First Experimental Result of Power Analysis Attacks on a FPGA Implementation of LEA

Yongdae Kim[1,2] and Hyunsoo Yoon[1]

[1] Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea
[2] The Attached Institute of ETRI, Daejeon, Korea
Email: yongdae.kim@kaist.ac.kr, hyoon@nslab.kaist.ac.kr

Abstract. The lightweight encryption algorithm (LEA) is a 128-bit block cipher introduced in 2013. It is based on Addition, rotation, XOR operations for 32-bit words. Because of its structure, it is useful for several devices to achieve a high speed of encryption and low-power consumption. However, side-channel attacks on LEA implementations have not been examined. In this study, we perform a power analysis attack on LEA. We implemented LEA with 128-bit key size on FPGA in a straightforward manner. Our experimental results show that we can successfully retrieve a 128-bit master key by attacking a first round encryption.

Keywords: Power Analysis Attack, LEA, ARX, Correlation Power Analysis

1 Introduction

A new category of cryptanalysis, known as a power analysis attack, was introduced by P. Kocher et al. in 1996 [1]. Power analysis attacks exploits not only intended information (e.g., input and output from cryptographic devices), but also power consumption related to secret information (e.g., secret keys) from cryptographic devices [2]. This can be accomplished by efficiently comparing power analysis attacks to conventional cryptanalysis. Thus, investigating not only cryptographic algorithms, but also their concrete implemenations is critical.

A new block cipher, called lightweight encryption algorithm (LEA), was introduced in 2013 [3]. It is a 128-bit block cipher designed for fast encryption based on an addition-rotation-XOR (ARX) design technique. Its round encryption and key schedule are used exclusively by several ARX operations. It has three types of key size: 128, 192, and 256 bits. This size determines the number of round iterations. In [3], researchers investigated several existing attacks for block ciphers such as differential, linear, and boomerang attacks. They focused only on security evaluation at the algorithm level. LEA in software and hardware outperformed other block ciphers [3]. However, security evaluations have not been performed on LEA implementations. Therefore, examing its security strength against power analysis attacks is essential.

LEA does not contain S-Box components such as other block ciphers (e.g., AES). Instead, it employs ARX operations such as modular addition, rotation and XOR operations. Several studies have been conducted on side-channel attacks on ARX-based cryptographic algorithms [4], [5], [6]. In [5], the authors conducted an attack on the Skein algorithm using a 64-bit words addition operation. Although they failed to retrieve all subkeys, their experimental results showed that the modular addition in the Skein algorithm is vulnerable to side-channel attacks. The sutides in [4], and [6], investigated the ARX-based SHA-3 cryptographic algorithms, BLAKE and CubeHash. The researchers demonstrated theoretically the selection function and the number of power analysis that must be conducted. In addition, they implemented the algorithms on a microprocessor. No previous study has conducted security evaluations on ARX-based block ciphers such as LEA, in a hardware implementation against power analysis attacks.

In this study, we demonstrate power analysis attacks on a hardware implementation of LEA. We employ correlation power analysis [7], and present details on the method we use with the selection function for LEA. Based on the the selection function, we present a practical method for retrieving a 128-bit secret key during an attacking of the first round encryption of LEA. Our experimental results indicate that deploying countermeasures to prevent attacks on hardware implementations is essential.

2 LEA Algorithm

In 2013, Hong et al. introduced a new 128-bit block cipher, known as LEA, for lightweight encryption [3]. LEA is an ARX-based cryptographic algorithm and consists of three key sizes of 128, 192, and 256 bits. In this study, we focus on the 128-bit key length. LEA for 128-bit key length possesses 24 rounds, and thus, contains 24 round keys, denoted as $RK_i (0 \leq i \leq 23)$.

