Shadow Block Ciphers with Minimal Data

Anda Che[1] and Shahram Rasoolzadeh[2]

[1] School of Cryptology, University of Chinese Academy of Science, Beijing, China,
cheanda22@mails.ucas.ac.cn
[2] Chair for Symmetric Cryptography, Ruhr University Bochum, Bochum, Germany,
shahram.rasoolzadeh@rub.de

Abstract. SHADOW is a family of lightweight block ciphers introduced by Guo, Li, and Liu in 2021, with SHADOW-32 having a 32-bit block size and a 64-bit key, and SHADOW-64 having a 64-bit block size and a 128-bit key. Both variants use a generalized Feistel network with four branches, incorporating the AND-Rotation-XOR operation similar to the SIMON family for their bridging function.

This paper reveals that the security claims of the SHADOW family are not as strong as suggested. We present a key recovery attack that can retrieve the sequence of round keys used for encryption with *only* two known plaintext/ciphertext pairs, requiring time and memory complexity of $2^{43.23}$ encryptions and $2^{21.62}$ blocks of memory for SHADOW-32, and complexity of $2^{81.32}$ encryptions and $2^{40.66}$ blocks of memory for SHADOW-64. Notably, this attack is independent of the number of rounds and the bridging function employed.

Furthermore, we critically evaluate one of the recent cryptanalysis on SHADOW ciphers and identify significant flaws in the proposed key recovery attacks. In particular, we demonstrate that the distinguisher used in impossible differential attacks by [LLCW23] is ineffective for key recovery, despite their higher claimed complexities compared to ours.

Keywords: SHADOW· Invariants · Key Schedule

1 Introduction

There is an increasing demand for secure and efficient cryptographic primitives to safeguard the growing number of resource-constrained devices that are becoming integral to our daily lives. Notable examples include RFID tags and nodes in sensor networks, which are key components of the Internet of Things (IoT). The SHADOW family of lightweight block ciphers, introduced by Guo et al. [GLL21], is designed to secure data transmission in IoT environments. SHADOW employs a combination of AND-Rotation-XOR operations and a generalized Feistel structure, with round operations that share similarities with those in the SIMON block cipher family [BSS+13].

While the designers evaluated the security of SHADOW against various types of cryptanalysis, Kim et al. [KSK+23] identified a structural invariant in the round functions of SHADOW. Using this invariant, they demonstrated that it is possible to recover the 64-bit key of (full-round) SHADOW-32 with a complexity of 2^{48} encryptions and the 128-bit key of (full-round) SHADOW-64 with a complexity of 2^{96} encryptions.

Subsequently, Mirzaei et al. [MAA24] introduced an integral attack targeting 14 out of the 16 rounds of SHADOW-32. Their attack requires 2^{31} chosen plaintext-ciphertext pairs, $2^{56.43}$ encryptions, and 2^{24} bytes of memory.

Later, Liu et al. [LLCW23] by applying the structural invariant discovered in [KSK+23], proposed impossible differential distinguishers for both versions of SHADOW and extended these distinguishers to a full-round key recovery attack. Their approach requires 2^{30} chosen plaintext-ciphertext pairs, 2^{48} encryptions, and 2^{43} blocks of memory for SHADOW-32, and 2^{61} chosen plaintext-ciphertext pairs, 2^{96} encryptions, and 2^{96} blocks of memory for SHADOW-64. However, as discussed in Section 5, this attack has technical flaws and is not feasible for key recovery.

More recently, Li et al. [LLB+24] proposed a differential attack on full-round SHADOW ciphers. Their attack has a complexity of 2^{15} chosen plaintext-ciphertext pairs and $2^{50.69}$ times solving quadratic equations for SHADOW-32, and $2^{32.6}$ chosen plaintext-ciphertext pairs and $2^{101.5}$ times solving quadratic equations for SHADOW-64. As mentioned in [LLB+24], solving these quadratic equations can be more time-consuming than performing a single SHADOW encryption.

Our Contribution. In this paper, we examine the properties of the SHADOW round functions and key schedule. Building on the structural invariant discovered by Kim et al. [KSK+23] and the additional properties presented in this work, we demonstrate that the SHADOW-32 block cipher can be compromised with a practical time complexity, and SHADOW-64 block cipher can be compromised with a not-out-of-reach time complexity. Our attacks are independent of the number of rounds used for encryption, require only two known plaintext-ciphertext pairs and a time and memory complexity of $2^{43.23}$ encryptions and $2^{21.62}$ blocks of memory for SHADOW-32, and $2^{81.32}$ encryptions and $2^{40.66}$ blocks of memory for SHADOW-64.

Outline of the Paper. We bring the specification of SHADOW in Section 2. In Section 3, we represent the structural invariant property on SHADOW. We present a key recovery attack on full-round SHADOW in Section 4. Then, in Section 5 we show the flaws in the attack of [LLCW23]. We conclude our work in Section 6.

2 Shadow Specification

SHADOW is a family of lightweight block ciphers with the following characteristics:

– SHADOW-32: 32-bit block size, 64-bit key size, and 16 rounds.
– SHADOW-64: 64-bit block size, 128-bit key size, and 32 rounds.

