RESEARCH ARTICLE OPEN ACCESS

Countermeasures Against Double Fault Attacks Using Reconfigurable Array Architecture

Mr. N. Vel Murugesh Kumar
Asst. Professor(SG)
Saveetha Engineering College
Thandalam, Chennai
Velmurugeshkumar@saveetha.ac.in

Ms. A. Babisha
Asst. Professor(OG)
Saveetha Engineering College
Thandalam, Chennai
babisha@saveetha.ac.in

Abstract :

With the increasing accuracy of fault injections, it has become possible to inject two faults into specific circuit regions precisely at a certain time. Unfortunately, most existing fault attack countermeasures are based on the single fault assumption, and it is, therefore, very difficult to resist double fault attacks. Both the time and spatial locations of the injections should be controlled precisely in the double fault attacks to form a pair of exploitable faults. Reconfigurable array architecture (RAA) has the ability to introduce spatial and time randomness by dynamic reconfiguration, which can alleviate the threat of double fault attacks. This project analyzes the double fault attack issues in the fault injection phase systematically. An evaluation model, named injection effort model (IEM), is proposed to quantify the efforts of a successful fault injection. In IEM, the real injection process is described mathematically using the probability method, so that a theoretical basis can be provided for the corresponding countermeasure design. Based on the concept of spatial and time randomization, three countermeasures RBR, RPS, RDI are implemented on RAA for the purpose of decreasing the implementation overhead under the premise of ensuring the security. Experiments are carried out to analyze the relationship between the resistance and the overhead using Advanced Encryption Standard (AES) and Data Encryption Standard (DES). It could be shown that when the three countermeasures are used in combination, a higher degree of randomness can increase the resistance to a larger extent.

Keywords:

Injection Effort Model; Reconfigurable Array Architecture; Advanced Encryption Standard; Data Encryption Standard

I. INTRODUCTION

Fault injections has significantly improved in recent years. For example, the spot size of laser injections has already reached the logic gate level and the time accuracy has been of sub-nanosecond scale. In this way, it is possible to precisely inject two faults into specific circuit regions at a certain time during one execution of the cryptographic algorithms (the encryption of one data block for example). Most existing fault attack countermeasures cannot resist such kind of double fault attacks. This is because the most commonly used redundancy countermeasures, which use redundant computation (achieved by duplicating the function or recomputing for example) and comparison to check for errors, are based on the single fault assumption [4]–[6]. In the single fault assumption, it is believed that the attacker is not able to precisely inject two specified faults during one execution. However, with the increasing accuracy of fault injections, two identical faults can be injected into the main processing path (i.e. original calculation logic or process) and the redundant path (i.e. additional calculation logic or process to provide information for the comparison) simultaneously. This will disable the comparison operation and lead to a failure of detecting the error.

Fault injection is a technique for improving the coverage of a test by introducing faults to test code paths. Error handling code path is often used with stress testing and is widely considered to be an important part of developing robust software. Robustness testing is a type of fault injection commonly used to test for vulnerabilities in communication interfaces such as protocols, command line parameters, or APIs. The propagation of a fault through to an observable failure follows a well-defined cycle. When executed, a fault may cause an error, which is an invalid state within a system boundary. An error may cause further errors within the system boundary, therefore each new error acts as

a fault, or it may propagate to the system boundary and be observable. When error states are observed at the system boundary they are termed failures. This mechanism is termed the fault-error-failure cycle and is a key mechanism in dependability. Fault attacks introduce errors in cryptographic algorithms to obtain faulty ciphertexts in order to reveal the cipher key. Double fault attacks disable the comparison process by injecting two identical faults so that the faulty ciphertexts can still be achieved when redundancy countermeasures are adopted. The principles of fault attacks, i.e. the specific ways to analyze the faulty ciphertexts, are actually the same for double fault attacks and traditional single fault attacks. The fault model, which defines the size (bit or byte level), the location of the faults, determines the basic analysis methods of fault attacks. As many other related articles, the fault attacks are classified into two types based on the size of errors in the fault model. They are the faults occurring at the bit level (single-bit errors) and at the byte level (single-byte errors). The safe error attacks which aim at setting the intermediate variable to a particular known value are not considered. The reason is that the value of faults is more likely to be random for the common methods of double fault attacks including optical fault injections and electromagnetic injections. Faults induced on software are considered transient and such transient faults can be viewed from various angles.

