SoK: Fault Injection Attacks on Cryptosystems

TingHung Chiu
kennychiu0818@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Wenjie Xiong
wenjiex@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

ABSTRACT

Fault injection attacks are a powerful technique that intentionally induces faults during cryptographic computations to leak secret information. This paper provides a survey of fault attack techniques on different cryptosystems. The fault attack consists of two main components: fault modeling, which examines methods for injecting faults in a target device, and fault analysis, which analyzes the resulting faulty outputs to deduce secrets in each cryptosystem. We first categorize various fault attack methods by fault model and fault analysis. We then give examples of the various fault attacks on symmetric key cryptosystems and public key cryptosystems. This paper aims to provide a background on fault attack research and directions for further study on securing real-world cryptosystems against fault injection attacks.

ACM Reference Format:
TingHung Chiu and Wenjie Xiong. 2023. SoK: Fault Injection Attacks on Cryptosystems. In *Hardware and Architectural Support for Security and Privacy 2023 (HASP '23), October 29, 2023, Toronto, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3623652.3623671

1 INTRODUCTION

Cryptography plays a critical role in guaranteeing the security in digital infrastructure today. The confidentiality and integrity of sensitive data are built on cryptosystems like encryption algorithms and signature schemes. While the security of these primitives is proven rigorously, the implementation can still be vulnerable to attacks. Fault injection attacks are a type of attack that maliciously induces fault during the computations to breach security.

The first fault injection attack was introduced in the late 90s [4], which used Differential Fault Analysis (DFA) to recover secrets. By carefully injecting faults during the computations, attackers are able to obtain secret information by analyzing the faulty output. According to the existing research, this powerful attack technique has demonstrated the ability to break the widely used cryptosystems such as recovering secret keys in Advanced Encryption Standard (AES) [16] and TLS singing key [26].

There are many different techniques to inject faults, e.g., laser [34], voltage glitching [12], and rowhammer [26] using the software, and many different cryptosystems can be affected. Prior research either only focuses on the hardware layer to study how to inject faults in the system [14, 31, 42] or only focuses on attacking individual cryptographic implementations [16, 26]. Under a given attacker model, it is not easy to know what cryptosystems are affected. *Anubhab Baksi et al.* [2] provides a survey of fault attacks with a focus only on symmetric key cryptosystems. In this paper, we aim to provide a survey across different fault models and different cryptosystems. We want to bridge the gap between studies on hardware fault injection techniques and studies on the analysis of cryptosystems.

In this paper, Section 2, we give an overview of fault injection attacks and present a detailed analysis of the fault models. Section 3 describes the concept of previous fault attacks on symmetric key cryptosystems, and Section 4 with fault attacks on public key cryptosystems. Section 5 provides a summary of the fault models of the attacks on the different cryptosystems.

2 OVERVIEW OF FAULT INJECTION ATTACKS

Fault injection attacks allow adversaries to derive secret information by intentionally inducing faults during cryptographic processing. As introduced in the previous chapter, fault attacks can be divided into two main components. The first involves inducing the fault in the target device. A fault model describes the type of faults injected for a given method. The second focuses on fault analysis, examining the corrupted outputs after fault injection to reveal secrets through techniques like differential analysis. In this section, we will categorize and analyze common fault attack approaches, considering both fault models and fault analysis methods.

2.1 Existing Fault Injection Methods

To process the fault attack, the attacker first needs to induce the fault during the cryptographic operation. They can exploit various fault injection methods to achieve the objective. We categorize the fault injection methods into three types.

The attackers can inject fault by directly contacting the target device, such as a Voltage [9] or clock glitch [13]. A voltage glitch refers to a spike in short duration in the voltage supply to a device. The voltage fluctuation may cause unexpected behaviors that could induce the fault during the operation. Besides, the Clock provides the timing signals that synchronize the digital logic operations, and the clock glitch may disrupt the circuit timing and cause errors.

Another type is that attackers are near the device without direct contact, such as EM pulse [32], Laser [36], or heating [18]. EM pulse is a burst of electromagnetic energy that can disrupt or even damage the electronic device, and this kind of energy could cause the fault to occur without contact with the device. Like an EM pulse, the Laser could also inject the fault by the device's malfunction resulting from the localized heating and burning. Heating is using any method to increase the device's temperature, resulting in the device malfunctioning to induce the fault. Although these attacks do not need direct contact during the fault injection, depackaging is

HASP '23, October 29, 2023, Toronto, Canada

ACM ISBN 979-8-4007-1623-2/23/10.
https://doi.org/10.1145/3623652.3623671

sometimes needed, e.g., for laser [39]. The attacker typically needs to have physical access to the target device.

The last type of fault injection method is software-induced error. For example, rowhammer [22] could flip the particular bit(s) in DRAM arrays by repeatedly accessing the adjacent row of the target bits in the memory, causing the faulty output to derive the secrets. Not all the DRAM cells are vulnerable to rowhammer, thus, attacks leveraging rowhammer usually have a profiling phase to select the memory location for fault injection, and manipulate the page mapping to map the victim address to the desired memory location. Such software-induced errors can occur in cloud servers and other environments with shared memory if the attacker's software can share the same system as the victim.

2.2 Fault Models

There are various methods to induce fault in the hardware using different physical effects. However, during the Fault attack analysis for cryptosystems, the attacker cares about the fault type instead of the way to inject the faults. Thus, we introduce the fault model in this section. We categorize existing fault attacks based on the "Fault Model," which refers to modeling the induced fault in the algorithm using the following four properties.

Target Which hardware component is impacted by the fault attack, e.g., arithmetic-logic unit, SRAM, DRAM, etc..

Precision How precisely can the attacker choose the fault location or the time to inject the fault.

Repeatability Whether the attacker can repeatedly induce the identical fault.

Feasibility Practical property (cost, accuracy of time and position, technical skill requirement) for inducing the fault.

2.2.1 Fault Target. In existing research, different types of physical attacks will result in the fault by perturbing different devices. For instance, in the voltage glitching attack in [3], the fault target is the execution unit; in the rowhammer attack in [22, 26], the fault target device is DRAM.