The overall structure of both ciphers is the same up to the size of the words used in encryption, and all the round functions are identical except for the values of the round keys. It applies a generalized Feistel network with four branches and actually each round of SHADOW consists of two Fiestel rounds, both applying the same bridging functions but with different shuffling between the branches.

We will use $(X_0^r, X_1^r, X_2^r, X_3^r)$ and $(Y_0^r, Y_1^r, Y_2^r, Y_3^r)$ to denote the state values and $(K_0^r, K_1^r, K_2^r, K_3^r)$ to denote the round key values in the r-th round where each X_i^r, Y_i^r and K_i^r are w-bit words with $w = 8$ for SHADOW-32 and $w = 16$ for SHADOW-64.

The bridging function F uses a similar function as in the SIMON family of block ciphers:

$$F(X) = (X \lll 1) \wedge (X \lll 7) \oplus (X \lll 2),$$

where $\wedge$ and $\oplus$ denote the bit-wise AND and XOR operations, respectively, and $X \lll i$ denotes the rotation of the w-bit word to the left by i bits.

In the SHADOW encryption process, the plaintext $P = (P_0, P_1, P_2, P_3)$ is initialized as the first intermediate state $(X_0^0, X_1^0, X_2^0, X_3^0)$. The encryption then proceeds through a fixed number of rounds R, where each round consists of two Feistel rounds. The round function is defined as follows:

$$\begin{cases} Y_0^r = F(X_0^r) \oplus X_1^r \oplus K_0^r \\ Y_1^r = X_0^r \\ Y_2^r = F(X_2^r) \oplus X_3^r \oplus K_1^r \\ Y_3^r = X_2^r \end{cases} \quad \text{and} \quad \begin{cases} X_0^{r+1} = Y_2^r \\ X_1^{r+1} = F(Y_0^r) \oplus Y_1^r \oplus K_2^r \\ X_2^{r+1} = Y_0^r \\ X_3^{r+1} = F(Y_2^r) \oplus Y_3^r \oplus K_3^r \end{cases}$$

Here, $(X_0^r, X_1^r, X_2^r, X_3^r)$ represents the input state, and $(Y_0^r, Y_1^r, Y_2^r, Y_3^r)$ denotes the intermediate state between the two Feistel rounds in the r-th round of SHADOW. After applying the round function for R times, the final state is $(X_0^R, X_1^R, X_2^R, X_3^R)$ and the ciphertext $C = (C_0, C_1, C_2, C_3)$ is then produced as $(X_2^R, X_1^R, X_0^R, X_3^R)$. Note that the exchange in the position of X_0^R and X_2^R for the ciphertext arises from omitting the second permutation of the branches in the final round. We depict two consecutive rounds of SHADOW in Figure 1.

SHADOW-4w uses a key of length $8w$ bits. Based on the size of the plaintext block, the cipher employs two similar key schedules to generate the round keys.

For SHADOW-32, the key is represented as a 64-bit bit-array $(k_0, \ldots, k_{63})$. In each round r, the state of the key is updated as follows:

1. The bits $(k_3, k_4, k_5, k_6, k_7)$ are XORed with the round constant $(c_0, c_1, c_2, c_3, c_4)$, which represents the binary form of the round counter.

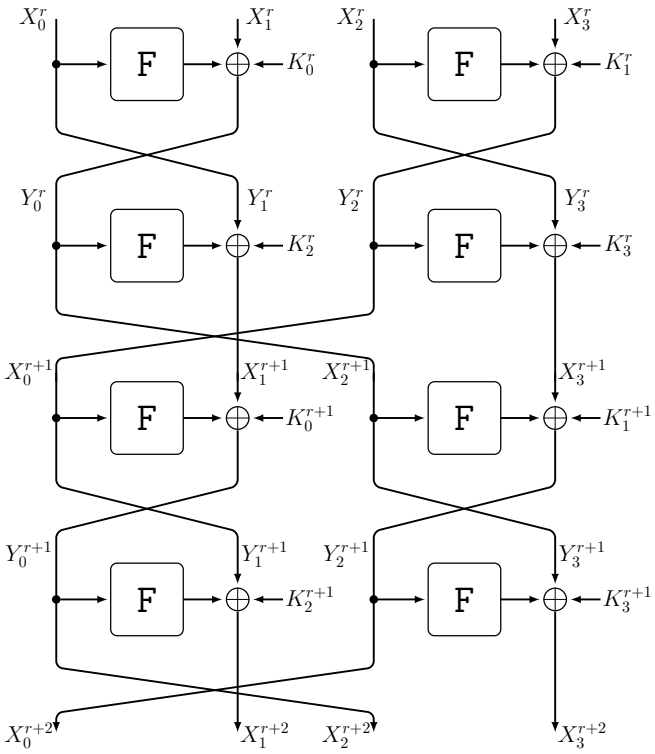

Fig. 1. Two Consecutive Rounds of SHADOW.

