Fault Sensitivity Analysis

Yang Li[1], Kazuo Sakiyama[1], Shigeto Gomisawa[1], Toshinori Fukunaga[2], Junko Takahashi[1,2], and Kazuo Ohta[1]

[1] Department of Informatics, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
`{liyang,saki,g-shigeto-lfat,junko,ota}@ice.uec.ac.jp`
[2] NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-1 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
`{fukunaga.toshinori,takahashi.junko}@lab.ntt.co.jp`

Abstract. This paper proposes a new fault-based attack called the Fault Sensitivity Analysis (FSA) attack, which unlike most existing fault-based analyses including Differential Fault Analysis (DFA) does not use values of faulty ciphertexts. Fault sensitivity means the critical condition when a faulty output begins to exhibit some detectable characteristics, *e.g.*, the clock frequency when fault operation begins to occur. We explain that the fault sensitivity exhibits sensitive-data dependency and can be used to retrieve the secret key. This paper presents two practical FSA attacks against two AES hardware implementations on SASEBO-R, PPRM1-AES and WDDL-AES. Different from previous work, we show that WDDL-AES is not perfectly secure against setup-time violation attacks. We also discuss a masking technique as a potential countermeasure against the proposed fault-based attack.

Keywords: Side-channel attacks, Fault Sensitivity Analysis, AES, WDDL

1 Introduction

Nowadays, the security of cryptographic devices such as smart cards is threatened by side-channel attacks that retrieve secret information from side-channel leakages such as power consumption and electromagnetic radiation. The most studied fault-based attack is Differential Fault Analysis (DFA) proposed by Biham and Shamir in 1997 [1]. The DFA attacks have been actively studied in [2, 4, 6–8, 10, 12–14] and [16].

Generally, DFA attacks retrieve the key based on information of the characteristics of the injected fault and the values of the faulty ciphertexts. In this paper, a faulty ciphertext represents the output after a transient fault is injected, *i.e.*, the output of the original cryptographic algorithm using a faulty intermediate value as an intermediate input. On the other hand, when an action of fault injection is performed, we generally call the output a *faulty output*. A faulty output could be a faulty ciphertext, a fault-free ciphertext when the fault injection fails, or a nonsense value for implementations with fault attack countermeasures.

This paper proposes a new fault-based attack called Fault Sensitivity Analysis (FSA) attack. We notice that in the process of fault injection, there are other types of information that are available to attackers, which we call fault sensitivity. The fault sensitivity is a condition where the faulty output begins to exhibit some detectable characteristics. For example, when gradually increasing the intensity of the fault injection, attackers can discern the critical condition where a fault begins to occur or the fault becomes stable. Similar to most side-channel attacks, if the relationship between the fault sensitivity and the processed sensitive data is known, the FSA attacks can retrieve the secret information from a cryptographic device.

This paper explains the general attack procedures and attack requirements for the FSA attacks. To prove the validity of the FSA attacks, this paper first presents a detailed FSA attack example against PPRM1-AES [9] (1 stage Positive Polarity Reed-Muller) implemented in ASIC mounted on the Side-channel Attack Standard Evaluation Board (SASEBO-R) [5]. For the PPRM1 S-box, based on its structure and a simulation, we explain that there is a correlation between the faulty sensitivity and the Hamming weight of the input signals for the S-box. In the FSA attack against PPRM1-AES herein, the 128-bit key can be retrieved with less than 50 plaintexts.

We note that the FSA attack has the potential to threaten many DFA-resistant implementations, since it does not require the value of the faulty ciphertext. For example, FSA can be applied to Wave Dynamic Differential Logic (WDDL) [15], which was said to be naturally immune to the DFA attacks based on setup-time violation [11]. This paper also describes an FSA attack against WDDL-AES on SASESO-R. Based on experimentation, we find that the fault sensitivity for a WDDL combinational logic tree can be correlated with the values of one bit of the output signal. By retrieving 3 out of 16 key bytes, we show that a practical ASIC implementation of WDDL-AES is not perfectly secure against the FSA attack based on setup-time violations.

Compared to DFA, FSA does not restrict the injected fault to a small subspace by assuming that only a few bits or bytes are disturbed. On the other hand, the masking technique, which is shown not to be effective against the DFA attacks [3], is a potential countermeasure against FSA attacks.

This paper is organized as follows. In Section 2, we briefly review the previous work on fault-based attacks. Section 3 describes the general principle and attack procedures of FSA. We also describe a detailed FSA attack against PPRM1-AES. Finally, we discuss the attack requirements and countermeasures against FSA. In Section 4, we explain the FSA attack against WDDL-AES. Section 5 concludes this paper.

2 Preliminaries

This section reviews several common fault injection techniques and presents the attack assumptions and requirements for DFA.

2.1 Common Fault Injection Techniques

In [2], the common fault injection techniques are listed as spike attacks, glitch attacks, optical attacks, and electromagnetic perturbations attacks. The spike and glitch attacks are likely to be simpler to implement than others since they disturb the external power supply or the external clock, respectively. An illegal power supply or illegal clock will cause a setup-time violation since flip-flops are triggered before the output signals are fixed to a correct value. Compared to spike attacks, it is easier to control the exact time of a fault injection for the glitch attack. Therefore, we use the glitch attack to perform fault injections in this paper.

