Faulty Clock Detection for Crypto Circuits Against Differential Faulty Analysis Attack

Pei Luo and Yunsi Fei

Department of Electrical and Computer Engineering
Northeastern University, Boston, MA 02115

Abstract. Differential fault analysis attack is a kind of serious threat to cryptographic devices. Previous protection schemes for crypto devices are not designed specifically against this kind of attacks. At the same time, previous schemes either incur large resource overhead or complex design work for different process technology. In this paper, we propose a method which can be easily implemented either in FPGAs or integrated circuits to detect the glitches in system clock. Results show that the proposed method can detect glitches efficiently while needs very few system resource and this method will not involve complex design work for different process technology.

Keywords: AES, differential fault analysis, side-channel attacks

1 introduction

Cryptographic applications are vulnerable to fault injection attacks. Differential Fault Analysis (DFA) was introduced by Biham et. al. on the Data Encryption Standard (DES) [1]. The authors in [2] showed that the attackers are able to break the AES-128 with only 2 faulty ciphertexts, assuming the fault occurs between the antepenultimate and the penultimate MixColumns. In [3], the authors show that inducing a random fault anywhere in one of the four diagonals of the state matrix at the input of the eighth round of the cipher leads to the deduction of the entire AES key. What's more, even if the fault induction corrupts two or three diagonals, 2 and 4 faulty ciphertexts are enough to uniquely identify the correct key.

In [3], the authors show that clock glitches can be used to inject faults into cryptographic devices. They run real-time fault injection using clock glitching via less sophisticated and less costly instruments on Xilinx FPGA platform and their results show that clock glitches can also be a means to induce internal faults. The authors in [4] also demonstrated the effectiveness of frequency injection attacks on a secure microcontroller and a highly secure FPGA chip.

The authors in [5] even propose a new fault-based attack called the Fault Sensitivity Analysis (FSA) attack. They make use of values of faulty ciphertexts and they show that faulty output may exhibit some detectable characteristics and such characteristics can be used to retrieve the secret key. They show that FSA attacks can be even used to break some protected AES implementations. After that, authors in [6] show that FSA and the Correlation Collision Attack [7] can be combined to create a even stronger attack method and the result shows that the proposed method can be used to attack some protected AES implementations efficiently.

To protect the cryptographic devices from such DFA attacks and improve the reliability of the system, many different schemes have been implemented. For AES specifically, different kinds of method have been proposed. For example, some simple linear error correction codes are added to the system and the redundancy are used to detect the injected faults [8,9]. At the same time, some works proposed to add another copy of AES such that the results of these two copies can be compared to find the differences. Because the DFA attacks always inject faults only into the last several rounds of AES, some works propose that reverse operations of the last several rounds can be implemented after the encryption/decryption to check the result.

In [10], the authors proposed a new method to detect the glitches instead of using redundancy to improve the security level against DFA attacks. In their method, a non-logic buffer-based delay chain is inserted, and then by monitoring the delay along the delay chain, a possible clock glitch based DFA can be detected.

In this paper, we propose a new method to monitor the clock signals and detect glitches in the system. The proposed scheme only involves very few redundancy of the existing circuits and the proposed method can be implemented very conveniently.

The following of this paper is organized as following: In Section 2, we revisit the conception of DFA attacks and previous protection methods. In Section 3, we introduce the conception of the proposed detection method and discuss its advantages over the previous schemes. In Section 4, we show the simulation results of the proposed methods, and compare the proposed scheme and the previous protection schemes based on the synthesis results. In Section 5, we conclude the paper.

2 Clock glitch fault injection and the protection schemes

2.1 Clock glitch fault injection

Previous papers demonstrated that faults can be injected between MixColumns in round 8 and SubBytes in round 10, DFA based on these injected faults can recover the last round key and thus recover the secret key.

Previous models all focus on the SubBytes of the last round and we can simplify the models shown as in Figure 1. For DFA attacks, the attacker first gets a group of (P, C, K_{10}), in which P is a random plaintext and C is the corresponding ciphertext with the last round key as K_{10}. With another pair of faulty pair (P, C', K_{10}), in which P'_{10} is faulty input of the last round caused by clock glitch.

