IEICE TRANS. INF. & SYST., VOL.E92–D, NO.1 JANUARY 2009

PAPER

A Chosen-IV Key Recovery Attack on Py and Pypy

Takanori ISOBE[†*a)], **Toshihiro OHIGASHI**[†**], **Hidenori KUWAKADO**[††], ***and*** **Masakatu MORII**[††], ***Members***

SUMMARY In this paper, we propose an effective key recovery attack on stream ciphers Py and Pypy with chosen IVs. Our method uses an internal-state correlation based on the vulnerability that the randomization of the internal state in the KSA is inadequate, and it improves two previous attacks proposed by Wu and Preneel (a WP-1 attack and a WP-2 attack). For a 128-bit key and a 128-bit IV, the WP-1 attack can recover a key with 2^{23} chosen IVs and time complexity 2^{72}. First, we improve the WP-1 attack by using the internal-state correlation (called a P-1 attack). For a 128-bit key and a 128-bit IV, the P-1 attack can recover a key with 2^{23} chosen IVs and time complexity 2^{48}, which is $1/2^{24}$ of that of the WP-1 attack. The WP-2 attack is another improvement on the WP-1 attack, and it has been known as the best previous attack against Py and Pypy. For a 128-bit key and a 128-bit IV, the WP-2 attack can recover a key with 2^{23} chosen IVs and time complexity 2^{24}. Second, we improve the WP-2 attack by using the internal-state correlation as well as the P-1 attack (called a P-2 attack). For a 128-bit key and a 128-bit IV, the P-2 attack can recover a key with 2^{23} chosen IVs and time complexity 2^{24}, which is the same capability as that of the WP-2 attack. However, when the IV size is from 64 bits to 120 bits, the P-2 attack is more effective than the WP-2 attack. Thus, the P-2 attack is the known best attack against Py and Pypy.

key words: *cryptanalysis, stream cipher, Py, Pypy, eSTREAM, key recovery attack*

1. Introduction

Symmetric-key cryptography, which uses the same key for encryption and decryption, provides security (e.g. confidentiality and integrity) to data or network. Stream cipher is a class of symmetric-key cryptography, and it is suitable for high-speed applications (e.g. applications for large-scale data processing). A typical stream cipher encrypts a plaintext by XORing it with a pseudo-random sequence (called a keystream) that is generated from a secret key and an initialization vector (IV). The core of the stream cipher is to generate the keystream from the secret key and the IV.

Py [1] is a software-oriented stream cipher, and it was designed by Biham and Seberry in April 2005. Py was submitted to the ECRYPT stream cipher project (eSTREAM) [2], which is a project to identify a portfolio of promising new stream ciphers and takes multi-year effort from 2004 to 2008. The speed of Py is more than 2.5 times faster than that of RC4 [3] on a Pentium Ⅲ processor. RC4 is a widely-used software-oriented stream cipher. Py uses a variable-length key and a variable-length key IV. The key size varies from 1 byte to 256 bytes. The IV size varies from 1 byte to 64 bytes. The recommended parameters for security are that the key size is up to 32 bytes and the IV size is up to 16 bytes. The Py algorithm consists of a key scheduling algorithm (KSA) and a pseudorandom number generation algorithm (PRGA).

Security of a stream cipher is analyzed under the assumption that a part of a keystream is given. Since Biham, who is one of designers of Py, is known as a researcher who did significant works, Py attracted the attention of many cryptanalysts. As the cryptanalysis of Py, a distinguishing attack, which distinguishes a keystream from a random stream, was proposed by Paul, Preneel, and Sekar in December 2005 [4]. Their attack was improved by Crowley in January 2006 [5]. In order to resist these distinguishing attacks, a new version of Py, called Pypy, was proposed by Biham and Seberry in March 2006 [6]. Wu and Preneel pointed out a weakness of the IV setup algorithms of Py and Pypy in August 2006 [7]. The IV setup algorithms of Py and Pypy are identical. The weakness is that two keystreams generated from the chosen IVs can be identical with a high probability. They used the weakness to construct a key recovery attack in September 2006 [8] (called a WP-1 attack). In the WP-1 attack, if the IV size is more than 9 bytes, $(IVsizeb-9)$ bytes of the key can be recovered with $(IVsizeb-4)\times 2^{19}$ chosen IVs, where $IVsizeb$ stand for the size of the IV in bytes. For a 128-bit key and a 128-bit IV, which are recommended parameters for security [1], 7 bytes of the key can be recovered with 2^{23} chosen IVs. Thus, for a 128-bit IV, the WP-1 attack can recover a 128-bit key with 2^{23} chosen IVs and time complexity 2^{72}. Another key recovery attack against Py and Pypy was proposed by Wu and Preneel in February 2007 [9] and May 2007 [10] (called a WP-2 attack). The fundamental idea of the WP-2 attack is same as the WP-1 attack. For a 128-bit key and a 128-bit IV, 13 bytes of the key can be recovered with 2^{23} chosen IVs. Thus, for 128-bit IV, the WP-2 attack can recover a 128-bit key with 2^{23} chosen IVs and time complexity 2^{24}.

In this paper, we first improve the WP-1 attack [8] (called a P-1 attack). The P-1 attack can recover more 3-byte key information than the WP-1 attack by using the internal-state correlation. The P-1 attack has new two effective processes as compared to those of Wu and Preneel. These two

Manuscript received February 6, 2008.
Manuscript revised August 8, 2008.

[†]The authors are with the Graduate School of Science and Technology, Kobe University, Kobe-shi, 657–8501 Japan.
[††]The authors are with the Graduate School of Engineering, Kobe University, Kobe-shi, 657–8501 Japan.
[*]Presently, with the System Technologies Laboratories, Sony Corporation.
[**]Presently, with the Information Media Center, Hiroshima University.
a) E-mail: Takanori.Isobe@jp.sony.com

DOI: 10.1587/transinf.E92.D.32

processes are called an expansion process and a comparison process, respectively. In the P-1 attack, if the IV size is more than 7 bytes, $(IVsizeb-6)$ bytes of the key can be recovered with $(IVsizeb-4)\times 2^{19}$ chosen IVs. For a 128-bit key and a 128-bit IV, 10 bytes of the key can be recovered with 2^{23} chosen IVs. Thus, for a 128-bit IV, the P-1 attack can recover a 128-bit key with 2^{23} chosen IVs and time complexity 2^{48}, which is $1/2^{24}$ of that of the WP-1 attack. Second, we improve the WP-2 attack [9] by using the internal-state correlation as well as the P-1 attack (called a P-2 attack). For a 128-bit IV, the P-2 attack can recover a 128-bit key with 2^{23} chosen IVs and complexity 2^{24}, which is the same capability as that of the WP-2 attack. However, when the IV size is from 8 bytes to 15 bytes, the P-2 attack is more effective than the WP-2 attack. The P-2 attack is the known best attack against Py and Pypy.