2.2 DFA and Attack Requirements

In 1997, Biham and Sharmir first proposed the concept of the DFA attack and applied it to DES [1]. Since then, the DFA attack has been the most discussed fault-based attack. DFA assumes that attackers are in physical possession of the cryptographic device, and can use it to obtain two ciphertexts for the same plaintext and secret key[3]. One of the ciphertexts is a fault-free ciphertext denoted by C, and the other denoted by C' is the result after some computational fault is injected. DFA further assumes that the attackers know some characteristics of the injected fault, *e.g.*, only several bits or bytes are disturbed in a specific round operation.

In the DFA attacks, the attackers first make a key guess, K_g. Then fault-free intermediate value I and faulty intermediate value I' are calculated based on (C, K_g) and (C', K_g), respectively. Subsequently, the attackers check whether $I \oplus I'$ satisfies the characteristics of the injected fault. Repeating these procedures for multiple pairs of (C, C'), the attackers can finally identify the secret key.

Generally, there are two major requirements for the DFA attacks.

– First, the DFA attack requires the value of faulty ciphertext C'. A faulty ciphertext is the output of the original cryptographic algorithm using the faulty intermediate value as the intermediate input. In the case of WDDL circuits under the fault injections caused by setup-time violations, this requirement cannot be satisfied [11].
– Second, attackers need to know some characteristics of the injected fault; however, the characteristics of the injected fault cannot be judged from the values of C and C'. Only when the actual injected fault is the expected one, can the DFA attackers identify the secret key.

[3] For simplicity, we consider only the encryption process.

3 FSA Proposal

In this section, we explain the general principle of the proposed FSA, and discuss the attack scenarios, attack requirements, and countermeasures to it. In a general discussion concerning the FSA attack, we present a detailed FSA procedure using PPRM1-AES as a case study.

3.1 General Principle of FSA

In the same way as in the DFA case, we assume that the attackers are in physical possession of the device. Starting from a condition where a correct ciphertext is obtained, the attacker gradually increases the intensity to which he disturbs the power supply or external clock. While doing so, there must be a moment where the success rate of the fault injection is non zero and a moment where the success rate is 1.

We call these critical conditions where the faulty output exhibits some detectable characteristics fault sensitivity. The fault sensitivity information can be observed and recorded by attackers and can be utilized as new side-channel information if it exhibits sensitive-data dependency. Consequently, we propose a new side-channel analysis FSA that utilizes the leakage of the fault sensitivity to retrieve secret information.

3.2 Data-Dependency of Fault Sensitivity

Since the transitions of signals in a device are data-dependent, it is natural to believe that the fault sensitivity is data-dependent. For faults caused by the setup-time violation, we explain the data-dependency for the timing delays of signals.

General Mechanism We use AND, OR and XOR gates as examples to explain the general mechanism of the data-dependency of the signal timing delay. In the following analysis, T_X denotes the timing delay for a signal X.

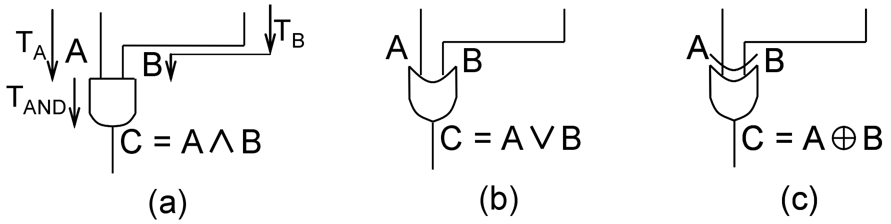

Fig. 1. Examples for data-dependency for fault sensitivity.

For a two-input AND gate as shown in Fig. 1 (a), we assume $T_A < T_B$. If signal A is logic 0, signal C is determined after signal A arrives at the AND gate. As a result,

$T_C = T_A + T_{AND}$, where T_{AND} is the timing delay caused by the AND gate. On the other hand, if signal A is logic 1, signal C will be determined after signal B arrives, so that $T_C = T_B + T_{AND}$.

Similarly, for a two-input OR gate as shown in Fig. 1 (b), we still assume $T_A < T_B$. If signal A is logic 1, $T_C = T_A + T_{OR}$, otherwise $T_C = T_B + T_{OR}$. In a word, for a two-input AND/OR gate, the input signal with a shorter timing delay is the selector for the timing delay of the output signal. However, for an XOR gate as shown in Fig. 1 (c), the timing delay of the output signal is decided by the maximum timing delay of its input signals without data-dependency.

We call the maximum timing delay among all the output signals for a combinational logic tree the critical timing delay. The fault sensitivity for setup-time violation is dependent on the critical timing delay. Since the timing delays of intermediate signals are data-dependent, the critical timing delay is also data-dependent. Once a circuit is physically decided, the data-dependency of the fault sensitivity is also physically fixed. Attackers can analyze the data-dependency based on the structure of the circuit, software simulation, or implementation of the circuit.

