出力破棄の時刻を利用した ABSG の解析

恩賀　嶺　　　　森井　昌克

神戸大学大学院工学研究科
657-8501 神戸市灘区六甲台町 1-1

onga@stu.kobe-u.ac.jp, mmorii@kobe-u.ac.jp

あらまし　圧縮関数は擬似乱数生成器やストリーム暗号等に用いられており，安全性の一部を保障している．ABSG はストリーム暗号 DECIM v2 に利用されている圧縮関数である．我々は ABSG の出力が捨てられた時刻の観測が可能であると想定した時，入力系列の一部を回復する手法を提案した．本稿ではこの時刻の間隔に着目して解析を行い，より多くの入力系列が回復可能であることを示す．また，入力系列を探索する際に計算量を削減する手法を明らかにする．

Analysis of ABSG using the timing that the outputs are dropped

Rei Onga　　　　Masakatu Morii

Graduate School of Engineering, Kobe University
1-1, Rokkodai, Nada-ku, Kobe-shi, 657-8501 Japan

onga@stu.kobe-u.ac.jp, mmorii@kobe-u.ac.jp

Abstract A compression function is used for a pseudo-random generator and stream cipher, and it ensures these security. ABSG is a compression function used by a stream cipher DECIM v2. We proposed the technique to recover the exact value of ABSG inputs under the assumption that we can get the timing such that the ABSG output is dropped. This assumption enables us to correlate the timing of the ABSG input and that of the buffer output. Observing the buffer output, we can recover the ABSG inputs. In this paper, focusing attention on the interval of the timing that the ABSG output is dropped, we clarify that more ABSG inputs can be recovered. Furthermore, we propose a technique to reduce the computational complexity to recover the ABSG input sequence using the correlation among the related ABSG inputs.

1　Introduciton

A compression function is used by a pseudo-random generator and a stream cipher in order to guarantee these security. For this reason, the recovery of the inputs of the compression function from its outputs must be difficult. Researchers have been analyzing the compression function to check the difficulty.

The Shrinking Generator (SG) is a compression function proposed by Coppersmith et al. in 1993 [1], which uses two linear feedback shift registers (LFSR). One LFSR controls the outputs of the other one. The Self-Shrinking Generator (SSG) was proposed by Meier et al. in 1994 [2]. The structure of SSG is simpler than SG because SSG requires only one LFSR. Bit-Search Generator (BSG), which is faster than SSG, was proposed by Gouget et al. in 2004 [3]. BSG requires only an 3n-bit on average for outputting an n-bit sequence. Gouget et al. reported that there was a weakness in BSG and proposed two compression functions ABSG and MBSG [4]. ABSG is used in stream cipher DECIM v2 and ensures the security of this cipher [5].

Loe et al. proposed the timing attack which reduces the candidates of the ABSG input sequence from the ABSG output one [6]. They assumed that they can get the timing such that the buffer becomes full before generating a keystream. It implies the Side Channel Attack against ABSG. The timing informs us to the number of inputs required to fill up the buffer before generating the keystream. Because the size of the buffer is known, we can get the number of candidates input sequence by calculating the total number of a combination. On the estimation of an 128-bit input sequence, 2^{128} candidates of input sequence are reduced to 2^{80} when they estimate an 128-bit input sequence.

In [7], we proposed a technique to recover the exact value of bits input to ABSG by introducing a new assumption. When ABSG outputs one bit, a buffer stores the bit if it is not full, otherwise, the bit is discarded. We assumed that we can get this timing. This assumption enables us to correlate the timing of the ABSG input and that of the buffer output. Observing a buffer output, we can recover the exact value of an ABSG input. In this situation, we could recover about 8.33% of the input sequence of ABSG.

In this paper, we further investigate the mutual dependency among the related ABSG inputs under the new assumption. We focus on the interval of the timing that the ABSG output is dropped. When the interval is a specific length, we can recover more ABSG inputs and can reduce the computational complexity to recover the ABSG input sequence. In our simulation, we could recover about 2.08% of the input sequence by this technique. Combining with [7], the recovery rate could be 10.41%.