Depending on the hardware structures, we can categorize the existing fault attack research by the fault target aspect and divide it into two main parts: processor Core and Memory/storage. Different memories are vulnerable to different fault injection methods due to the materials used. For example, DRAM is vulnerable to radiation [21, 40], while the disk is almost radiation-hard. Regarding the cryptosystem implementation, the target devices are related to where the cryptography parameters are stored and computed. For instance, the fault attack method aimed to derive TLS signing keys stored in DRAM targets the DRAM [26], and the device that processes RSA-CRT keeps the secret parameter in EEPROM [6]. According to the fault target, the attacker should model the fault depending on the physical conditions, such as which devices are vulnerable or where the key parameters are stored.

2.2.2 Precision. Precision is the property that indicates how precisely the attacker can choose the fault value or when to inject the fault. We categorize fault attacks by spatial resolution, which indicates how precise the fault value is, and by temporal resolution, which indicates how precise the attacker should control when the fault is injected.

Table 1: Precision Aspect of Fault Model

Precision Type		Victim Algorithm
Spatial resolution	Bit	AES [3, 16], ECDSA[26]
	Multi-Bit	AES [11, 16], DSA [27], ECDSA [17]
	Random	RSA [41], PQC [38]
Temporal resolution	Rough	ECDSA[26], DSA [27]
	Precise	AES [3, 11, 16], RSA[41], ECC[7] ECDSA [17], PQC [38]

Table 2: Summary of different attack methods.

	Cost	Accuracy		Technical skill requirement
		Spatial	Temporal	
Clock glitch	Low	Global	High	Low
Voltage glitch	Low	Global	Medium	Low
EM pulse	Low	Global	Medium	Low
Laser beam	High	Local	High	High
Heating	Low	Global	X	Low
Row Hammer	Low	Local	Medium	Medium

Fault models can be categorized by spatial resolution into three categories: bit model, multi-bit model, and random fault model. The bit model [3] is that the adversary induces the fault by flipping one particular bit to recover the secret. The multi-bit model [16] is that the attack induces the fault by flipping one or multiple bits, owing to the attacker can only roughly control the impacted bit location. The random fault model [41] is that the attacker induces the fault but cannot control the value change by the fault injection, and is the most common one in practice.

We can also categorize the fault models by temporal resolution into precise and rough. If injecting the fault needs to occur during a particular operation, then we call it "precise." However, if the time of injecting the fault will not affect the fault injection attack results, then we call it "rough." Table 1 shows the spatial and temporal resolution of the known attacks on cryptosystems.

We have listed various physical attacks in Table 2. The different physical attacks will have different accuracy. For instance, the Clock glitch is the fault inject method that inserts the short glitch into the clock; therefore, it will have high temporal accuracy, and the fault will impact the whole device so that the spatial accuracy will be global. Heating is the method that heats the device and makes the device malfunction. The attacker can not control the time that fault occurred; therefore, the temporal accuracy is too low to be ignored, and it can impact the whole device so that the spatial accuracy will be global as well.

2.2.3 Repeatability. Repeatability is the property that describes whether the fault model can inject the identical fault in each process of fault attack. For instance, in [41], the attacker using the physical attack injects the permanent fault resulting from the damage or alters the part of the device, then each fault attack process will inject the identical fault. In [26], the attacker uses the row hammer to flip a particular bit in the memory with a careful profiling phase. This kind of physical attack method is also repeatable due to the ability to inject identical faults. Meanwhile, many fault models are not repeatable and induce random or multi-bit faults. For instance, in [16], the author injected the multi-bit fault in the intermediate ciphertext in AES. This kind of fault will not be identical in each attack.

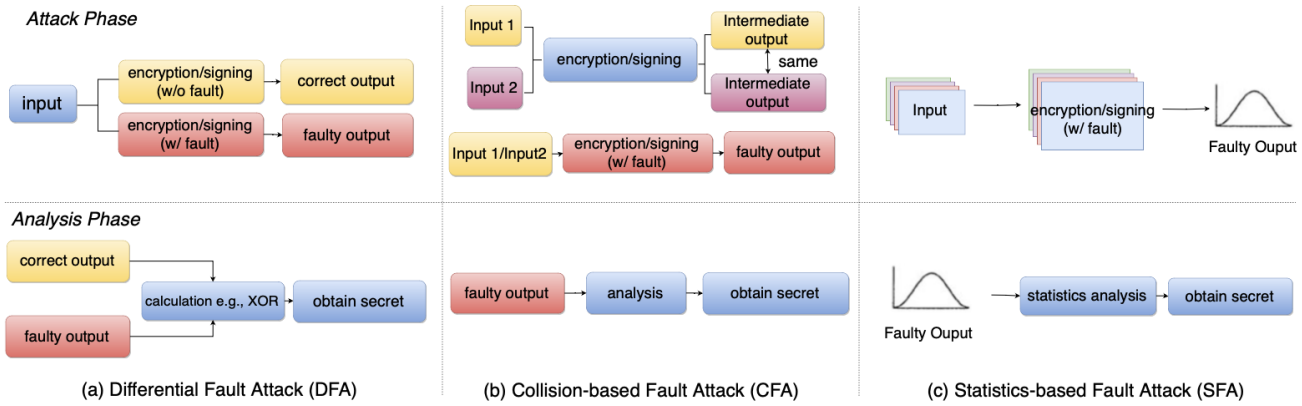

Figure 1: Overview of Fault analysis method

2.2.4 Feasibility. There are various attack methods. e.g., EM pulse, voltage spike, laser beam. The requirements of each method are also different. The feasibility is based on those requirements, such as cost, technical requirements, etc. For example, the Laser beam is costly but has a high accuracy of time, and the fault injection location and the clock glitch have low cost and low accuracy of position but still have high accuracy of time. Table 2 lists several physical attack methods that are commonly used.

2.3 Fault Attack Analysis Methods

The second component of a fault injection attack is fault analysis. We can categorize the fault analysis methods into three main types, *Differential Fault Attack* (DFA), *Collision Fault Attack* (CFA), and *Statistics Fault Attack* (SFA), as shown in Figure 1.

DFA is the most commonly used fault attack method, which injects a fault and compares the difference between the correct and the faulty outputs to extract useful information for recovering confidential data. Figure 1 (a) shows the overview of Differential Fault Attack. We will show more examples of DFA in Sections 3 and 4.