2. The bits $(k_{56}, k_{57}, k_{58}, k_{59}, k_{60}, k_{61}, k_{62}, k_{63})$ are updated using the following equations which is called the `NX` operation:

$$\begin{cases} k_{56} \leftarrow k_{56} \wedge (k_{56} \oplus k_{62}) \\ k_{57} \leftarrow k_{57} \wedge (k_{57} \oplus k_{63}) \\ k_{58} \leftarrow k_{58} \wedge (k_{58} \oplus k_{56} \oplus k_{62}) \\ k_{59} \leftarrow k_{59} \wedge (k_{59} \oplus k_{57} \oplus k_{63}) \\ k_{60} \leftarrow k_{60} \wedge (k_{60} \oplus k_{58} \oplus k_{56} \oplus k_{62}) \\ k_{61} \leftarrow k_{61} \wedge (k_{61} \oplus k_{59} \oplus k_{57} \oplus k_{63}) \\ k_{62} \leftarrow k_{62} \wedge (k_{62} \oplus k_{60} \oplus k_{58} \oplus k_{56} \oplus k_{62}) \\ k_{63} \leftarrow k_{63} \wedge (k_{63} \oplus k_{61} \oplus k_{59} \oplus k_{57} \oplus k_{63}) \end{cases}$$

3. Each bit of the key state is then permuted according to the permutation P_{64} defined in Table 1. Specifically,

$$k_i \leftarrow k_{P_{64}(i)} \quad \forall i$$

Note that this bit-wise permutation, can also be considered as a nibble-wise permutation on the nibbles of the key state.

Table 1. The Permutation in the SHADOW-32 Key Schedule.

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

4. After these operations, the round keys are produced as follows:

$$\begin{cases} K_0^r = (k_{00}, k_{01}, k_{02}, k_{03}, k_{08}, k_{09}, k_{10}, k_{11}) \\ K_1^r = (k_{04}, k_{05}, k_{06}, k_{07}, k_{12}, k_{13}, k_{14}, k_{15}) \\ K_2^r = (k_{16}, k_{17}, k_{18}, k_{19}, k_{24}, k_{25}, k_{26}, k_{27}) \\ K_3^r = (k_{20}, k_{21}, k_{22}, k_{23}, k_{28}, k_{29}, k_{30}, k_{31}) \end{cases}$$

For SHADOW-64, the key is represented as a 128-bit bit-array $(k_0, \ldots, k_{127})$. In each round r, the state of the key is updated as follows:

1. The bits $(k_2, k_3, k_4, k_5, k_6, k_7)$ are XORed with the round constant $(c_0, c_1, c_2, c_3, c_4, c_5)$, which represents the binary form of the round counter.
2. The bits $(k_{104}, \ldots, k_{127})$ are updated using the following equations which is called the `NX` operation:

$$\begin{cases} k_{104} \leftarrow k_{104} \wedge (k_{104} \oplus k_{126}) \\ k_{105} \leftarrow k_{105} \wedge (k_{105} \oplus k_{127}) \\ k_{106} \leftarrow k_{106} \wedge (k_{106} \oplus k_{104} \oplus k_{126}) \\ k_{107} \leftarrow k_{107} \wedge (k_{107} \oplus k_{105} \oplus k_{127}) \\ k_{108} \leftarrow k_{108} \wedge (k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k_{109} \leftarrow k_{109} \wedge (k_{109} \oplus k_{107} \oplus k_{105} \oplus k_{127}) \\ k_{110} \leftarrow k_{110} \wedge (k_{110} \oplus k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k_{111} \leftarrow k_{111} \wedge (k_{111} \oplus k_{109} \oplus k_{107} \oplus k_{105} \oplus k_{127}) \\ \ldots \\ k_{126} \leftarrow k_{126} \wedge (k_{126} \oplus k_{124} \oplus \ldots \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k_{127} \leftarrow k_{127} \wedge (k_{127} \oplus k_{125} \oplus \ldots \oplus k_{107} \oplus k_{105} \oplus k_{127}) \end{cases}$$

3. Each bit of the key state is then permuted according to the permutation P_{128} defined in Table 2. Specifically,

$$k_i \leftarrow k_{P_{128}(i)} \quad \forall i$$

Table 2. The Permutation in the SHADOW-64 Key Schedule.

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	56	57	58	59	60	61	62	63	64	65	66	67

i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	116	117	118	119	68	69	70	71	72	73	74	75	76	77	78	79

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	z	85	86	87	88	89	90	91	92	93	94	95
$P(i)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

i	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
$P(i)$	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Again, this bit-wise permutation can also be viewed as a nibble-wise permutation of the key state nibbles.