Data-dependency of Critical Timing Delay for PPRM1 S-box As a case study, we analyze the data-dependency of the critical timing delay for the PPRM1 S-box based on its structure and a simulation.

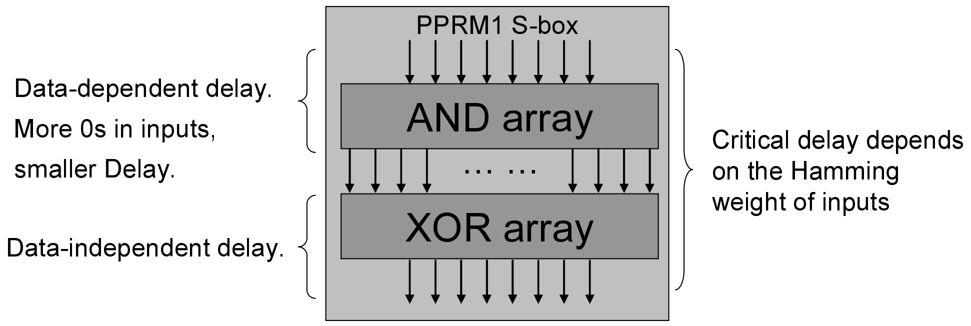

Fig. 2. Structure of PPRM1 S-box and data-dependency for the critical timing delay.

PPRM1 (AND-XOR logic) was proposed by Morioka and Satoh at CHES 2002 for a low power AES design [9]. Although PPRM1 is not likely to be used in a practical implementation, its straight-forward structure makes it a perfect attack target for the case study. As shown in Fig. 2, for the S-box of PPRM1, the input signals go through an AND gate array and an XOR gate array to become the output signals. For the AND gate array, the timing delays of the output signals are dependent on the values of the input signals. For the XOR gate array, the timing delays of the output signals are not data-dependent. In general, the structure of the PPRM1 S-box indicates the dependency between the critical timing delays and the values of the input signals.

For the AND gate array, each logic 0 input signal has a probability for decreasing the critical timing delay. Consequently, the more 0s in the input signals, the bigger the possibility that the critical timing delay of the S-box becomes shorter. In conclusion, statistically the critical timing delay of the PPRM1 S-box should be correlated with the Hamming weight of the input signals. Specifically, the input signals with a higher Hamming weight make the PPRM1 S-box more sensitive to a fast clock.

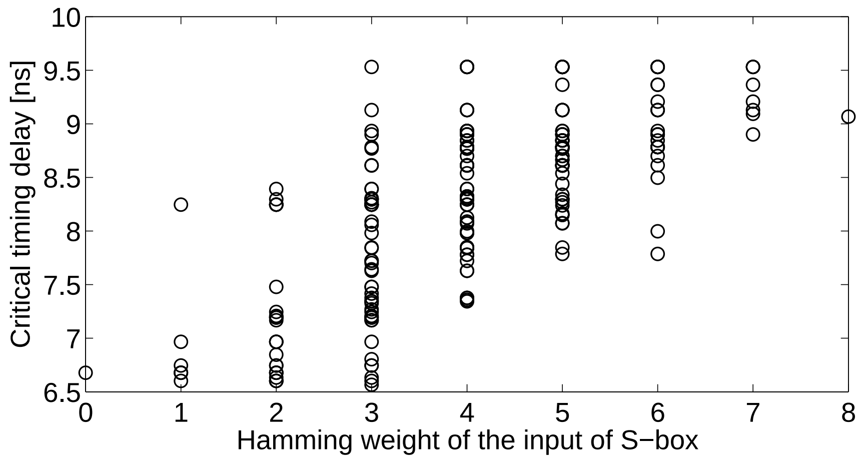

Fig. 3. Relationship between the critical timing delay and Hamming height of the input of PPRM1 S-box.

To confirm this, we simulate the transitions and timing delays of signals using the Verilog-HDL codes for the PPRM1 S-box [5] by Xilinx. For each possible input of the S-box, we obtain the corresponding critical timing delay where the initial values of all wires are reset to logic 0. As shown in Fig. 3, the correlation coefficient between the critical timing delay and the Hamming weight of the input signals is approximately 0.71. Later we show a similar correlation existing in the ASIC implementation of PPRM1 that can lead us to a practical FSA attack.

3.3 General FSA Attacks Scenarios

Without loss of generality, this paper only shows the case in which the rounds near the output are attacked.

