A Chosen IV Attack Using Phase Shifting Equivalent Keys against DECIM v2

Hidehiko Nakagami, Ryoichi Teramura, Toshihiro Ohigashi, Hidenori Kuwakado, and Masakatu Morii

Kobe University, 1-1 Rokkodai, Kobe 657-8501, Japan
{0414358t, teramura}@stu.kobe-u.ac.jp, ohigashi@m.ieice.org, {kuwakado, mmorii}@kobe-u.ac.jp

Abstract. DECIM v2 is a stream cipher submitted to the ECRYPT stream cipher project (eSTREAM) and ISO/IEC 18033-4. No attack against DECIM v2 has been proposed yet. In this paper, we propose a chosen IV attack against DECIM v2 using a new equivalent key class. Our attack can recover an 80-bit key with a time complexity of $2^{79.56}$ when all bits of the IV are zero. This result is the best one on DECIM v2.

Keywords: cryptanalysis, equivalent keys, stream cipher, DECIM v2, eSTREAM

1 Introduction

Keys are called equivalent keys if ciphertexts generated from these keys have equivalence. In general, equivalent keys generate same ciphertexts. In the stream ciphers, keys that generate pseudo-random sequences (called keystreams) of different phase are also called equivalent keys since the ciphertext is made by XORing a plaintext to a keystream. We call such equivalent keys *phase shifting equivalent keys*.

In 2006, phase shifting equivalent keys on Grain v1 [1] are discussed by Ö. Küçük [2]. Recently, key recovery attacks using phase shifting equivalent keys have been proposed. These attacks search an key and phase shifting equivalent keys in parallel for speeding up an exhaustive key search. Two key recovery attacks against Grain v1 using phase shifting equivalent keys were independently proposed by Isobe et al. [3] in September 2007 and by C. De Cannière et al. [4] in February 2008 [1]. When all bits of an initialization vector (IV) are one, the attack of Isobe et al. can recover an 80-bit key with a time complexity of $2^{78.4}$ or $2^{78.7}$, and the attack of C. De Cannière et al. can recover the key with a time complexity of 2^{79}. The IV is a public value, and is changed in each encryption session. Moreover, Isobe et al. have given a lot of IVs for the attack [5]. Although key

[1] The approach of these attacks are different. We think that the efficient of the key recover attack can be improved with a combination of attacks of [3] and [4].

recovery attacks against Grain v1 using phase shifting equivalent keys have been discussed, applying the attack to other stream ciphers has not been discussed yet.

In this paper, we discuss a key recovery attack against DECIM v2 [6] using phase shifting equivalent keys. DECIM v2 is a hardware oriented stream cipher, which was designed by C. Berbain et al. It uses an 80-bit key K and a 64-bit IV IV, and has submitted to the ECRYPT stream cipher project (eSTREAM) [7] and ISO/IEC 18033-4. Now, no efficient attack against DECIM v2 has been proposed yet. First, we show that any $(K,\, IV)$ pair has a 1-bit phase shifting equivalent key $(\hat{K},\, \hat{IV})$ with a probability of 1/8 on DECIM v2. Next, we propose a key recovery attack against DECIM v2 using these equivalent keys. When $IV = (0, 0, \ldots, 0)$ is used, our attack can recover an 80-bit key with a time complexity of $2^{79.56}$. Our attack is a first one against DECIM v2.

This paper is organized as follows: DECIM v2 is described in Sect. 2. In Sect. 3. we define phase shifting equivalent keys, and show that conditions for such equivalent keys. Sect. 4. we give conditions for phase shifting equivalent keys of DECIM v2, and calculate the probability that any $(K,\, IV)$ has a phase shifting equivalent key. Sect. 5. we propose a key recovery attack against DECIM v2 using phase shifting equivalent keys.