2.1 Structure of LEA

Fig. 1 shows that the i-th round function with a 128-bit key length. $T^i_{0,1,2,3}$ represents four 32-bit sub-keys of the i-th round key RK_i. These keys are provided by the key scheduling component. Each round function is calculated as follows:

$$X_0^{i+1} \leftarrow ROL_9\left((X_0^i \oplus T_0^i) \boxplus (X_1^i \oplus T_1^i)\right) \tag{1}$$

$$X_1^{i+1} \leftarrow ROR_5\left((X_1^i \oplus T_2^i) \boxplus (X_2^i \oplus T_1^i)\right) \tag{2}$$

$$X_2^{i+1} \leftarrow ROL_3\left((X_2^i \oplus T_3^i) \boxplus (X_3^i \oplus T_1^i)\right) \tag{3}$$

$$X_3^{i+1} \leftarrow X_0^i, \tag{4}$$

where $ROL_i(x)$ and $ROR_i(x)$ denote the i-th left and right rotations, respectively, on a 32-bit value x. A 128-bit plaintext P is denoted as a concatenation of 32-bit words, such that $P = (X_0^0, X_1^0, X_2^0, X_3^0)$. In the same manner, a ciphertext C is represented as $C = (X_0^{23}, X_1^{23}, X_2^{23}, X_3^{23})$.

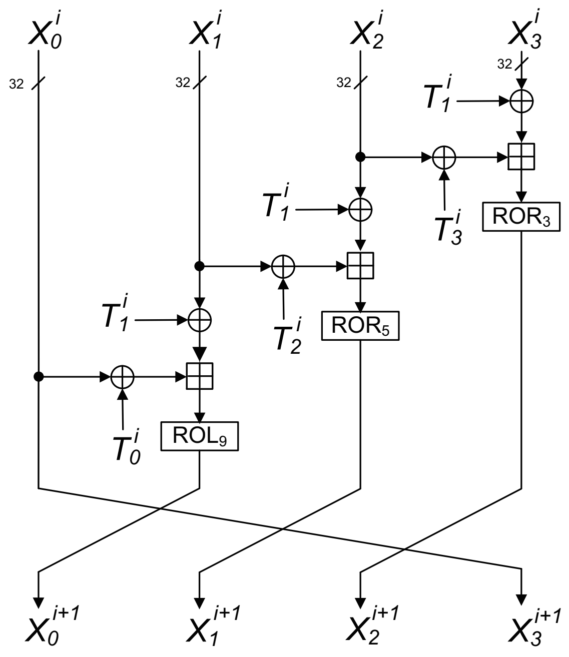

Fig. 1. LEA Structure

2.2 Key Schedule

In the key schedule of LEA, the following four constants are used:

$$\delta[0] = \text{0xc3efe9db} \tag{5}$$
$$\delta[1] = \text{0x44626b02} \tag{6}$$
$$\delta[2] = \text{0x79e27c8a} \tag{7}$$
$$\delta[3] = \text{0x78df30ec}. \tag{8}$$

A 128-bit master key K is represented as a concatenation of 32-bit words as follows:

$$K = (T_0^{-1}, T_1^{-1}, T_2^{-1}, T_3^{-1}) \tag{9}$$

Based on the master key, a 192-bit round key $RK_i = (T_0^i, T_1^i, T_2^i, T_1^i, T_3^i, T_1^i)$ is generated for $0 \leq i \leq 23$ as follows:

$$T_0^i \leftarrow ROL_1\left(T_0^{i-1} \boxplus ROL_i(\delta[\text{i mod 4}])\right) \tag{10}$$
$$T_1^i \leftarrow ROL_3\left(T_1^{i-1} \boxplus ROL_{i+1}(\delta[\text{i mod 4}])\right) \tag{11}$$
$$T_2^i \leftarrow ROL_6\left(T_2^{i-1} \boxplus ROL_{i+2}(\delta[\text{i mod 4}])\right) \tag{12}$$
$$T_3^i \leftarrow ROL_1 1\left(T_3^{i-1} \boxplus ROL_{i+3}(\delta[\text{i mod 4}])\right) \tag{13}$$
$$RK_i \leftarrow (T_0^i, T_1^i, T_2^i, T_1^i, T_3^i, T_1^i). \tag{14}$$

Each 32-bit subkey of the round keys is calculated independently of other 32-bit subkey, so if we can retrieve a 32-bit subkey (e.g. T_0^i), then we can easily compute a part of a master key, T_0^{-1}.

3 Power Analysis Attack on LEA

In this section, we present how we retrieve a 128-bit master key by attacking the first round encryption. In this study, we use correlation power analysis (CPA) based on a linear relationship between measured power consumption and an intermediate value. The intermediate value is usually calculated by means of a selection function. There are several ways to map the value based on power model, such as the hamming weight, the hamming distance model, etc.

3.1 Selection Function

In this study, we use the hamming distance model as a selection function, because our target implementation is a hardware platform and the hamming distance model is usually well correlated for a hardware implementation [8].

