applied sciences

Review

Hardware Countermeasures Benchmarking against Fault Attacks

Francisco Eugenio Potestad-Ordóñez [1,2,*], Erica Tena-Sánchez [1,2], Antonio José Acosta-Jiménez [1,3], Carlos Jesús Jiménez-Fernández [1,2] and Ricardo Chaves [4]

1 Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC; Universidad de Sevilla), 41092 Sevilla, Spain; erica@imse-cnm.csic.es (E.T.-S.); acojim@imse-cnm.csic.es (A.J.A.-J.); cjesus@imse-cnm.csic.es (C.J.J.-F.)
2 Departamento de Tecnología Electrónica, Escuela Politécnica Superior, Universidad de Sevilla, Virgen de Africa 7, 41011 Sevilla, Spain
3 Departamento de Electrónica y Electromagnetismo, Facultad de Física, Universidad de Sevilla, 41012 Sevilla, Spain
4 INESC-ID, IST, Universidade de Lisboa, 1049-001 Lisboa, Portugal; ricardo.chaves@tecnico.ulisboa.pt
* Correspondence: potestad@imse-cnm.csic.es

Abstract: The development of differential fault analysis (DFA) techniques and mechanisms to inject faults into cryptographic circuits brings with it the need to use protection mechanisms that guarantee the expected level of security. The AES cipher, as a standard, has been the target of numerous DFA techniques, where its security has been compromised through different formulations and types of fault injections. These attacks have shown vulnerabilities of different AES implementations and building blocks. Consequently, several solutions have been proposed that provide additional protection to cover the identified vulnerabilities. In this paper, an extensive analysis has been carried out covering the existing fault injection techniques, the types of faults, and the requirements needed to apply DFA. Additionally, an analysis of the countermeasures reported in the literature is also presented, considering the protection provided, the type of faults considered, and the coverage against fault attacks. The eight different types of fault that allow us to perform DFAs on the AES cipher have been differentiated, as well as the vulnerabilities of the cipher. On the other hand, two comparisons have been made considering frequency penalty vs. area and fault coverage vs. area and frequency overhead. A metric has been proposed to compare the fault coverage of all the proposed solutions. To conclude, a final analysis is presented discussing the key aspects when choosing a particular solution and the possible development of new countermeasures to provide further protection against DFA.

Keywords: AES; hardware countermeasure; benchmarking; trade-off; fault attack; DFA

Citation: Potestad-Ordóñez, F.E.; Tena-Sánchez, E.; Acosta-Jiménez, A.J.; Jiménez-Fernández, C.J.; Chaves, R. Hardware Countermeasures Benchmarking against Fault Attacks. *Appl. Sci.* **2022**, *12*, 2443. https://doi.org/10.3390/app12052443

Academic Editors: Guy Gogniat, Vianney Lapotre and Maria Mushtaq

Received: 5 January 2022
Accepted: 24 February 2022
Published: 26 February 2022

1. Introduction

Nowadays, determining and minimizing vulnerabilities in cryptographic devices has become a scientific challenge. This is mainly due to the constant development and improvement of analysis techniques and attacks that are able to exploit the vulnerabilities of existing implementations in order to access the device secret information. In addition to this, there has been a large increase in the number of applications that require the use of devices with significant resource and power constraints. An example of this is the increasing usage of the Internet of Things (IoT). In [1], the authors describe the importance of analysing security in the IoT, considering recent research work on the different stages of the IoT security solution. Finally, ref. [2] discusses the risks that exist if attacks on embedded applications used in this area are not considered.

Although different techniques exist to compromise the security of the cryptographic devices, herein we will focus on the so-called active attacks. Active attacks have been extensively studied since the paper presented by Bonet et al. [3], where the authors presented a fault injection attack on the Rivest–Shamir–Adleman (RSA) cryptosystem. These types of attack can compromise the security of cryptographic devices by comparing the relationship between the correct and the faulty output of the circuit when a fault is injected during its operations. On the one hand, fault injection is carried out by means of Fault Injection Attack (FIA) techniques such as non-invasive (the device is not modified such as glitch induction in the clock signal) or invasive (the device is manipulated such as stripping of the circuit using a laser beam). On the other hand, the analysis between the correct and faulty output of the circuit is performed by means of Differential Fault Analysis (DFA). This analysis allows attackers to obtain information related to the secret key by comparing the correct and faulty ciphertexts of the circuit.

This technique has been used successfully in different cryptographic algorithms. Among the block ciphers presented over the years, the Advanced Encryption Standard (AES) [4] cipher has received the most attention because it is the standard chosen by the National Institute of Standards and Technology (NIST), it is widely used, and it is implemented as a standard in a large number of technologies. This cipher has been subjected to numerous differential analyses and attacks with the aim of compromising its security [5–17].