CFA is the fault analysis in which the attacker will first find the different inputs that generate the identical intermediate output, called *Collision*, then use these inputs to process the encryption or signing operation by inducing the fault and obtaining the faulty output. After that, the attacker can analyze the faulty output to obtain the secret. Unlike DFA, the attacker who used CFA will induce the fault in the beginning of the cipher operations. Figure 1 (b) shows the overview of the Collision Fault Attack. In [5], *Johannes Blömer et al.* provide the collision-based fault attack on AES, which induces the fault during the cryptographic operation, and analyzing the two inputs that will produce identical intermediate output to recover the secrets. Notice that collision-based fault attacks require the attacker to obtain the intermediate output.

SFA is the method that relies on inducing faults to cause computational errors, gathering statistics about the errors, and mathematically analyzing the statistics to extract secrets. Unlike CFA, which uses carefully chosen input, SFA only needs to randomly induce the fault and, using statistics methods like Bayesian inference, clustering, etc., to analyze a large number of faulty outputs and obtain the secret. Figure 1 (c) shows the overview of Statistics Fault Attack. In [15], *Thomas Fuhr et al.* provide the statistic-based fault attack on AES. Assume the attacker is able to inject fault on the specific byte during the *AddRoundKey* operation in round 9 or round 8. Then, analyze the distribution of faulty outputs to recover the AES key. With SFA, the attacker can obtain the AES key with only faulty outputs.

3 FAULT ATTACKS ON SYMMETRIC KEY CRYPTOSYSTEM

Symmetric Key Cryptosystem is a cryptographic method that uses the same key for encryption and decryption. The characteristic of the symmetric key cryptosystem is that the operation is faster and more efficient (less computation power) than a public key system. Due to its high efficiency, *Symmetric Key Cryptosystem* is used for file and data storage encryption or network layer data authentication, such as TLS and SSL, which are used to encrypt network communication. Advanced Encryption Standard (AES) [10] became the dominant symmetric key cryptographic algorithm that replaced DES [37] and 3DES [25] and is used for securing various protocols such as VPNs, HTTPs, and TLS. Therefore, network communication will be severely impacted if the AES is vulnerable. For AES, the attacker targets to derive the round key, whose lengths can be 128, 192, or 256 bits.

A number of research papers explore the methods of fault attack on symmetric keys, with the majority focusing on the DFA on AES. Figure 2 provided the overview of DFA on AES that is mentioned in [3, 11, 16, 28]. For instance, in [16], *Christophe Giraud* introduces two types of DFA on AES. One is the Bit-Fault Attack, which uses the faults that occur on only one bit of the temporary cipher result before the final round (M^9) in AES to obtain the last round key (Figure 3). This attack operates independently on each byte, so if we induce a fault on only one bit on several bytes of M^9, it can

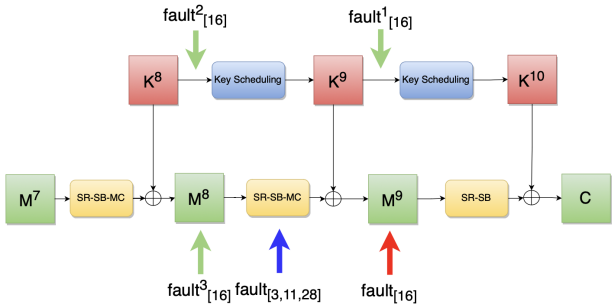

Figure 2: Overview of DFA on AES in [3, 11, 16, 28]. In the fault injection positions illustrated, the red arrow represents the location utilized for the first type of DFA described in [16], while the green arrow denotes the position used for the second form of DFA from the same source. The blue arrow represents the fault injection location employed in the DFA in [3, 11, 28]

Table 3: Fault attack on AES

	Fault	
	Time	Position
[16]	Round 9	Any byte of M^9
	Round 9 after *AddRoundKey*	one of last 4 bytes of K^9
	Round 8 after *AddRoundKey*	one of last 4 bytes of K^8
	Round 8 after *AddRoundKey*	K^8_1,K^8_2,K^8_6,K^8_7,K^8_8,K^8_{11},K^8_{12},or K^8_{13}
[3]	Round 9 before *MixColumn*	Any byte of M^8
[28]	Round 9 before *MixColumn*	Any byte of M^8
[11]	Round 9 before *MixColumn*	Any byte of M^8

reduce the number of faulty ciphertexts required to obtain the key. However, it may be difficult to induce a fault on only one bit during a precise event.

Another attack in [16] uses a fault model based on inducing a fault, which may change a whole byte of a temporary result. This method only works on AES-128, and operates in 3 steps: (1) A fault is introduced on K^9 to obtain the last 4 bytes of K^9, as in Figure 4; (2) A fault is introduced on K^8 to obtain another 4 bytes of K^9, as in Figure 5; (3) A fault is introduced on M^8 to obtain the AES key, K, as in Figure 6. Through the process of this attack, the adversary can recover the secret key and compromise the message.

In [28] and [3], the authors provide the fault attack on SPN structures, e.g., AES, which concept similar to the step 3 of the second DFA method in [16]. Both attack methods could not directly recover the secret by calculating the difference between correct and faulty output. Instead, the attacker needs to test the possible values individually to obtain the final correct secret. However, unlike [16], there is no restriction on the fault location in [28]. The attacker is assumed to be able to control the fault occurring on one byte but does not know the exact fault position. Therefore, the possible value set in [16] is 255 ($e \in 0, \ldots, 255$), and the possible value set in [28] is 255n ($e \in 0, \ldots, 255$,n is the byte-length of ciphertext). Figure 6 is the overview of the fault attack method, and Alg.1 shows the fault analysis in [28].