The four 32-bit subkeys of the first round key, T_0, T_1, T_2, T_3, are our primary targets for retrieval by CPA. For simplicity in the remainder of this paper, we omit the symbol i to represent a round. Fig. 1 reveals that targeting only a single 32-bit subkey is difficult. Instead, we must target two pairs of subkeys for each CPA: (T_0, T_1), (T_1, T_2), and (T_1, T_3). In addition, we assume that plaintext and ciphertext are known value. Therefore, we can calculate the hamming distance value between X_0^0 and $X_0^1 = ROL_9\left((X_0^0 \oplus T_0) \boxplus (X_1^0 \oplus T_1)\right)$ by deducing the T_0 and T_1 values. We must then produce a $2^{32} \times 2^{32}$ hamming distance value for every T_0 and T_1 value. However, applying this method to accomplish a power analysis attack is not feasible.

Instead of calculating the 32-bit hamming distance value, we divide the subkeys pair (T_0, T_1) into four 8-bit partial subkeys. We then produce four types of 8-bit hamming distance values for each z-th $(0 \le z \le 3)$ partial subkey h_z as follows:

$$\begin{aligned} h_z = ROR_9(&ROL_9(X^0_{0[(8(z+1)-1)-8z]}) \oplus \\ &ROL_9((X^0_{0[(8(z+1)-1)-8z]} \oplus T_{0[(8(z+1)-1)-8z]}) \boxplus \\ &(X^0_{1[(8(z+1)-1)-8z]} \oplus T_{1[(8(z+1)-1)-8z]}))), \end{aligned} \tag{15}$$

where $X_{i[a-b]}$ denotes a bit string cut from the a-th to the b-th bit of X_i. Each type of hamming distance value can be calculated by estimating two 8-bit partial subkeys. For example, h_1 is produced for all possible values of $T_{0[7-0]}$ and $T_{1[7-0]}$ from Eq. 15.

The number of possible candidates for the pair is $2^8 \times 2^8 = 65536$, derived from $(T_{0[7-0]}, T_{1[7-0]}) = (0,0), (0,1), \cdots, (0,255), \cdots, (255,255)$. Using h_z, we can derive four 8-bit partial subkey from the lowest byte $(T_{0[7-0]}, T_{1[7-0]})$ to the highest byte $(T_{0[31-24]}, T_{1[31-24]})$ sequentially by CPA. In addition, the remaining subkey pair (T_1, T_2) and $(T_1, T3)$ can be derived using the same method.

3.2 Key Guessing

Because of the number of possible key candidates for each pair is 65536, we must conduct CPA 65536 times for each key candidates. In addition, because of the structure of the LEA encryption, the hamming distance value is the same for two key pairs. For example, two pairs of keys, $(T_{0[7-0]}, T_{1[7-0]})$ and $(T'_{0[7-0]}, T'_{1[7-0]})$, always produce the same hamming distance value. In other words, one additional pairs of values always exists that satisfies the following:

$$\begin{array}{c} h_1 = h'_1 \\ \begin{array}{c}(X^0_{0[7-0]} \oplus T_{0[7-0]})\boxplus \\ (X^0_{1[7-0]} \oplus T_{1[7-0]})\end{array} = \begin{array}{c}(X^0_{0[7-0]} \oplus T'_{0[7-0]})\boxplus \\ (X^0_{1[7-0]} \oplus T'_{1[7-0]}).\end{array} \end{array} \tag{16}$$

We determined that a necessary and sufficient condition for Eq. 16 is $(T'_{0[7-0]}, T'_{1[7-0]}) = (T_{0[7-0]}, T_{1[7-0]})$ or $(T'_{0[7-0]}, T'_{1[7-0]}) = (T_{0[7-0]} \oplus 2^7, T_{1[7-0]} \oplus 2^7)$. To prove the condition, we simplify the problem as the following theorem.