Collection of Fault Sensitivity Information In a practical FSA attack, attackers first need to collect the fault sensitivity information. For simplicity, we use an unspecific parameter called the fault injection intensity denoted by F. When $F = 0$, no

Algorithm 1 Collection of Fault Sensitivity Information

```
**Inputs:** The number of different plaintexts: $N$
**Outputs:** Ciphertexts: $CT[i]$, Critical fault injection intensity: $F^C[i]$
for $i = 1$ to $N$ do
    Generate a random plaintext $PT[i]$
    Reset fault injection intensity $F \leftarrow 0$
    $CT[i] \leftarrow Enc(PT[i], F)$ (No fault injection)
    repeat
        Increase $F$ by a little
    until $Enc(PT[i], F) \neq CT[i]$
    $F^C[i] \leftarrow F$
end for
```

fault injection is performed. An increase in F represents an increase in the intensity for the fault injection, *e.g.*, a decrease in the power supply and a shortening of the clock period. Then we denote the output of the encryption for plaintext PT with fault injection intensity F as $Enc(PT, F)$. We use the intensity where a fault begins to occur as critical fault injection intensity F^C, then the fault sensitivity information is collected according to the procedures in Alg. 1. In Alg. 1, the critical fault injection intensity information is collected by gradually increasing the intensity and checking whether the output is still the same as the fault-free ciphertext.

The Key Retrieval Procedure We assume that the attackers can use ciphertexts $CT[i]$ and a key guess, K_g, to predict critical fault injection intensity F^C using a function denoted by $f_{F_g^C}$. Algorithm 2 shows the basic procedures for the key retrieval calculation where $\rho(A, B)$ denotes the absolute value of the Pearson correlation coefficient between A and B. The correlation peak among all possible key guesses is expected to be the same as the correct one.

Algorithm 2 Key Retrieval Procedure

```
**Inputs:** Bit length of (sub-)key: $t$, Ciphertexts: $CT[i]$, Critical fault injection intensity: $F^C[i]$
**Outputs:** Key
for $K_g = 0$ to $2^t - 1$ do
    for $i = 1$ to $N$ do
        $F_g^C[i] \leftarrow f_{F_g^C}(CT[i], K_g)$
    end for
    $Cor[K_g] \leftarrow \rho(F^C, F_g^C)$
end for
$Key \leftarrow K_g$ where $Cor[K_g]$ is the maximum
```

3.4 FSA Attack Scenarios Against PPRM1-AES

In this section, we propose an FSA attack scenario against 128-bit PPRM1-AES. We denote the calculation results of the i-th round of AES by H^i and $i \in [1, 10]$. In the last round of AES, the MixColumns operation is omitted. Each byte of H^9 is substituted by an S-box, and the 10th round key, K^{10}, is added to become the corresponding byte for ciphertext H^{10}. Since the byte-wise calculations in the last round of AES are independent from each other, there are 16 independent combinational logic trees. As a result, each byte of the ciphertext is an independent indicator of whether a fault is injected in its combinational logic tree. Therefore, even through all the S-boxes are calculated in parallel for the PPRM1-AES on SASEBO-R, the 16 bytes of K^{10} can be attacked independently in the FSA attack.

Based on the analysis in Section 3.2, when we inject the fault during the last round of PPRM1-AES by shortening the corresponding clock period, the fault sensitivity is correlated with H^9. Attackers can set the basic attack target as a byte of K^{10}, then the FSA attacks can be applied as shown in Algs. 1 and 2, where $f_{F^C_g}$ is the Hamming Weight of $InvSbox(CT[i] \oplus K_g)$.

FSA Attack Results Against PPRM1-AES This section shows the FSA attack results against PPRM1-AES implemented in ASIC mounted on the SASEBO-R [5]. The detailed experimental setup and parameter settings are shown in Appendix A.

In Fig. 4, we show 16 sub-figures for the correlation coefficients against key guesses for 16 bytes of K^{10} when 360 plaintexts are used. We note that in the practical attack, all 16 key bytes of K^{10} are attacked in parallel, *i.e.*, for each plaintext we collect fault sensitivity data for every combinational logic tree. Figure 5 shows the number of correct key bytes against the number of used plaintexts. We found that full key recovery for the FSA attack against PPRM1-AES requires less than 50 plaintexts.

3.5 Attack Requirements and Countermeasures for FSA

There are two requirements for a practical FSA attack. First, attackers must understand the data-dependency for the fault sensitivity. Even through the sensitive data and the fault sensitivity may not have a clear correlation, as long as attackers have a template for the data-dependency of the fault sensitivity, FSA can retrieve the secret key. Second, the secret key must be able to be divided and attacked independently so that Alg. 2 can be finished in a practical amount of time. All the software implementations where S-boxes are sequentially calculated satisfy the second requirement. Furthermore, most of the parallel implementations without countermeasures satisfy the second requirement as well.

Compared to the DFA attacks, the FSA attacks do not require that the injected fault be restricted to a small subspace by assuming that only a few bits or bytes are

