SCIS 2019 2019 Symposium on Cryptography and Information Security
Shiga, Japan, Jan. 22 - 25, 2019
The Institute of Electronics, Information and Communication Engineers

Fault Injection, Simple Power Analysis, and Power Glitch Attacks against FPGA-implemented Xoroshiro128+

Nakjun Choi* Jeeun Lee† Kwangjo Kim*†

Abstract: Random Number Generators (RNGs) are playing an important role in providing a uniqueness of many protocols in various fields such as IoT, Artificial Intelligence, Database, and Information Security in ICT systems. However, since most of the RNGs are implemented in a physical chip, they are always vulnerable to the risk of side channel attack such as timing attack and fault injection attack, etc. In this paper, we attempt to execute side channel attacks on Xoroshiro128+ which is a Pseudo-Random Number Generator (PRNG) designed by Vigna and Blackman. Using the Xoroshiro128+ source code, we implemented this PRNG over the Field Programmable Gate Array (FPGA). We also verify the security of Xoroshiro128+ and suggest countermeasures against side channel attacks using ChipWhisperer-Pro

Keywords: Random Number Generator, Side Channel Attack, Xoroshiro128+, ChipWhisperer

1 Introduction

1.1 Motivation

Random Number Generators (RNGs) are playing an important role in providing a uniqueness of many protocols in various fields such as IoT, Artificial Intelligence, Database, and Information Security in ICT systems. It is especially important to develop high-entropy RNG with security because RNG is essential in the security industry applying cryptographic technology. Accordingly, the development of a True Random Number Generator (TRNG) [1, 2], which generates a complete random number using a physical phenomenon, has been actively studied. In recent years, quantum random number generators (QRNGs) [3, 4] have also been developed using unpredictable quantum mechanics such as photons traveling through semi transparent mirrors, nuclear radiation sources, and quantum mechanical noise of electronic circuits.

However, since most of the RNGs are implemented in a physical chip, they are always vulnerable to the risk of side channel attack such as timing attack and fault injection attack, etc. While RNGs have a number of hardware sources (e.g, electromagnetic waves, computation time, hardware sound, etc.) that can provide useful information to an attacker, RNG's protection systems have obvious limitations. Therefore, since RNG can not cope with all types of side channel attacks, researches [5, 6, 7, 8] that have succeeded in effective side channel attacks against RNGs are being continuously announced.

* Graduate School of Information Security, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. {*cnj*8160, *kkj*}@kaist.ac.kr
† School of Computing, KAIST. 291, Daehak-ro, Yuseong-gu, Daejeon, South Korea 34141. {*jeeun.lee*, *kkj*}@kaist.ac.kr

Based on this point, we decided to attempt several side channel attacks to verify the security of hardware-based PRNG. In order to proceed our experiment, we chose xoroshiro128+ [9, 10] as an attack target. xoroshi ro128+ is a relatively recent version of the xorshift family and is a PRNG with high entropy. To ensure that most randomness tests can be passed, Marsaglia, Vigna and others have developed xorshiro128+ to complement the xorshift. Nevertheless, if hardware-based xoroshiro128+ loses randomness due to side channel attacks, this can be an opportunity to remind modern society of the importance of countermeasures against side channel attacks.

Since xoroshiro128+ source code [11] is provided by the author, we implemented the source code on the FPGA board and then executed various side channel attacks on the board. One method of attack seems to be insufficient to obtain useful information from hardware devices. To analyze as many different hardware sources as possible, we used ChipWhisperer-Pro (CW12 00) [12], an open source embedded security analysis platform. We measured the power consumption required to operate xoroshiro128+ with this platform and executed a power glitch attack [13]. In addition, the skipping fault attack [14] was executed to check the expected random number outputs when the attack on xoroshiro128+ was a success.

In this paper, we describe RNGs and hardware-based side channel attacks and introduce several attack cases that have been executed on PRNGs. We also verify the security of xoroshiro128+ using the FPGA board and ChipWhisperer-Pro. Afterwards, we give our own evaluation of the experiment and suggest countermeasures to prevent side channel attacks. Finally, we summarize the results of the study and discuss the importance of a countermeasures against side channel attacks in RNGs.

1.2 Outline of the Paper

Section 2 discusses several side channel attacks and the attack target of this paper, xoroshiro128+. Section 3 introduces previous work about side channel attack cases on TRNGs. Sections 4 and 5 describe the experimental environment and specific attack methods, respectively. Section 6 evaluates the results of our experiment and Section 7 presents appropriate countermeasures against our attacks. Finally, Section 8 concludes our study and describes future works.

2 Background

2.1 Random Number Generator

A random number is a randomly selected number within a defined range. A random number generated by the RNG should not be able to predict the next random number; should be free of correlation between the consecutive two random numbers; should not be biased toward either. However, in a real computer environment, it is difficult to obtain complete random numbers because random numbers are generated in a deterministic way. Since the random number obtained by the deterministic method is not a true random number, it is called a pseudo-random number, and the code for generating a random number in software is called a Pseudo-Random Number Generator (PRNG).