Reconfigurable array architecture (RAA) is fully equipped with the capability described above: it has the ability to modify the structure of the device through dynamic reconfiguration. Many researches have been done to explore design methods of RAA for cryptographic implementations. This is because RAA can provide flexibility for security applications while still maintaining high energy efficiency (i.e. throughput/power) at the same time. The model, named injection effort model (IEM), gives a comprehensive analysis of the injection phase and further divides this real attack process into the search stage and continuous injection stage. Unlike traditional fault attack countermeasures which are aimed at detecting faults passively, the methods in this paper reduce the probability of successful injections by introducing randomness. A complete solution against double fault attacks, including two spatial randomization methods named round based relocation (RBR) and register pair swap (RPS), one time randomization method named random delay insertion (RDI), are implemented on RAA and evaluated by IEM. Experimental results show that these countermeasures can be implemented on the existing RAA through adding a small amount of random control logic, and the double fault resistance of the system is improved dramatically with low overhead.

II. MOTIVATION

As one kind of physical attacks, fault attacks pose serious threats to hardware security through maliciously injecting faults. In this way, it is possible to precisely inject two faults into specific circuit regions at a certain time during one execution of the cryptographic algorithms. Most existing fault attack countermeasures cannot resist such kind of double fault attacks. This is because the most commonly used redundancy countermeasures, which use redundant computation and comparison to check for errors, are based on the single fault assumption. In the single fault assumption, it is believed that the attacker is not able to precisely inject two specified faults during one execution. However, with the increasing accuracy of fault injections, two identical faults can be injected into the main processing path and the redundant path simultaneously. This will disable the comparison operation and lead to a failure of detecting the error. Hence it is required to develop some countermeasures to resist the double fault attacks.

III. RELATED WORK

Fault injection is a technique for improving the coverage of a test by introducing faults to test code paths. Error handling code path is often used with stress testing and is widely considered to be an important part of developing robust software. Robustness testing is a type of fault injection commonly used to test for vulnerabilities in communication interfaces such as protocols, command line parameters, or APIs. The propagation of a fault through to an observable failure follows a well-defined cycle. When executed, a fault may cause an error, which is an invalid state within a system boundary. An error may cause further errors within the system boundary, therefore each new error acts as a fault, or it may propagate to the system boundary and be observable. When error states are observed at the system boundary they are termed failures. This mechanism is termed the fault-error-failure cycle and is a key mechanism in dependability. A significant amount of researchers has gone into exploring infection methods for block ciphers. In earlier studies, deterministic computations, which mainly consist of simple linear operations such as swapping or exclusive-OR, are used to accomplish infection functions. These operations are easy to implement and only induce relatively low overheads. However, due to the determinacy of the infection, if the function details are known by the attacker, the fault diffusion pattern is still obtainable by modifying the attack method. Here, the security of these countermeasures relies on the confidentiality of the methods themselves. In order to solve this problem, randomness is introduced into the infection countermeasures so that the attack invariant can be negated by unpredictability. For example, the random execution of dummy cipher rounds or random multiplicative masks can be used . However, various

targeted attacks are developed against most of these countermeasures, which indicates that security vulnerabilities still exist in these random-enhanced methods. One of the primary reasons for this is the tight coupling between the infection and algorithm calculations. In this case, the weaknesses of the infective computation, such as the biased, repeated, or incomplete masking of the ciphertext, may be easier to explore by taking a combined mathematical analysis of the algorithm as a back door.

Fault attacks introduce errors in cryptographic algorithms to obtain faulty ciphertexts in order to reveal the cipher key. Double fault attacks disable the comparison process by injecting two identical faults so that the faulty ciphertexts can still be achieved when redundancy countermeasures are adopted. The principles of fault attacks, i.e. the specific ways to analyze the faulty ciphertexts, are actually the same for double fault attacks and traditional single fault attacks. The fault model, which defines the size (bit or byte level), the location of the faults, determines the basic analysis methods of fault attacks. As many other related articles, the fault attacks are classified into two types based on the size of errors in the fault model. They are the faults occurring at the bit level (single-bit errors) and at the byte level (single-byte errors). The safe error attacks which aim at setting the intermediate variable to a particular known value are not considered. The reason is that the value of faults is more likely to be random for the common methods of double fault attacks including optical fault injections and electromagnetic injections. Faults induced on software are considered transient and such transient faults can be viewed from various angles.