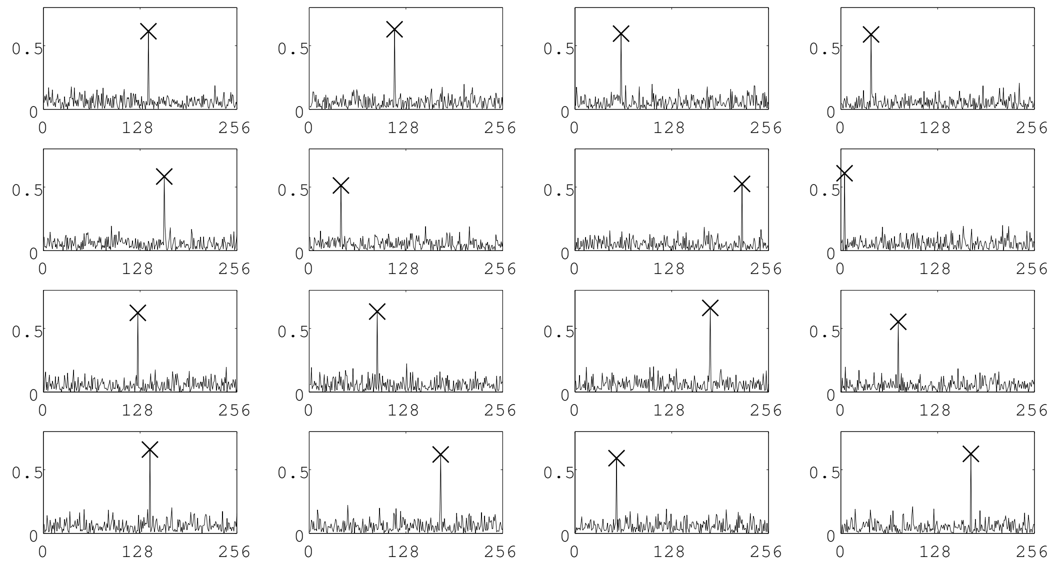

Fig. 4. Attack results for PPRM1-AES using 360 plaintexts. Each sub-figure corresponds to a key byte. The key guess is represented on the horizontal axis. The correlation coefficient between the critical fault injection intensities and Hamming weight of the input is represented on the vertical axis. Each correct key byte is marked by an $\times$.

disturbed. Even though the DFA attacks are likely to require fewer fault injections to retrieve the secret key [8, 10, 12, 16], it requires that the attacker have the ability to inject the expected fault in the first place. Since attackers have the device, there is no limitation on how many times that the non-invasive fault injections are performed; however, it requires much knowledge or investigation to inject the expected faults.

Furthermore, the FSA attacks do not require the values for faulty ciphertexts. So for the conventional fault-based attack countermeasures that provide a nonsense output or halt the calculation when a computational fault is detected, the FSA attack is still a potential threat. The fault sensitivity information is still available to attackers since they can distinguish whether or not a fault is injected. However, if the S-boxes are calculated in parallel, the fault detection is no longer byte-wise independent. The collected information only corresponds to the most sensitive part. Since the second requirement for the FSA attacks is not satisfied, it is difficult to retrieve the full key using the FSA attacks.

On the other hand, we note that the masking technique, which is shown not to be effective against the DFA attacks [3], is likely to be an effective countermeasure against the FSA attacks. Once all of the sensitive values are masked by uniformly distributed random numbers, the data-dependency between the intermediate values and the fault sensitivity can no longer be used in the key retrieval.

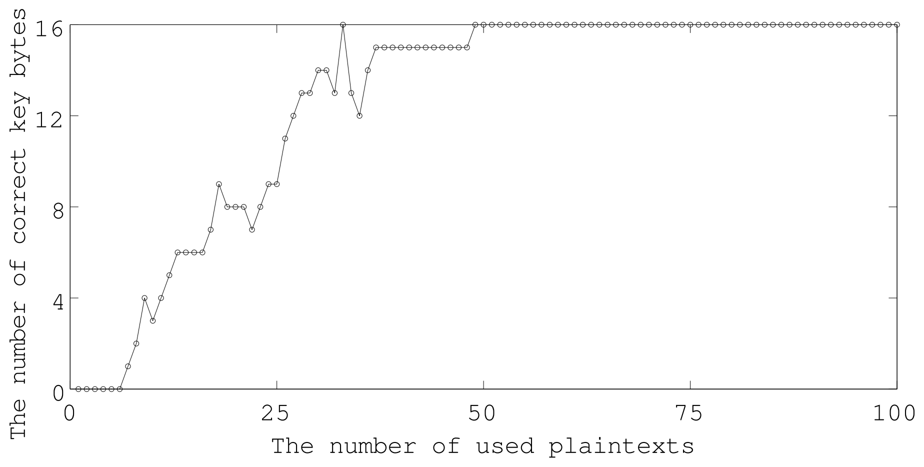

Fig. 5. Number of correct key bytes vs. number of used plaintexts.

4 FSA Attacks Against WDDL-AES

Since the faulty output for WDDL circuit has no information regarding the key, it is concluded that the WDDL circuit is naturally protected from the setup-time violation attacks in FDTC 09 [11]. We note that WDDL-AES satisfies the attack requirements for FSA, so that theoretically it is potentially vulnerable to the FSA attacks.