True-Random Number Generator (TRNG), one of the RNGs, generates random numbers by sampling random phenomena that can not be predicted using physical phenomena rather than computer programs. Recently, Quantum-Random Number Generator (QRNG) has been developed using unpredictable characteristics of quantum mechanics. Since QRNG shows higher randomness than PRNG, researches related to QRNG are being studied widely. Quantis RNG [15] and QRNG Chip of ID Quantique Co., Ltd. of Korea [16, 17] are known to be the popular QRNG products. Random numbers generated by RNG are analyzed by randomness test such as NIST STS [18], dieharder [19] or bitstream to verify randomness.

2.2 Timing Attack

A timing attack uses the method to obtain useful information by measuring the time required for algorithm operation, and it is the easiest method to execute among side channel attacks. Timing attacks, first introduced in [20], are classified as passive and non-invasive attacks, such as power analysis attacks. Timing attacks can be attempted in a relatively simple way, but it is possible to defend against most timing attacks in a way that deliberately creates a time delay in the computation process.

2.3 Power Analysis Attack

A power analysis attack first introduced in [21] is classified as passive and non-invasive attack because it does not directly damage or modify the attack target. Power analysis attacks are divided into Simple Power Analysis (SPA) and Differential Power Analysis (DPA). SPA is an attack method that obtains useful information such as secret key by measuring and analyzing the power change which is consumed in the operation of the encryption device. Also, DPA uses a method of estimating the secret key by comparing the statistical characteristics of the measured power consumption. In order to prevent power analysis attack, it is necessary to use a masking technique which gives randomness to the power consumed in the operation or hides the intermediate value of the full execution.

2.4 Glitch Attack

A glitch attack [13] is an attack on a hardware device that contains a cryptographic processor, and a smart card is the primary attack target. Glitch attacks analyze the circuitry of a hardware device and add unpredictable behavior to the device by adding glitch signals to external power or clock signals. In the case of a glitch attack on a smart card, the internal clock of the smart card may increase, resulting in a fatal result that the authentication process is bypassed. In order to prevent glitch attacks, circuitry must be configured irregularly so that hardware devices are not easily analyzed.

2.5 Xoroshiro128+

xoroshiro128+ [9] is a PRNG intended as a successor to xorshift [22]. xorshift is a PRNG produced by Marsaglia in 2003 and it can generate sequences of $2^{32} - 1$ integers or $2^{128} - 1$ integers depending on the type. It takes the next random number by repeatedly performing xor and Bit-shift operations within the sequence. xorshift has been widely used due to its relatively fast speed and simple code configuration, but it has the disadvantage of not passing some randomness tests [23]. To solve this problem, Vigna proposed xorshift128+ [24], which complemented xorshift in 2017. xorshift128+ has a maximum period of $2^{128} - 1$ and shows very fast computation speed. Vigna asserted that xorshift128+ would pass the BigCrush of TESTU01 [25], which xorshift could not pass. However, Lemire claimed that xorshift128 + still does not pass the BigCru sh test [26].

xoroshiro128+ is a recently proposed PRNG among the xorshift family and is based on xorshift128+. It was created by Vigna in collaboration with Blackman and uses shift/rotate-based linear transformations. As the name suggests, xoroshiro128+ generates a random number through a single xor, a single shift, and double rotate operations, respectively. All researchers can easily get the xoroshiro128+ code because the designs have dedicated all copyright and related rights to this software [11]. We also used the code provided by the author to proceed with our study. xoroshiro128+ shows extremely high entropy, but the authors acknowledge that xoroshiro128+ has a very mild Hamming-weight dependency making their test fail after 8 TB of output [27]. Therefore, it is necessary to study that can overcome these problems.

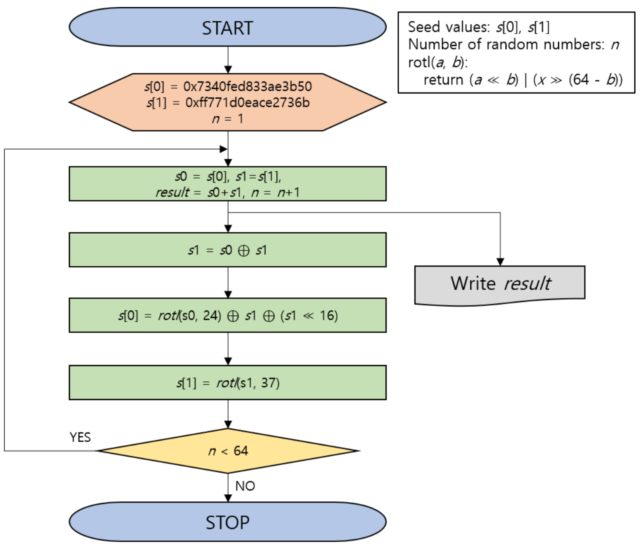
START
Seed values: s[0], s[1]
Number of random numbers: n
rotl(a, b):
return (a ≪ b) | (x ≫ (64 - b))
s[0] = 0x7340fed833ae3b50
s[1] = 0xff771d0eace2736b
n = 1
s0 = s[0], s1=s[1],
result = s0+s1, n = n+1
s1 = s0 ⊕ s1
Write result
s[0] = rotl(s0, 24) ⊕ s1 ⊕ (s1 ≪ 16)
s[1] = rotl(s1, 37)
YES
n < 64
NO
STOP