4. After these operations, the round keys are produced as follows:

$$\begin{cases} K_0^r = (k_{00}, k_{01}, k_{02}, k_{03}, k_{04}, k_{05}, k_{06}, k_{07}, k_{16}, k_{17}, k_{18}, k_{19}, k_{24}, k_{25}, k_{26}, k_{27}) \\ K_1^r = (k_{08}, k_{09}, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{24}, k_{25}, k_{26}, k_{27}, k_{28}, k_{29}, k_{30}, k_{31}) \\ K_2^r = (k_{32}, k_{33}, k_{34}, k_{35}, k_{36}, k_{37}, k_{38}, k_{39}, k_{48}, k_{49}, k_{50}, k_{51}, k_{56}, k_{57}, k_{58}, k_{59}) \\ K_3^r = (k_{40}, k_{41}, k_{42}, k_{43}, k_{44}, k_{45}, k_{46}, k_{47}, k_{52}, k_{53}, k_{54}, k_{55}, k_{60}, k_{61}, k_{62}, k_{63}) \end{cases}$$

3 Structural Invariant on Shadow

In this section, we present a structural invariant for SHADOW encryption, as outlined in the following theorem. We emphasize that this property on SHADOW was first demonstrated in [KSK+23].

Theorem 1. *For* SHADOW *encryption, the following properties hold:*

$$\begin{cases} Y_0^r \oplus Y_2^r \oplus Y_0^{r+2k} \oplus Y_2^{r+2k} = \bigoplus_{i=1}^{k} \left(K_2^{r+2i-1} \oplus K_3^{r+2i-1} \oplus K_0^{r+2i} \oplus K_1^{r+2i} \right) \\ Y_1^r \oplus Y_3^r \oplus Y_1^{r+2k} \oplus Y_3^{r+2k} = \bigoplus_{i=0}^{k-1} \left(K_2^{r+2i} \oplus K_3^{r+2i} \oplus K_0^{r+2i+1} \oplus K_1^{r+2i+1} \right) \end{cases}$$

Equivalently:

$$\begin{cases} X_0^r \oplus X_2^r \oplus X_0^{r+2k} \oplus X_2^{r+2k} = \bigoplus_{i=0}^{k-1} \left(K_2^{r+2i} \oplus K_3^{r+2i} \oplus K_0^{r+2i+1} \oplus K_1^{r+2i+1}\right) , \\ X_1^r \oplus X_3^r \oplus X_1^{r+2k} \oplus X_3^{r+2k} \oplus F(X_0^r) \oplus F(X_2^r) \oplus F(X_0^{r+2k}) \oplus F(X_2^{r+2k}) \\ \qquad\qquad = \bigoplus_{i=0}^{k-1} \left(K_0^{r+2i} \oplus K_1^{r+2i} \oplus K_2^{r+2i+1} \oplus K_3^{r+2i+1}\right) . \end{cases}$$

Consequently:

$$\begin{cases} P_0 \oplus P_2 \oplus C_0 \oplus C_2 = \bigoplus_{i=0}^{R/2-1} \left(K_2^{2i} \oplus K_3^{2i} \oplus K_0^{2i+1} \oplus K_1^{2i+1}\right) , \\ P_1 \oplus P_3 \oplus C_1 \oplus C_3 \oplus F(P_0) \oplus F(P_2) \oplus F(C_0) \oplus F(C_2) \\ \qquad\qquad = \bigoplus_{i=0}^{R/2-1} \left(K_0^{2i} \oplus K_1^{2i} \oplus K_2^{2i+1} \oplus K_3^{2i+1}\right) . \end{cases} \tag{1}$$

Proof. To prove the invariant, we analyze the SHADOW encryption process over multiple rounds. First, consider the second half of the r-th round and the first half of the $(r+1)$-th round. We have the following equations:

$$\begin{cases} X_0^{r+1} = Y_2^r \\ X_1^{r+1} = F(Y_0^r) \oplus Y_1^r \oplus K_2^r \\ X_2^{r+1} = Y_0^r \\ X_3^{r+1} = F(Y_2^r) \oplus Y_3^r \oplus K_3^r \end{cases} \quad \text{and} \quad \begin{cases} Y_0^{r+1} = F(X_0^{r+1}) \oplus X_1^{r+1} \oplus K_0^{r+1} \\ Y_1^{r+1} = X_0^{r+1} \\ Y_2^{r+1} = F(X_2^{r+1}) \oplus X_3^{r+1} \oplus K_1^{r+1} \\ Y_3^{r+1} = X_2^{r+1} \end{cases}$$

Combining these two sets of equations, we obtain:

$$\begin{cases} Y_0^{r+1} = F(Y_2^r) \oplus F(Y_0^r) \oplus Y_1^r \oplus K_2^r \oplus K_0^{r+1} \\ Y_1^{r+1} = Y_2^r \\ Y_2^{r+1} = F(Y_0^r) \oplus F(Y_2^r) \oplus Y_3^r \oplus K_3^r \oplus K_1^{r+1} \\ Y_3^{r+1} = Y_0^r \end{cases}$$

Adding the first and third equations, and the second and fourth equations, we get:

$$\begin{cases} Y_0^{r+1} \oplus Y_2^{r+1} = Y_1^r \oplus Y_3^r \oplus K_2^r \oplus K_3^r \oplus K_0^{r+1} \oplus K_1^{r+1} \\ Y_1^{r+1} \oplus Y_3^{r+1} = Y_0^r \oplus Y_2^r \end{cases}$$