4.1 WDDL "Protected" Against Setup-Time Violation Attacks

WDDL was proposed by Tiri and Verbauwhede at DATE 2004 as a hiding countermeasure for power analysis. As a representative of the Dual-Rail Precharge Logic, each WDDL gate comprises two complementary operations. Every signal in WDDL has two complementary wires (true, false) as well, where the true wire has the actual value of the signal and the false wire has the complementary value. The logic values of two wires for a signal are either $(1, 0)$ or $(0, 1)$. Each clock cycle is divided into two phases, precharge and evaluation. In the precharge phase, all of the wires are set to be the precharge value, which is assumed to be 0 in this paper. In the evaluation phase, each pair of wires is set back to the logic values as either $(1, 0)$ or $(0, 1)$. As a result, exactly half of the wires will transit from 0 to 1, and the other half remain at 0 during each evaluation phase. Since the number of bit transitions is independent from the processed data, a WDDL circuit is likely to consume a constant amount of power for each clock cycle.

Under the fault injection caused by the setup-time violations, the two wires of a faulty signal in WDDL can only be $(0, 0)$. Furthermore, an input faulty signal, $(0, 0)$, is likely to spread to all of the output signals for a WDDL combinational logic tree.

In the case of the fault injection at the beginning of the 8th round used in [10,12,16], attackers can only obtain a faulty output with all 0s. Since faulty ciphertext C' is not available for DFA attackers, it is concluded that WDDL is naturally immune to the setup-time violation attacks. However in this work, we show that in a practical implementation of WDDL-AES it is vulnerable to the FSA attacks based on setup-time violation.

4.2 Data-Dependency of Fault Sensitivity for WDDL-AES

For the WDDL-AES on SASEBO-R, we try to use the implementation itself to obtain the data-dependency of the fault sensitivity. With full knowledge of the secret key, we performed fault injections that shorten the evaluation period for the last round of AES. Then we found that the critical fault injection intensity of each byte is correlated with the value of one single bit of the ciphertext byte. Although the correlation between fault sensitivity information and the ciphertexts cannot be used for key retrieval, we understand the fault sensitivity for the WDDL circuits are dependent on its output.

4.3 Practical FSA Attack Against WDDL-AES

As a practical attack, we performed another fault injection that shortens the evaluation period for the 9th round of AES, since the data-dependency based on H^9 can be used in the key retrieval. A modified key retrieval algorithm shown in Alg. 3 is applied. In Alg. 3, $f_{getbit}(A, b)$ represents a vector comprising the b-th bit of each element of A. The attack results after using 1200 plaintexts are shown in Fig. 6. The 6th key byte and the 11th key byte of K^{10} can be identified clearly. Also the correlation coefficient peak for the 4th key byte corresponds to the correct key as well.

Algorithm 3 Key Retrieval Procedure for WDDL-AES

```
Inputs: Bit length of (sub-)key: $t$, Ciphertexts: $CT[i]$, Critical fault injection intensity: $F^C[i]$
Outputs: Key
for $K_g = 0$ to $2^t - 1$ do
    for $i = 1$ to $N$ do
        $F^C_g[i] \leftarrow InvSbox(CT[i] \oplus K_g)$
    end for
    for $bit = 0$ to $7$ do
        $BitCor[bit] \leftarrow \rho(f_{getbit}(F^C_g, bit), F^C)$
    end for
    $Cor[K_g] \leftarrow Max(BitCor[bit])$
end for
$Key \leftarrow K_g$ where $Cor[K_g]$ is the maximum
```

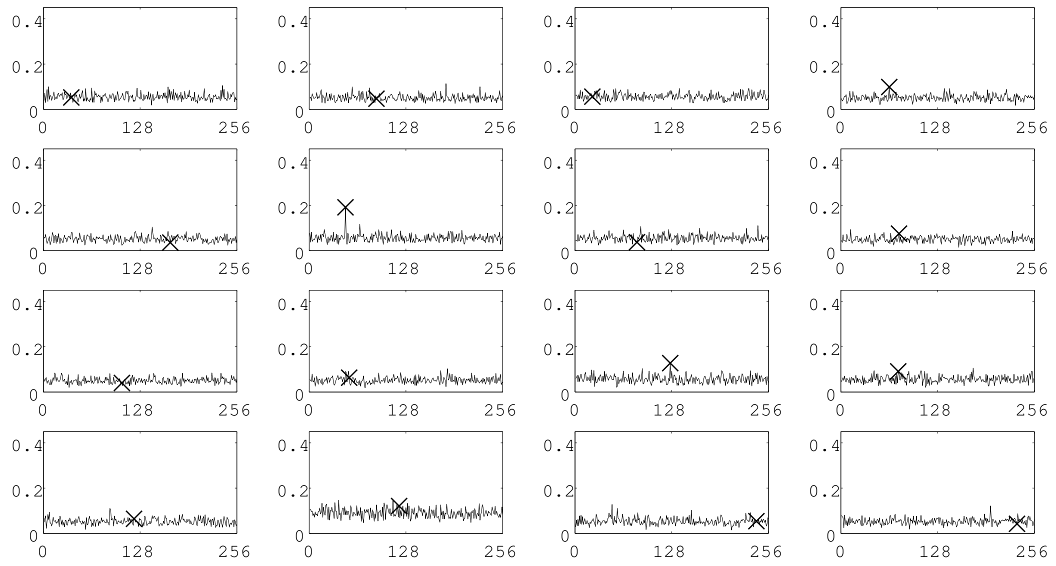
Fig. 6. Attack results for WDDL-AES using 1200 plaintexts. Each sub-figure corresponds to a key byte. The key guess is represented on the horizontal axis. The maximum 8-bit-based correlation coefficient is represented on the vertical axis. Each correct key byte is marked by an ×.