IV PROPOSD WORK

An evaluation model targeting at the injection phase and a set of randomization countermeasures are proposed in this project to respond to the threat of double fault attacks. The model, named injection effort model (IEM), gives a comprehensive analysis of the injection phase and further divides this real attack process into the search stage and continuous injection stage. Unlike traditional fault attack countermeasures which are aimed at detecting faults passively, the methods in this proposed work reduce the probability of successful injections by introducing randomness. A complete solution against double fault attacks, including two spatial randomization methods named round based relocation (RBR) and register pair swap (RPS), one time randomization method named random delay insertion (RDI), are implemented on RAA and evaluated by IEM. These countermeasures can be implemented on the existing RAA through adding a small amount of random control logic to improve the double fault resistance of the system.

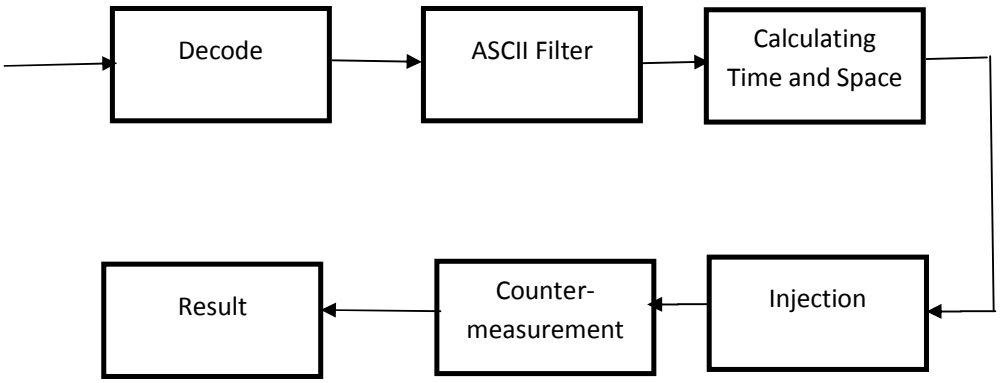

Figure 1 Proposed System Architecture

The overall architecture of the proposed system is shown in figure 4.1. Messages from client are decoded by using recursive traversal decoding technique. Then the decoded message is sent to the ASCII filter. ASCII filter reads the file bits. Then the time and space is calculated based on the file size. After calculating the time and space the virus is injected into the original file using Injection Effort Model (IEM). After the injection the injected file space and time is calculated. By using the countermeasures (RBR, RDI, RPS) the virus is detected. Then the virus is removed and original message is sent to the receiver using Reconfigurable Array Architecture Algorithm.

ALGORITHM

Reconfigurable Array Architecture Algorithm

***Step* 1.** If instruction sequence s_a is a subsequence of instruction sequence s_b, then exclude s_a. The rationale for excluding s_a is that if s_a satisfies some characteristics of programs, s_b also

satisfies these characteristics with a high probability.

Step 2. If instruction sequence s_a merges to instruction sequence s_b after a few instructions and s_a is no longer than s_b, we exclude s_a. It is reasonable to expect that s_b will preserve s_a's characteristics.

Step 3. For some instruction sequences, when they are executed, whichever execution path is taken, an illegal instruction is inevitably reached. We say an instruction is inevitably reached if two conditions hold. One is that there are no cycles (loops) in the EIFG of the instruction sequence; the

other is that there are no external address nodes in the EIFG of the instruction sequence.

A) RECURSIVE TRAVERSAL DECODING

In this module disassemble the request from a certain address until the end of the request is reached or an illegal instruction opcode is encountered. There are two traditional disassembly algorithms: linear sweep and recursive traversal. The linear sweep algorithm begins disassembly at a certain address and proceeds by decoding each encountered instruction. The recursive traversal algorithm also begins disassembly at a certain address, but it follows the control flow of instructions. Here recursive traversal algorithm is used, because it can obtain the control flow information during the disassembly process.
Calculate space by using the original file size using the formula,

Space = File size in bits / 1000
File size in bits = File Size * 8.