Repeating these equations for another round, we obtain the following two-round equations:

$$\begin{cases} Y_0^{r+2} \oplus Y_2^{r+2} = Y_0^r \oplus Y_2^r \oplus K_2^{r+1} \oplus K_3^{r+1} \oplus K_0^{r+2} \oplus K_1^{r+2} \\ Y_1^{r+2} \oplus Y_3^{r+2} = Y_1^r \oplus Y_3^r \oplus K_2^r \oplus K_3^r \oplus K_0^{r+1} \oplus K_1^{r+1} \end{cases}$$

By repeating these equations for k times, we derive the following equations for $2k$ consecutive rounds:

$$\begin{cases} Y_0^r \oplus Y_2^r \oplus Y_0^{r+2k} \oplus Y_2^{r+2k} = \bigoplus_{i=1}^{k} \left(K_2^{r+2i-1} \oplus K_3^{r+2i-1} \oplus K_0^{r+2i} \oplus K_1^{r+2i}\right) \\ Y_1^r \oplus Y_3^r \oplus Y_1^{r+2k} \oplus Y_3^{r+2k} = \bigoplus_{i=0}^{k-1} \left(K_2^{r+2i} \oplus K_3^{r+2i} \oplus K_0^{r+2i+1} \oplus K_1^{r+2i+1}\right) \end{cases}$$

By substituting the definitions of Y^r and Y^{r+2k}, we get:

$$\begin{cases} X_1^r \oplus X_3^r \oplus X_1^{r+2k} \oplus X_3^{r+2k} \oplus F(X_0^r) \oplus F(X_2^r) \oplus F(X_0^{r+2k}) \oplus F(X_2^{r+2k}) \oplus \\ K_2^r \oplus K_3^r \oplus K_2^{r+2k} \oplus K_3^{r+2k} = \bigoplus_{i=1}^{k} \left(K_2^{r+2i-1} \oplus K_3^{r+2i-1} \oplus K_0^{r+2i} \oplus K_1^{r+2i}\right) \\ \\ X_0^r \oplus X_2^r \oplus X_0^{r+2k} \oplus X_2^{r+2k} = \bigoplus_{i=0}^{k-1} \left(K_2^{r+2i} \oplus K_3^{r+2i} \oplus K_0^{r+2i+1} \oplus K_1^{r+2i+1}\right) \end{cases}$$

Thus, we have established the properties as claimed. For the relation between the plaintext and ciphertext words, we set r to zero and k to $R/2$, assuming that R is always an even integer. □

Remark 1. In the SHADOW encryption, for any plaintext-ciphertext pair, the following phrases are always constant:

$$\begin{cases} P_0 \oplus P_2 \oplus C_0 \oplus C_2 \\ P_1 \oplus P_3 \oplus C_1 \oplus C_3 \oplus F(P_0) \oplus F(P_2) \oplus F(C_0) \oplus F(C_2) \end{cases}$$

In [KSK+23], the authors applied the invariant described in Remark 1 for a distinguishing attack on SHADOW ciphers. In this attack, the adversary, given access to two known plaintext-ciphertext pairs, can determine if these pairs are encrypted using the SHADOW cipher. This enables the attacker to distinguish the SHADOW cipher from a random permutation with a success probability of $1-2^{-2w}$, where w represents the word size: $1-2^{-16}$ for SHADOW-32 and $1-2^{-32}$ for SHADOW-64.

Additionally, the authors of [KSK+23] leveraged the properties in Theorem 1 to slightly accelerate the exhaustive search for the correct key used in the encryption. To achieve this, they query two known plaintext-ciphertext pairs. Using one of the pairs, they compute the values on the left side of Equation (1). They then guess the full-length key K, compute the round keys using SHADOW's key schedule, and substitute these round keys into the right side of Equation (1). They check if these values match those computed using the first plaintext-ciphertext pair. If there is no match, the guessed key is incorrect. Otherwise, they use the computed round keys to encrypt the two queried plaintexts and verify if the results match the corresponding ciphertexts.

Because SHADOW-$4w$ uses an $8w$-bit key, the exhaustive search attack in [KSK+23] requires 2^{8w} evaluations of the key schedule. Since computing the key schedule is as computationally intensive as encrypting a plaintext, the time complexity of their attack is only marginally better than a simple exhaustive search attack.

In the next section, we will present an accelerated exhaustive search for SHADOW that offers significantly better complexity than a straightforward exhaustive search.

4 Key Recovery Attacks on Shadow

In this section, we discuss several techniques for key recovery attacks on SHADOW ciphers. We begin by analyzing basic properties of the key schedule and demon-

strate that it is feasible to recover the full-length key with 2^{6w} SHADOW encryptions, utilizing 2^{3w} blocks of $4w$ bits (which is the same as the block size of the encryption). Next, we delve deeper into the properties of the key schedule to enhance the efficiency of our attack. We show that by using only two known plaintext-ciphertext pairs, $2^{21.62}$ blocks of memory and $2^{43.23}$ encryptions for SHADOW-32, and $2^{40.66}$ blocks of memory and $2^{81.32}$ encryptions for SHADOW-32, and it is possible to recover the sequence of round keys used for encryption.