Figure 1: The flowchart of the xoroshiro128+ algorithm that generates 64 random numbers: $s[0]$ and $s[1]$ are initial seed values; n is the number of random numbers.

xoroshiro128+ generates random numbers using two initial seeds. To verify the randomness of xoroshiro128+, we fixed the seed values (s[0] = 0x7340fed833ae3b50, s[1] = 0xff771d0eace2736b) and generated a total of 64 consecutive random numbers. Fig. 1 shows the flowchart of the xoroshiro128+ algorithm we used in our experiment. However, the random number may be biased if the seed values are insufficient because xoroshiro128+ generates random numbers in deterministic ways using xor, rotate, and shift operations.

3 Previous Work

Since most of the RNGs are implemented in a physical chip, they are always vulnerable to the risk of side channel attacks such as timing attack and fault injection attack, etc. Even if RNG has a protection system, there are many types of side channel attacks that can be executed on the hardware. Therefore, studies [5, 6, 7, 8] that have successfully attacked RNG were published. This section introduces several side channel attack cases that have been executed on TRNG in recent years.

3.1 Fault Injection Attack on TRNG

In 2009, Markettos *et al.* announced a frequency injection attack that is valid for ring oscillator-based TRNGs [5]. They injected a sinusoidal wave signal into the power supply of a smart card or a secure microcontroller to intentionally modify the operating conditions of the ring oscillator and obtain a biased output signal. In order to compare and analyze the reduction of randomness, Markettos *et al.* summarized the randomness test results in Table 1 after the sinusoidal wave signal injection. NIST STS pass rate is about 15% and dieharder test is about 32%, which shows that the frequency injection attack has been successfully executed.

NIST STS	**Pass**	**-**	**-**	**Fail**
No injection	187	-	-	1
Injection	28	-	-	160
Dieharder	**Pass**	**Poor**	**Weak**	**Fail**
No injection	86	6	6	9
Injection	28	16	5	58

Table 1: Result of NIST STS and Dieharder test: After 24.04MHz frequency injection into the Smartcard, the number of NIST STS passes decreased from 187 to 28, and the number of diehard passes decreased from 86 to 28.

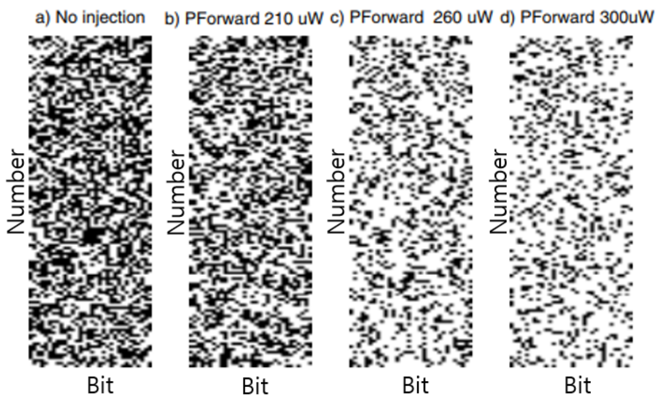

Figure 2: The result of electromagnetic attack on TRNG: Each sample is composed of 120 successive 32-bit frames (black and white squares correspond to 1 and 0, respectively).

3.2 Electromagnetic Active Attack on TRNG

In 2012, Bayon *et al.* announced an electromagnetic attack [6] that improves Markettos's frequency injection attack [5]. In general, electromagnetic (EM) attacks are non-invasive attacks that analyze the characteristics of EM radiation emitted by cryptographic devices. However, Bayon *et al.* demonstrated active attack that use probe to inject EM into the cryptographic device. This attack also did not require direct access to the the power pad of attack target.

Fig. 2 shows the output of random numbers of TRNG as a bit-stream. We can observe the decrease of randomness as the stronger electromagnetic is injected, and the effectiveness of the attack can be confirmed. They have also successfully proved that TRNG's random number can be fully controlled through additional experiments, such as manipulating the bit-stream to make certain sentences appear.

Since the attacker can easily extract the secret key information of the encryption algorithm implementation by taking the exact value of the random number through these attacks, research for increasing the side channel attack security of the RNG should be actively progressed.