We published the idea of the P-1 attack in December 2006 [11] (in eSTREAM) and February 2007 [12] (in SASC 2007). Later, the WP-2 attack was proposed by Wu and Preneel in February 2007 [9] (in SASC 2007) and May 2007 [10] (in Eurocrypt 2007). In this paper, we show the P-2 attack which is an improvement on the WP-2 attack for the first time.

We explain why we focus on Py and Pypy. We have the following opinion from our experience of stream-cipher analysis: if the internal-state size is much larger than the key size, the KSA needs much computational cost in order to map the key information to the internal state randomly. In other words, if the KSA of a stream cipher is designed to be faster in general to achieve good key agility and the randomization of the large internal state is not sufficient, then bytes of the internal-state tend to correlate each other. We think that attacks based on the internal-state correlation are effective against such stream ciphers, where the internal-state correlation indicates that a part of the internal state includes information on the other internal state. Recall Py and Pypy. Since the internal-state size of Py and Pypy is 1300 bytes and the key size is at most 32 bytes, the internal-state size is much larger than the key size. Indeed, the internal-state size of Py and Pypy is much larger than that of other stream ciphers. Moreover, we think that the computational cost for randomizing the internal state in the KSA of Py and Pypy is not sufficiently-large, although the KSA has good key agility. Hence, we considered that Py and Pypy were especially vulnerable to an attack based on the above opinion.

This paper is organized as follows. Section 2 describes the Key and IV Setup of Py and Pypy. Section 3 describes the WP-1 attack. Section 4 shows the P-1 attack. Section 5 explains the WP-2 attack and shows the P-2 attack. Section 6 concludes this paper.

2. Description of Py and Pypy

Py and Pypy are stream ciphers that use a variable-length key and an IV. The key size varies from 1 byte to 256 bytes. The IV size varies from 1 byte to 64 bytes. The recommended parameters for security are that the key size is up to 32 bytes and the IV size is up to 16 bytes. Internal states of Py and Pypy contain two arrays P and Y and a 4-byte variable s. P is an array of 256 bytes that contains a permutation of all the values $0,\ldots,255$, and Y is an array of 260 4-byte words indexed as $-3,\ldots,256$. The Py and Pypy algorithms consist of the KSA and PRGA. In the KSA, the key and IV are expanded into internal states of Py and Pypy. In the PRGA, a keystream is generated from internal states of Py and Pypy. Py and Pypy have the same KSAs and different PRGAs. In the each step of the PRGA, Py generates an 8-byte keystream, while Pypy generates only a 4 byte keystream to resist the distinguishing attacks [4], [5]. The KSA consists of a Key setup, an IV setup1, and an IV setup2. The Key setup initializes Y with the key. The IV setup1 initializes P, s, and EIV with Y and the IV, where EIV is an array with the same size as IV. The IV setup2 updates Y, P, and s with EIV. We now provide a description of the Key setup and the IV setup1 and omit the description of the IV setup2 and the PRGA, since our attacks use only the Key setup and the IV setup1.

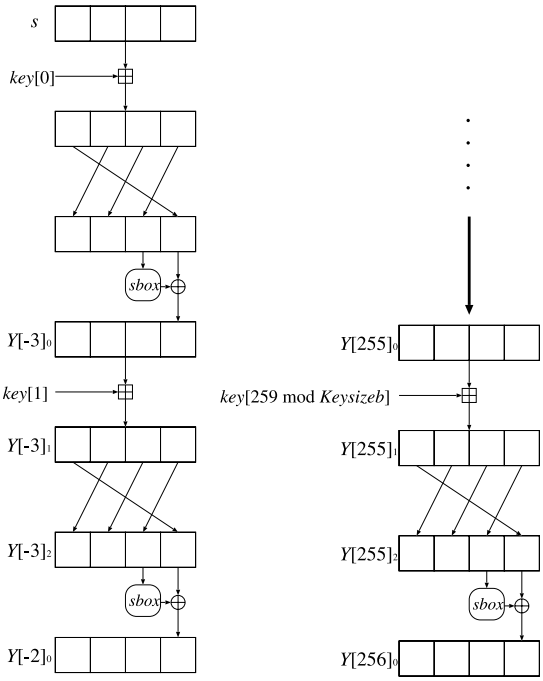

Fig. 1 Key setup.

Our descriptions of the Key setup and the IV Setup have some differences from the original ones to explain our attack effictually.

2.1 Key Setup

In the Key setup, Y is initialized with the key. Figure 1 shows the structure of the Key setup. The main process in the Key setup is divided into three processes listed as follows:

$$Y[i]_1 = (Y[i]_0 + key[(i+4) \bmod Keysizeb]) \bmod 2^{32}, \quad (1)$$

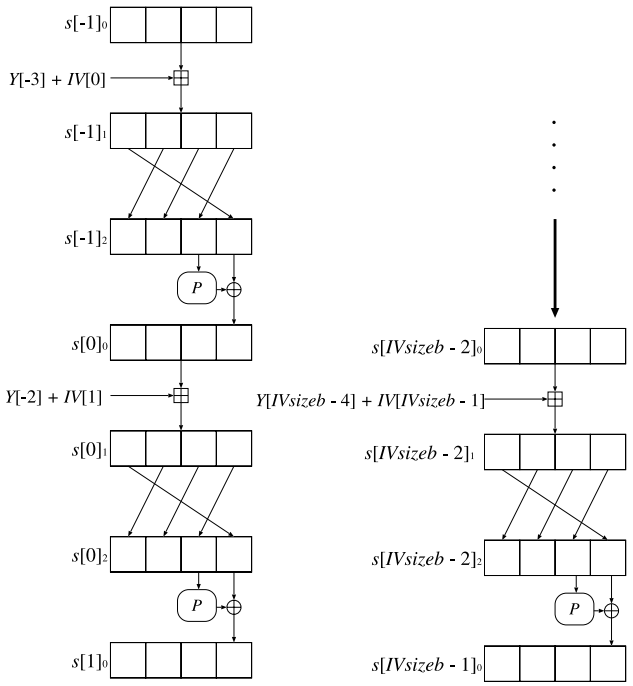

Fig. 2 IV setup1.

$$Y[i]_{2,j} = Y[i]_{1,(j-1) \bmod 4} \ (j = 0, 1, 2, 3), \tag{2}$$

$$Y[i+1]_{0,j} = \begin{cases} Y[i]_{2,0} \oplus sbox[Y[i]_{2,1}] \ (j = 0) \\ Y[i]_{2,j} \ (j = 1, 2, 3), \end{cases} \tag{3}$$

where $Y[i]_0$, $Y[i]_1$, and $Y[i]_2$ are the outputs of each process, $key[i]$ is the value of the $(i+1)$-th byte of the key, $Keysizeb$ is the size of the key in bytes, $Y[i]_{l,j}$ is the value of the $(j+1)$-th byte of $Y[i]_l$, $sbox[\cdot]$ is a substitution box that accepts a 1-byte input and yields a 1-byte output.