2 Description of DECIM v2

DECIM v2 uses an 80-bit key $K = (K_0, \cdots, K_{79})$ and a 64-bit IV $IV = (IV_0, \cdots,\ IV_{63})$, where K_i and IV_i are 1-bit variables. An internal state S consists of a 192-bit linear feedback shift register (LFSR) $S = (x_0, x_1, \ldots, x_{191})$, where x_i is a 1-bit variable. In addition, DECIM v2 has a nonlinear filter function f and an irregular decimation mechanism (called the ABSG) and a Buffer.

The algorithm of DECIM v2 is split into a key-scheduling algorithm (KSA) and a pseudo-random generation algorithm (PRGA). The KSA initializes the internal state using K and IV. The PRGA generates a pseudo-random sequence (called a keystream) $Z = (z_0, z_1, \ldots)$ from an initial state of the PRGA, which is an internal state when the KSA was completed, where z_i is a 1-bit variable. A ciphertext/plaintext is obtained by XORing a plaintext/ciphertext to the keystream.

We describe the KSA and the PRGA. In order to distinguish between the KSA and the PRGA, we use different symbols. The internal state of the KSA is denoted by S^*, and that of the PRGA is denoted by S.

2.1 Pseudo Random Generation Algorithm

The internal state of the PRGA at time t denotes $S_t = (x_{0,t}, x_{1,t}, \ldots, x_{191,t})$. The PRGA has an update function $PRGA\_Nextstate(\ )$ and an output function $Output(\ )$. $PRGA\_Nextstate(\ )$ at time t updates S_{t-1} to S_t, and $Output(\ )$ generates the keystream from the internal state.

We describe the process of $PRGA\_Nextstate(\,)$. The internal state S_{t-1} is updated as follows:

$$x_{i,t} = \begin{cases} x_{i+1,t-1} & \text{if } i = 0, \dots,\ 190, \\ lv_{t-1} & \text{if } i = 191, \end{cases} \tag{1}$$

where

$$\begin{aligned} lv_{t-1} = {} & x_{189,t-1} \oplus x_{188,t-1} \oplus x_{169,t-1} \oplus x_{156,t-1} \oplus x_{155,t-1} \oplus x_{132,t-1} \\ & \oplus x_{131,t-1} \oplus x_{94,t-1} \oplus x_{77,t-1} \oplus x_{46,t-1} \oplus x_{17,t-1} \\ & \oplus x_{16,t-1} \oplus x_{5,t-1} \oplus x_{0,t-1}. \end{aligned} \tag{2}$$

We describe the process of $Output(\,)$. First, the nonlinear filter function f outputs a 1-bit variable v_{t-1} using the internal state S_{t-1} as follows [2] :

$$v_{t-1} = \begin{cases} 0 & \text{if } X = 0, 3, \\ 1 & \text{if } X = 1, 2, \end{cases} \tag{3}$$

where $X = (x_{191,t-1} + x_{186,t-1} + x_{178,t-1} + x_{172,t-1} + x_{162,t-1} + x_{144,t-1} + x_{111,t-1} + x_{104,t-1} + x_{65,t-1} + x_{54,t-1} + x_{45,t-1} + x_{28,t-1} + x_{13,t-1}) \bmod 4$. Second, a 1-bit variable y_{t-1} is obtained from v_{t-1} and S_{t-1} as follows:

$$y_{t-1} = v_{t-1} \oplus x_{1,t-1}. \tag{4}$$

Third, the ABSG outputs the keystream $(z_0, z_1, \dots)$ from the sequence $Y = (y_0, y_1, \dots)$. The algorithm of the ABSG is given as follows:

```
Input: $(y_0, y_1, \dots)$
Set: $i \leftarrow 0;\ j \leftarrow 0;$
Repeat the following steps:
    1. $e \leftarrow y_i,\quad z_j \leftarrow y_{i+1};$
    2. $i \leftarrow i + 1;$
    3. while $(y_i = \overline{e})$ $i \leftarrow i + 1;$
    4. $i \leftarrow i + 1;$
    5. output $z_j;$
    6. $j \leftarrow j + 1;$
```

The keystream $(z_0, z_1, \dots)$ are stored in the buffer. Since the Buffer is a mechanism for the implementation, we omit its detail. Figure 1 shows the processes of the PRGA of DECIM v2.