4.1 The First Key Recovery Attack

In each operation of the SHADOW key schedule, key variables with even indices never interact with key variables with odd indices. Therefore, the entire key schedule, which operates on $8w$ bits, can be divided into two similar smaller key schedules, each operating on $4w$ bits: one updates the key state variables with even indices, and the other updates the key state variables with odd indices. Note that these two smaller key schedules, aside from the round constants, are identical.

Moreover, since the round key bits share the same odd-even parity as the key state variables, we can split Equation (1) into two parts: one w-bit condition that depends only on the key state variables with even indices, and another w-bit condition that depends only on the key state variables with odd indices.

By applying this separation to both the key schedule and Equation (1), we can reduce the computation time required to find candidates for the entire $8w$-bit key that satisfy the $2w$-bit condition specified in Equation (1).

The procedure for this attack is as follows: We perform the following steps separately for the $4w$ bits of the key corresponding to the even indices and again for the $4w$ bits corresponding to the odd indices. First, we guess a value for the $4w$ bits of the key corresponding to the even/odd indices. Using the half of the key schedule that operates on the variables with even/odd indices, we compute the entire half of the round keys with even/odd indices. With these round key variables, we can then check the w-bit condition of Equation (1) for the even/odd indices. If the guessed $4w$-bit key value satisfies the condition, we store it in memory. On average, there are 2^{3w} candidate values for each $4w$-bit part of the key corresponding to the even/odd indices.

Subsequently, any combination of a $4w$-bit key value from the memory of candidates for the even indices and a $4w$-bit key value from the memory of candidates for the odd indices represents a candidate for the entire $8w$-bit key that satisfies Equation (1). For each such full-length key candidate, we compute the encryption for the two known plaintexts and verify if the outputs match the corresponding ciphertexts. If they do, we have a candidate for the correct key.

The complexity of this attack involves computing the half key schedule twice for 2^{4w} guesses and storing the candidates in two separate memories of 2^{3w} blocks of $4w$ bits each. In the second part of the attack, we perform $2^{3w} \times 2^{3w} = 2^{6w}$ encryptions to verify the full-length key candidates.

Note that instead of storing the $4w$-bit candidates for the even and odd indices separately, we can use a single memory table to store candidates for the

Table 3. The NX Mapping in the SHADOW-32 Key Schedule.

$k_{56}k_{58}k_{60}k_{62}$	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$k'_{56}k'_{58}k'_{60}k'_{62}$	0000 0000 0010 0001 0100 0001 0100 0010 1000 0001 1000 0010 1000 0100 1010 0101

even indices of the key. Then, when we find a candidate for the odd indices of the key, we perform an exhaustive search by combining this value for the odd indices and any value for the even indices of the key stored in the memory. While this approach makes the algorithm more complex, it reduces the memory complexity of the attack to a single memory of 2^{3w} blocks.

4.2 The Second Key Recovery Attack

In this subsection, we delve deeper into the operations of the key schedule to exploit new properties that enhance the complexity of our key recovery attack.

First, we analyze the NX operation, which operates on the last 8 bits in SHADOW-32 and on the last 24 bits in SHADOW-64. As previously mentioned, all key schedule operations, including NX, can be divided into two smaller parts: one operating on even-indexed bits and the other on odd-indexed bits. In the case of NX, it splits into two identical functions but applied to different inputs.

Here, we focus on the portion of NX that operates on the even bits, specifically on 4 bits of $(k_{56}, k_{58}, k_{60}, k_{62})$ in SHADOW-32 and on 12 bits of $(k_{104}, k_{106}, \ldots, k_{124}, k_{126})$ in SHADOW-64. For instance, in SHADOW-32, the operation is defined as follows:

$$\begin{cases} k'_{56} = k_{56} \wedge (k_{56} \oplus k_{62}) \\ k'_{58} = k_{58} \wedge (k_{58} \oplus k_{56} \oplus k_{62}) \\ k'_{60} = k_{60} \wedge (k_{60} \oplus k_{58} \oplus k_{56} \oplus k_{62}) \\ k'_{62} = k_{62} \wedge (k_{62} \oplus k_{60} \oplus k_{58} \oplus k_{60} \oplus k_{62}) \end{cases}$$

We demonstrate the look-up table of this mapping in Table 3. Clearly, this mapping is not bijective. Specifically, there are only 7 distinct outputs out of the 16 possibilities.

This means that even though the master 64-bit key for SHADOW-32 encryption is chosen randomly, due to the non-bijectivity of the NX operation in the first round, it can produce at most $(7/16)^2 \cdot 2^{64} = 7^2 \cdot 2^{56}$ possible different set of round keys. Note that the power two comes from the separate considerations for even-indexed and odd-indexed bits. This results in an entropy loss of $(8 - 2\log_2 7) \approx 2.39$ bits in the key information used in the encryption.

In the first round, the index of 8 bits from the master key those go through NX operation are $56, \ldots, 63$. In the second round, the index of 8 bits from the master key those go through this operation are $8, \ldots, 15$ which are not involved in the first round. Consequently, these 8 bits, despite being chosen randomly to initialize the master key, can only produce 7^2 different possibilities due to the NX operation in the second round. This results in an additional entropy loss

of approximately 2.39 bits. Thus, the total number of possible different sets of round keys in the key schedule of SHADOW-32 is at most $(7/16)^4 \cdot 2^{64} = 7^4 \cdot 2^{48}$.