First, s is obtained from the key during the preprocessing. Second, the values of $Y[i]_0$ $(-3 \le i \le 256)$ are decided from s and Eqs. (1)–(3), and the values are outputted to the IV setup as $Y[i]$ $(-3 \le i \le 256)$. In this paper, we omit the description of the preprocessing stage.

2.2 IV Setup1

In the IV setup1, P, s, and EIV are initialized with the IV and Y. Figure 2 shows the structure of the IV setup1. P can be expressed as

$$v = IV[0] \oplus Y[0]_{0,2}, \tag{4}$$

$$d = (IV[1 \bmod IVsizeb] \oplus Y[1]_{0,2}) | 1, \tag{5}$$

$$P[i] = sbox[v + i \cdot d], \ i \in \{0, 1, \ldots, 255\}, \tag{6}$$

where v and d are 1-byte variables, $IV[i]$ is the value of the $(i+1)$-th byte of the IV, and $IVsizeb$ is the size of the IV in bytes.

The main process in the IV setup1 is divided into four processes, which are expressed as

$$s[i]_1 = (s[i]_0 + Y[i-2] + IV[i+1]) \bmod 2^{32}, \tag{7}$$

$$s[i]_{2,j} = s[i]_{1,(j-1) \bmod 4} \ (j = 0, 1, 2, 3), \tag{8}$$

$$EIV[i+1] = P[s[i]_{2,1}], \tag{9}$$

$$s[i+1]_{0,j} = \begin{cases} s[i]_{2,0} \oplus P[s[i]_{2,1}] \ (j = 0) \\ s[i]_{2,j} \ (j = 1, 2, 3), \end{cases} \tag{10}$$

where $s[i]_0$, $s[i]_1$, and $s[i]_2$ are the outputs of each process.

The values of $EIV[i]$ $(0 \le i \le IVsizeb - 1)$ can be determined from the abovementioned equations. $s[-1]_0$ is given by

$$s[-1]_0 = ((v \ll 24) \oplus (d \ll 16) \oplus (P[254] \ll 8) \oplus (P[255])) \oplus (Y[-3] + Y[256]). \tag{11}$$

$EIV[i]$ $(0 \le i \le IVsizeb - 1)$ are initialized by using Eqs. (7)–(10). Next, $EIV[i]$ $(0 \le i \le IVsizeb - 1)$ are updated by using

$$s'[i]_1 = (s'[i]_0 + Y[255 - i] + IV[i+1]) \bmod 2^{32}, \tag{12}$$

$$s'[i]_{2,j} = s'[i]_{1,(j-1) \bmod 4} \ (j = 0, 1, 2, 3), \tag{13}$$

$$EIV'[i+1] = EIV[i+1] + P[s'[i]_{2,1}], \tag{14}$$

$$s'[i+1]_{0,j} = \begin{cases} s'[i]_{2,0} \oplus P[s'[i]_{2,1}] \ (j = 0) \\ s'[i]_{2,j} \ (j = 1, 2, 3), \end{cases} \tag{15}$$

where s' and EIV' indicate the updated values of s and EIV respectively. $s'[-1]$ is given as $s[IVsizeb - 1]_0$.

3. WP-1 Attack

We describe the WP-1 attack [8] to understand the P-1 attack in Sect. 4 and the WP-2 attack [9], [10] in Sect. 5.1. The WP-1 attack is based on the weakness of the IV setup algorithm of Py and Pypy. The weakness is that two keystreams generated from the chosen IVs can be identical with a high probability [7].

3.1 Identical Keystreams

From Eqs. (4)–(6), only 15 bits of the IV ($IV[0]$ and $IV[1]$) are used to initialize the array P. For an IV pair, if the 15 bits are identical, then the resulting arrays P are the same. From [7], if IV_1 and IV_2 have only a two-byte difference and satisfy the following four conditions, then this type of an IV pair results in identical keystreams with a probability $2^{-23.2}$.

Condition 1: $IV_1[i] \oplus IV_2[i] = 1 \quad (1 \le i)$,

Condition 2: $IV_1[i+1] \ne IV_2[i+1] \quad (1 \le i)$,

Condition 3: The least significant bit of $IV_1[i]$ is 1 $\quad (1 \le i)$,

Condition 4: $IV_1[j] = IV_2[j] \quad (0 \le j < i, \ i+1 < j \le IVsizeb - 1)$,

where i is a fixed value.

For the abovementioned IV pair, if two keystreams are identical, then the two $s[i+1]_0$ are the same. This implies

$$(P[B(s[i-1]_{0,0} + IV_1[i] + Y[-3+i]_{0,0})]$$

$$\oplus B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_{i1})) + 256 + IV_1[i+1]$$
$$= (P[B(s[i-1]_{0,0} + IV_2[i] + Y[-3+i]_{0,0}))]$$
$$\oplus B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_{i2})) + IV_2[i+1], \quad (16)$$

where $B(x)$ is a function that provides the least significant byte of x, and ξ_{i1} and ξ_{i2} are the carry bits introduced by $IV[i]$ and $Y[-3+i]$. The probability obtaining $\xi_{i1} = \xi_{i2}$ is very close to 1, since $IV[i]$ has a negligible effect on ξ_{i1} and ξ_{i2}.

3.2 Recovery of a Part of Y from the Identical IV Pairs

From Eq. (16) such that $s[i-1]_{0,0} = s[i-1]_{0,3}$, we can recover values of $B(s[i-1]_{0,0} + Y[-3+i]_{0,0})$ and $B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i)$. Equation (16) such that $s[i-1]_{0,0} = s[i-1]_{0,3}$ can be generated if the first i bytes of all the IVs are the same. From [8], if there are seven Eqs. (16), the values of $B(s[i-1]_{0,0} + Y[-3+i]_{0,0})$ and $B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i)$ can be recovered.

After recovering the values of $B(s[i-1]_{0,0} + Y[-3+i]_{0,0})$ and $B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i)$ for $i \geq 1$, $s[i]_{0,0}$ can be recovered as follows:

$$s[i]_{0,0} = P[B(s[i-1]_{0,0} + IV[i]^{\theta} + Y[-3+i]_{0,0})]$$
$$\oplus B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i), \quad (17)$$

where $IV[i]^{\theta}$ is a fixed IV. In the WP-1 attack, we use arbitrary fixed IV^{θ}. When differences are introduced in $IV[i]$ and $IV[i+1]$, all the first i bytes of each IV are chosen to be identical to those of IV^{θ}.