[2] Note that Eq. (3) and $v_{t-1} = f(a_1, \dots, a_{13}) = \bigoplus_{1 \le i < j \le 13} a_i a_j \bigoplus_{1 \le i \le 13} a_i$ are equivalent, where $(a_1, \dots, a_{13}) = (x_{191,t-1}, x_{186,t-1}, x_{178,t-1}, x_{172,t-1}, x_{162,t-1}, x_{144,t-1}, x_{111,t-1}, x_{104,t-1}, x_{65,t-1}, x_{54,t-1}, x_{45,t-1}, x_{28,t-1}, x_{13,t-1})$.

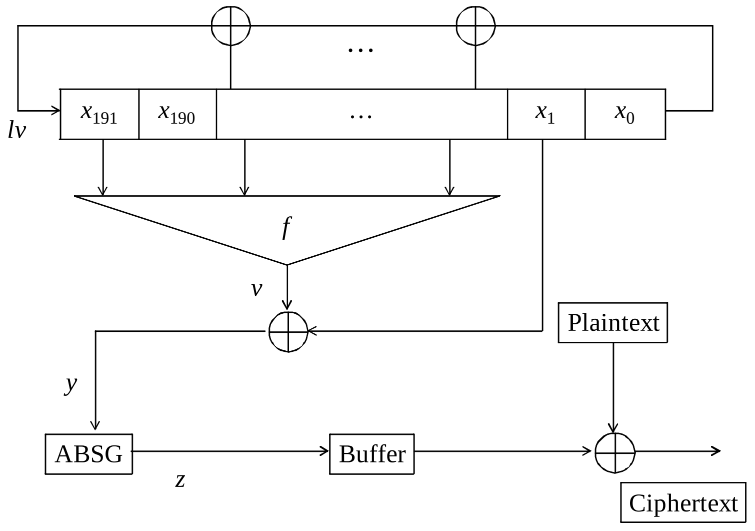

Fig. 1. PRGA of DECIM v2.

2.2 Key Scheduling Algorithm

The internal state of the KSA at time t denotes $S_t^* = (x_{0,t}^*, x_{1,t}^*, \ldots, x_{191,t}^*)$. First, S^* is initialized by K and IV as follows:

$$x_{i,0}^* = \begin{cases} K_i & \text{if } i = 0, \ldots, 79, \\ K_{i-80} \oplus IV_{i-80} & \text{if } i = 80, \ldots, 143, \\ K_{i-80} \oplus IV_{i-144} \oplus IV_{i-128} \oplus IV_{i-112} \oplus IV_{i-96} & \text{if } i = 144, \ldots, 159, \\ IV_{i-160} \oplus IV_{i-128} \oplus 1 & \text{if } i = 160, \ldots, 191. \end{cases} \quad (5)$$

Next, S_{t-1}^* is updated by $KSA\_Nextstate(\ )$. $KSA\_Nextstate(\ )$ is defined as

$$x_{i,t}^* = \begin{cases} x_{i+1,t-1}^* & \text{if } i = 0, \ldots, 190, \\ lv_{t-1}^* \oplus v_{t-1}^* & \text{if } i = 191. \end{cases} \quad (6)$$

lv_{t-1}^* and v_{t-1}^* are obtained from S_{t-1}^* by using Eqs. (2) and (3).

After $KSA\_Nextstate(\ )$ is executed for $t = 1, 2, \ldots, 768$, an obtained S_{768}^* is set to S_0, which is an initial state of the PRGA. Figure 2 shows the processes of the KSA of DECIM v2.