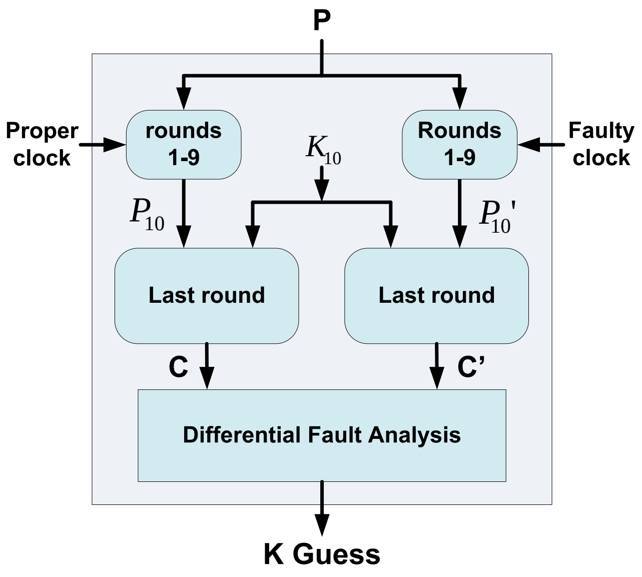

Fig. 1. Differential fault analysis model

The attacker makes an assumption of the last round key bytes K_i^{10}, in which i denotes the faulty bytes of C' comparing with C. For K_i^{10}, the attacker can recompute the last round and gets the corresponding last round input P_{10} and P'_{10}.

$$\begin{cases} P_{10} = AES_{10}^{-1}(C, K_i^{10}) \\ P'_{10} = AES_{10}^{-1}(C', K_i^{10}) \end{cases} \tag{1}$$

For some fault injection models, the attacker can only flip one bit thus P_{10} and P'_{10} have only one bit difference. For more complex faulty models, there will be more faulty bytes and the relationship of these faulty bytes can be used to extract the last round key bytes. Details of more complex DFA models and the attacking schemes can be found in [3,5,2].

2.2 Protection method

The general protection methods based on area redundancy is shown as in Figure 2. The method proposed in [8] is almost the same as Figure 2, they used two error detection modules to detect the errors in nonlinear parts and linear parts separately. The nonlinear protection scheme proposed in [8,11] appends a cubic computation module after the linear compressor and linear predictor and this scheme can also be simplified as in Figure 2 . For [9], the proposed CRC protection method separates the liner protection schemes into separated steps for MixColumns, ShiftRows and AddRoundKey operations.

At the same time, protection schemes based on duplicating the AES modules can also be simplified to Figure 2 by thinking the predictor as another copy of the AES modules. So without loss of generality, we use Figure 2 to discuss the protection methods based on area redundancy.

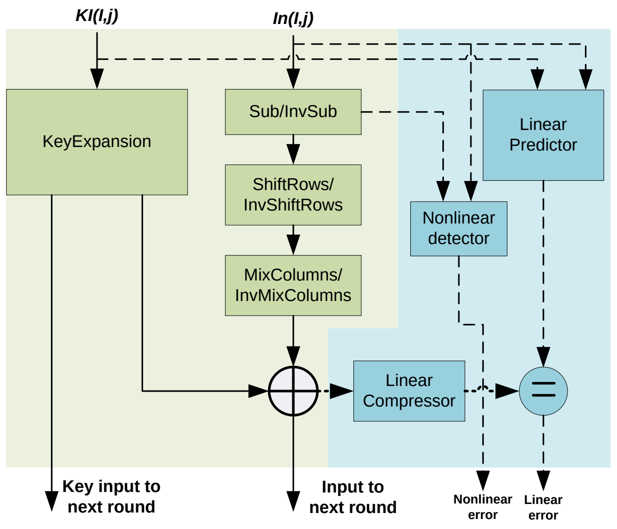

Fig. 2. Protection schemes against DFA based on redundancy and error detection codes

This kind of protection schemes can detect the injected faults when the results of the predictor and the compressor don't match. This method is useful when the faults are injected in only one of either the original circuits or the protection circuits, or the faults injected in both parts result in different errors. But for faults injected by clock glitches, the faults can happened at the input of these two parts, which means they have the same faulty input and thus generate the same faulty output, then it's very probable that the faults caused by clock glitches will be hidden.

3 Proposed clock glitch detection method

Different from the previous protection schemes which either need large resource overhead or need special consideration of technology in IC design phase, we propose a kind of clock glitch detection method which is easy to implement and can make good use of existing resource. More importantly, the designer can configure the precision of the system to meet different threshold.