Calculate time by using the original file size using the formula,

Time = File Size / 120

At the time of transmission the packet sizes are calculated. This is known as the physical size. Physical size is calculated by using the formula,

Physical Size = Logical Size * 1.5

Bandwidth is the amount of data that can be transmitted in a fixed amount of time. It is calculated by using the formula,

Bandwidth = File Size / 14.4

Backup window size per local space (BSWL) is the no. of bits transmitted. It is the maximum workload for transmission. It is calculated by using the formula,

BSWL = T + (C/B) * (PL /L)

Where,

T – Transfer time
C – Chunk
B – Bandwidth
PL - Physical Size
L – Logical Size.

B) *FAULT ATTACK BY INJECTION*

This module defines the size (bit or byte level), the location (which intermediate variable of the cipher to affect) and the logical effect (how the intermediate variable is affected) of the faults, determines the basic analysis methods of fault attacks. It is divided into the search stage and the continuous injection stage. In this, the calculation steps in the algorithm which can produce particular kinds of faulty bits are defined as sensitive points (SPs).Then the search stage is the process to find the precise time and spatial locations of the injections to excite these SPs. The continuous injection stage is defined as the procedure to inject a sufficient number of faulty bits when adopting the injection parameters obtained by the search stage. A set of randomization countermeasures is also implemented on a RAA platform.

C)DETECTION MODULE

This module use divide-and-conquer method, i.e. divide the data region into many equal units and search individually. After the injection the occurrence of virus is detected. This is done by calculating the file size, once the data is transmitted. The difference in file size calculated before and transmitting data, shows that the information is attacked.

D) FAULT RESISTANCE

Countermeasure is to map one SP onto different AEs (or AE pairs) randomly among multiple executions. The solution against double fault attacks, including two spatial randomization countermeasures and one time countermeasure. The space countermeasure, Round Based Relocation(RBR) is used to verify the spatial location. The time countermeasure Random Delay Insertion (RDI) is used to verify the time of SP. Register Pair Swap(RPS) is used to change the correspondence between the computing units and registers.

V EXPERIMENTAL EVALUATION

a}Before Injection

When the message is sent from client to server, initially there is no injection in original file. So the original file space and time is calculated.

b)After Injection

Virus is added to the original message and the time and space is calculated.

c)Space

Space calculation of an algorithm is the amount of memory it needs to run to completion.
Space = File Size in bits / 1000
File size in bits = File size * 8

Table 1 Space calculations

The average value of Space calculated for before and after injection are graphically represented in figure 2,

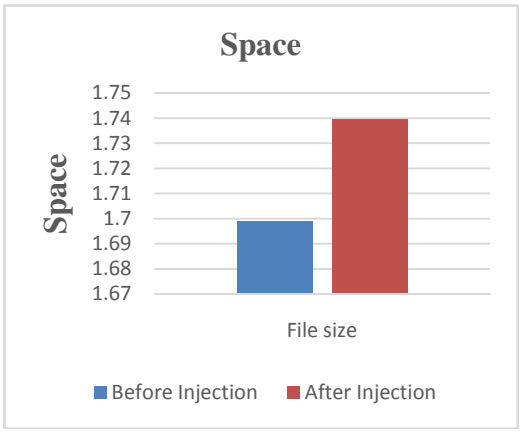

Figure 2 Space Graph

d)Transfer Time

Here transfer time is calculated. The Transfer time is the estimated time for the completion of a data transmission. Because most data transfers are not fixed, transfer times may increase and decrease as the data transmission is occurring.

Time = File Size / 120

Table 2 Transfer Time Calculation

File Size	Transfer Time	
	Before Injection	After Injection
381.63	3.1802	3.2556
159.13	1.326	1.3572
96.44	0.8036	0.8233
Average	1.7699	1.8120

File Size	Space	
	Before Injection	After Injection
381.63	3.053	3.1254
159.13	1.273	1.3028
96.44	0.7715	0.7904
Average	1.6991	1.7395