We believe that there are two reasons for this correlation. First, since the S-box of WDDL-AES is not based on S-boxes with clear gate arrays, even though the fault sensitivity for WDDL-AES is dependent on the input signals there is no clear correlation as in the case for PPRM1-AES. Second, assuming that the two complementary wires for the critical path have different timing delays, then the fault sensitivity will be correlated with this output signal since only $1 \rightarrow 0$ could occur for WDDL circuits under setup-time violations. For example, we assume that the timing delay of the true wire is longer than that for the false wire. As a result, the calculation with output signal (1,0) is more sensitive to a fast clock than the one with output signal (0,1), which leads to the correlation we observed in the experiments.

Compared to a single-rail circuit such as PPRM1, it is harder to apply the FSA attack to WDDL. Each path in the WDDL combinational logic tree has two wires that are supposed to have the same timing delay. However, practically the timing delays for two complementary wires cannot be exactly the same, so that the vulnerability to FSA attacks for practical WDDL implementations still exists. The unexploited key bytes may become exploitable by using a more precise experimental setup.

Another difficulty in attacking WDDL is that, since the fault injection is performed in the 9th round, there is influence from the key schedule and the MixColumns. In the proposed attack, we find that several bytes have the same fault sensitivity when we inject the fault in the 9th round, which indicates that the fault signal affects these bytes at the same time. This kind of fault is difficult to use in the FSA attacks.

As future work, we plan to investigate in more detail the fault sensitivity of WDDL circuits.

5 Conclusions

This paper proposed a new fault-based attack called Fault Sensitivity Analysis, which has lower attack requirements than those for Differential Fault Analysis. The FSA attacks are based on the dependency between the sensitivity data and the critical conditions where faulty outputs begin to exhibit detectable characteristics. Two practical FSA attacks against ASIC implementations of AES were shown in the paper. For PPRM1-AES, less than 50 plaintexts were needed to retrieve the full key. For WDDL-AES, which was shown to be immune to DFA attacks based on setup-time violation, the proposed FSA attack successfully retrieved 3 out of 16 key bytes with 1200 plaintexts.

6 Acknowledgement

The authors would like to thank the anonymous referees for their valuable comments. This research was partially supported by the Strategic International Cooperative Program (Joint Research Type), Japan Science and Technology Agency.

References

1. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In B.S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.
2. J. Blömer and J.-P. Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In *Financial Cryptography FC 2003*, volume 2742 of *LNCS*, pages 162–181. Springer Berlin, 2003.
3. A. Boscher and H. Handschuh. Masking Does Not Protect Against Differential Fault Attacks. In L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, *FDTC*, pages 35–40. IEEE Computer Society, 2008.
4. P. Dusart, G. Letourneux, and O. Vivolo. Differential Fault Analysis on A.E.S., 2003. Cryptology ePrint Archive, Report2003/010.
5. Research Center for Information Security (RCIS). Side-channel Attack Standard Evaluation Board (SASEBO). http://www.rcis.aist.go.jp/special/SASEBO/CryptoLSI-en.html.
6. C. Giraud. DFA on AES, 2003. Cryptology ePrint Archive, Report2003/008.
7. Y. Li, S. Gomisawa, K. Sakiyama, and K. Ohta. An Information Theoretic Perspective on the Differential Fault Analysis against AES, 2010. Cryptology ePrint Archive, Report2010/032.
8. A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh. A Generalized Method of Differential Fault Attack Against AES Cryptosystem. In L. Goubin and M. Matsui, editors, *Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 4249 of *Lecture Notes in Computer Science*, pages 91–100. Springer, 2006.
9. S. Morioka and A. Satoh. An Optimized S-Box Circuit Architecture for Low Power AES Design. In B. S. Kaliski Jr., Çetin K. Koç, and C. Paar, editors, *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2523 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2002.

10. D. Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In *AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 421–434. Springer, 2009.
11. S. Guilley T. Graba N. Selmane, S. Bhasin and J.-L. Danger. WDDL is Protected Against Setup Time Violation Attacks. In *FDTC*, pages 73–83. IEEE Computer Society, 2009.
12. G. Piret and J.-J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In C. D. Walter, Çetin K. Koç, and C. Paar, editors, *Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.
13. D. Saha, D. Mukhopadhyay, and D. RoyChowdhury. A Diagonal Fault Attack on the Advanced Encryption Standard, 2009. Cryptology ePrint Archive, Report2009/581.
14. K. Sakiyama, T. Yagi, and K. Ohta. Fault Analysis Attack against an AES Prototype Chip Using RSL. In M Fischlin, editor, *Proceedings of Cryptographer's Track at the RSA Conference (CT-RSA'09)*, pages 429–443, 2009.
15. K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *DATE*, pages 246–251. IEEE Computer Society, 2004.
16. M. Tunstall and D. Mukhopadhyay. Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault, 2009. Cryptology ePrint Archive, Report2009/575.