3.1 Details of the proposed method

Denote the system clock used in the cryptographic system as *clock* and assume there is another clock source *clk* which has higher frequency in the system. Source *clk* can be used to measure the width of the target *clock*. The structure of the proposed scheme is shown as in Figure 3.

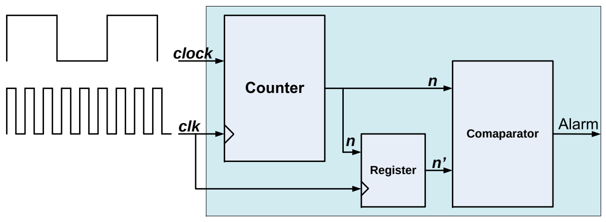

Fig. 3. The block diagram of the clock glitch detection circuit

Denote the counter result as n_0^H for the previous logic 1, which means the number of cycles of *clk* while *clock* = 1 in previous cycle. Meanwhile, define n_0^L the number of cycles of *clk* for *clock* = 0 in previous cycle. At the same time, use n_1^H and n_1^L to denote the width of the logic 1 and logic 0 of current cycle, shown as in Figure 4.

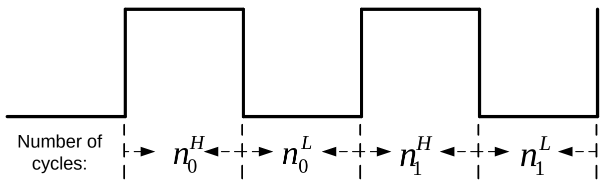

Fig. 4. Counter of the clock glitch detection circuit

If the target *clock* and the reference *clk* are both stable, then:

$$\begin{cases} n_0^L = n_1^L \\ n_0^H = n_1^H \end{cases} \tag{2}$$

For signals with 50% duty cycle, we have:

$$n_0^L = n_1^L = n_0^H = n_1^H \tag{3}$$

If the clock is not constant, then the width of logic '0' and logic '1' of *clock* are not equal, which means:

$$\begin{cases} n_0^L \neq n_1^L \\ n_0^H \neq n_1^H \end{cases} \tag{4}$$

3.2 Implementation of the proposed scheme

From the above analysis, the key point of this design is to have a stable high frequency reference clock source *clk*. For FPGA and some other embedded systems, such high frequency stable clock source can be obtained through external clock source such as oscillator and clock generator devices. Instead of using

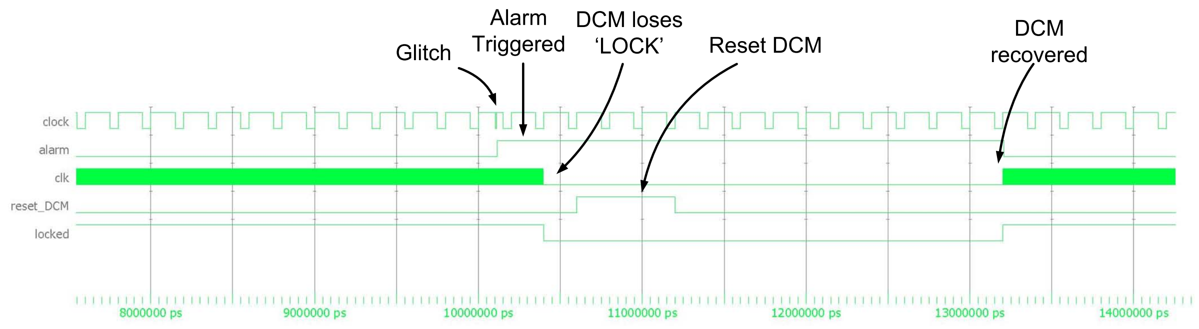

Fig. 5. Simulation result of the proposed scheme

external clock sources, clock generator can be designed inside the ICs in design phase. At the same time, a phase-locked loop (PLL) can also be implemented inside for higher frequency clock generation.

A substitution method to get *clk* is to use the existing resource to generate higher frequency using the target *clock*. Take FPGA as an example, a digital clock manager (DCM) or a PLL can be used to generate higher frequency with *clock* as the input clock. Using this method, only one clock source is needed. The Digital Clock Manager (DCM) primitive in Xilinx FPGA parts is used to implement delay locked loop, digital frequency synthesizer, digital phase shifter, or a digital spread spectrum [12]. The PLL in FPGA is used to generate multiple clocks with defined phase and frequency relationships to a given input clock [13,14].