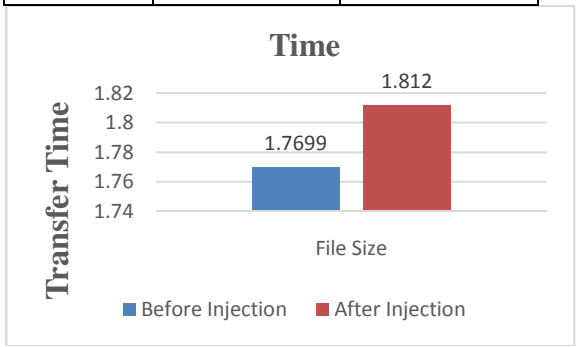

Figure 3 Transfer Time Graph

e) Server

The Injected virus are detected and removed by using Reconfigurable Array Architecture Algorithm. And send only the correct message to the receiver. Table 5.3 shows the server receiving file size occupying space and the transfer time, which is same as that of original file size occupying space and transfer time.

Table 3 Received Space And Time

File Size	Space	Time
212.4	1.6991	1.7699

The server receiving file size occupying memory and transfer time is graphically represented in figure 4,

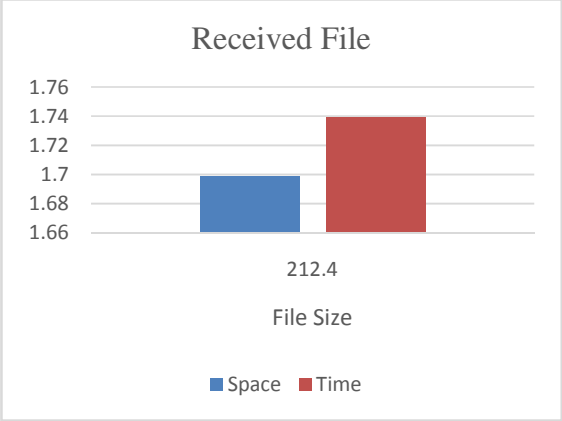

Figure 4 Received space and time graph

VI CONCLUSION

IEM is proposed to respond the threat of double fault attacks. In IEM, the increase of fault numbers does not affect the basic form, but only the value of variables, this means the basic derivation can be extended to the situation of three or more faults. The modelling of the search stage and continuous injection stage is consistent with the fault injection process, so the basic analysis method of IEM is not limited to RAA. The double fault attack countermeasures, used to reduce the probability of success injection through time and spatial randomization, so the countermeasures can also be applied against multi-fault attacks.

REFERENCES

[1] Battistello and C. Giraud, "Fault analysis of infective AES computations," in Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC), Aug. 2013, pp. 101–107.

[2] R. Beat, P. Grabher, D. Page, S. Tillich, and M. Wojcik, "On reconfigurable fabrics and generic side-channel countermeasures," in Proc. 22nd Int. Conf. Field Program. Logic Appl. (FPL), Aug. 2012, pp. 663–666

[3] J. Blömer and J.-P. Seifert, "Fault based cryptanalysis of the advanced encryption standard (AES)," in Financial Cryptography (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2003,pp. 162–181.

[4] B. Gierlichs, J.-M. Schmidt, and M. Tunstall, "Infective computation and dummy rounds: Fault protection for block ciphers without check before-output," in Progress in Cryptology—LATINCRYPT. Heidelberg, Germany: Springer, 2012, pp. 305–321.

[5] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in Cryptographic Hardware and Embedded Systems—CHES (Lecture Notes in Computer Science). Berlin, Germany: Springer, Jan. 2011, pp. 33–4.

[6] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, "Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller," in Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC), Aug. 2013, pp. 77–88.

[7] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight high performance fault detection scheme for the advanced encryption standard using composite fields," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 1, pp. 85–91, Jan. 2011.

[8] C. Roscian, J.-M. Dutertre, and A. Tria, "Frontside laser fault injection on cryptosystems—Application to the AES," in Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST), Jun. 2013, pp. 119–124.

[9] D. Saha, D. Mukhopadhyay, and D. RoyChowdhury, "A diagonal fault attack on the advanced encryption standard," Cryptology ePrint Archive, Tech. Rep. 2009/581, 2009. [Online].

[10] J. G. J. van Woudenberg, M. F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," in Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC), Sep. 2011, pp. 91–99.

[11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. Journal of Cryptographic Engineering, 1(1):5–27, March 2011.

[12] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized Electromagnetic Analysis of Cryptographic Implementations. In CT-RSA, Lecture Notes in Computer Science, pages 231–244. February 2012.