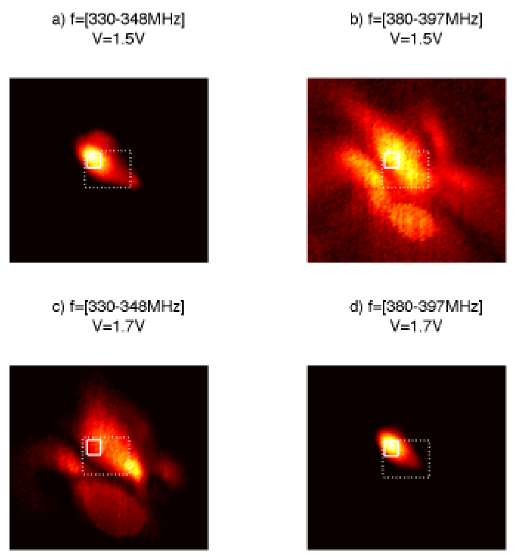

Figure 3: EM device maps: The white dotted rectangle represents the FPGAs DIE and the solid rectangle represents the location of the TRNGs

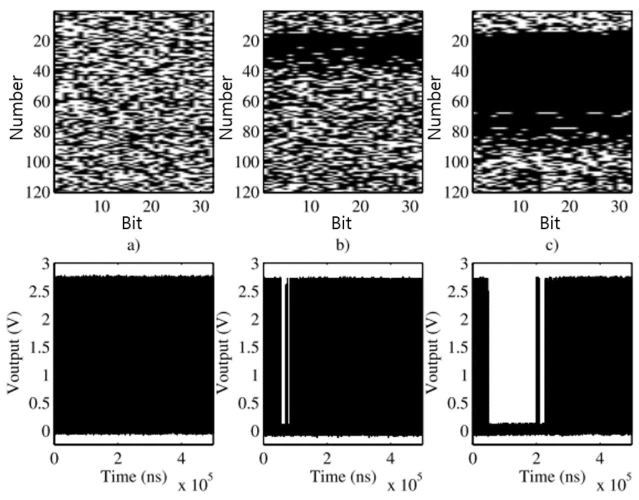

Figure 4: The result of glitch attack on TRNG: For case (a)no power glitch was inserted, for case (b)a power glitch of length 62.5μs was inserted, and for case (c)a power glitch of length 187.5μs was inserted.

3.3 Electromagnetic Passive Attack on TRNG

Bayon *et al.* published another paper on the EM attack in 2013 [7]. Unlike previous active attack that was injected with EM radiation, this paper executed passive attack that analyzed the EM emanation of cryptographic device. Bayon's goal was to retrieve as much information as possible on the self timed ring oscillator-based TRNG. Therefore, they supposed that the attacker could determine the working frequencies of the TRNGs and their locations on the chip. Because ring oscillator frequency depends dynamically on the power supply voltage and temperature, they measured the EM emanations and working frequencies by changing the pwoer supply voltage. Especially, EM emanations of the device were analyzed point by point using a frequency analysis, and it appeared in the form of a map. Fig. 3 is the EM device maps provided by Bayon. Before measuring the EM emanation, they expected EM emanation to appear in the upper left corner of the DIE only for a couple of conditions (Power = $1.5\,V$, frequency = [330 MHz - 348 MHz]) and (Power = $1.7\,V$, frequency = [380 MHz-397 MHz]). Similarly, Fig. 3(a) and Fig. 3(d) clearly show that they obtained the expected results. Using this attack, Bayon *et al.* demonstrated that it is possible to locate the TRNG on the chip and to find their corresponding frequencies.

3.4 Glitch Attack on TRNG

In 2015, Martin *et al.* published a paper on power and clock glitch attacks on self timed ring oscillator-based TRNGs [8]. They experimented with increasing the time to inject power glitch after lowering the base power of the TRNG from $1.2V$ to $0.7V$ for a successful power glitch attack. Fig. 4 shows the variation in random numbers due to glitch attacks. Fig. 2(a) is a bit-stream representation of the output of random numbers before an attack is executed, and Fig. 4(b)-(c) shows the output of random numbers after injecting power glitch signals for 62.5μs and 187.5μs, respectively. We can see that the longer the injection time of glitch, the less the randomness of TRNG. They also injected a clock signal of 40MHz instead of a 20MHz clock signal for clock glitch attacks and demonstrated that they can have complete control over the output of random numbers.

4 Our Experimental Setup

Because attacks on hardware devices are affected by the experimental environment, it is necessary to clearly specify the experimental environment. Therefore, in this section, we describe our experimental environment in detail and specify the values of the parameters. In addition, we provide a brief description of the attack scenario.