3 Phase Shifting Equivalent Keys

3.1 Description

In stream ciphers, a w bits of a keystream is outputted with updating an internal state S in every time. The keystream Z generated from a key and an IV pair

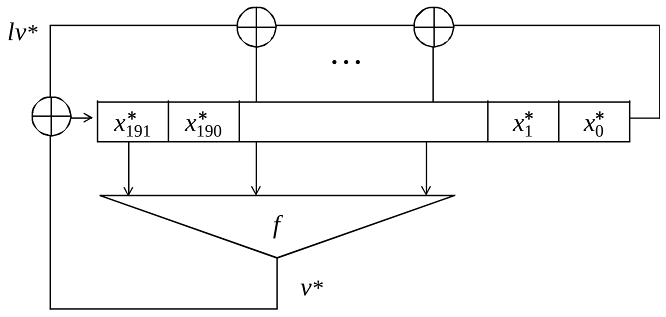

Fig. 2. KSA of DECIM v2.

(K, IV) is expressed as $Z = (z_0, z_1, \dots)$, where z_i is a w-bit variable. We consider the case that the internal state S_n, which is generated from (K, IV), is equal to the initial state of the PRGA $\hat{S}_0$, which is generated from the different key and IV pair $(\hat{K}, \hat{IV})$, where n is a positive integer. This case is expressed as follows:

$$S_n = \hat{S}_0. \tag{7}$$

If Eq. (7) holds, the keystream $\hat{Z}$, which is generated from $(\hat{K}, \hat{IV})$, is

$$\hat{Z} = (\hat{z}_0, \hat{z}_1, \dots) = (z_n, z_{n+1}, \dots). \tag{8}$$

$\hat{Z}$ is the nw-bit shifted keystream from Z. Then, (K, IV) and $(\hat{K}, \hat{IV})$ are called phase shifting equivalent keys.

3.2 Conditions for Phase Shifting Equivalent Keys

First, we present conditions for obtaining phase shifting equivalent keys of a stream cipher that has similar update functions of the KSA and the PRGA. We give the following theorem for phase shifting equivalent keys.

Theorem 1 *If (K, IV) and $(\hat{K}, \hat{IV})$ satisfy following conditions, then $S_n = \hat{S}_0$ holds.*

Condition 1: $\hat{S}^*_0$ and S^*_n are identical.
Condition 2: The function $PRGA\_Nextstate(\ )$ of (K, IV) at time $t = 1, 2, \dots, n$ and the function $KSA\_Nextstate(\ )$ are identical.

Proof. From Condition 1, $\hat{S}^*_0 = S^*_n$ holds. Let T be the number of calling $KSA\_Nextstate(\ )$ in the KSA. Since identical update function is used in the KSA for all $t = 1, 2, \dots, T - n$, the initial state of the PRGA S_0 satisfies $\hat{S}^*_{T-n} = S^*_T = S_0$. When Condition 2 holds, $PRGA\_Nextstate(\ )$ of (K, IV) at time $t = 1, 2, \dots, n$ and the function $KSA\_Nextstate(\ )$ of $(\hat{K}, \hat{IV})$ at time $t = T - n + 1, T - n + 2, \dots, T$ are identical. Then, $\hat{S}^*_T = \hat{S}_0 = S_n$ holds. □