Both PLL and DCM can be used to generate higher frequency *clk* using the target *clock*. The existed PLL and DCM module in FPGAs can be use to generate higher frequency. For example, there are up to 12 DCM modules in Virtex 5 FPGAs and each can be used to generate utmost $32X$ higher frequency, and cascaded DCMs can be used to generate even higher frequency output [15].

Assume one DCM module is used to generate $32X$ frequency *clk*, then for stable *clock*, $n_0^L = n_1^L = n_0^H = n_1^H = 16$. If there is a glitch in *clock* shown as in Figure 6. Then it's obvious that:

$$\begin{cases} n_0^L \neq n_1^L,\ n_1^L \neq n_2^L \\ n_0^H \neq n_1^H,\ n_1^H \neq n_2^H \end{cases} \tag{5}$$

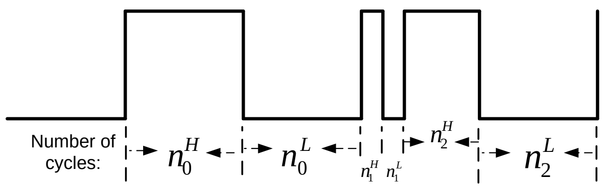

Fig. 6. When glitch happens in *clock*

For PLL and DCM, if there is a glitch in *clock*, then the glitch will be detected but the frequency generation module will lose *LOCK* and some operations are needed to reset the module. The details will be explained and simulated in Section 4.

3.3 Advantage of the proposed scheme

From the discussion in Section 3.1, the first advantage of the proposed scheme comparing with the previous schemes is that the proposed schemes has low resource requirement. The proposed scheme can be easily implemented in FPGAs and the resource overhead is very low. For ICs, only another higher frequency clock source or a clock management module (PLL, etc) is needed.

Another advantage of the proposed scheme is that it's very easy to reconfigure the module according to the precision requirement of the system. The method proposed in [10] requires to inject a non-logic buffer-based delay chain to the circuits and the delay cannot be changed after production. What's more, the delay and the delay chain design are affected by the process technology and this improve the design difficulty and complexity.

Meanwhile, our method can be easily implemented in both FPGAs and ICs, and this scheme is highly reconfigurable. First of all, it's easy to control the frequency of reference clock *clk*, thus higher precision can be achieved by using higher frequency *clk*. Secondly, our scheme has configurable threshold of clock variation. Assume that the clock has variation and the width of *clock* is not constant, we note that if the difference between n_0^L and n_1^L, the difference between n_0^H and n_1^H are smaller than λ, then the variation is acceptable and it's not caused by DFA. Which means that if the following equations hold, then it's not a DFA caused glitch:

$$\begin{cases} |n_0^L - n_1^L| < \lambda \\ |n_0^H - n_1^H| < \lambda \end{cases} \tag{6}$$

In conclusion, the proposed scheme is easy to implement and it's easy to reconfigure the detection module according to different requirements and implementations. What's more, the proposed scheme needs much less resource than previous methods.

4 Implementation and simulation results

To verify the functionality of the proposed scheme, we use Virtex-5 FPGA and its internal DCM to implement the proposed scheme. The simulation result is shown as in Figure 5.

The target *clock* is running at 5 MHz and its duty cycle is 75%. We use DCM to generate the higher frequency clock *clk* with the frequency 160 MHz. From the result we can see that:

– The proposed scheme can detect glitches efficiently and it will trigger the alarm when glitch detected;
– The proposed scheme can recover monitoring very soon after losing "lock" of the target clock.

To compare the overheads of the proposed glitch detection scheme and previous AES protection schemes, we implement a glitch detection scheme based on the original AES using PLL in Verilog. We model glitch detection scheme and the above different protection schemes in Verilog and synthesized in Cadence Encounter RTL Compiler with the Nangate 45nm Opencell library version v2009_07. The designs were placed and routed using Cadence Encounter. The latency, the area overhead of the protection schemes were estimated using Concurrent Current Source (CCS) model under typical operation condition assuming a supply voltage of 1.1V and a temperature of 25 Celsius degree. The synthesis results for the schemes are shown in Table 1 and Figure 7.