A Experimental Setup for Fault Sensitivity Analysis using Clock Glitch

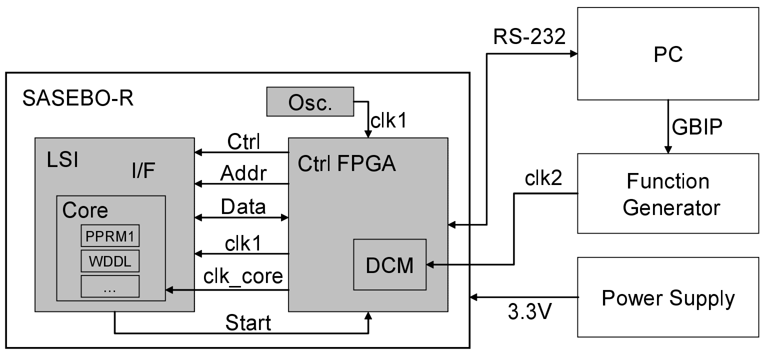

Fig. 7. Experimental setup for fault sensitivity analysis.

The experimental setup for FSA is shown in Fig. 7. The fault injection technique used in the proposed attack is the clock glitch. We use two clock supplies in the experiment system. The first clock supply, $clk1$, is generated by a 24 MHz oscillator and provided to the control FPGA and the I/F of the LSI to ensure that they work appropriately. The second clock supply, $clk2$, is generated from a function generator that is controlled by a PC through GBIP. By multiplying $clk2$, higher frequency clock clk_{hf} is generated using the Digital Clock Manager (DCM) inside the control FPGA. Then, based on $clk2$, clk_{hf}, and the start signal from the LSI core, we use the control FPGA to generate a special clock, clk_{core}, which is provided to the LSI core. Most

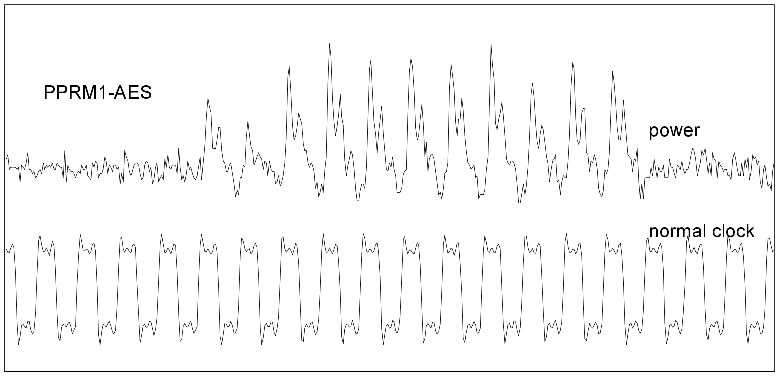

Fig. 8. Power trace of PPRM1-AES without fault injection (above) and clock supply without glitch (bottom).

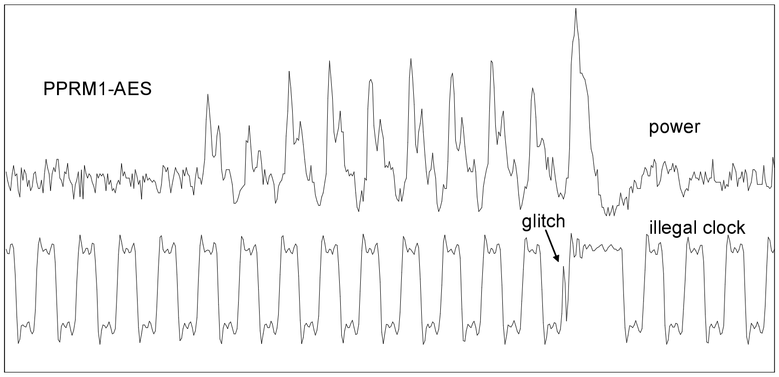

Fig. 9. Power trace of PPRM1-AES with fault injection (above) and clock supply with a glitch (bottom).

cycles of clk_{core} are the same as those for $clk2$, except one cycle is the same as the clk_{hf}, which triggers the computational fault at the time we want. Figures 8 and 9 show the power traces of PPRM1-AES and clocks clk_{core} without fault injection and with fault injection, respectively.

In order to reduce the total number of fault injections for a successful FSA attack, we first employ a binary search to determine a relatively high frequency for clk_{hf} that does not trigger any fault. Then we increase the frequency of clk_{hf} step-by-step and record the critical frequency for each byte of ciphertext. In the experiments, the period of clk_{hf} is decreased by approximately 35 picoseconds in each step. Furthermore, every plaintext is repeatedly used, until all of the bytes of ciphertext have been disturbed into a faulty value. In the worst case in the proposed attacks, a plaintext must be repeatedly used 120 times. In the experiment, we choose these parameters to make sure that 1) the recorded fault sensitivity has informative variations and 2) the level of efficiency in collecting the fault sensitivity information is tolerable. These parameters can be optimized to lead a more efficient FSA attack.