Since fault attacks and differential analysis studies are numerous, so are fault detection schemes. A large number of related works focus on the AES cipher, as in the case of this work. This paper aims to analyze the different fault injection techniques and the types of fault they can produce, to analyze the characteristics of the DFAs presented on the AES cipher, to summarize the most relevant protection schemes, and to analyze them considering fault coverage, resource consumption, and throughput. To this end, the different FIA techniques are analyzed to meet the requirements of the DFAs, considering the fault level, requirements for a successful attack, and the fault location control. A complete and detailed classification of the different DFA methods on the AES cipher has been carried out, defining weaknesses, requirements, and attacks times. To compare all countermeasures as a whole, it is necessary to consider that the attacker is capable of achieving any type of fault, being the most powerful attacker. This makes it possible to determine whether the countermeasures studied offer an appropriate level of protection depending on the types of fault that may occur in relation to those that they are able to detect. This analysis is a guide for choosing among the many proposals, depending on the level of protection required for a given application and environment.

Note that in this paper, we focus on analysing attacks on physical implementations of private key encryption algorithms and on proposed hardware solutions to detect such attacks and prevent secret encrypted and exchanged information from being compromised. To this end, attackers must have access to the devices that implement the encryption algorithms in hardware.

There are some reviews about differential analysis and countermeasures in hardware implementations such as [18–20] or software such as [21]. In the case of [18] an analysis of security issues and possible countermeasures in IoT applications is carried out. In this work, the main weaknesses in IoT environments are analysed, and the application of encryption algorithms such as AES, Data Encryption Standard (DES), or Rivest–Shamir–Adleman (RSA) are proposed as countermeasures. However, this work does not consider the vulnerabilities that are present in this type of algorithms, such as the AES cipher, the possible attacks or solutions that allow them to be detected. In [19], an extensive analysis of the different types of DFA applied to the AES cipher is presented and some of the solutions proposed to detect flaws are presented. In this work, some of the DFAs reported in the literature on the AES cipher are considered, considering single bit, single byte and multiple byte fault types. However, the analysis of attack types by fault insertion is not considered, as well as the comparison between fault coverage and occupied area of the physical implementations. The comparison between the most relevant solutions against attacks is not presented, considering only a few solutions. A comparison of cost versus security of countermeasures against fault attacks is presented in [20]. In this work, different solutions based on the use of redundant spaces for fault detection are analysed.

However, the comparison of fault types, attack techniques, DFAs and fault coverage analysis versus delay and area is not carried out. Finally, an analysis of countermeasures applied on the AES cipher from the software point of view is carried out in [21]. Different solutions and their comparisons are presented, but it should be considered that they are not comparable to those studied in this work because they do not deal with attacks on physical implementations or hardware countermeasures.

The rest of the paper is organized as follows. Section 2 presents a study of the different FIA techniques through which it is possible to achieve the requirements needed to be able to apply the DFA on the AES. Section 3 presents in summary form a description of the AES cipher, a classification of the types of faults useful for DFAs is presented, and the vulnerabilities of the AES cipher against DFAs are described in detail. Section 4 presents the different countermeasures considered, performing a classification of all of them, and making a comparison between implementation costs and fault coverage. Section 5 presents a comparison between the different protection schemes and the current state of the art according to the security level. Section 6 presents a discussion from the point of view of security and resource trade-offs. Finally, Section 7 presents the conclusions.

2. Fault Injection Attack Techniques

In the context of crypto circuit vulnerability analysis, FIAs are the existing techniques on which DFAs are based to achieve the fault injections and faulty outputs necessary to recover the secret key. In order to perform these attacks, sometimes extensive knowledge of the device and the cryptographic system is required. Once the device has been manipulated, the output of the device must be different from the correct output. Additionally, the effectiveness of the attack depends on the fault model applied. The fault model is a representation of how the fault affects the behaviour of the device, making a hypothesis of how many bits have been modified and what is the impact of the modification on them. The most important fault injection techniques are briefly described below.

As a general example of attacks on hardware implementations, Figure 1 can be found. In this figure, two main blocks can be distinguished, the physical attacks on the physical implementations and the analysis of the behaviour of the circuits after the attacks. The first one consists of carrying out attacks on ASIC or FPGA implementations and comparing the behaviour with the correct operation performed on hardware or software. On the other hand, there is the analysis of the results obtained after the application of the FIAs, where, depending on the DFA used, one type of fault or another is exploited in order to recover the secret key by means of cryptanalysis.

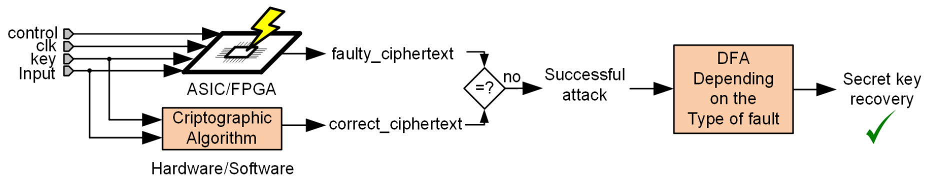

Figure 1. General representation of attack using FIA and DFA.