We introduce the IV difference at the $(i+1)$-th and $(i+2)$-th bytes. The first $i+1$ bytes of each IV are identical to those of IV^{θ}. Then, we recover the value of $B(s[i]_{0,0} + Y[-2+i]_{0,0})$. From the value of $B(s[i]_{0,0} + Y[-2+i]_{0,0})$ and $s[i]_{0,0}$, we determine the value of $Y[-2+i]_{0,0}$. Figure 3 shows the method for recovering a part of Y from s.

3.3 Generating the Equations

Here, we demonstrate a method for obtaining several equations such that $s[i-1]_{0,0} = s[i-1]_{0,3}$.

To ensure that the same values of $s[i-1]_{0,0}$ and $s[i-1]_{0,3}$ appear in these equations, we need to fix the values of $IV[j]$ $(0 \leq j < i)$. Let the least significant bit of $IV[i]$ and $IV[i+1]$ choose all the 512 values and $IV[j]$ $(0 \leq j < i)$ choose $IV^{\theta}[j]$ $(0 \leq j < i)$. Let $IV[j]$ $(i+2 \leq j \leq IVsize - 1)$ choose the optional fixed values. Then we obtain $2^{16} (\approx 255 \times 255)$ desired IV pairs. These 512 IVs are termed as the desired IV group. From [7], this type of IV pair results in identical keystreams with a probability $2^{-23.2}$. We obtain $2^{-7.2} (= 2^{-23.2} \times 2^{16})$ identical keystream pairs from one desired IV group. We modify the value of the 7 most significant bits of $IV[i]$ and 3 bits of $IV[i+2]$; we can then obtain $2^{10} (= 2^7 \times 2^3)$ desired IV groups. From these desired IV groups, we obtain $7 (= 2^{10} \times 2^{-7.2})$ Eqs. (16). There are $2^{19} (= 2^7 \times 2^3 \times 2^9)$ IVs used in the attack. With $(IVsizeb - 4) \times 2^{19}$ IVs, we can recover $B(s[i-1]_{0,0} + Y[-3+i]_{0,0})$, $B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i)$, and $s[i]_{0,0}$ $(2 \leq i \leq IVsizeb - 3)$ for IV^{θ}. Thus, we can recover the value of $Y[-3+i]_{0,0}$ $(3 \leq i \leq IVsizeb - 3)$.

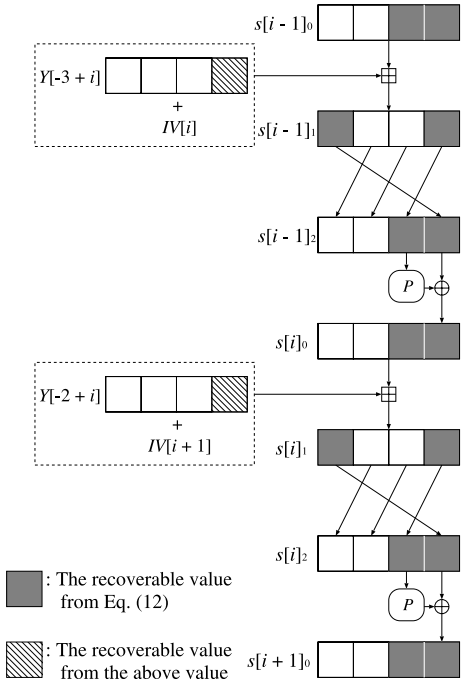

Fig. 3 Recovery of a part of Y from s.

3.4 Key Recovery

We now show how to recover the key from $Y[-3+i]_{0,0}$ $(3 \leq i \leq IVsizeb - 3)$. From Eqs. (1)–(3), the relation between the key and Y is given by

$$(B(Y[-3+i]_{0,0} + key[i+1] + \xi'_i)$$
$$\oplus sbox[B(Y[-3+i+3]_{0,0} + key[i+4])]$$
$$= Y[-3+i+4]_{0,0}, \quad (18)$$

where ξ'_i is the carry bit introduced by $key[i+2]$ and $key[i+3]$; it is computed by using $\xi'_i \approx (key[i+2] + Y[-3+i+1]_{0,0}) \ll 8$. The value of ξ'_i is 0 with a probability approximately 0.5.

When $Y[-3+i]_{0,0}$ $(3 \leq i \leq IVsizeb - 3)$ are known, Eq. (18) becomes an expression that relates $key[i+1]$ and $key[i+4]$ for $(3 \leq i \leq IVsizeb - 7)$. Once we correctly guess the values of $key[4], key[5]$, and $key[6]$, we can determine the other key bytes $key[j]$ $(6 < j \leq IVsizeb - 3)$. Thus, the values of $key[j]$ $(4 \leq j \leq IVsizeb - 3)$ can be restricted to 2^{24} values. This indicates that $(IVsize - 9)$ bytes of the key constitutes the leaked information.

From the description given above, in the WP-1 attack, $(IVsizeb - 9)$ bytes of the key can be recovered with $(IVsizeb - 4) \times 2^{19}$ chosen IVs. Thus, time complexity for recovering a key is $2^{8\times(Keysizeb-(IVsizeb-9))}$. For a 128-bit key and a 128-bit IV, time complexity for recovering a key is 2^{72} $(= 2^{8\times(16-(16-9))})$.

4. P-1 Attack

We improve the WP-1 attack with the internal-state correlation. The WP-1 attack and the P-1 attack are different in the process after recovering a part of a Y. The P-1 attack comprises new two effective processes using the internal-state correlation. These two processes are called an expansion process and a comparison process, respectively. In the expansion process, $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb-3)$ are expanded into the candidate values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$. In the comparison process, the values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ are determined by comparing the candidate values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ with $s[i]$ $(2 \le i \le IVsizeb-6)$ recovered from Eqs. (16) and (17). From these processes, the P-1 attack can use the values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ for recovering the key while the WP-1 attack use only $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb-3)$. By using the values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$, $key[j]$ $(4 \le j \le IVsizeb-3)$ can be recovered. Thus, the P-1 attack can recover more 3-byte key information than the WP-1 attack.