4.1 Environment

In our experiment, side channel attack was implemented using ChipWhisperer-Pro (CW1200) which is an open source embedded security analysis platform that generates FPGA-based pulses to enable a variety of side channel attacks, including clocks and voltage fault injection attacks. The xoroshiro128+ used in the experiment was implemented by uploading the source code to the CW308T XMEGA target board, one of the programmable FPGA boards. Fig. 5 shows the ChipWhisperer equipment and the FPGA board we actually used in our experiment. We also used ChipWhisperer Capture open source software, which is basically provided by NewAE Technology, to record and analyze power consumption.

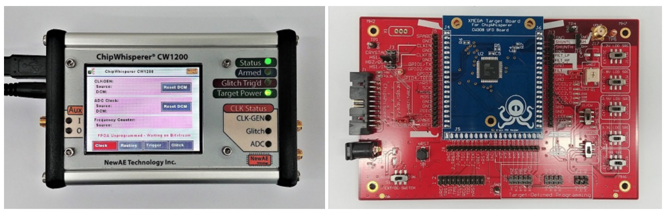
Figure 5: ChipWhisperer-Pro (CW1200), CW308 UFO board and CW308T XMEGA target board

4.2 Attack Scenario

Our final goal is to make xoroshiro128+ lose randomness. Because xoroshiro128+ performs many operations such as xor, rotate, shift, and so on, it is possible to have unwanted results if problems occur during operation. We deliberately interrupt the xoroshiro128+'s operation to output the wrong random number and compare it to the correct value. However, if the initial seed value is changed every time, the accuracy of the random number can not be verified. Therefore, we set two initial seed values as 0x7340fed833ae3b50 and 0xff771d0eace2736b, respectively.

In order to interrupt the xoroshiro128+'s operation, it is first necessary to figure out where each operation is performed. We measured the amount of power consumed during the operation of xoroshiro128+ through SPA. The power consumption graph can be very useful to find out where each operation inside xoroshiro128+ is performed.

After that, we executed the fault skipping attack and found the most likely point to lose randomness. We also output the expected result of the random number which can be obtained by successful attack in the form of bit-stream, and we compare it with the correct random number. Finally, we executed a power glitch attack to reduce xoroshiro128+'s randomness.

5 Side channel Attacks on Xoroshiro128+

5.1 Skipping Fault Analysis

Algorithm 1 Xoroshiro128+

```
1:  procedure NEXT( )
2:      $s0 \leftarrow s[0]$
3:      $s1 \leftarrow s[1]$
4:      $result \leftarrow s0 + s1$
5:
6:      $s1 \leftarrow s0$ xor $s1$
7:      $s[0] \leftarrow$ rotl($s0$, 24) xor $s1$ xor ($s1$ left-shift 16)
8:      $s[1] \leftarrow$ rotl($s1$, 37)
9:
10:     return $result$
11: end procedure
```

xoroshiro128+ consists of three functions: next(), jump(), and long_jump(). next() is a function that generates random numbers using xor, rotate, and shift operations on the seed value, and jump() and long_jump() are functions that generate random numbers by repeating the next() function several times. In our experiment, only the next() function was used to generate random numbers.

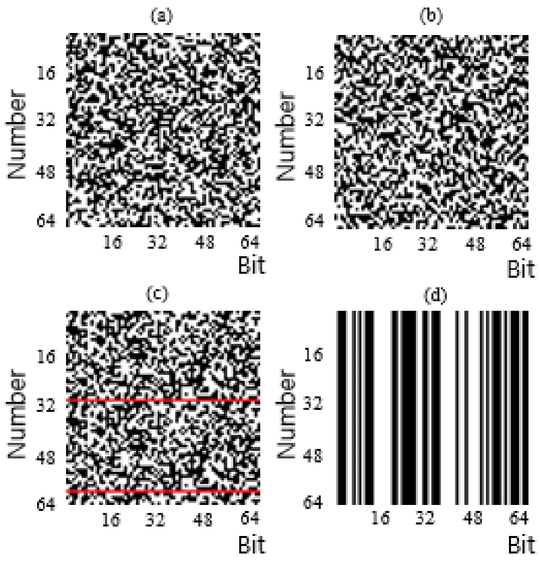

Figure 6: xoroshiro128+ bit-stream from skipping attack: (a)no skipping; (b)rotate and shift operations skip; (c)the second rotate operation skip; (d)two rotate and shift operations skip.

The detailed algorithm of the next() function is shown in the pseudo-code above. As mentioned earlier, the value of random numbers resulting from a single call in the next() function is always constant because we fixed the initial seed values (s[0] = 0x7340fed833ae3b50, s[1] = 0xff771d0eace2736b). However, because of the xoroshiro128+ algorithm that uses the previous results for the next operation, calling the next() function multiple times causes different random numbers to occur. We repeated the next() function 64 times to generate a total of 64 random numbers and expressed them in bit-stream form to confirm their randomness in Fig. 6(a).

xoroshiro128+ shows relatively sufficient randomness before being attacked. Before we executed the power glitch attack, we decided to print out the results of what would happen if xoroshiro128+ was interrupted during the computation process. To do this, we have attempted a skipping attack that skips certain operating lines in the source code. Fig. 6(b) shows the random number output when skipping the 7th line of the pseudo-code. Compared with Fig. 6(a), there is no significant difference in randomness. This shows that we can not get the desired outcome even if we hit the 7th line using a power glitch attack.