2 ABSG and Buffer

ABSG is a bit generator that searches for a pattern in the input bitstream and outputs a shorter bitstream. ABSG receives the input sequence $Y = (y_0, y_1, ...)$ and generates the output sequence $Z = (z_0, z_1, ...)$. Buffer stores

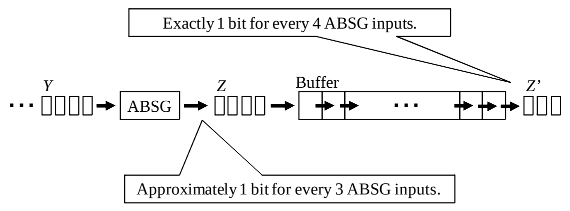
Exactly 1 bit for every 4 ABSG inputs.
Y
ABSG
Z
Buffer
Z'
Approximately 1 bit for every 3 ABSG inputs.

Fig. 1: The process of ABSG.

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

Fig. 2: The algorithm of ABSG.

Z and generates a keystream $Z' = (z'_0, z'_1, ...)$. But if the buffer is full, the ABSG output bit is not added into the queue, i.e. it is dropped. We call this timing "drop timing". Figure 1 shows the process of ABSG.

2.1 ABSG

ABSG receives an 1-bit with each clock and outputs approximately an 1-bit per 3 clocks. The algorithm of ABSG is shown in Fig. 2. From the mechanism of ABSG, we know two important properties.

Property 1. The minimum interval of the ABSG output is 2 clocks.
Referring to Fig. 2, at the step2 and step4 the cycle is incremented. If $y_t = e$ at the step3, the incrementation is omitted. Therefore, the minimum interval is derived when $y_t = e$ at the step3, and then the number of cycles is two.

Property 2. The input patterns for the corresponding output bit is summarized in Table 1, where "0^{α}" stands for the string of 0 with length α.

Table 1: How to compress the input sequence.

Input Pattern	Output Bit
$\{00, 10^{\alpha}1\}$	0
$\{11, 01^{\alpha}0\}$	1

Table 1 shows that substrings of the form 00, 101, 1001, ... are compressed to 0, while substrings of the form 11, 010, 0110, ... are compressed to 1. From this property, if we know the values of output bits and α, we can recover the ABSG inputs.

2.2 Buffer

Before generating a keystream, the buffer stores the ABSG outputs until it is full. Then, the internal state t in Fig. 2 is incremented by the repetition of the algorithm of ABSG. For convenience, we define the keystream generation phase starts at $t = 1$. Thus, the process that the buffer stores the ABSG outputs until it is full is performed at the duration $t \in \{..., -1, 0\}$.

The buffer implemented in DECIM v2 consists of an 32-bit internal state $B_{\tilde{t}} = (b_{0,\tilde{t}}, b_{1,\tilde{t}}, ..., b_{31,\tilde{t}})$, where $b_{i,\tilde{t}}$ is an 1-bit variable at a timing $\tilde{t}$. Buffer stores the ABSG outputs and generates keystream $z'_{\tilde{t}}$. The correlation between t and $\tilde{t}$ is given as follows;

$$\tilde{t} = \left\lfloor \frac{t-1}{4} \right\rfloor. \quad (1)$$

Buffer outputs a bit which is stored in the buffer $b_{0,\tilde{t}}$ as a keystream at $t = 4T$ ($T = 1, 2, ...$). At the time, $b_{31,\tilde{t}}$ becomes empty and the other stored buffer values are shifted as follows;

$$b_{i,\tilde{t}+1} = b_{i+1,\tilde{t}} \text{ for } 0 \leq i \leq 30. \quad (2)$$

3 Conventional Analysis

We proposed the technique to recover the exact value of ABSG inputs if we know the drop timing and the timing that the buffer becomes full before generating a keystream [7].