4.1 Expansion Process

Theorem 1: If $Y[i]_0$ and $Y[i-1]_{0,0}$ are known, then $key[(i+3) \bmod Keysizeb]$, $Y[i-1]_{0,1}$, $Y[i-1]_{0,2}$, and $Y[i-1]_{0,3}$ can be determined from

$$Y[i-1]_{2,j} = \begin{cases} Y[i]_{0,0} \oplus sbox[Y[i]_{0,1}] \ (j=0) \\ Y[i]_{0,j} \ (j=1,2,3), \end{cases} \quad (19)$$

$$Y[i-1]_{1,j} = Y[i-1]_{2,(j+1) \bmod 4} \ (j=0,1,2,3), \quad (20)$$

$$key[(i+3) \bmod Keysizeb] = B(Y[i-1]_{1,0} - Y[i-1]_{0,0}), \quad (21)$$

$$Y[i-1]_0 = (Y[i-1]_1 - key[(i+3) \bmod Keysizeb]) \bmod 2^{32}. \quad (22)$$

Proof. Eqs. (19)–(22) are derived from Eqs. (1)–(3). First, $Y[i-1]_1$ is determined from $Y[i]_0$, Eq. (19), and Eq. (20). Second, $key[(i+3) \bmod Keysizeb]$ is determined from $Y[i-1]_{0,0}$, $Y[i-1]_{1,0}$, and Eq. (21). Finally, $Y[i-1]_0$ is determined from $Y[i-1]_1$, $key[(i+3) \bmod Keysizeb]$, and Eq. (22). Hence, $Y[i-1]_{0,1}$, $Y[i-1]_{0,2}$, and $Y[i-1]_{0,3}$ can be determined. □

The expansion process is executed after recovering a part of Y from the identical IV pair in the WP-1 attack. In the expansion process, we obtain the candidate values of $Y[-3+i]_{0,1}$, $Y[-3+i]_{0,2}$, $Y[-3+i]_{0,3}$ $(3 \le i \le IVsizeb-3)$, and $key[j]$ $(4 \le j \le IVsizeb-3)$ from $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb-3)$ by going the Key setup algorithm backward. The algorithm of the expansion process is given below. Figure 4 shows the expansion process.

Algorithm 1:

Input: $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb-3)$.

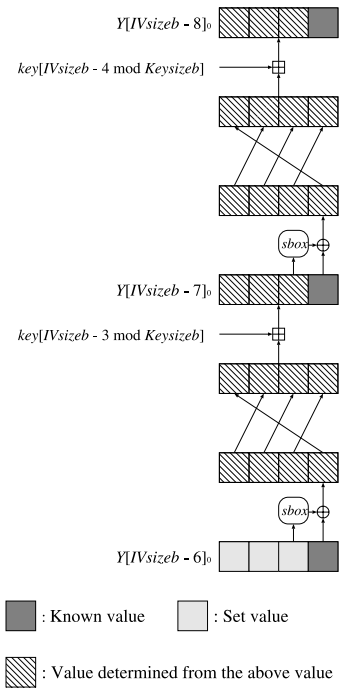

Fig. 4 Expansion process.

Output: The candidate values of $Y[-3+i]_{0,1}$, $Y[-3+i]_{0,2}$, $Y[-3+i]_{0,3}$ $(3 \le i \le IVsizeb-3)$, and $key[j]$ $(4 \le j \le IVsizeb-3)$.

Step 1: If $Y[IVsizeb-6]_{0,1}$, $Y[IVsizeb-6]_{0,2}$, and $Y[IVsizeb-6]_{0,3}$ have been set to all 2^{24} values, terminate this algorithm. Otherwise, set those values that have not been set yet to $Y[IVsizeb-6]_{0,1}$, $Y[IVsizeb-6]_{0,2}$, and $Y[IVsizeb-6]_{0,3}$.

Step 2: Substitute 0 into *num*.

Step 3: Determine $key[((IVsizeb-6)-num+3) \bmod Keysizeb]$, $Y[(IVsizeb-6)-num-1]_{0,1}$, $Y[(IVsizeb-6)-num-1]_{0,2}$, and $Y[(IVsizeb-6)-num-1]_{0,3}$ from Eqs. (19)–(22), $Y[(IVsizeb-6)-num]_0$, and $Y[(IVsizeb-6)-num-1]_{0,0}$ (Theorem 1).

Step 4: Add 1 to *num*.

Step 5: If $num = IVsizeb-6$, go to Step 1. Otherwise, go to Step 3.

From the above algorithm, if $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb-3)$ are known, once we correctly guess the values of $Y[IVsizeb-6]_{0,1}$, $Y[IVsizeb-6]_{0,2}$, and $Y[IVsizeb-6]_{0,3}$, we can determine the values of $key[j]$ $(4 \le j \le IVsizeb-3)$ and $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$. This implies that the values of $key[j]$ $(4 \le j \le IVsizeb-3)$ and $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ can be restricted to only 2^{24} values.

4.2 Comparison Process

Theorem 2: If $s[i]_{0,0}$, $s[i]_{0,1}$, $s[i+1]_{0,0}$, $s[i+1]_{0,1}$, $s[i+2]_{0,0}$, $s[i+2]_{0,1}$, $Y[i-2]$, $Y[i-1]$, $Y[i]$, $IV[i+1]$, $IV[i+2]$, and $IV[i+3]$ are known, $s[i+2]_{1,3}$ can be determined from Eqs. (7), (8), and (10) as follows:

$$s[i]_1 \bmod 2^{16} = (s[i]_0 \bmod 2^{16} + Y[i-2] \bmod 2^{16} + IV[i+1]) \bmod 2^{16}, \quad (23)$$

$$s[i+1]_1 \bmod 2^{24} = (s[i+1]_0 \bmod 2^{24} + Y[i-1] \bmod 2^{24} + IV[i+2]) \bmod 2^{24}. \quad (24)$$

Proof. Eqs. (23) and (24) are derived from Eq. (7). First, $s[i+1]_{0,2}$ is determined from $s[i]_{0,0}$, $s[i]_{0,1}$, $Y[i-2]$, $IV[i+1]$, Eq. (8), Eq. (10), and Eq. (23). Second, $s[i+2]_{0,2}$ and $s[i+2]_{0,3}$ are determined from $s[i+1]_{0,0}$, $s[i+1]_{0,1}$, $s[i+1]_{0,2}$, $Y[i-1]$, $IV[i+2]$, Eq. (8), Eq. (10), and Eq. (24). Finally, $s[i+2]_1$ is determined from $s[i+2]_0$, $Y[i]$, $IV[i+3]$, and Eq. (7). Hence, $s[i+2]_{1,3}$ can be determined. □

The comparison process is executed after the expansion process. In the comparison process, we determine the correct values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ from 2^{24} candidate values and recover the values of $key[j]$ $(4 \le j \le IVsizeb-3)$.

For IV^θ, $B(s[i-1]_{0,0} + Y[-3+i]_{0,0})$, $B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i)$, and $s[i]_{0,0}$ $(2 \le i \le IVsizeb-3)$ are known. Hence, $s[i-1]_{1,0}$, $s[i-1]_{1,3}$, $s[i]_{0,0}$, and $s[i]_{0,1}$ $(2 \le i \le IVsizeb-3)$ can be determined from Eqs. (7), (8), and (10). From Theorem 2, $s[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$ can be recovered from $s[i]_{0,1}$, $s[i]_{0,0}$ $(2 \le i \le IVsizeb-3)$, $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$, and IV^θ. Thus, by comparing $s[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$ recovered from Eqs. (16) and (17) with $s^*[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$, we can determine the correct values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$, where $s^*[i]$ indicates the value determined from the candidate values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$.

The algorithm of the comparison process is given below. Figure 5 shows the comparison process.