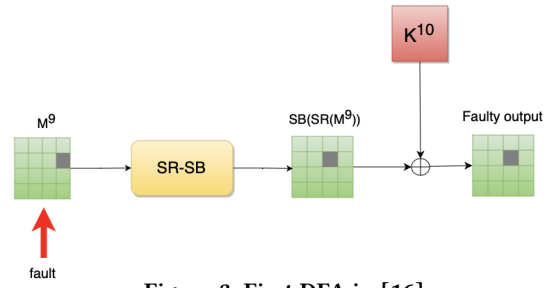

Figure 3: First DFA in [16]

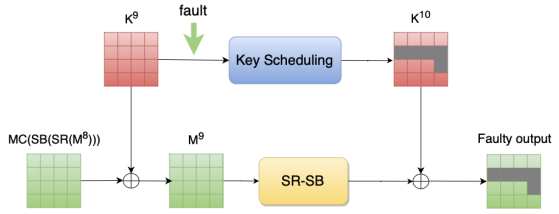

Figure 4: step 1 of second DFA in [16]

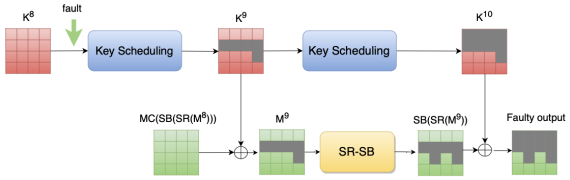

Figure 5: step 2 of second DFA in [16]

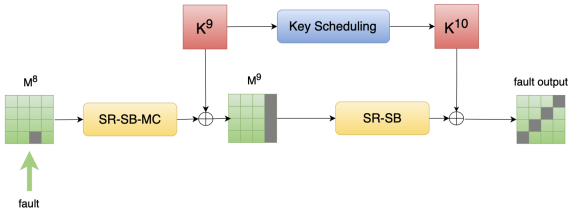

Figure 6: step 3 of second DFA in [16]

Algorithm 1 Fault analysis on AES [28]

Input: $\hat{C}$ the possible differences at the intermediate output of M_9
Output: L the correct byte of AES

1: generate correct output C, and faulty output $\hat{C}$ with input P
2: guess round key K
3: $V = (SB^{-1}SF^{-1}(C \oplus K)) \oplus (SB^{-1}SF^{-1}(\hat{C} \oplus K))$
4: **if** $V \in \hat{C}$ **then**
5: add V to L
6: **end if**
7: choose new P, and back to Line 1, but the round key guesses only go through the list L, and if $V \notin \hat{C}$, then remove V from L, repeat until there is only one value in L
8: **return** L

[11] also provides DFA on AES. However, the authors provided another position to insert the fault to derive the key. In the previous paper, [16], most of the fault position is not affected by *MixColumns* (except the third step of the second type of DFA). On the other hand,

[11] provided the attack that injects the fault after *ShiftRow* before *MixColumns* (blue arrow in Figure 2). Hence, the fault value will be impacted by *MixColumns*, which calculates the recovery key more complicated. Note that even though the fault in the second type of DFA mentioned in [16] is also impacted by *MixColumns*, the calculation is more straightforward due to its previous step, which already derives part of the key. The attack injects the fault to temporary ciphertext after *ShiftRow* in Round 9. The authors also use $C \oplus D$ to derive the bytes impacted by fault value. As analysis and calculation with the bytes differ between correct and faulty ciphertext, we can get four sets of possible values of the bytes impacted by fault. Then, we can get the single value and recover the key by intersecting the sets obtained by iterating the analysis and calculation. [11] demonstrates that not only can inducing the fault to the Round key derive the secret key, but inducing the fault in the temporary ciphertext can also recover the secret key, even if the fault has been impacted by *MixColumns*.

4 FAULT ATTACK ON PUBLIC KEY CRYPTOSYSTEM

Public Key Cryptosystem (PKCs) is a cryptographic system with key pairs, public key and private key. The public key is published, and the private key is kept secret. The PKCs are usually used for key exchange and authenticating the messages by signing operation and verification, presented in many scenarios such as web certification or data signature. Therefore, if the PKCs are attacked, the data integrity will be broken. The most commonly used PKCs algorithm is RSA, ECDSA, and ECC.

4.1 Fault Attacks on RSA and Mitigation

The basic RSA [30] is a cryptographic algorithm whose security depends on the difficulty of factoring the product of large prime numbers. If the attacker can obtain one of the prime numbers, he/she can break RSA. *Chinese Remainder Theorem* (CRT) is the theorem that could be used to speed up the calculation of RSA. CRT can result in approximately a fourfold increase in the algorithm's computational efficiency and is widely used in RSA implementations. However, this theorem could make RSA vulnerable to fault inaction attacks. Here, we discuss the attacks on RSA-based signatures, in which the victim uses a secret key to sign a message. For the case of RSA-CRT, there are two moduli $s_p (= m^d \bmod p)$, $s_q (= m^d \bmod q)$ and the final signature $s = CRT(s_p, s_q)$. Here, p and q are two large prime numbers, m is the message that we want to sign, and d is the secret decryption key. The attacker can obtain the prime numbers from the two moduli.

DFA on RSA Suppose the attacker induced the fault during the modulo reduction operation ($s_p = s'_p \bmod p$) and obtained the faulty result $\hat{s_p}$. The faulty $\hat{s_p}$ and the correct s_q will result in faulty $\hat{s}$, and the attacker can factorize the n (which is secret to the attacker) with Equation (1) [41]. If the attacker can only get a signature and its correlative message, then with a faulty and a correct signature, the attacker can obtain q using Equation (2) [41].

$$q = gcd((\hat{s} - s) \bmod n, n) \tag{1}$$

$$q = gcd((\hat{s}^e - m) \bmod n, n) \tag{2}$$

Algorithm 2 Shamir's Implementation [35]

Input: $m\ p, q, d, q^{-1} \bmod p, m$
Output: s the correct signature
1: select random prime r
2: $p' = p \cdot r, q' = q \cdot r$ ▷ memory fault position
3: $d'_p = d \bmod (p-1) \cdot (r-1), d'_q = d \bmod (q-1) \cdot (r-1)$
4: $s'_p = (m \bmod p')^{d'_p} \bmod p', s'_q = (m \bmod q')^{d'_q} \bmod q'$
5: $s_p = s'_p \bmod p, s_q = s'_q \bmod q$ ▷ memory/operation fault position
6: **if** $s'_p \equiv s'_q \bmod r$ **then**
7: $s = s_q + ((s_p - s_q) \cdot (q_{-1} \bmod p) \bmod p) \cdot q$
8: **else**
9: **return** *check fail*
10: **end if**
11: **return** s

Other than injecting faulty in $\hat{s_p}$, in [26], the attacker injects a single-bit fault into the signing key d, which changes it to $\hat{d} = d + \Delta_d$, then obtains the faulty output $\hat{s} = m^{\hat{d}} \bmod n$. With $\hat{s}$, the attacker can recover the singing key by trying the value $\Delta_d = \pm 2^i, i \in 0, \ldots, n$, until $(\hat{s} m^{-\Delta_d})^e = m \bmod n$, where e is the public exponent.