Table 2: The classification of the timing t when the drop is occurred.

$b_{31,\tilde{t}} \leftarrow z_n$	z_{n+1} is dropped
$4T + 1$	$4T + 3, 4T + 4$
$4T + 2$	$4T + 4$
-3	$-1, 0, 1, 2, 3, 4$
-2	$0, 1, 2, 3, 4$
-1	$1, 2, 3, 4$
0	$2, 3, 4$

3.1 Drop Timing

After $b_{31,\tilde{t}}$ becomes empty at $t = 4T$, we suppose the case such that ABSG outputs during the timing $t \in \{4T+1, 4T+2, 4T+3, 4T+4\}$. When ABSG outputs at $t = 4T + 1$, we define the bit z_n which stands for the (n+1)-th output value, ABSG may output further one bit z_{n+1} at $t \in \{4T+3, 4T+4\}$ from the Property 1. In such a case, the bit z_{n+1} is discarded as z_n has been already assigned to $b_{31,\tilde{t}}$. Similarly, when ABSG outputs z_n at $t = 4T + 2$, the buffer may output z_{n+1} at $t = 4T + 4$ and it is discarded.

There are the other cases that the drop of the ABSG output is occurred. Though the buffer becomes full before generating a keystream at $t = 0$, the buffer may have been already full at timing $t \in \{-3, -2, -1\}$ because the buffer checks for every four clocks if it is full or not. If the buffer becomes full at $t = -3$, an ABSG output at $t \in \{-1, 0\}$ is discarded. However, ABSG does not generate a keystream at $t = 0$, $b_{31,\tilde{t}}$ does not become empty at that times. Thus, if the buffer becomes full at $t = -3$, the ABSG output at $t \in \{-1, 0, 1, 2, 3, 4\}$ is discarded. Similarly, the other cases can be enumerated. Table 2 summarizes the cases of timing t.

3.2 Recovering the ABSG input

If we get a drop timing, we can recover an ABSG input. When z_{n+1} is dropped at t, the value of y_{t+1} is certainly stored in $b_{31,\tilde{t}}$. The value stored in $b_{31,\tilde{t}}$ is shifted as shown in Eqs. (1) and (2), and some times later, the buffer

Table 3: The correlation between the drop timig and the ABSG inputs.

drop timing	recoverable y_t
$4T+3$	y_{4T+4}
$4T+4$	y_{4T+5}

outputs the value as a keystream. Observing the keystream, we can recover an ABSG input.

For example, we suppose z_{n+1} is discarded at $t = 4T + 3$. At the time in ABSG internal states, the value of y_{4T+3} is assigned to e, and that of y_{4T+4} to z_{n+2}. Until $y_t = e$, ABSG does not output z_{n+2}, we define the timing $t = \acute{t}$. By the way, the buffer outputs a bit as a keystream and updates their internal states at $t = 4T + 4$. Because Property 1 shows $\acute{t} \geq 4T + 5$, z_{n+2} is stored in the buffer. Since the value is output as z'_{T+32}, the observation of z'_{T+32} gives us the exact value of y_{4T+4}. Table 3 summarizes the correlation between the drop timing and the ABSG input which could be recovered.

4 Proposed Analysis

We focus on the interval of the drop timing. When the interval is a specific length, we can recover more ABSG inputs and can reduce the computational complexity to recover the ABSG input sequence.

4.1 Recovering more ABSG inputs

Due to Property 1, when the ABSG output is dropped at $t = 4T + 3$, the next two outputs of ABSG may be occurred at $t = 4T + 5$ and $t = 4T + 7$. In this case, the ABSG output at $t = 4T + 7$ is dropped. Thus, we can see that the minimum interval of the drop timing is 4 clocks. At that time, we can recover more ABSG inputs. There are two cases that the interval of the drop timing becomes 4 clocks.