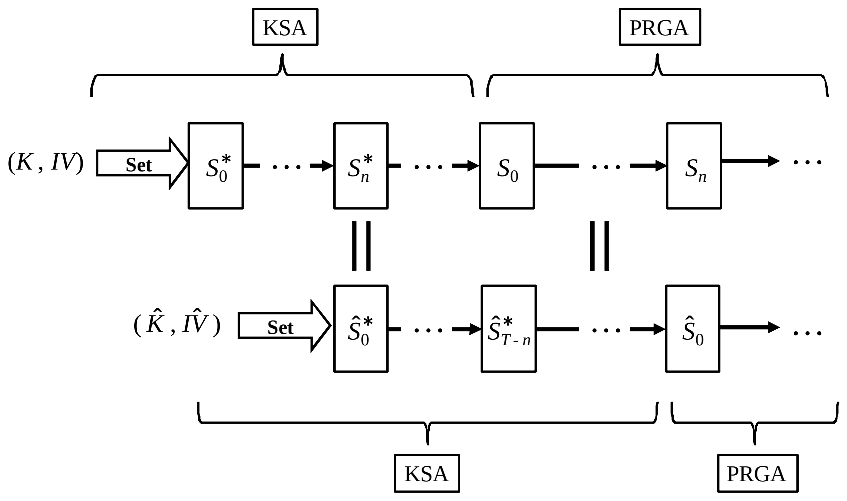

Fig. 3. Conditions of the phase shifting equivalent keys.

Figure 3 shows that Condition 1 and Condition 2 of Theorem 1.

Next, we calculate a probability that any (K, IV) pair has a phase shifting equivalent key $(\hat{K}, \hat{IV})$. Let P_1 be a probability that Condition 1 holds, and P_2 be a probability that Condition 2 holds. Suppose that Condition 1 and Condition 2 are independent. Then, P_f, which is the probability that any (K, IV) pair has a phase shifting equivalent key, is given as

$$P_f = P_1 \cdot P_2. \tag{9}$$

4 Phase Shifting Equivalent Keys of DECIM v2

4.1 Conditions for 1-bit Phase Shifting Equivalent Keys

We discuss conditions for 1-bit phase shifting equivalent keys of DECIM v2. In DECIM v2, Condition 1 of Theorem 1 for 1-bit phase shifting equivalent keys is written as follows:

$$S_1^* = \hat{S}_0^*. \tag{10}$$

In addition, Condition 2 of that is written as follows:

$$PRGA\_nextstate(S_0) = KSA\_nextstate(\hat{S}_{767}^*). \tag{11}$$

We consider conditions for satisfying Eq. (10). From Eqs. (5) and (6), S_1^* is given as follows:

$$x_{i,1}^* = \begin{cases} K_{i+1} & \text{if } i = 0, \dots, 78, \\ K_{i-79} \oplus IV_{i-79} & \text{if } i = 79, \dots, 142, \\ K_{i-79} \oplus IV_{i-143} \oplus IV_{i-127} \oplus IV_{i-111} \oplus IV_{i-95} & \text{if } i = 143, \dots, 158, \\ IV_{i-159} \oplus IV_{i-127} \oplus 1 & \text{if } i = 159, \dots, 190, \\ lv_0^* \oplus v_0^* & \text{if } i = 191. \end{cases} \quad (12)$$

A 1-bit phase shifting equivalent key $(\hat{K}, \hat{IV})$ is defined as

$$\hat{K}_i = \begin{cases} K_{i+1} & \text{if } i = 0, \dots, 78, \\ K_0 \oplus IV_0 & \text{if } i = 79, \end{cases} \quad (13)$$

$$\hat{IV}_i = \begin{cases} IV_{i+1} & \text{if } i = 0, \dots, 62, \\ IV_0 \oplus IV_{16} \oplus IV_{32} \oplus IV_{48} & \text{if } i = 63. \end{cases} \quad (14)$$

Then, from Eqs. (5) and (12)–(14), two conditions for satisfying Eq. (10) are given as follows:

$$IV_0 \oplus IV_{32} \oplus K_0 = 1, \quad (15)$$

$$IV_0 \oplus IV_{16} \oplus IV_{48} \oplus 1 = lv_0^* \oplus v_0^*. \quad (16)$$