Existing Mitigations There are two implementations that are developed to disable the RSA-CRT against CRT-based hardware fault attack; one is Shamir's implementation [35], and the other is Infineon's implementation [1].

Shamir's implementation assumes that the input of algorithm are $m\ p, q, d$, and a precalculated value of $q^{-1} \bmod p$. Each time a signature is generated, the algorithm will first generate a random prime r and then compute p', q', d'_p, and d'_q (Line 2-3). Subsequent calculations produce the intermediate value s'_p,s'_q, and the partial signature s_p and s_q (Line 4-5). The algorithm checks whether $s'_p \equiv s'_q \bmod r$. When the fault occurs, s'_p or s'_q be changed, the checking will fail. If the above checking is correct, then the sequential value s_p and s_q will be regarded as error-free and employ Garner's algorithm [24, p.612-613] to derive the signature s (Line 7).

However, in [1], *C. Aumüller et al.* provided that if the fault occurred in p, which makes the attacker obtain faulty p' and d'_p, the fault would not be detected, and the check operation would still be true in Shamir's implementation [35]. Therefore, they provided another implementation. Infineon's implementation fixes the flaw of Shamir's one by not storing the d directly. In Infineon's implementation, it is assumed that the input are $m, p, q, d_p (= d \bmod (p-1))$, $d_q (= d \bmod (q-1))$, and $q^{-1} \bmod p$. Similar to Shamir's implementation, each time a signature is generated, the algorithm will first generate a random prime r and compute p',q', d'_p, and d'_q, which $random_1$ is a random integer (Line 1-13). s'_p, s'_q and partial signature s_p, s_q are sequentially computed (Line 14-15). The algorithm will check whether $p' \bmod p = 0$ and $d'_p \bmod (p-1) = d_p$ to circumvent the flaw in Shamir's method (Line 5) respect to q. Moreover, it will also check whether $s \bmod p = s_p$ and $s \bmod q = s_q$ (Line 16) and $s_{pr}^{d_{pr}} \equiv s_{qr}^{d_{qr}}$ (Line 21).

In [41], *Sung-Ming Yen et al.* provide two kinds of attack methods that can against the two above countermeasures. One is to induce a fault into an important modulo-reduction operation. Another is

Algorithm 3 Infineon's Implementation [1]

Input: m $p, q, d_p, d_q, q^{-1} \mod p$
Output: s the correct signature
1: select random prime r
2: $p' = p \cdot r$
3: $d'_p = d_p + random_1 \cdot (p-1)$
4: $s'_p = m^{d'_p} \mod p'$
5: **if** $\neg(p' \mod p = 0$ and $d'_p \mod (p-1) = d_p)$ **then**
6: **return** *check fail*
7: **end if**
8: $q' = q \cdot r$
9: $d'_q = d_q + random_1 \cdot (q-1)$
10: $s'_q = m^{d'_q} \mod q'$
11: **if** $\neg(q' \mod q = 0$ and $d'_q \mod (q-1) = d_q)$ **then**
12: **return** *check fail*
13: **end if**
14: $s_p = s'_p \mod p, s_q = s'_q \mod q$ ▷ memory /operation fault position
15: $s = s_q + ((s_p - s_q) \cdot (q_{-1} \mod p) \mod p) \cdot q$
16: **if** $\neg(s \mod p = s_p$ and $s \mod q = s_q)$ **then**
17: **return** *check fail*
18: **end if**
19: $s_{pr} = s'_p \mod r, s_{qr} = s'_q \mod r$
20: $d_{pr} = d'_p \mod (r-1), d_{qr} = d'_q \mod (r-1)$
21: **if** $s_{pr}^{d_{pr}} \equiv s_{qr}^{d_{qr}} \mod r$ **then**
22: **return** s
23: **else**
24: **return** *check fail*
25: **end if**

considered the memory access fault on some stored value. Both attack methods are focused on the RSA-CRT algorithm.

Attacking the operations in the processor: with Shamir's countermeasure, when the attacker induces computational faults during the reduced modulo operation ($s_p = s'_p \mod p$), which following the calculation of s'_p and s'_q, the attacker would pass the check operation, and obtain the fault output $\hat{s}$, which making factorize of n becomes possible. As previously stated, this can occur under two circumstances, utilizing equations (1) and (2). The checking schemes, which check $s'_p \equiv s'_q \mod r$, are ineffective in detecting such computational and memory access faults. Consequently, the attack can successfully bypass Shamir's countermeasure. With Infineon's countermeasure, two check operations will be analyzed in the following. One operation checks whether $s \mod p = s_p$ and whether $s \mod q = s_q$ (Line 16). However, this checking only checks if the operation of calculating s is correct; in other words, even if the output is fault s, this check operation will always be true as long as the calculation is correct. The other operation checks whether $s_{pr}^{d_{pr}} \equiv s_{qr}^{d_{qr}}$ (Line 21). This operation depends on s'_p,s'_q, and r, which we assume to be error free. Therefore, the fault occurred in calculation of s_p or s_q would not be detected.

Attacking the storage: The second type of attack mentioned in [41] is attacking the memory, which causes a memory access fault. The attacker can induce the memory access fault in the same position as the operation fault mentioned above. Moreover, the attacker can induce a permanent fault, in which the attacker permanently corrupts or damages the device. The following two cases are considered in [41].

Algorithm 4 NAF-based Repeated Doubling on Elliptic Curve [7]

Input: P on E, k $(1 < k < ord(P))$ in non-adjacent form, n the binary length of k
Output: kP on E
1: $Q_n = O$ ▷ O denotes the point at infinity
2: **for do***i* from $n-1$ down to 0 do
3: $Q'_i = 2Q_{i+1}$
4: **if** $k_i = 1$ **then** ▷ fault inject position
5: $Q_i = Q'_i + P$
6: **else if** $k_i = -1$ **then**
7: $Q_i = Q'_i - P$
8: **else**
9: $Q_i = Q'_i$
10: **end if**
11: **end for**
12: **if** $Q_0 \notin E$ **then**
13: $Q_0 = O$
14: **end if**
15: **return** Q_0

First, if the fault is induced on the storage of p or$q$, then this kind of fault can only be detected by $s \equiv s'_p \pmod p$, due to $s \not\equiv s_p \pmod{\hat{p}}$ and $s \not\equiv s_p \pmod p$(with $\hat{q}$). Another case is that permanent fault is induced on d_p or d_q in Alg.3. The attacker can obtain the faulty $\hat{s_p}$ by inducing a fault on d_p. Therefore, the attacker can factorize n by the faulty signature $\hat{s} = CRT(\hat{s_p}, s_q)$ using Equation 1 or 2, and such permanent fault will not be detected by checks in Alg.3.