Algorithm 2:

Input: $s[i-1]_{1,0}, s[i-1]_{1,3}, s[i]_{0,1}, s[i]_{0,0}$ $(2 \le i \le IVsizeb-3)$, the candidate values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$, and IV^θ.

Output: $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$.

Step 1: Set one candidate value of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ to $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$.

Step 2: From Theorem 2, determine $s^*[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$ from the candidate values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$, $s[i]_{0,1}$, $s[i]_{0,0}$ $(2 \le i \le IVsizeb-3)$, and IV^θ.

Step 3: Compare $s^*[i+2]_{1,3}(2 \le i \le IVsizeb-6)$ with $s[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$ recovered from Eq. (16). If $s^*[i+2]_{1,3} = s[i+2]_{1,3}$ $(4 \le i \le IVsizeb-3)$, output the candidate values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb-3)$ as the correct values. Otherwise, go to Step 1.

We discuss $s^*[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$ and $s[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$ recovered from Eq. (16) in Step 3. If the values set in Step 1 are correct, then $s^*[i+2]_{1,3} = s[i+2]_{1,3}$ $(2 \le i \le IVsizeb-6)$. Theoretically, since the probability that $s^*[4]_{1,3} = s[4]_{1,3}$ is 1/256, on comparing $s^*[4]_{1,3}$ with $s[4]_{1,3}$, we can determine 2^{16} candidate values from 2^{24} candidate values. Supposing that the probabilities of $s^*[4]_{1,3} = s[4]_{1,3}$, $s^*[5]_{1,3} = s[5]_{1,3}$, and $s^*[6]_{1,3} = s[6]_{1,3}$ are independent, we can determine 2^8 candidate values by comparing $s^*[4]_{1,3}$ with $s[4]_{1,3}$ and $s^*[5]_{1,3}$ with $s[5]_{1,3}$. We can determine one candidate value by comparing $s^*[4]_{1,3}$ with $s[4]_{1,3}$, $s^*[5]_{1,3}$ with $s[5]_{1,3}$, and $s^*[6]_{1,3}$ with $s[6]_{1,3}$.

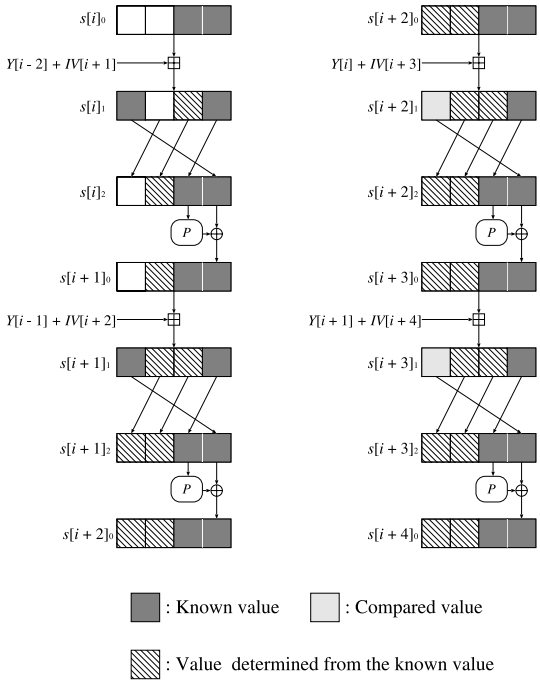

Fig. 5 Comparison process.

From the simulation for 10^4 keys, we were able to determine $65536.019 \approx 2^{16}$ candidate values by comparing $s^*[4]_{1,3}$ with $s[4]_{1,3}$, and $255.084 \approx 2^8$ candidate values by comparing $s^*[4]_{1,3}$ with $s[4]_{1,3}$ and $s^*[5]_{1,3}$ with $s[5]_{1,3}$, and 1.384 candidate values by comparing $s^*[4]_{1,3}$ with $s[4]_{1,3}$, $s^*[5]_{1,3}$ with $s[5]_{1,3}$, and $s^*[6]_{1,3}$ with $s[6]_{1,3}$. If the number of comparisons is more than three, only one value can be identified.

From the algorithm given above, we determine the values of $Y[-3+i]_0$ $(3 \le i \le IVsize-3)$ and $key[j](4 \le j \le IVsize-3)$. If $IVsize \ge 10$, these values can be restricted to one value from more than two comparisons. If $IVsize = 9$, these values can be restricted to 2^8 values from two comparisons. If $IVsize = 8$, these values can be restricted to 2^{16} values from one comparison.

4.3 Evaluation

In the P-1 attack, $(IVsizeb-6)$ bytes of the key can be recovered with $(IVsizeb-4) \times 2^{19}$ chosen IVs. Thus, time complexity for recovering the key is the sum of that of recovering $(IVsizeb-6)$ bytes of the key with $(IVsizeb-4) \times 2^{19}$ chosen IVs and that of exhaustive remaining key search. Since time complexity of the exhaustive remaining key search is very larger than that of recovering $(IVsizeb-6)$ bytes of the

Table 1 Time complexity for recovering a 128-bit (16-byte) key.

$IVsizeb$	8	9	10	11	12	13	14	15	16
The WP-1 attack [8]	2^{128}	2^{128}	2^{120}	2^{112}	2^{104}	2^{96}	2^{88}	2^{80}	2^{72}
The P-1 attack	2^{120}	2^{112}	2^{96}	2^{88}	2^{80}	2^{72}	2^{64}	2^{56}	2^{48}
The WP-2 attack [9], [10]	2^{128}	2^{128}	2^{112}	2^{96}	2^{80}	2^{64}	2^{48}	2^{32}	2^{24}
The P-2 attack	2^{112}	2^{96}	2^{64}	2^{48}	2^{32}	2^{24}	2^{24}	2^{24}	2^{24}

key with $(IVsizeb - 4) \times 2^{19}$ chosen IVs, time complexity for recovering a key is approximated to that of exhaustive remaining key search which is $2^{8\times(Keysizeb-(IVsizeb-6))}$.

Time complexity of the WP-1 attack [8] and that of the P-1 attack for a 128-bit key are shown in Table 1. Time complexity of the WP-1 attack can be calculated from Sect. 3.4 in this paper. For a 128-bit key and a 128-bit IV, which are recommended parameters for security [1], the P-1 attack can recover the key with 2^{23} chosen IV and time complexity 2^{48} $(= 2^{8\times(16-(16-6))})$. In addition, when $IVsizeb = 8$ and 9, the WP-1 attack is ineffective; in this case, the P-1 attack can recover the key with lower time complexity than the exhaustive key search.

5. Improvement on the WP-2 Attack

In this section, we first explain the WP-2 attack [9], [10] which is the improvement of the WP-1 attack. Second, we propose the P-2 attack which is the improvement on the WP-2 attack. The P-2 attack comprises new two effective processes using the internal-state correlation as well as the P-1 attack. By using these processes, when the IV size is from 8 bytes to 15 bytes, the P-2 attack is more effective than the WP-2 attack.