This technique does not apply to other rounds because the 8 bits affected by the `NX` operation in the third round are indexed $20, \ldots 27$, which have already been used in the first and second rounds.

By utilizing this property in our key recovery attack, we can improve the complexity of the attack. Instead of guessing all 32 bits of the key corresponding to even/odd indices, we only need to consider $7^2 \cdot 2^{24}$ possibilities for the even/odd indices of the round keys. With these round key variables, we then check the 8-bit condition of Equation (1) for the even/odd indices. On average, there are $7^2 \cdot 2^{16}$ candidate values for each part of the round keys corresponding to the even/odd indices. By combining one candidate from each part, we compute the encryption for the two known plaintexts and verify if the outputs match the corresponding ciphertexts. If they match, we have a candidate for the set of round keys.

Thus, by applying this technique, we reduce the complexity of the attack to $7^4 \cdot 2^{32} \approx 2^{43.23}$ encryptions and using $7^2 \cdot 2^{16} \approx 2^{21.62}$ blocks of memory.

In the case of SHADOW-64, the `NX` operation can be split into two parts: one operating on 12 even-indexed bits and one on 12 odd-indexed bits. Here, we focus on the portion of `NX` that operates on the even bits. Specifically, this involves 12 bits of $(k_{104}, k_{106}, \ldots, k_{124}, k_{126})$ in SHADOW-64. This operation can be expressed as following:

$$\begin{cases} k'_{104} = k_{104} \wedge (k_{104} \oplus k_{126}) \\ k'_{106} = k_{106} \wedge (k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{108} = k_{108} \wedge (k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{110} = k_{110} \wedge (k_{110} \oplus k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{112} = k_{112} \wedge (k_{112} \oplus k_{110} \oplus k_{108} \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{114} = k_{114} \wedge (k_{114} \oplus k_{112} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{116} = k_{116} \wedge (k_{116} \oplus k_{114} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{118} = k_{118} \wedge (k_{118} \oplus k_{116} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{120} = k_{120} \wedge (k_{120} \oplus k_{118} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{122} = k_{122} \wedge (k_{122} \oplus k_{120} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{124} = k_{124} \wedge (k_{124} \oplus k_{122} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \\ k'_{126} = k_{126} \wedge (k_{126} \oplus k_{124} \oplus \ \ldots \ \oplus k_{106} \oplus k_{104} \oplus k_{126}) \end{cases}$$

We compute the value for $(k'_{104}, k'_{106}, \ldots, k'_{124}, k'_{126})$ corresponding for each 2^{12} possible value of $(k_{104}, k_{106}, \ldots, k_{124}, k_{126})$, and observe that the output of this mapping can only have 322 distinct values out of the 2^{12} possiblities.

Since this operation is non-bijective, even though the master 128-bit key for SHADOW-64 is chosen randomly, the operation in the first round can produce at most $(322/2^{12})^2 \cdot 2^{128} = 161^2 \cdot 2^{106}$ possible different sets of round keys. This results in an entropy loss of approximately $(22 - 2\log_2 161) \approx 7.34$ bits in the key information used in the encryption.

While in the first round, the indices of 24 bits from the master key that go through the NX operation are $104, \ldots, 127$, in the second round, the indices of 24 bits from the master key that are processed by this operation are $8, \ldots, 31$, which were not involved in the first round. Consequently, these 24 bits, despite being chosen randomly to initialize the master key, can only produce 322^2 different possibilities due to the NX operation in the second round. This results in an additional entropy loss of approximately 7.34 bits. Thus, the total number of possible different sets of round keys in the key schedule of SHADOW-64 is at most $(322/2^{12})^4 \cdot 2^{128} = 161^4 \cdot 2^{84} \approx 2^{113.32}$.

Again, this technique does not apply to other rounds because the 24-bit indices from the master key that go through the NX operation in the third round are $36, \ldots, 55$ and $108, \ldots, 111$, which have already been used in the first and second rounds.

Similar to the case for SHADOW-32, here as well, we can improve the attack complexity by utilizing this property. Instead of guessing all 64 bits of the key corresponding to the even/odd indices, we only need to consider $161^2 \cdot 2^{42}$ possibilities for the even/odd indices of the round keys. With these round key variables, we then check the 16-bit condition of Equation (1) for the even/odd indices. On average, there are $161^2 \cdot 2^{26}$ candidate values for each part of the round keys corresponding to the even/odd indices. By combining one candidate from each part, we compute the encryption for the two known plaintexts and verify if the outputs match the corresponding ciphertexts. If they match, we have a candidate for the set of round keys.

Thus, by applying this technique, we reduce the complexity of the attack to approximately $161^4 \cdot 2^{52} \approx 2^{81.32}$ encryptions and using $161^2 \cdot 2^{26} \approx 2^{40.66}$ blocks of memory.