4.2 Fault Attacks based on ECC

Elliptic Curve Cryptosystem (ECC) [23] based on the elliptic curves under finite fields. The complexity of ECC is based on the fact that it is hard to recover the point that has performed the scalar multiplication, and it is called the elliptic curve discrete log problem.

The elliptic curve E and points on E can be expressed by a variety of coordinate representations, e.g. affine coordinates and projective coordinates. The *Sign Change Attack* mentioned in [7] is represented by projective Weierstraß equation $y^2z \equiv x^3 + Axz^2 + Bz^3 \mod p$.

The final objective of attack methods that target cryptosystems based on elliptic curves, e.g., the ElGamal cryptosystem, is to recover the secret factor k due to the crucial computation of these cryptosystems is the scalar multiplication of public base point P with k. Moreover, there are some implementations to speed up the operation of scalar multiplication, which is called *Non-Adjacent Form*(NAF) [8, 20], and [7] attack a left-to-right version (Alg. 4).

DFA on repeated doubling algorithm on ECC: *Johannes Blömer et al.* provided fault attack method on ECC, which they called *Sign Change Attack* [7]. The attacker induce the *Sign Change Fault* on E, which changes the sign of the y-coordinate of faulty point e.g., $Q'_i \rightarrow -Q'_i (Q'_i, -Q'_i \in E)$. In this attack, the attacker does not know in which loop iteration the fault was induced, and it is possible that the fault that occurred in each iteration is identical.

According to [7], all parameters in Line 3-4 in Alg. 4 can be attacked with *Sign Change Attack*. First, we talk about the attack on Q'_i in Line 4 during the i^{th} loop iteration $(0 \leq i \leq n-1)$. The faulty value $\hat{Q}$ is obtained by induced sign change fault in Q'_i, which

Algorithm 5 LSBs version sign change attack [7]

Input: n the length of the secret key k in non-adjacent form, Q the correct result, m a parameter for acceptable amount of offline work
Output: k with probability at least 1/2
Phase 1: collect faulty output $\hat{Q}$
1: get c faulty output from Alg.4 ($c = (n/m) \cdot log(2n)$)
Phase 2: recover secret key bits
2: $s = -1$ the number $s+1$ of known bits of k
3: **while** $s < n-1$ **do**
4: $L = 2\sum_{j=0}^{s} k_j 2^j P$
5: **for** $r = 1$ to m **do**
6: **for** $x = (x_{s+1}, x_{s+2}, \ldots, x_{s+r})$ in non-adjacent form **do**
7: $T_x = L + 2\sum_{j=s+1}^{s+r} x_j 2^j P$
8: **for** $\forall \hat{Q} \in S$ **do**
9: **if** $T_x - \hat{Q} = Q$ **then**
10: $k_{s+1} = x_{s+1}, \ldots, k_{s+r} = x_{s+r}$,
11: $s = s + r$, continue at Line 3
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: if no $\hat{Q}$ satisfies $T_x - \hat{Q} = Q$, $k_{s+1} = 0$ and $s = s+1$
17: **end while**
18: *If* $Q \neq kP$ **return** *failure*
19: **return** k

$\hat{Q} = -Q + 2L_i(k)$. ($L_i(k) = \sum_{j=0}^{i} k_j 2^j P$). The only unknown part is $L_i(k)$. Therefore, if only a small number of the signed bits k_0, $k_1, \ldots, k_i$ used in $L_i(k)$ is unknown, these signed bits of k can be recovered starting from the LSBs. Besides, the recovery operation can also start from MSBs due to $Q = L_i(k) + \sum_{j=i+1}^{n-1} k_j 2^j P$. In [7], the authors use the LSBs version and assume Q and $\hat{Q}$ are known.

Alg. 5 shows how LSBs version sign change attack works on Q'_i. First, we need to obtain c (which value has been proofed to recover the k with probability at least 1/2) the faulty output $\hat{Q}$ (Line 1-2). Then, recover the secret key bits by Q and $\hat{Q}$ obtained in Phase 1 (Line 3-19). Note that we assume the most significant bit $k_{n-1} = 1$. Otherwise, n cannot be uniquely defined. Therefore, $s < n-1$ (Line 4). The *Sign Change Attack* on P can also used to recover the secret key and is similar to the Alg.5.

To induce the sign change attack in the non-adjacent-form-based algorithm, which speeds up the scalar multiplication calculation, we can use all the physical attacks that can target conditional decisions, e.g., power spikes or clock glitches. These attacks can force the conditional statement to choose the wrong branch, which means adding $-P$ instead of P or vice versa. However, this kind of attack can only induce the fault in P. To induce the sign change fault to the intermediate variable, such as Q'_i, we need to use the physical attacks that target the ALU, the unit used to implement the non-adjacent-form based calculation.

DFA on ECDSA: There is a digital signature scheme that is based on ECC called ECDSA [19]. Like other digital signature schemes, the main goal of attacking the ECDSA is to recover the secret key d. In [26], *Koksal Muset al.* introduce an attack method known as *Jolt*, specifically designed to target implementations of signature schemes used to deploy the TLS protocols. This attack uses faulty signatures derived by inducing faults when generating signatures

Algorithm 6 ECDSA with faulty d

Input: $M \in 0,1*$ a message, the faulty private key $\hat{d} = d + \Delta_d$ ▷ fault inject position
Output: Faulty signature $(r, \hat{s})$
1: choose random key $k \in Z_n*$
2: $R = kP$, $R_x = r = (kP)_x$
3: $\hat{s} = k^{-1}(H(M) + \hat{d}r) \mod n$ ▷ $H(M)$ hash value of M
4: **return** $(r, \hat{s})$

Algorithm 7 ECDSA faulty signature verification

Input: $M \in 0,1*$ a message, $(r, \hat{s})$ faulty signature, $Q \in E$ public key
Output: *Reject*
1: $\hat{w} = \hat{s}^{-1} \mod n$
2: $\hat{u}_1 = H(M)\hat{w} \mod n$
3: $\hat{u}_2 = r\hat{w} \mod n$
4: $\hat{R} = \hat{u}_1 P + \hat{u}_2 Q$
5: $\hat{r} = \hat{R}_x$
6: **return** *Reject*

and can recover the 256-bit ECDSA secret signing key with less than a thousand faulty signatures. Moreover, with minor modifications, the Jolt attack is compatible with other signature schemes, such as RSA signatures.