Theorem 1.

$$(a \oplus x) + (b \oplus y) = (a \oplus z) + (b \oplus w) \ \mathit{mod}\ 2^n \ , \forall a, b \in \{0,1\}^n \tag{17}$$

$$\Leftrightarrow (x = z \ \mathit{and}\ y = w) \ \mathit{or}\ (x \oplus z = 2^{n-1} \ \mathit{and}\ y \oplus w = 2^{n-1}). \tag{18}$$

Proof. i) First, we show that Eq. 18 implies Eq. 17 based on the as following:

$$\begin{aligned} &(a \oplus x) + (b \oplus y) \text{ mod } 2^n \\ &\quad = (a \oplus x) + (b \oplus y) + 2^n \text{ mod } 2^n \\ &\quad = (a \oplus x) + 2^{n-1} + (b \oplus y) + 2^{n-1} \text{ mod } 2^n \\ &\quad = (a \oplus x \oplus 2^{n-1}) + (b \oplus y \oplus +2^{n-1}) \text{ mod } 2^n \\ &\quad = (a \oplus z) + (b \oplus w) \text{ mod } 2^n. \end{aligned}$$

ii) Next, we prove that Eq. 17 implies Eq. 18 is equivalent to the following equation:

$$\begin{aligned} &(x \neq z \text{ or } y \neq w) \text{ and } (x \oplus z \neq 2^{n-1} \text{ or } y \oplus w \neq 2^{n-1}) \\ &\quad \Rightarrow (a \oplus x) + (b \oplus y) \neq (a \oplus z) + (b \oplus w) \text{ mod } 2^n \ , \exists a, b \in \{0,1\}^n \end{aligned} \tag{19}$$

Eq. 19 can be proved by contradiction. Four possible cases exist as follows:

- Case 1 : $(x \neq z)$ and $(x \oplus z \neq 2^{n-1})$
- Case 2 : $(x \neq z)$ and $(y \oplus w \neq 2^{n-1})$
- Case 3 : $(y \neq w)$ and $(y \oplus w \neq 2^{n-1})$
- Case 4 : $(y \neq w)$ and $(x \oplus z \neq 2^{n-1})$

Case 1 We assume that Eq. 17 is true and let b be y and w. We then derive

$$\begin{aligned}(a \oplus x) + (y \oplus y) &= (a \oplus z) + (y \oplus w), \text{ and}\\ (a \oplus x) + (w \oplus y) &= (a \oplus z) + (w \oplus w).\end{aligned} \tag{20}$$

By substitution, we derive the following:

$$\begin{aligned}(a \oplus x) - (a \oplus z) &= (a \oplus z) - (a \oplus x) \bmod 2^n\\ 2 \cdot (a \oplus x) &= 2 \cdot (a \oplus z) \bmod 2^n.\\ a \oplus x &= a \oplus z \bmod 2^{n-1}.\end{aligned}$$

Finally, we derive $x = z \oplus 2^{n-1}$ or $x = z$. However, this contradicts the supposition of the Case 1. Thus, the supposition is false.

Case 2 We assume that Eq. 17 is true. From the proof of the Case 1, we then derive $x = z \oplus 2^{n-1}$. If we let a as x, we then derive the following:

$$\begin{aligned}(x \oplus x) + (b \oplus y) &= (z \oplus 2^{n-1} \oplus z) + (b \oplus w) \bmod 2^n\\ b \oplus y &= 2^{n-1} + (b \oplus w) \bmod 2^n\\ b \oplus y &= 2^{n-1} \oplus b \oplus w \bmod 2^n.\end{aligned}$$

Therefore, $y = w \oplus 2^{n-1}$ which is a contradiction of the supposition of the Case 2.

Case 3, 4 If we change x to y, z to w, and a to b, Casee 3 and 4 become the same as Cases 1 and 2, respectively.

4 Experimental Results

We implemented the LEA algorithm with 128-bit key length on the Side-channel Attacks Standard Evaluation BOard (SASEBO-GII) [9] in a straightforward manner. The LEA algorithm was processed by means of Xilinx FPGA (Virtex 5) on the board. In addition, the round keys were generated by an on-the-fly method. Thus, each round of the LEA was processed in a cycle. We captured power traces to include in the first two rounds of encryption. This was easily performed by determining the range of the first two rounds from the traces by observing two distinct power consumption patterns. Table 1 shows the details of our experimental environment.