5.1 WP-2 Attack

The WP-2 attack uses the same chosen IVs as the WP-1 attack. From [7], for these chosen IV, if two keystreams are identical, we obtain Eq. (16) and

$$\begin{aligned}&(P[B(s'[i-1]_{0,0} + IV_1[i] + Y[256-i]_{0,0})]\\ &\oplus B(s'[i-1]_{0,3} + Y[256-i]_{0,3} + \xi_{i1})) + 256 + IV_1[i+1]\\ &= (P[B(s'[i-1]_{0,0} + IV_2[i] + Y[256-i]_{0,0}))]\\ &\quad \oplus B(s'[i-1]_{0,3} + Y[256-i]_{0,3} + \xi_{i2})) + IV_2[i+1].\end{aligned} \tag{25}$$

The WP-2 attack uses these two equations while the WP-1 attack uses only Eq. (16). By applying an attack similar to the WP-1 attack, we can recover $B(s'[i-1]_{0,0} + Y[256-i]_{0,0})$, $B(s'[i-1]_{0,3} + Y[256-i]_{0,3} + \xi_i)$, and $s'[i]_{0,0}$ $(2 \le i \le IVsizeb - 3)$ as well as $B(s[i-1]_{0,0} + Y[-3+i]_{0,0})$, $B(s[i-1]_{0,3} + Y[-3+i]_{0,3} + \xi_i)$, and $s[i]_{0,0}$ $(2 \le i \le IVsizeb - 3)$ for IV^θ. Thus, we can recover the values of $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb - 3)$ and $Y[256-i]_{0,0}$ $(3 \le i \le IVsizeb - 3)$.

When $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb - 3)$ and $Y[256-i]_{0,0}$ $(3 \le i \le IVsizeb - 3)$ are known, Eq. (18) becomes the expression that relates to $key[j+1]$ and $key[j+4]$ $(3 \le j \le IVsizeb - 7,\ 262 - ivsizeb \le j \le 252)$. Thus, there are $2 \times (IVsizeb - 9)$ expressions (18) linking the key bytes. For a 128-bit key and a 128-bit IV, 14 expressions (18) can be obtained: 7 expressions (18) linking $key[j]$ and $key[j+3]$ $(4 \le j \le 10)$, and another 7 expressions (18) linking $key[j]$ and $key[j+3 \bmod 16]$ $(7 \le j \le 13)$. There are 13 bytes involved in these 14 expressions (18). Since these 14 expressions are sufficient to recover the involved 13 key bytes, time complexity for recovering a key is 2^{24}. For $IVsizeb \le 15$, since the number of expressions (18) is less than that of involved key bytes, time complexity for recovering a key is $2^{8\times(Keysizeb-2\times(IVsizeb-9))}$. For a 128-bit key and a 120-bit IV, time complexity for recovering a key is 2^{32} $(= 2^{8\times(16-2\times(15-9))})$. The number of the chosen IVs used for the attack is $(IVsizeb - 4) \times 2^{19}$ as well as the WP-1 attack.

5.2 P-2 Attack

The P-2 attack has the expansion process and the comparison process after recovering a part of a Y in the WP-2 attack as well as the P-1 attack.

In the expansion process, we obtain 2^{24} candidate values of $Y[-3+i]_{0,1}, Y[-3+i]_{0,2}, Y[-3+i]_{0,3}$ $(3 \le i \le IVsizeb - 3)$, and $key[j]$ $(4 \le j \le IVsizeb - 3)$ from $Y[-3+i]_{0,0}$ $(3 \le i \le IVsizeb - 3)$ by using Algorithm 1. Moreover, we can obtain 2^{24} candidate values of $Y[256-i]_{0,1}, Y[256-i]_{0,2}, Y[256-i]_{0,3}$ $(3 \le i \le IVsizeb - 3)$, and $key[j \bmod 16]$ $(23 - IVsizeb \le j \le 16)$ from $Y[256-i]_{0,0}$ $(3 \le i \le IVsizeb - 3)$ by using the algorithm similar to Algorithm 1. Thus, the values of $key[j]$ $(4 \le j \le IVsizeb - 3)$ and $Y[-3+i]_0$ $(3 \le i \le IVsizeb - 3)$, and $key[j \bmod 16]$ $(23 - IVsizeb \le j \le 16)$ and $Y[256-i]_0$ $(3 \le i \le IVsizeb - 3)$ can be restricted to only 2^{24} values respectively.

In the comparison process, we determine the values of $key[j]$ $(4 \le j \le IVsizeb - 3)$ and $key[j \bmod 16]$ $(23 - IVsizeb \le j \le 16)$. From the P-1 attack, we can determine the correct values of $Y[-3+i]_0$ $(3 \le i \le IVsizeb - 3)$ and recover the values of $key[j]$ $(4 \le j \le IVsizeb - 3)$. For IV^θ, $B(s'[i-1]_{0,0} + Y[256-i]_{0,0})$, $B(s'[i-1]_{0,3} + Y[256-i]_{0,3} + \xi_i)$, and $s'[i]_{0,0}$ $(2 \le i \le IVsizeb - 3)$ are known from the WP-2 attack. Hence, $s'[i-1]_{1,0}, s'[i-1]_{1,3}, s'[i]_{0,0}$, and $s'[i]_{0,1}$ $(2 \le i \le IVsizeb - 3)$ can be determined from Eqs. (12), (13), and (15). $s'[i+2]_{1,3}$ $(2 \le i \le IVsizeb - 6)$ can be recovered from $s'[i]_{0,1}, s'[i]_{0,0}$ $(2 \le i \le IVsizeb - 3)$, $Y[256-i]_0$ $(3 \le i \le IVsizeb - 3)$, and IV^θ. By using the algorithm similar to Algorithm 2, we can determine the correct values of $Y[256-i]_0$ $(3 \le i \le IVsizeb - 3)$ and recover the values of $key[j \bmod 16]$ $(23 - IVsizeb \le j \le 16)$.

From the algorithm given above, we determine the values of $key[j] (4 \le j \le IVsize - 3)$ and $key[j \bmod 16]$ $(23 -$

$IVsizeb \leq j \leq 16$). If $IVsize \geq 10$, these values can be restricted to one value from more than two comparisons. If $IVsize = 9$, these values can be restricted to 2^8 values from two comparisons. If $IVsize = 8$, these values can be restricted to 2^{16} values from one comparison.