Case 1 ABSG outputs are discarded at $t = 4T + 3$ and $t = 4T + 7$.
In this case, the ABSG inputs y_{4T+4} and y_{4T+8} can be recovered from Table 3, and the two timings informs us that the ABSG output at $t = 4T + 1$ and $t = 4T + 5$ are stored in the buffer from the Property 1 and Table 2. Furthermore, the following correlations among the related ABSG inputs are obtained from the Property 2.

$$y_{4T+1} = y_{4T+2}, \tag{3}$$
$$y_{4T+3} = y_{4T+4}, \tag{4}$$
$$y_{4T+5} = y_{4T+6}. \tag{5}$$

Here, using Eq. (4) we can recover y_{4T+3} because we know the value of y_{4T+4}.

Case 2 ABSG outputs are discarded at $t = 4T + 4$ and $t = 4T + 8$.
Similar to the case 1, the ABSG input y_{4T+5} and y_{4T+9} can be recovered, and the ABSG outputs at $t = 4T+6$ is stored in the buffer. The following correlations among the related ABSG inputs are obtained.

$$y_{4T+4} = y_{4T+5}, \tag{6}$$
$$y_{4T+6} = y_{4T+7}. \tag{7}$$

Then, we can recover y_{4T+4}.

4.2 Reducing the computational complexity

From the drop timing and the interval of the timing, we find the correlations among the related ABSG inputs. Using the correlations, we can reduce the computational complexity to recover the ABSG input sequence. There are three cases that the correlation is occurred.

Case 1 ABSG output is discarded at $t = 4T + 3$.
In this case, we know that the ABSG output at $t = 4T + 1$ is stored in the buffer from the Property 1 and Table 2. Here, we can find the correlation of Eq. (3) from the Property 2.

Case 2 ABSG outputs are discarded at $t = 4T + 4$ and $t = 4T + 8$.
As discussed in Sect.4.1, we can find the correlation of Eq. (7).

Case 3 ABSG outputs are discarded at $t = 4T+3$ and $t = 4T+8$.

In this case, the ABSG input y_{4T+4} and y_{4T+9} can be recovered from Table 3, and the ABSG output at $t = 4T+1$ is stored in the buffer from the Property 1 and Table 2. In addition, the ABSG output at $t = 4T+5$ or $t = 4T+6$ is also stored in the buffer.

- $t = 4T+5$.
  ABSG inputs have the correlations of Eqs. (4) and (8) from the Property 2.

$$y_{4T+5} = y_{4T+7} = \overline{y_{4T+6}}. \quad (8)$$

From Eqs. (4) and (8), the following equation can be derived.

$$\begin{aligned}&\{y_{4T+3}, y_{4T+5}, y_{4T+7}\}\\ &= \{y_{4T+4}, \overline{y_{4T+6}}, \overline{y_{4T+6}}\}. \quad (9)\end{aligned}$$

- $t = 4T+6$.
  ABSG inputs have the correlations Eqs. (10) and (11) from the Property 2.

$$y_{4T+3} = y_{4T+5} = \overline{y_{4T+4}}, \quad (10)$$
$$y_{4T+6} = y_{4T+7}. \quad (11)$$

From Eqs. (10) and (11), the following equation can be derived.

$$\begin{aligned}&\{y_{4T+3}, y_{4T+5}, y_{4T+7}\}\\ &= \{\overline{y_{4T+4}}, \overline{y_{4T+4}}, y_{4T+6}\}. \quad (12)\end{aligned}$$

Comparing Eqs. (9) and (12), we know that y_{4T+5} is equal to $\overline{y_{4T+6}}$ or $\overline{y_{4T+4}}$. Here, remember that $\overline{y_{4T+4}}$ is known because we can recover y_{4T+4}. Therefore, when we guess $y_{4T+6} = y_{4T+4}$, we can uniquely determine y_{4T+5}, hence, the recovery process of y_{4T+5} can be skipped. If such a case is often occurred, the computational complexity can be reduced accordingly.