We consider a condition for satisfying Eq. (11). From Eqs. (1) and (6), $KSA\_Nextstate(\ )$ and $PRGA\_Nextstate(\ )$ are identical if $v_{t-1}^* = 0$ holds in the KSA. Thus, the condition for satisfying Eq. (11) is as follows:

$$\hat{v}_{767}^* = 0. \quad (17)$$

If Eqs. (15)–(17) hold, then $\hat{Y}$ is 1-bit shifted sequence from Y, that is $\hat{Y} = (\hat{y}_0, \hat{y}_1, \hat{y}_2, \dots) = (y_1, y_2, \dots)$ holds. Note that Y and $\hat{Y}$ are not keystreams but inputs of the ABSG. Keystreams Z and $\hat{Z}$ are obtained from Y and $\hat{Y}$ by the ABSG. In the ABSG, Z is synchronized with $\hat{Z}$ after the time that symbols "00" or "11" are found in Y. After the synchronization, Z and $\hat{Z}$ are promised same sequences except the first bit. Thus, Z and $\hat{Z}$ have equivalence. Moreover, Z and $\hat{Z}$ are specific sequences, which are $(0, 0, 0, \dots)$ or $(1, 1, 1, \dots)$, before the synchronization. Therefore, Z and $\hat{Z}$ generated from 1-bit phase shifting equivalent keys can be detected.

The cases that the phase of Z and $\hat{Z}$ does not shift exist. For example, if (y_0, y_1, y_2) is $(0, 0, 0)$ or $(1, 1, 1)$, then Z is equal to $\hat{Z}$. It is means that such (K, IV) and $(\hat{K}, \hat{IV})$ are typical equivalent keys. Moreover, if (y_0, y_1, y_2) is $(0, 0, 1)$ or $(1, 1, 0)$, then Z is equal to $\hat{Z}$ except z_0.

4.2 Rate of 1-bit Phase Shifting Equivalent Keys

We calculate a rate that any (K, IV) pair has a 1-bit phase shifting equivalent key $(\hat{K}, \hat{IV})$ by using Eq. (9).

Suppose that (K, IV) is selected randomly. Then, a probability that Eq. (15) holds is 1/2. Moreover, since v_0^* is a 1-bit pseudo random variable, a probability that Eq. (16) holds is 1/2. Therefore, the probability that Eq. (10) holds, which is P_1, is given as follows:

$$P_1 = \frac{1}{4}. \tag{18}$$

Since $\hat{v}_{767}^*$ is a 1-bit pseudo random variable, a probability that Eq. (17) holds is 1/2. Therefore, the probability that Eq. (11) holds, which is P_2, is given as follows:

$$P_2 = \frac{1}{2}. \tag{19}$$

Finally, a rate that any (K, IV) pair has a 1-bit phase shifting equivalent key, which is P_f, is given as follows:

$$P_f = P_1 \cdot P_2 = \frac{1}{8}. \tag{20}$$

5 A Key Recovery Attack against DECIM v2 using 1-bit Phase Shifting Equivalent Keys

In this section, we propose a key recovery attack against DECIM v2 using 1-bit phase shifting equivalent keys. We focus on 1-bit phase shifting equivalent keys with same IV, namely, $IV = \hat{IV}$ holds. When $IV = \hat{IV}$ holds, we can search K and $\hat{K}$ in parallel. It means that a key can be found faster than an exhaustive key search when such an IV is used.

5.1 Phase shifting equivalent keys with same IV

We discuss 1-bit phase shifting equivalent keys that $IV = \hat{IV}$ holds. From Eq. (14), the condition for $IV = \hat{IV}$ is given as follows:

$$IV_i = \begin{cases} IV_{i+1} & \text{if } i = 0, \dots, 62, \\ IV_0 \oplus IV_{16} \oplus IV_{32} \oplus IV_{48} & \text{if } i = 63. \end{cases} \tag{21}$$

Equation (21) holds only when $IV = (0, 0, \dots, 0)$ is used. Thus, we focus on 1-bit phase shifting equivalent keys when $IV = (0, 0, \dots, 0)$ is used.