Fig. 2 shows the power consumption trace that includes the first round of the LEA from our LEA implementation. We captured 50,000 traces using randomly generated plaintext. In addition, the number of sample points for power traces in each cycle is:

$$\frac{1 \times 10^9 \text{ points/s}}{24 \times 10^6 \text{ cycles/s}} \simeq 41. \tag{21}$$

Table 1. Experimental environment

Target Module	Xilinx FPGA (Virtex 5)
Digital Oscilloscope	LeCroy WaveRunner 6Zi
Sampling Frequency	1GSa/s
Probe (power)	Coaxial cable (50-Ohm)
Amp.	10-1000MHz LNA

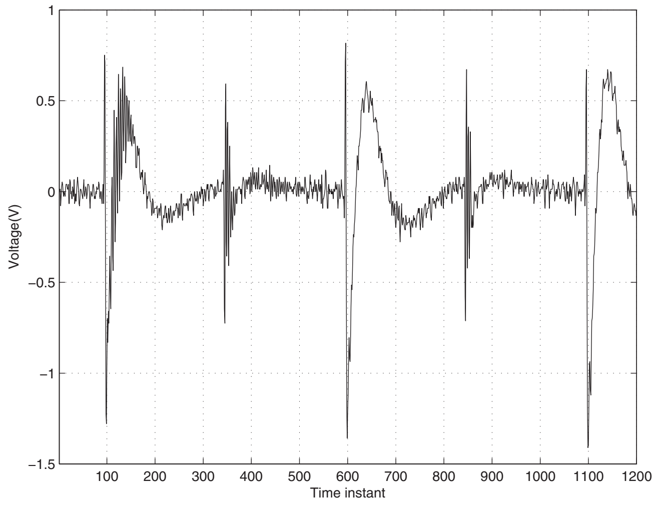

Fig. 2. Power consumption from the LEA implementation

Fig. 3 (a) shows CPA results based on 50,000 power traces for $(T_{0[7-0]}|T_{1[7-0]})$. The X-axis indicates time and the Y-axis indicates the correlation coefficient for each possible key guess. The black line represents correlation peaks for the correct concatenated two partial subkeys, whereas gray lines show wrong key guesses. Fig. 3 (a) shows that distinguishing the correct key is possible. Only one correct key seems to exist. However, another key exists that have the same value as the absolute value of the correlation coefficient shown in Fig. 3 (b). In addition, another key exists that has the same value of correlation as we mentioned in the previous section. We confirmed the theorem based on the result of CPA.

From Fig. 3 (b), the correct key pairs for $T_{0[7-0]}$ and $T_{1[7-0]}$ are $(T_{0[7-0]}|T_{1[7-0]}) = (0x2B, 0xA3)$ and $(T_{0[7-0]}|T_{1[7-0]}) = (0xAB, 0x23)$, respectively, which have the same absolute value of correlation, 0.08847. In the same manner, we retrieved two sets of key pairs for $(T_{1[7-0]}, T_{2[7-0]})$, and $(T_{2[7-0]}, T_{3[7-0]})$. We do not yet know which is the correct key pair..

We first retrieved two possible correct key pairs at the $[7-0]$ bit position. We calculated the hamming distance value for the next bit position, $[15-8]$. In addition, we already know two possible values at $[7-0]$. Therefore, generating

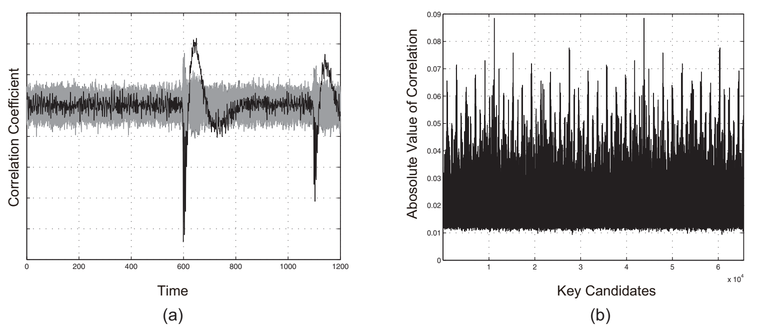

Fig. 3. CPA Result : (a) Corelation on time domain, (b) Absolute value of correlation on key candidates