Table 1. Comparison between different protection schemes

Protection Schemes	Overheads	
	Area (um^2)	Power mW
Original	13987.9	20.4
Duplication	25893.0	65.18
Linear[8]	18299.5	30.61
Robust[11]	36711.2	136.7
$CRC4:1$[9]	19504.5	52.01
$CRC8:1$[9]	19804.5	52.97
Glitch	14392.5	22.76

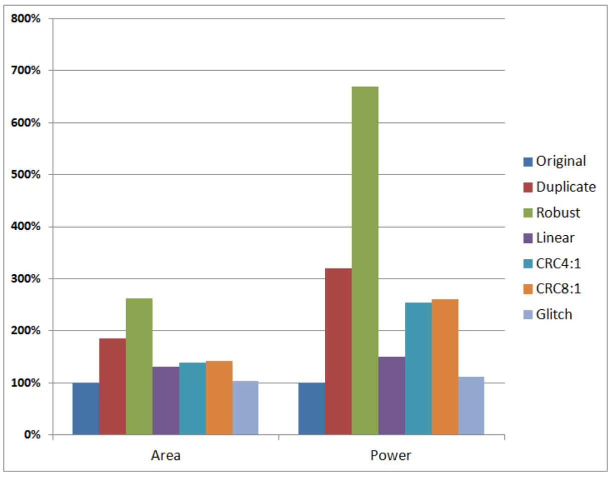

Fig. 7. Overhead comparison of different protection schemes

From the synthesis results shown as in Table 1 and Figure 7, it's obvious that the proposed glitch detection scheme needs much less resource overhead than the previous proposed schemes. For example, the proposed scheme only need about 103% area of the original AES scheme while it consumes only about 112% power of the original AES implementation. At the same time, the previous protection schemes such as the duplication protection method even needs about 185% area and about 320% power resource of the original AES implementation.

5 Conclusion

In this paper, we propose a simple method to detect clock glitches and the results show that the proposed scheme can detect glitches in clock efficiently while needs few resource overhead. Compare with previous schemes, the proposed scheme is designed specifically for clock glitch detection and it's highly reconfigurable. The proposed scheme involves no complex work in design phase for different process technology.

The simulation and synthesis results show that the proposed scheme can detect the clock glitches efficiently and it needs much fewer resource than the previous protection schemes.

References

1. E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems,” in *Advances in CryptologyCRYPTO’97.* Springer, 1997, pp. 513–525.
2. G. Piret and J.-J. Quisquater, “A differential fault attack technique against spn structures, with application to the aes and khazad,” in *Cryptographic Hardware and Embedded Systems-CHES 2003.* Springer, 2003, pp. 77–88.
3. D. Saha, D. Mukhopadhyay, and D. R. Chowdhury, “A diagonal fault attack on the advanced encryption standard.” *IACR Cryptology ePrint Archive*, vol. 2009, p. 581, 2009.
4. S. Skorobogatov, “Synchronization method for sca and fault attacks,” *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 71–77, 2011.
5. Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, “Fault sensitivity analysis,” in *Cryptographic Hardware and Embedded Systems, CHES 2010.* Springer, 2010, pp. 320–334.
6. A. Moradi, O. Mischke, C. Paar, Y. Li, K. Ohta, and K. Sakiyama, “On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting,” in *Cryptographic Hardware and Embedded Systems–CHES 2011.* Springer, 2011, pp. 292–311.
7. A. Moradi, O. Mischke, and T. Eisenbarth, “Correlation-enhanced power analysis collision attack,” in *Cryptographic Hardware and Embedded Systems, CHES 2010.* Springer, 2010, pp. 125–139.
8. M. Karpovsky, K. J. Kulikowski, and A. Taubin, “Differential fault analysis attack resistant architectures for the advanced encryption standard,” in *Smart Card Research and Advanced Applications VI.* Springer, 2004, pp. 177–192.
9. C.-H. Yen and B.-F. Wu, “Simple error detection methods for hardware implementation of advanced encryption standard,” *Computers, IEEE Transactions on*, vol. 55, no. 6, pp. 720–731, 2006.
10. H. Igarashi, Y. Shi, M. Yanagisawa, and N. Togawa, “Concurrent faulty clock detection for crypto circuits against clock glitch based dfa,” in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on.* IEEE, 2013, pp. 1432–1435.
11. M. Karpovsky, K. J. Kulikowski, and A. Taubin, “Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard,” in *Dependable Systems and Networks, 2004 International Conference on.* IEEE, 2004, pp. 93–101.
12. *DS485: Digital Clock Manager (DCM) Module.*
13. *DS622: Phase Locked Loop (PLL) Module (v2.00a).*
14. *Altera Phase-Locked Loop (Altera PLL) Megafunction User Guide.*
15. *UG190: Virtex-5 FPGA User Guide.*