First, we disscuss conditions for 1-bit phase shifting equivalent keys $\hat{K}$ under the condition that $IV = (0, 0, \dots, 0)$. When $IV = (0, 0, \dots, 0)$ is used, a condition for $\hat{K}$ is given from Eqs. (13) and (15) as follows:

$$\hat{K}_{79} = K_0 = 1. \tag{22}$$

Moreover, a condition for $\hat{K}$ is given from Eq. (16) as follows:

$$lv_0^* \oplus v_0^* = 1. \tag{23}$$

In order to satisfy Eq. (23), it is necessary to satisfy $lv_0^* = 1 \wedge v_0^* = 0$ or $lv_0^* = 0 \wedge v_0^* = 1$. We consider a case of $lv_0^* = 1 \wedge v_0^* = 0$. When $lv_0^* = 1 \wedge v_0^* = 0$, the following conditions for K is given from Eqs. (2), (3), and (5) and $IV = (0, 0, \dots, 0)$.

$$K_{77} \oplus K_{76} \oplus K_{75} \oplus K_{52} \oplus K_{51} \oplus K_{46} \oplus K_{17} \oplus K_{16} \oplus K_{14} \oplus K_5 \oplus K_0 = 0, \tag{24}$$

$$(K_{65} + K_{64} + K_{54} + K_{45} + K_{31} + K_{28} + K_{24} + K_{13}) \bmod 4 = 2 \text{ or } 3. \tag{25}$$

By using Eq. (13), Eqs. (24) and (25) are rewritten as follows:

$$\hat{K}_{79} \oplus \hat{K}_{76} \oplus \hat{K}_{75} \oplus \hat{K}_{74} \oplus \hat{K}_{51} \oplus \hat{K}_{50} \oplus \hat{K}_{45} \oplus \hat{K}_{16} \oplus \hat{K}_{15} \oplus \hat{K}_{13} \oplus \hat{K}_4 = 0, \tag{26}$$

$$(\hat{K}_{64} + \hat{K}_{63} + \hat{K}_{53} + \hat{K}_{44} + \hat{K}_{30} + \hat{K}_{27} + \hat{K}_{23} + \hat{K}_{12}) \bmod 4 = 2 \text{ or } 3. \tag{27}$$

In addition, we consider a case of $lv_0^* = 0 \wedge v_0^* = 1$. When $lv_0^* = 0 \wedge v_0^* = 1$, the following conditions for $\hat{K}$ is given in a manner similar to the case of $lv_0^* = 1 \wedge v_0^* = 0$.

$$\hat{K}_{79} \oplus \hat{K}_{76} \oplus \hat{K}_{75} \oplus \hat{K}_{74} \oplus \hat{K}_{51} \oplus \hat{K}_{50} \oplus \hat{K}_{45} \oplus \hat{K}_{16} \oplus \hat{K}_{15} \oplus \hat{K}_{13} \oplus \hat{K}_4 = 1, \tag{28}$$

$$(\hat{K}_{64} + \hat{K}_{63} + \hat{K}_{53} + \hat{K}_{44} + \hat{K}_{30} + \hat{K}_{27} + \hat{K}_{23} + \hat{K}_{12}) \bmod 4 = 0 \text{ or } 1. \tag{29}$$

Next, we show a method to recover K from $\hat{K}$. From Eqs. (13) and (22), a following relation of between K and $\hat{K}$ is given as

$$K_{(i+1) \bmod 80} = \hat{K}_i \qquad \text{for } \forall i \in \{0, 1, \dots, 79\}. \tag{30}$$

Therefore, an original key K can be recovered from the equivalent key $\hat{K}$ by Eq. (30).

5.2 Proposed Attack and its Time Complexity

We propose a chosen IV key recovery attack using 1-bit phase shifting equivalent keys. Our attack searches $\hat{K}$ using Eqs. (22) and (26)–(29) when $IV = (0, 0, \dots, 0)$ is used. Specifically, our attack searches 2^{78} candidates that satisfy Eqs. (22), (26), and (27) or Eqs. (22), (28), and (29). If K has $\hat{K}$ with a probability of $P_f = 1/8$, $\hat{K}$ can be found with a probability of one from the 2^{78} candidates. Then, K is recovered from $\hat{K}$ by Eq. (30). On the other hand, K is found from 2^{78} candidates with a probability of $P_s = 2^{78}/2^{80} = 1/4$. Since our attack can search K and $\hat{K}$ in parallel, K can be recovered more efficient than an exhaustive key search.