Signature Correction Attack (SCA) is a framework designed to correct faulty signatures generated via fault injection to recover the secrets. The Jolt attack is a SCA that needs to specify where to inject the faults, the method of correcting the faulty signatures, and the method of recovering the remaining bits on ElGamal type signatures.

Most implementations of ECDSA do not verify the signature, and thus, the attacker can inject faults into the secret key before the signing operation and then obtain the faulty signature to conduct DFA. Specifically, in the Jolt attack [26], there are three phases. The first phase is *Memory Profiling*, in which the adversary allocates the memory pages and runs the hammering experiments to identify vulnerable memory locations. The second phase is *Online Fault Injection*, in which the attacker injects faults during the ECDSA operations. Two locations can inject the fault mentioned in [26]. One is singing key $\hat{d} = d + \Delta_d$,which need to injected before generating the signature, and the other is nonce $\hat{k} = k + \Delta_k$, which needs to be injected after.

The last phase is *Offline Post-processing*, in which the SCA is used to recover the secret key bits. If faulty signatures are insufficient for the adversary to obtain the entire key, then use an exhaustive search. In the *Offline Post-processing Phase*, there are two cases to recover the secret value of ECDSA. One case is the fault induced on d after key generation. The attacker obtains the faulty signature by processing Alg. 6. The attacker can use $\hat{R} + \Delta_d \hat{u}_2 P = R$ (which derive by the Alg. 7) to find the Δ_d, then recover the d by Δ_d. Another case is inducing the fault on nonce value k after kP is computed. Since fault induced after r is generated, according to the Alg.6, the attacker will obtain the faulty signature $(r, \hat{s})$, with correct r and faulty s, and can find the fault $\Delta_k(\hat{k} = k + \Delta_k)$ by using the equation $\hat{R} = R + \Delta_k P$.

Table 4: Fault attack with cryptographic algorithm

Cryptosystems	Precision		Target	Method	
	Temporal	Spacial			
AES	precise	multi-bit	SRAM	Optical	[16]
	precise	bit	CPU core	Low voltage	[3]
	precise	multi-bit	x	x	[11]
RSA	precise	random	EEPROM	x	[41]
	rough	bit	DRAM	Rowhammer	[26]
DSA	rough	multi-bit	CPU register	Glitch	[27]
ECC	precise	sign change	execution unit	power spike/clock glitch	[7]
ECDSA	rough	multi-bit	DRAM	Rowhammer	[29]
	rough	multi-bit	DRAM	Rowhammer	[26]
	precise	skip the instruction	execution unit	spike attack	[33]

5 DISCUSSION

Known fault injection attacks use different fault models and fault analysis methods. In a previous survey paper [2], the author provides the different fault attack (DFA) methods on the symmetric key cryptosystems. Unlike [2], in this paper, we provide not only the attack on symmetric key cryptosystems but also the attack on public key cryptosystems. Moreover, we connect the cryptographic algorithm part with the physical fault model, which makes the reader understand the potential risk of the device and algorithm.

Table 4 lists cryptographic algorithms and the corresponding types of faults that we need to prevent. For instance, based on the precision aspect, we can notice that most fault attack methods require precise temporal resolution; therefore, the countermeasure that makes the time that the fault occurred hard to control can increase the complexity of the fault attack.

6 MITIGATION

At the algorithm level, all the fault analysis methods are based on faulty outputs. Therefore, if we check the output before showing the result, then it can reduce the possibility of a fault attack. Notice that this method can not prevent all kinds of fault attacks, such as *Safe Error Analysis*, which are used when the fault does not impact the output.

The direct way to prevent the fault attack is to reduce the possibility of the fault occurring at the hardware level. For instance, we can use a surge protector to reduce the impact of EM pulse or employ beam stop to prevent the Laser attack. Notice that this kind of mitigation requires knowledge of each type of physical attack, and no protection can prevent all kinds of physical attacks.

7 CONCLUSION

In this paper, we provide the concept of fault attack with two components: *Fault Model* and *Fault Analysis*. The fault Model part provides how the attacker will inject the fault and what kind of fault the attacker will induce. The fault Analysis part describes the analysis methods in different scenarios and the key point of it. It enables us to clearly understand how the existing fault attack method works and provides the key points of various fault attacks that can be focused on to mitigate the attack. Moreover, the most important part of this paper is we add the algorithm aspect, which provides a new view of the fault attack method.

REFERENCES

[1] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and J-P Seifert. 2003. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*. Springer, 260–275.

[2] Anubhab Baksi, Shivam Bhasin, Jakub Breier, Dirmanto Jap, and Dhiman Saha. 2022. A survey on fault attacks on symmetric key cryptosystems. *Comput. Surveys* 55, 4 (2022), 1–34.

[3] Alessandro Barenghi, Guido M Bertoni, Luca Breveglieri, Mauro Pellicioli, and Gerardo Pelosi. 2010. Fault attack on AES with single-bit induced faults. In *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 167–172.

[4] Eli Biham and Adi Shamir. 1997. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*. Springer, 513–525.

[5] Johannes Blömer and Volker Krummel. 2006. Fault based collision attacks on AES. In *International Workshop on Fault Diagnosis and Tolerance in Cryptography*. Springer, 106–120.

[6] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. 2003. A new CRT-RSA algorithm secure against bellcore attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*. 311–320.

[7] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. 2006. Sign change fault attacks on elliptic curve cryptosystems. In *Fault Diagnosis and Tolerance in Cryptography: Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006. Proceedings*. Springer, 36–52.