Fig. 6(c) shows the random number output when skipping the 8th line of the pseudo-code. At first glance, it doesn't seem to make a big difference in random numbers, but if we look closely, we can see that the same arrangement of random numbers is repeated based on

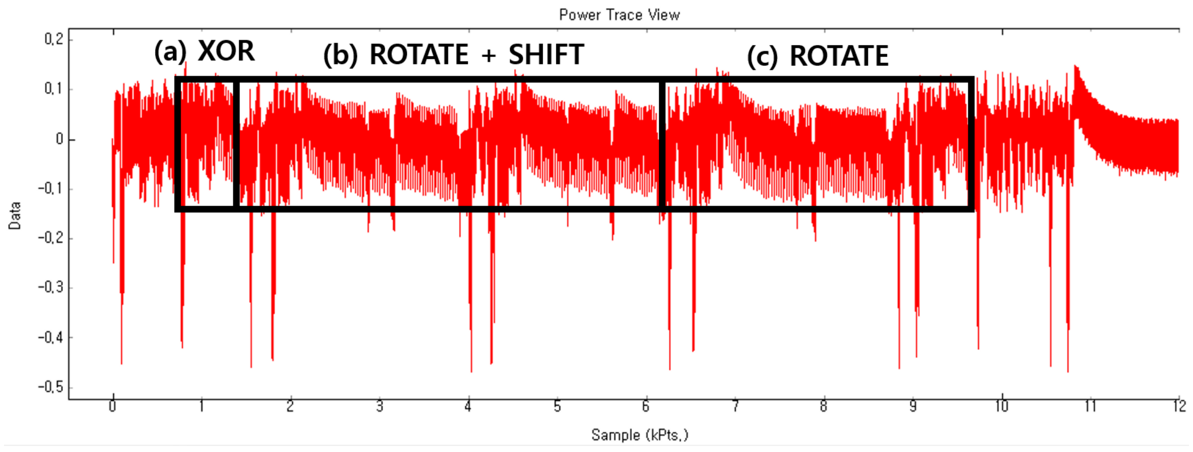

Figure 7: The result of SPA attack on xoroshiro128+: (a)800 to 1,400 traces are the points where xor is performed; (b)1,400 to 6,100 traces are the points where the rotate and shift operations are performed; (c)6,100 to 9,800 traces are the points where the second rotate operations are performed.

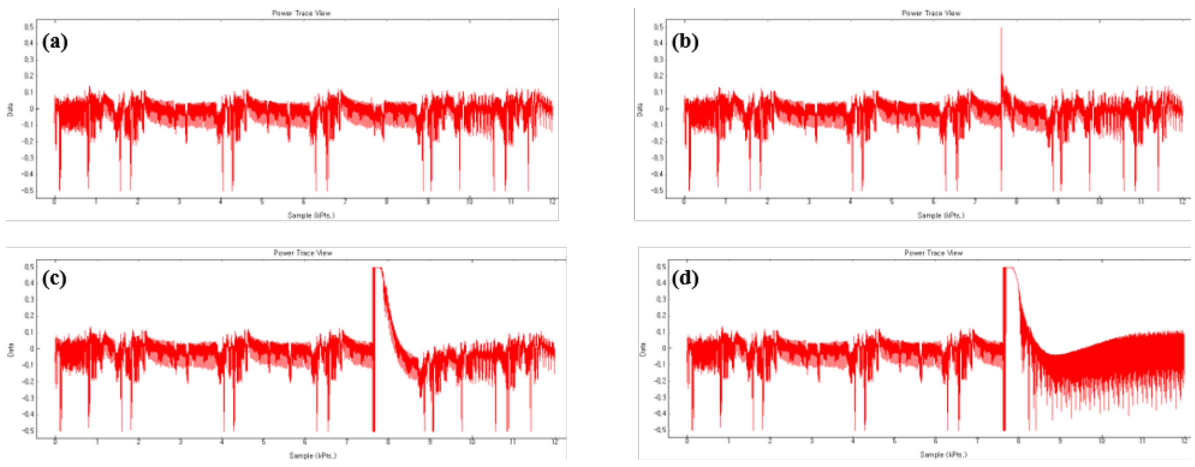

Figure 8: The result of a power glitch attack on xoroshiro128+: (a)no injection (b)a single injection (c)14 injections (d)15 or more injections

the red line. Seeing a particular pattern in bit-stream means that PRNG's randomness has been completely damaged, so we are expected to achieve a successful outcome if we attempt an attack on the 8th line of the pseudo-code.