– Temperature: Since chip manufacturers define the temperature range at which their device can work properly, taking the chip to its limits can intentionally cause faults within it. By setting the temperature of the chip to a value where write options work, but reads do not, or conversely, several attacks can be mounted. Depending on the vulnerability of the components to temperature, different types of fault can be achieved. Generally, the attacker does not have precise control over the type of faults that will be generated [22,23].
– White light: All electrical circuits are sensitive to light due to photoelectric effects. The current induced by photons can be used to induce faults if a circuit is exposed to intense light for a short period of time. This can be used as an inexpensive method of fault induction [24].

– Laser beam: A laser can reproduce a wide variety of faults. The effect produced is similar to white light, but the advantage of a laser over white light is the directionality that allows it to accurately target a small area of the circuit. On the other hand, they can also cause widespread faults. The works presented in [24,25] are examples of these attacks.
– Electromagnetic Fault Injection: Induction of an electromagnetic field (EM) can cause the malfunction of an encapsulated chip or change the contents of the memory. By inducing Foucault currents on the surface of the chip, they can produce multiple faults, including single bit faults [26].
– Power supply: An increase or decrease in the voltage supply to the chip above the tolerance level of the devices (typically 10%), can cause faults in the combinational operations or in the bits stored in the flip-flops. These faults can affect a part of the circuit or cause widespread faults [27].
– Clock glitches: In synchronous circuits, it is possible to introduce faults by reducing the time between two active edges of the clock signal. One or more flip-flops store a value before it is stable at its input. This technique has the characteristic that, if the time between two active edges is well controlled, then no generalized faults are introduced in the circuit, but only in those flip-flops whose inputs cross the paths with the greatest delay. This technique was used to obtain experimental results in the work presented by Street and Lafayette [28], where these attacks are applied to RSA and DES devices. Other works in which these techniques are used include [5,29–32].
– Combined approaches: These attacks include, as their name suggests, combinations of several of the attacks described above. Physical attacks can be carried out simultaneously by altering the clock signal and the power supply voltage. In this way, it is possible to achieve more advantageous scenarios for the injection of faults. It is also possible to combine two or more attacks using, for example, power consumption and electromagnetic emanations simultaneously [33,34].

The difference between this wide variety of attack types lies in the cost of their implementation and the necessary granularity of the attack, i.e., the range of fault types that can be achieved by the attacker. Attacks that allow the maximum fault insertion capacity (single-bit or set of bit faults desired in any part of the circuit) will be the costliest and, therefore, determine the most powerful attacker. The described attacks and their most important characteristics are summarized in Table 1.

Table 1. Fault Injection Techniques Classification.

Attack Type	References	Fault Level	Requirements	Fault Control
Temperature	[22,23]	byte	Control of the temperature limits	Low
White light	[24]	byte	Strip of the circuit and layout knowledge	Medium
Laser beam	[24,25]	bit	Strip of the circuit and layout knowledge	High
Electromagnetic fault injection	[26]	bit	Knowledge of the circuit	Medium
Power supply	[27]	byte	Fine control of the power supply	Low
Clock Glitches	[5,29–32]	bit	Clock cycle control and good signal generator	Medium
Combined approaches	[33,34]	bit	Combination of the previous ones	High

Each of these methods has its requirements and necessary control from the attacker. For example, by using a laser it is possible to inject single-bit faults, but it is also necessary to perform a circuit stripping and carry out a study to know its routing and component placement (and thus high level of knowledge and equipment is required). On the other hand, an attack performed by means of clock glitches does not require the manipulation of the circuit (requiring almost no knowledge of the circuit layout and very low cost of equipment), but it will not be possible to inject faults with the same precision as with a laser attack. Therefore, in this last example, the attacker will be less powerful when it comes to achieving the theoretical requirements to perform a DFA successfully.

3. Dfa on AES

Differential fault analysis is one of the most widely used techniques in the field of cryptanalysis [3]. This technique consists of assuming the fault injection into a cryptographic circuit, using any of the previously described FIAs, and comparing its faulty output with the one obtained when it works correctly. The comparison between the correct and faulty results obtained during the encryption or decryption process makes it possible to recover the secret information from the device. This analysis, performed on different block functions or cipher rounds, makes it possible to recover the secret key of the attacked cipher. In order to accurately analyze AES DFAs, the AES cipher and the types of faults that can occur during its operation are briefly described below.

3.1. Aes Block Cipher

The AES cipher is the NIST standard, using the Rijndael algorithm [4] with 128 bit input blocks. Depending on the key size (128, 192, or 256 bits) AES performs the input transformation over multiple rounds (10, 12, or 14, respectively).