5 Technical Flaws in the Previous Work by [LLCW23]

In this section, we critically examine the impossible differential attacks proposed by Liu et al. in [LLCW23] on full-round SHADOW block ciphers and identify flaws in their key recovery attacks.

Liu et al. [LLCW23] present an impossible differential distinguisher for any number of rounds of the SHADOW cipher. Their approach leverages the results from Theorem 1 to demonstrate the impossibility of certain differentials. Specifically, they show that the differential

$$(\texttt{10000000},\texttt{00000000},\texttt{00000000},\texttt{00000000}) \nrightarrow (\texttt{01000000},\texttt{00000000},\texttt{00000000},\texttt{00000000})$$

is impossible for any number of rounds in SHADOW-32. Similarly, they assert that the differential

$$(\texttt{1000000000000000},\texttt{0000000000000000},\texttt{0000000000000000},\texttt{0000000000000000}) \nrightarrow (\texttt{0100000000000000},\texttt{0000000000000000},\texttt{0000000000000000},\texttt{0000000000000000})$$

is impossible for any number of rounds in SHADOW-64.

To perform a key recovery attack, Liu et al. extend these differentials by one additional round on the ciphertext side. For SHADOW-32,

$$\Delta X_{15} = (\texttt{01000000},\texttt{00000000},\texttt{00000000},\texttt{00000000})$$

extends to

$$\Delta C = \Delta X_{16} = (\texttt{*0*00001},\texttt{**0*01**},\texttt{00000000},\texttt{00000000}),$$

with $\Delta C_0 \oplus \Delta C_2 = \texttt{*0*00001}$. For SHADOW-64,

$$\Delta X_{31} = (\texttt{0100000000000000},\texttt{0000000000000000},\texttt{0}\cdots\texttt{0},\texttt{0}\cdots\texttt{0})$$

extends to

$$\Delta C = \Delta X_{32} = (\texttt{*000000000*00001},\texttt{010*0000**0001**},\texttt{0}\cdots\texttt{0},\texttt{0}\cdots\texttt{0})$$

with $\Delta C_0 \oplus \Delta C_2 = (\texttt{*000000000*00001})$.

However, due to the structural invariant described in Equation (1), we have

$$\Delta C_0 \oplus \Delta C_2 = \Delta P_0 \oplus \Delta P_2\,.$$

In this case, $\Delta P_0 \oplus \Delta P_2$ is `10000000` for SHADOW-32 and `1000000000000000` for SHADOW-64. But, `*0*00001` cannot be equal to `10000000`, and `*000000000*00001` cannot be equal to `1000000000000000`. Thus, the differentials proposed by Liu et al. cannot be used for a key recovery attack, as no differential pair will match the required form.

Therefore, the attack methodology presented in [LLCW23] is fundamentally flawed. The proposed differentials cannot be exploited for key recovery due to the inherent contradiction with the structural invariant.

6 Conclusion

In this paper, we presented a comprehensive analysis of the key schedule operations in SHADOW block ciphers, focusing on the `NX` operation and its impact on key recovery attacks.

We introduce an attack capable of recovering the round key sequence using only two known plaintext-ciphertext pairs. This attack requires $2^{43.23}$ encryptions and $2^{21.62}$ blocks of memory for SHADOW-32, and $2^{81.32}$ encryptions with $2^{40.66}$ blocks of memory for SHADOW-64. Importantly, the attack's effectiveness does not depend on the number of rounds or the bridging function used.

Furthermore, we critically assessed a previous work of impossible differential attack by Liu et al. [LLCW23], identifying flaws in their proposed methods. We showed that their attacks, based on impossible differentials, cannot be applied effectively. This critique underscores the importance of thoroughly validating attack methodologies against cryptographic primitives.

References

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, page 404, 2013.

[GLL21] Ying Guo, Lang Li, and Botao Liu. Shadow: A Lightweight Block Cipher for IoT Nodes. *IEEE Internet Things J.*, 8(16):13014–13023, 2021.

[KSK+23] Sunyeop Kim, Myoungsu Shin, Seonkyu Kim, Hanbeom Shin, Insung Kim, Donggeun Kwon, Dongjae Lee, Seonggyeom Kim, Deukjo Hong, Jaechul Sung, and Seokhie Hong. Shining Light on the Shadow: Full-round Practical Distinguisher for Lightweight Block Cipher Shadow. *IACR Cryptol. ePrint Arch.*, page 1200, 2023.

[LLB+24] Yanjun Li, Hao Lin, Xinjie Bi, Shanshan Huo, and Yiyi Han. MILP-based Differential Cryptanalysis on Full-Round Shadow. *J. Inf. Secur. Appl.*, 81:103696, 2024.

[LLCW23] Yuting Liu, Yongqiang Li, Huiqin Chen, and Mingsheng Wang. Full-Round Impossible Differential Attack on Shadow Block Cipher. *Cybersecur.*, 6(1):52, 2023.

[MAA24] Atiyeh Mirzaie, Siavash Ahmadi, and Mohammad Reza Aref. Integral Cryptanalysis of Round-Reduced Shadow-32 for IoT Nodes. *IEEE Internet Things J.*, 11(6):10592–10599, 2024.