Fig. 6(d) shows the random number generated by skipping both 7th and 8th lines. Since only xor operations are repeated, the same random number is output each time. To achieve the same result as above, we executed SPA and power glitch attack on xoroshiro128+

5.2 Simple Power Analysis

To inject the power glitch signal into the appropriate time, it is necessary to find the specific time in which each operation is performed. We executed SPA on xoroshiro128+ using ChipWhisperer-Pro. Fig. 7 graphically shows the power consumed by xoroshiro128+ when it produces a single random number. Because this single waveform alone cannot find the point at which a particular operation is performed, the previously executed skipping attack was applied equally.

We measured power waveforms by skipping three lines of pseudo-code in sequence. After that, we compared the obtained data with each other and located the point where the specific operation was performed. In Fig. 7, we found that (a)800 to 1,400 traces on the graph were the point where xor was performed; (b)1,400 to 6,100 traces were the point where the rotate and shift operations were performed; (c)6,100 to 9,800 traces were the point where the second rotate operation was performed.

Because the rotate operation consists of several shift operations, Fig. 7(b)-(c) output of power waveforms are similar. SPA and skipping attack have successfully identified the point at which the operation is performed, but this information alone is not sufficient to reduce xoroshiro128+'s randomness. However, this information can be useful in executing power glitch attacks.

5.3 Power Glitch Attack

We attempted a power glitch attack based on the above information in order to break the randomness of xoroshiro128+. Because of the wide range of measured power waveforms, we needed to determine a specific point of attack in advance. We found that it was effective to skip the second `rotate` operation through the previous SPA and power glitch attacks. Therefore, we decided to inject power glitches at 7,500 to 8,000 traces of the power consumption graph.

The glitch signal used for our experiment was initialized at 10MHz, and the glitch width was initialized at 20% of the period. Also, the glitch offset was initialized at 10% of the period. Fig. 8(a) is a power consumption graph before injecting the power glitch signal, and Fig. 8(b) is the result of a single injection of the glitch signal set as above. Unusual power waveforms were detected at 7,600 trace, but random numbers were normally output. So we tried to increase the number of glitch signal injections from 1 to 14 to more actively interfere with xoroshiro128+.

Fig. 8(c) shows a power consumption waveform after 14 glitch signals were injected, which clearly shows a larger glitch signal compared to Fig. 8(b). Contrary to our expectations, however, the random numbers were still output normally. Fig. 8(d) is a power consumption waveform that increases the number of repetition to 15 or more. After injecting the glitch signal, no waveform at the back was normally detected, which means xoroshiro128+ stopped working. Too strong glitch signal was injected so that xoroshiro128+ not only failed to skip specific operations, but also terminated all of execution. In this case, we could not gain any random numbers. Terminating the hardware device is also a part of the side channel attack, but it is not the outcome we have expected. We have executed our attack by moving where the glitch signal was being injected, but we have not obtained the outcome that some operations have been skipped.

6 Evaluation

In our experiment, the power consumption of xoroshiro128+ was successfully measured using ChipWhisperer-Pro. The skipping attack also enabled us to determine where a particular operation was performed. However, skipping attack assumes that it is possible for an attacker to access and modify the source code of the target. We could easily check randomness by printing the results of skipping attack in a bit-stream form.

We initially set a relatively low glitch width and repeat counts and conducted the experiment. As a result, we identified an abnormally terminated operation of xoroshiro128+ when the glitch width exceeds 20% or the repeat count exceeds 14. We then set these values to the maximum glitch signal and injected them into most of the trace points. The glitch signal was easily observed in power consumption graph, but the computation of xoroshiro128+ worked normally. It seems that our power glitch attack on xoroshiro128+ was not valid. Therefore, further research is needed, such as further lowering the power at a particular point or executing glitch attacks using clock signals.

7 Countermeasure

7.1 Defending Simple Power Analysis

SPA is an attack method that measures and then analyzes the power changes consumed during the computation of cryptographic devices. Therefore, if the cryptographic devices have to handle large amounts of computation, it can be vulnerable to SPA. However, intentionally inserting additional computations inside cryptographic devices can help prevent SPA. In this case, the difference in power consumption between each operation is reduced so that cryptographic device can hide useful information.

7.2 Defending Glitch Attack

Although we did not gain useful information from the power glitch attack, many hardware devices are still vulnerable to glitch attacks. Injecting abnormal signals using clock or power signals will cause unexpected operation of the hardware device. To prevent this situation, circuits need to be configured irregularly so that hardware equipment is not easily analyzed. Furthermore, stopping the operation of a circuit when an abnormal signal is detected may not expose specific information to an attacker.