We calculate the time complexity of our attack. When our attack searches for 2^{78} candidates, the following four events occur: E_1 (K and $\hat{K}$ can be found), E_2 (only K can be found), E_3 (only $\hat{K}$ can be found), and E_4 (not found). Suppose that K and $\hat{K}$ are found independently. Then, probability that each event occurs

Table 1. Events when our attack searches 2^{78} candidates and probabilities that each event occurs.

Event	Probability that the event occurs	Recoverable keys
E_1	$P_s \cdot P_f = 1/32$	K and $\hat{K}$
E_2	$P_s \cdot (1 - P_f) = 7/32$	K
E_3	$(1 - P_s) \cdot P_f = 3/32$	$\hat{K}$
E_4	$(1 - P_s) \cdot (1 - P_f) = 21/32$	—

is given as Table 1. When E_1 occurs, a key K can be recovered if either K or $\hat{K}$ is found. In this case, a time complexity for recovering the key is 2^{77}. When E_4 occurs, a key K cannot be recovered from 2^{78} candidates. In this case, K is found from other $(2^{80} - 2^{78})$ candidates. Thus, a time complexity for recovering the key is 2^{80} when E_4 occurs. Then, the total time complexity for recovering the key is obtained as follows:

$$\begin{aligned} &\Pr(E_1) \cdot 2^{77} + \Pr(E_2) \cdot \cdot 2^{78} + \Pr(E_3) \cdot 2^{78} + \Pr(E_4) \cdot 2^{80} \\ &= \frac{1}{32} \cdot 2^{77} + \frac{7}{32} \cdot 2^{78} + \frac{3}{32} \cdot 2^{78} + \frac{21}{32} \cdot 2^{80} = 2^{79.56} \ (< 2^{80}). \qquad (31) \end{aligned}$$

Therefore, our attack can recover an 80-bit key with a time complexity of $2^{79.56}$.

6 Conclusion

This paper has presented the key recovery attack against DECIM v2 using phase shifting equivalent keys. Our attack can recover an 80-bit key with a time complexity of $2^{79.56}$ when $IV = (0, 0, \ldots, 0)$ is used. It means that DECIM v2 is not necessarily secure against the key recovery attack. Moreover, our attack can easily apply to DECIM-128 [8]. Applying our attack to other IVs is future work.

References

1. M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf`
2. Ö. Küçük, "Slide resynchronization attack on the initialization of Grain 1.0," eSTREAM, Report 2006/044, 2006, available at `http://www.ecrypt.eu.org/stream/papersdir/2006/044.ps`
3. T. Isobe, T. Ohigashi, H. Kuwakado, and M. Morii, "A chosen-IV attack against Grain," Proc. Information and Communication System Security, ICSS2007-3, pp.15–20, Sep. 2007.
4. C. De Cannière, Ö. Küçük, and B. Preneel "Analysis of Grain's initialization algorithm," Workshop Record of SASC 2008, pp.43–56, Feb. 2008.
5. T. Isobe, T. Ohigashi, H. Kuwakado, and M. Morii, "A key recovery attack with equivalent keys of stream cipher," Technical Report of IEICE, ISEC2007-110, pp.69–74, Nov. 2007.

6. C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "DECIM v2," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/decim/decim_p3.pdf`
7. eSTREAM, ECRYPT Stream Cipher Project, IST-2002-507932, available at `http://www.ecrypt.eu.org/stream/`
8. C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "DECIM-128," eSTREAM, available at `http://www.ecrypt.eu.org/stream/p3ciphers/decim/decim128_p3.pdf`