5.3 Evaluation

In the P-2 attack, we can recover the values of $key[j](4 \leq j \leq IVsize - 3)$ and $key[j \bmod 16]$ $(23 - IVsizeb \leq j \leq 16)$ with $(IVsizeb - 4) \times 2^{19}$ chosen IVs. For a 128-bit key and a 128-bit IV, $key[j \bmod 16]$ $(4 \leq j \leq 16)$ can be recovered with 2^{23} chosen IVs. Since remaining key values are $key[1]$, $key[2]$, and $key[3]$, time complexity for recovering a key is 2^{24}. Similarly, for a 128-bit key and an 88-bit IV, since remaining key values are $key[1]$, $key[2]$, $key[3]$, $key[9]$, $key[10]$, and $key[11]$, time complexity for recovering a key is 2^{48}.

Time complexity of the P-2 attack and that of the WP-2 attack [9], [10] for a 128-bit key are shown in Table 1. Time complexity of the WP-2 attack can be calculated from Sect. 5.1 in this paper. When the IV size is from 8 to 15 bytes, the P-2 attack is more effective than the WP-2 attack. In addition, when $IVsizeb = 8, 9$, the WP-2 attack is ineffective; in this case, the P-2 attack can recover the key with lower time complexity than the exhaustive key search and is more effective than the P-1 attack.

6. Conclusion

In this paper, we have proposed the P-1 attack and P-2 attack which are improvements on the WP-1 attack and WP-2 attack, respectively. These proposed attacks comprise new two effective processes using the internal-state correlation. As a result, the P-1 attack can recover more 3-byte key information than the WP-1 attack. For a 128-bit key and a 128-bit IV, the P-1 attack can recover the key with time complexity of 2^{48}, which is $1/2^{24}$ of that of the WP-1 attack. When $IVsizeb$ is from 8 to 15 bytes, the P-2 attack can recover the 128-bit key with less time complexity than the WP-2 attack. In particular, when $IVsizeb$ is from 13 to 16 byte, the P-2 attack can recover a 128-bit key with time complexity of 2^{24}. In Py and Pypy, since a variable-length IV is used, in some cases the IV may range in size from 8 bytes to 15 bytes. Thus, the P-2 attack is the known best attack against Py and Pypy.

Our attacks show that the attack based on the internal-state correlation is effective against a stream cipher such that the internal-state size is large and the randomization of the internal state in the KSA is not sufficient. Thus, the ideas of our attacks are applicable to Py-like ciphers.

Acknowledgements

We thank anonymous reviewers for useful comments. We also thank Associate Editor Shinsaku Kiyomoto for his helpful comments.

References

[1] E. Biham and J. Seberry, "Py(Roo): A fast and secure stream cipher using rolling arrays," eSTREAM, Report 2005/023, 2005. available at http://www.ecrypt.eu.org/stream/ciphers/py/py.ps
[2] http://www.ecrypt.eu.org/stream/
[3] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons (Computers), 1995.
[4] S. Paul, B. Preneel, and G. Sekar, "Distinguishing attack on stream cipher Py," eSTREAM, Report 2005/081, 2005. available at http://www.ecrypt.eu.org/stream/papersdir/081.pdf
[5] P. Crowley, "Improved cryptanalysis of Py," eSTREAM, Report 2006/010, 2006. available at http://www.ecrypt.eu.org/stream/papersdir/2006/010.pdf
[6] E. Biham and J. Seberry, "Pypy: Another version of Py," eSTREAM, Report 2006/038, 2006. available at http://www.ecrypt.eu.org/stream/papersdir/2006/038.pdf
[7] H. Wu and B. Preneel, "Attacking the IV setup of Py and Pypy," eSTREAM, Report 2006/050, 2006. available at http://www.ecrypt.eu.org/stream/papersdir/2006/050.pdf
[8] H. Wu and B. Preneel, "Key recovery attack on Py and Pypy with chosen IVs," eSTREAM, Report 2006/052, 2006. available at http://www.ecrypt.eu.org/stream/papersdir/2006/052.pdf
[9] H. Wu and B. Preneel, "Differential cryptanalysis of the stream ciphers Py, Py6 and Pypy," Workshop Record of SASC 2007, pp.326–339, Jan.-Feb. 2007.
[10] H. Wu and B. Preneel, "Differential cryptanalysis of the stream ciphers Py, Py6 and Pypy," Proc. EUROCRYPT 2007, Lecture Notes in Computer Science, vol.4515, pp.276–290, 2007.
[11] T. Isobe, T. Ohigashi, H. Kuwakado, and M. Morii, "How to break Py and Pypy by a chosen IV attack," eSTREAM, Report 2006/060, 2006. available at http://www.ecrypt.eu.org/stream/papersdir/2006/060.pdf
[12] T. Isobe, T. Ohigashi, H. Kuwakado, and M. Morii, "How to break Py and Pypy by a chosen IV attack," Workshop Record of SASC 2007, pp.340–352, Jan.-Feb. 2007.

Takanori Isobe received the B.E. and M.E. degrees from Kobe University, Japan, in 2006 and 2008, respectively. He joined the System Technologies Laboratories, Sony Corporation in 2008. His current research interests include information security and cryptography. He received the SCIS Paper Award from ISEC group of IEICE in 2008.

Toshihiro Ohigashi received the B.E. and M.E. degrees from the University of Tokushima, Japan, and the D.E. degree from Kobe University in 2002, 2004, and 2008, respectively. Since 2008, he has been an Assistant Professor in the Information Media Center, Hiroshima University. His current research interests include information security and cryptography. He received the SCIS 20th Anniversary Award from ISEC group of IEICE in 2003. He is a member of the Information Processing Society of Japan.

Hidenori Kuwakado received the B.E., M.E. and D.E. degrees from Kobe University in 1990, 1992, and 1999 respectively. He worked for Nippon Telegraph and Telephone Corporation from 1992 to 1996. From 1996 to 2002 he was a Research Associate in the Faculty of Engineering, Kobe University. From 2002 to 2007, he was an Associate Professor in the Faculty of Engineering, Kobe University. Since 2007, he has been an Associate Professor in Graduate School of Engineering, Kobe University. His research interests are in cryptography and information security.

Masakatu Morii received the B.E. degree in electrical engineering and the M.E. degree in electronics engineering from Saga University, Saga, Japan, and the D.E. degree in communication engineering from Osaka University, Osaka, Japan, in 1983, 1985, and 1989, respectively. From 1989 to 1990 he was an Instructor in the Department of Electronics and Information Science, Kyoto Institute of Technology, Japan. From 1990 to 1995 he was an Associate Professor at the Department of Computer Science, Faculty of Engineering at Ehime University, Japan. From 1995 to 2005 he was a Professor at the Department of Intelligent Systems and Information Science, Faculty of Engineering at the University of Tokushima, Japan. Since 2005, he has been a Professor at the Department of Electrical and Electronics Engineering, Faculty of Engineering at Kobe University, Japan. His research interests are in error correcting codes, cryptography, discrete mathematics and computer networks and information security. Dr. Morii is a member of the IEEE, the Information Processing Society of Japan and the Society of Information Theory and Its Applications.