8 Conclusion and Future Work

In this paper, we described RNGs and hardware-based side channel attacks and introduce several attack cases that have been executed on TRNGs. We also verified the security of xoroshiro128+ using the FPGA board and ChipWhisperer-Pro. We executed a SPA and skipping attack on xoroshiro128+, and we identified the randomness by expressing random numbers in a bit-stream form. Afterwards, we tried to execute a power glitch attack to damage the randomness of xoroshiro128+. Although the randomness was not reduced as desired, we confirmed that the randomness of the hardware-based PRNGs is likely to be lowered by various side channel attacks. Therefore, it is very important to develop and apply techniques to defend against various side channel attacks.

As future work, we will attempt not only a power glitch attack but also a clock glitch attack to disrupt the operation of xoroshiro128+. We will also conduct a study utilizing a number of hardware sources (e.g., electromagnetic waves, computing time, hardware sound, temperature, etc.) that makes FPGA-based PRNG vulnerable. In addition, we will expand the targets of our study to commercialized TRNGs and QRNGs.

Acknowledgement

This work was supported by ID Quantique Ltd. of Korea (IDQ) (Study on cryptographic strength and performance of quantum random number generator).

References

[1] J. Brown, R. Gao, Z. Ji, J. Chen, J. Wu, J. Zhang, B. Zhou, Q. Shi, J. Crowford, and W. Zhang, "A low-power and high-speed true random number generator using generated rtn," in *2018 IEEE Symposium on VLSI Technology*, pp. 95–96, IEEE, 2018.

[2] H. Fang, P. Wang, X. Cheng, and K. Zhou, "High speed true random number generator with a new structure of coarse-tuning pdl in fpga," *Journal of Semiconductors*, vol. 39, no. 3, p. 035001, 2018.

[3] X. Ma, X. Yuan, Z. Cao, B. Qi, and Z. Zhang, "Quantum random number generation," *npj Quantum Information*, vol. 2, p. 16021, 2016.

[4] F. Raffaelli, P. Sibson, J. E. Kennard, D. H. Mahler, M. G. Thompson, and J. C. Matthews, "A soi integrated quantum random number generator based on phase fluctuations from a laser diode," *arXiv preprint arXiv:1804.05046*, 2018.

[5] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Cryptographic Hardware and Embedded Systems-CHES 2009*, pp. 317–331, Springer, 2009.

[6] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson, and P. Maurine, "Contactless electromagnetic active attack on ring oscillator based true random number generator," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 151–166, Springer, 2012.

[7] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer, "Electromagnetic analysis on ring oscillator-based true random number generators," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pp. 1954–1957, IEEE, 2013.

[8] H. Martin, T. Korak, E. San Millán, and M. Hutter, "Fault attacks on strngs: Impact of glitches, temperature, and underpowering on randomness," *IEEE Transactions on information forensics and security*, vol. 10, no. 2, pp. 266–277, 2015.

[9] S. Vigna and D. Blackman, "Xoroshiro," *PRNG Shootout*, 2016, http://xoshiro.di.unimi.it/.

[10] D. Blackman and S. Vigna, "Scrambled linear pseudorandom number generators," *arXiv preprint arXiv:1805.01407*, 2018.

[11] "Xoroshiro128+." `http://vigna.di.unimi.it/xorshift/xoroshiro128plus.c`.

[12] "Chipwhisperer-pro." `https://wiki.newae.com/Main_Page,https://newae.com/tools/chipwhisperer/`.

[13] "Glitch attack." `https://www.pcmag.com/encyclopedia/term/43805/glitch-attack`.

[14] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[15] "Quantis rng." `https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator`.

[16] "Sk telecom qrng chip." `https://www.globalskt.com/home/info/2108`.

[17] "Quantis qrng chip." `https://www.idquantique.com/random-number-generation/products/quantis-qrng-chip`.

[18] "Nist sts." `https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final`.

[19] "dieharder." `https://webhome.phy.duke.edu/~rgb/General/dieharder.php`.

[20] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Annual International Cryptology Conference*, pp. 104–113, Springer, 1996.

[21] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*, pp. 388–397, Springer, 1999.

[22] G. Marsaglia *et al.*, "Xorshift rngs," *Journal of Statistical Software*, vol. 8, no. 14, pp. 1–6, 2003.

[23] D. Blackman and S. Vigna, "Scrambled linear pseudorandom number generators," *arXiv preprint arXiv:1805.01407*, 2018.

[24] S. Vigna, "Further scramblings of marsaglias xorshift generators," *Journal of Computational and Applied Mathematics*, vol. 315, pp. 175–181, 2017.

[25] "Testu01." `http://www.bitbabbler.org/test-data/TestU01.html`.

[26] D. Lemire and M. E. ONeill, "Xorshift1024*, xorshift1024+, xorshift128+ and xoroshiro128+ fail statistical tests for linearity," *Journal of Computational and Applied Mathematics*, vol. 350, pp. 139–142, To appear.

[27] "Testing hamming-weight dependencies." `http://prng.di.unimi.it/hwd.php`.