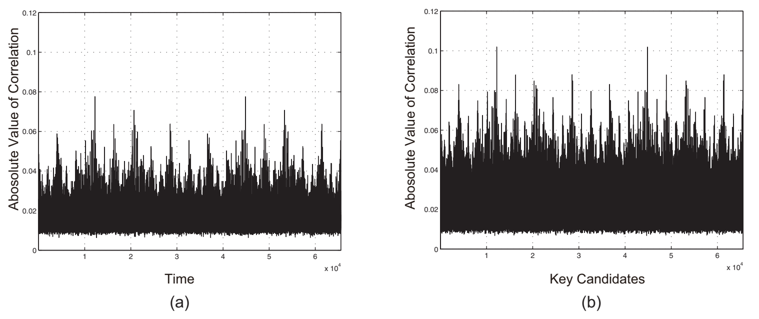

Fig. 4. CPA Result using two possible key candidates

two types of hamming distance values for the bit position $[15-0]$ is feasible. The possible number of key pairs is 2^{16}. We used the hamming distance value for the bit position $[15-0]$ instead of $[15-8]$ to determine the next key pairs. This method usually increases the correlation value in CPA [10]. However, we deployed a method to identify correct key pairs at $[7-0]$ and two possible correct key pairs at $[15-8]$ simultaneously.

Fig. 4 (a) and (b) show the CPA results of $(T_{0[15-8]}|T_{1[15-8]})$ using $(T_{0[15-8]}|T_{1[15-8]}) = (0x2B, 0xA3)$, and $(T_{0[15-8]}|T_{1[15-8]}) = (0xAB, 0x23)$, respectively. The peaks in Fig. 4 (b) are higher than those in Fig. 4 (a). Therefore, we can determine the correct key for $(T_{0[7-0]}|T_{1[7-0]})$. The absolute value of the correlation is 0.102 and 0.077 in Fig. 4 (a) and (b), respectively. In addition, the overall value for all key candidates are higher in Fig. 4 (b) than in (a). Other key pairs at different bit positions (such as $[15-8]$ and $[23-16]$) can be retrived in the same manner. Finally, we retrieved a correct $T_{0[23-0]}, T_{1[23-0]}, T_{2[23-0]}, T_{3[23-0]}$ and two possible correct key pairs for $T_{0[31-24]}, T_{1[31-24]}, T_{2[31-24]}, T_{3[31-24]}$. We then calculated two possible 128-bit master keys from a reversed key scheduling process. The correct master key from two keys was simply retrieved by brute force.

5 Conclusion

LEA is 128-bit lightweight block cipher introduced in WISA 2013. However, the authors investigated security evaluation of LEA on a theoretical basis only. LEA can be implemented in various platforms having throughput and a small size. We first investigated its security strength against power analysis attacks on a hardware implementation. Our results showed that LEA implementation reamins vulnerable to power analysis attacks. According to our research, this is the first experimental result of an LEA hardware implementation. Based on our results, implementing LEA with countermeasures is essential. For a future study, we plan to investigate other types of platforms and compare performance between countermeasure and non-countermeasure implementations.

References

1. P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," CRYPTO: Proceedings of Crypto, pp.388–397, 1999.
2. S. Mangard, E. Oswald, and T. Popp, "Power analysis attacks - revealing the secrets of smart cards," Springer, 2007.
3. D. Hong, J. keun Lee, D. chan Kim, D. Kwon, K. Ryu, and D. geon Lee, "LEA: A 128-bit block cipher for fast encryption on common processors," The 14th International Workshop Information Security Applications(WISA), pp.3–27, 2013.
4. M. Zohner, M. Kasper, M. Stottinger, and S. Huss, "Side channel analysis of the SHA-3 finalists," Design, Automation Test in Europe Conference Exhibition (DATE), pp.1012–1017, 2012.
5. C. Boura, S. Lévêque, and D. Vigilant, "Side-channel analysis of grøst and skein," Proceedings of the 2012 IEEE Symposium on Security and Privacy Workshops, pp.16–26, 2012.
6. O. Benot and T. Peyrin, "Side-channel analysis of six SHA-3 candidates," International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp.140–157, 2010.
7. E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp.16–29, 2004.
8. F.X. Standaert, E. Peeters, G. Rouvroy, and J.J. Quisquater, "An overview of power analysis attacks against field programmable gate arrays," Proceedings of the IEEE, pp.383–394, 2006.
9. Research Center for Information Security, "Side-channel Attack Standard Evaluation BOard (SASEBO)," http://www.rcis.aist.go.jp/special/SASEBO.
10. Y. Komano, H. Shimizu, and S. Kawamura, "BS-CPA: Built-in determined sub-key correlation power analysis," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, pp.1745–1337, 2010.