[8] Andrew D Booth. 1951. A signed binary multiplication technique. *The Quarterly Journal of Mechanics and Applied Mathematics* 4, 2 (1951), 236–240.

[9] Claudio Bozzato, Riccardo Focardi, and Francesco Palmarini. 2019. Shaping the glitch: optimizing voltage fault injection attacks. *IACR transactions on cryptographic hardware and embedded systems* (2019), 199–224.

[10] Joan Daemen and Vincent Rijmen. 2002. *The design of Rijndael*. Vol. 2. Springer.

[11] Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. 2003. Differential Fault Analysis on A.E.S. In *Applied Cryptography and Network Security*, Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Jianying Zhou, Moti Yung, and Yongfei Han (Eds.). Vol. 2846. Springer Berlin Heidelberg, Berlin, Heidelberg, 293–306.

[12] Jean-Max Dutertre, Jacques J.A. Fournier, Amir-Pasha Mirbaha, David Naccache, Jean-Baptiste Rigaud, Bruno Robisson, and Assia Tria. 2011. Review of fault injection mechanisms and consequences on countermeasures design. In *2011 6th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*. 1–6. https://doi.org/10.1109/DTIS.2011.5941421

[13] Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. 2011. An on-chip glitchy-clock generator for testing fault injection attacks. *Journal of Cryptographic Engineering* 1 (2011), 265–270.

[14] Mohammad Eslami, Behnam Ghavami, Mohsen Raji, and Ali Mahani. 2020. A survey on fault injection methods of digital integrated circuits. *Integration* 71 (2020), 154–163.

[15] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. 2013. Fault attacks on AES with faulty ciphertexts only. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 108–118.

[16] Christophe Giraud. 2005. Dfa on aes. In *Advanced Encryption Standard–AES: 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers 4*. Springer, 27–41.

[17] Christophe Giraud and Erik W Knudsen. 2004. Fault attacks on signature schemes. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings 9*. Springer, 478–491.

[18] Michael Hutter and Jörn-Marc Schmidt. 2014. The temperature side channel and heating fault attacks. In *Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers 12*. Springer, 219–235.

[19] Don Johnson, Alfred Menezes, and Scott Vanstone. 2001. The elliptic curve digital signature algorithm (ECDSA). *International journal of information security* 1 (2001), 36–63.

[20] Marc Joye and Sung-Ming Yen. 2000. Optimal left-to-right binary signed-digit recoding. *IEEE Trans. Comput.* 49, 7 (2000), 740–748.

[21] Johan Karlsson, Peter Liden, Peter Dahlgren, Rolf Johansson, and Ulf Gunneflo. 1994. Using heavy-ion radiation to validate fault-handling mechanisms. *IEEE micro* 14, 1 (1994), 8–23.

[22] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors. In *Proceeding of the 41st annual international symposium on Computer architecuture*. 361–372.

[23] Neal Koblitz. 1987. Elliptic curve cryptosystems. *Mathematics of computation* 48, 177 (1987), 203–209.

[24] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. 2018. *Handbook of applied cryptography*. CRC press.

[25] Ralph C Merkle and Martin E Hellman. 1981. On the security of multiple encryption. *Commun. ACM* 24, 7 (1981), 465–467.

[26] Koksal Mus, Yarkın Doröz, M Caner Tol, Kristi Rahman, and Berk Sunar. 2022. Jolt: Recovering TLS Signing Keys via Rowhammer Faults. *Cryptology ePrint Archive* (2022).

[27] David Naccache, Phong Q Nguyen, Michael Tunstall, and Claire Whelan. 2005. Experimenting with Faults, Lattices and the DSA. In *Public Key Cryptography-PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005. Proceedings 8*. Springer, 16–28.

[28] Gilles Piret and Jean-Jacques Quisquater. 2003. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In *Cryptographic Hardware and Embedded Systems-CHES 2003: 5th International Workshop, Cologne, Germany, September 8–10, 2003. Proceedings 5*. Springer, 77–88.

[29] Damian Poddebniak, Juraj Somorovsky, Sebastian Schinzel, Manfred Lochter, and Paul Rösler. 2018. Attacking deterministic signature schemes using fault attacks. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 338–352.

[30] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.

[31] Nadir K Salih, D Satyanarayana, Abdullah Said Alkalbani, and R Gopal. 2022. A survey on software/hardware fault injection tools and techniques. In *2022 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*. IEEE, 1–7.

[32] Jörn-Marc Schmidt and Michael Hutter. 2007. *Optical and em fault-attacks on crt-based rsa: Concrete results*. na.

[33] Jörn-Marc Schmidt and Marcel Medwed. 2009. A fault attack on ECDSA. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 93–99.

[34] Bodo Selmke, Johann Heyszl, and Georg Sigl. 2016. Attack on a DFA protected AES by simultaneous laser fault injections. In *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 36–46.

[35] A Shamir. 1997. Method and apparatus for protecting public key schemes from timing and fault attacks. In *EUROCRYPT'97*.

[36] Sergei P Skorobogatov and Ross J Anderson. 2003. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*. Springer, 2–12.

[37] Miles E Smid and Dennis K Branstad. 1988. Data encryption standard: past and future. *Proc. IEEE* 76, 5 (1988), 550–559.

[38] Yan Bo Ti. 2017. Fault attack on supersingular isogeny cryptosystems. In *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*. Springer, 107–122.

[39] Jasper GJ Van Woudenberg, Marc F Witteman, and Federico Menarini. 2011. Practical optical fault injection on secure microcontrollers. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 91–99.

[40] Tomá Vanát, Jan Pospíil, Filip Kríek, Jozef Ferencei, and Hana Kubátová. 2015. A System for Radiation Testing and Physical Fault Injection into the FPGAs and Other Electronics. In *2015 Euromicro Conference on Digital System Design*. 205–210. https://doi.org/10.1109/DSD.2015.98

[41] Sung-Ming Yen, Sangjae Moon, and Jae-Cheol Ha. 2003. Hardware fault attack on RSA with CRT revisited. In *Information Security and Cryptology—ICISC 2002: 5th International Conference Seoul, Korea, November 28–29, 2002 Revised Papers 5*. Springer, 374–388.

[42] Haissam Ziade, Rafic A Ayoubi, Raoul Velazco, et al. 2004. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.* 1, 2 (2004), 171–186.