4.3 Simulation Result

First, we checked how often the ABSG outputs are dropped. In this simulation, we used 100,000 patterns of secret keys and generated an 10,000-byte keystream. From the experimental result, we found the average probability that the ABSG outputs is dropped is about 25%. In addition, we investigated the proportion of each case described in Sect. 4.1 and Sect. 4.2 to all the number of dropped bits. Table 4 shows the result.

Table 4: Proportion of each case.

Drop Timing	Proportion
$4T+3$	38.2%
$4T+4$	61.8%
$4T+3$ and $4T+7$	9.55%
$4T+4$ and $4T+8$	15.5%
$4T+3$ and $4T+8$	9.54%

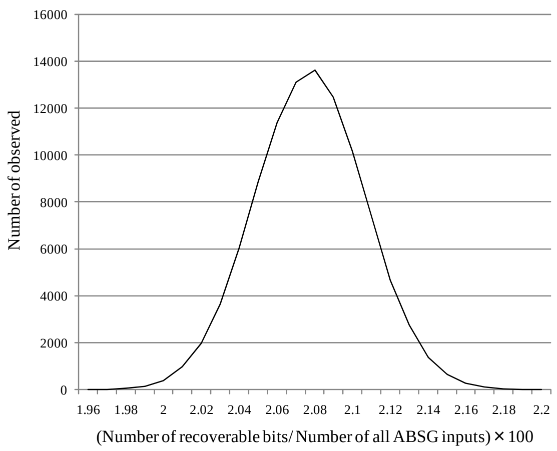

Fig. 3: Recovering the ABSG inputs.

Next, we evaluated the effectiveness of the proposed recovery technique presented in Sect. 4.1. We counted the recoverable number of ABSG inputs from all ABSG inputs, the result are shown in Fig. 3. Figure 3 suggests that we could recover 1.96–2.20% of the ABSG inputs, in average 2.08%. Combining with the technique proposed in [7], the recovery rate could be increased about 10.41%.

5 Conclusion

ABSG is a compression function used by a stream cipher DECIM v2. The ABSG outputs may be dropped because the speed of ABSG outputting is faster than that of buffer outputting. We had showed that we could recover the exact values of ABSG inputs by assuming that we could get the timing that the ABSG outputs were dropped as the Side Channel Attack.

In this paper, we focused on the interval of the timing that the ABSG outputs are dropped. First, we showed that we could recover more ABSG inputs when the interval is a specific length. Second, we reduced computational complexity to recover the ABSG input sequence. Finally, we showed simulation results to estimate how often we can use the new technique and how many bits could be recovered. The result indicated that we could recover in average about 10.41% of the ABSG inputs.

Acknowledgements

We thank Dr. Minoru Kuribayashi for useful discussions.

References

[1] D. Coppersmith, H. Krawczyk, and Y. Mansour, "The Shrinking Generator," *Proc. CRYPTO'93*, Lecture Note in Computer Science, vol. 773, pp.22–39, 1993.

[2] W. Meier and O. Staffelbach, "The Self-Shrinking Generator," *Proc. EUROCRYPT'94*, Lecture Note in Computer Science, vol. 950, pp.205–214, 1994.

[3] A. Gouget and H. Sibert, "The Bit Search Generator" In *The State of the Art of Stream Ciphers: Workshop Record, Brugge, Belgium*, pp.60–68, 2004.

[4] A. Gouget, H. Sibert, C. Berbain, N. Courtois, B. Debraize, and C. Mitchell, "Analysis of the Bit-Search Generator and Sequence Compression Techniques," *Proc. FSE2005*, Lecture Note in Computer Science, vol. 3557, pp.196–214, 2005.

[5] 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`

[6] C. W. Loe and K. Khoo, "Side Channel Attacks on Irregularly Decimated Generators" *Proc. ICISC 2007*, Lecture Note in Computer Science, vol. 4817, pp.116–130, 2007.

[7] R. Onga and M. Morii, "Analysis of DECIM v2 using the timing ABSG output is dropped" *Proc. Joint Workshop on Information Security 2009* (*JWIS 2009*), 2009.