The round process consists of processing the (16 byte) input state S through the operations SubBytes(), ShiftRows(), MixColums() and AddRoundKey(), as illustrated in Figure 2 for a 128 bit key.

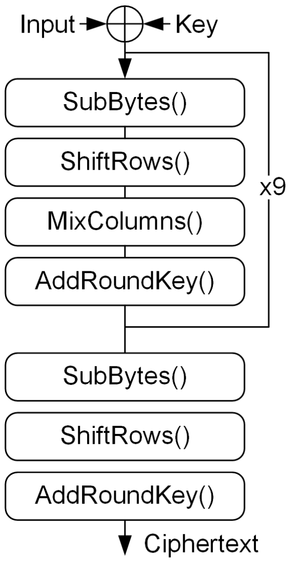

Figure 2. Schematic representation of AES.

In the case of the 128-bit AES cipher, the cipher operates with 128-bit data matrixes, both for the key and for the plaintext. The 128 bits of data in each matrix are divided into 16 blocks of 8 bits. This encryption algorithm can be divided into two blocks, state transformation block and key expansion block. The first block applies different transformations to the state divided into 10 rounds. First, the state is loaded with the plaintext values, and then the AddRoundKey() function is applied to it along with the first key value. Then, each of the intermediate functions of the algorithm is applied to the state, transforming it in each round, obtaining at the end of the rounds the ciphertext.

On the other hand, there is the key expansion. This block is used to generate the intermediate keys used in each round. Each of the keys in each round is obtained from the previous expanded key. Thus, in each round, the state will be mixed with a different key.

AddRoundKey()—Performs the XOR operation between the state matrix and the expanded key of each round. SubBytes()—In this function the transformation of the state matrix with the Sbox8 of the AES byte by byte is applied. ShiftRows()—This function rotates each row of the state matrix. MixColumns()—Multiplies the internal state by a fixed data matrix defined by the algorithm.

3.2. Type of Faults

The types of faults that are considered in the theoretical models of DFA that can be achieved experimentally are briefly described below and listed in Table 2. These faults have been classified according to the requirements of the theoretical models of the DFA. For example, it is necessary to differentiate between faults inserted in multiple bits because they can occur in the same byte in an odd or even number, or on the contrary, they can occur between different bytes in an odd or even number. In the case D, the injection of a single-byte is one that considers the change of a whole byte, not considering the reset of all its values. In case F, the faults considered are those that are randomly produced, i.e., those that can be of any type without any control over them or their repetitiveness.

The type of fault denoted as A can be considered the most difficult to achieve due to the complexity of changing the value to a single bit of the cipher state, for example, between the operations during encryption process, at a given clock cycle and which must be repeated in different attacks for the same cipher input. On the contrary, the fault denoted as F can be considered the simplest to achieve, since, being a random fault, it can occur at any clock cycle and location.

Table 2. Type of Possible Faults Classification.

Type of Fault	Description
A	Single Bit Fault
B.1	Multiple Bits in the same Byte (odd)
B.2	Multiple Bits in the same Byte (even)
C.1	Multiple Bits in different Bytes (odd)
C.2	Multiple Bits in different Bytes (even)
D	Single Byte Fault
E	Multiple Byte Faults
F	Random Faults

Thus, the attacker capable of obtaining faults of any type is considered the most powerful. On the other hand, the attacker who is only able to inject one type of fault or faults of type F will be the one who has less power when it comes to achieving the theoretical scenarios and obtaining information about the circuit, since his control over the generation of faults is minimal.

When considering the fault injection into the AES cipher, depending on where the fault occurs (the state matrix, or on the S-box(), or KeySchedule() operation) the capacity of the attacker varies notably, since the successful insertion of faults will depend on the type of implementation of the AES cipher used. For example, in the case of an attack targeting the KeySchedule() generated and stored in advance, it will be difficult to inject faults inside it. On the other hand, on a rolled implementation of the AES cipher, performing a single round each clock cycle forces the attacker to optimize the fault injection in order to make them occur herein on specific clock cycle. Note that, in order to study and compare the total coverage of each of the countermeasures, we will consider that the attacker is capable of achieving any type of fault in any scenario.

3.3. Aes Vulnerabilities

The use of differential fault analysis on the AES cipher [5–17] has shown its vulnerability to this type of analysis. The vulnerabilities found can be classified according to the block or function where the attack is carried out (state matrix, S-box() or KeySchedule()), the type of fault injected (single fault, multiple fault, etc.), or the round on which the attack must be carried out. Table 3 presents a classification of the different types of cryptanalysis reported in the literature where the DFA on the AES cipher is carried out.

Table 3. DFA on AES Classification.

Reference	Attack on State	Attack on S-Box	Attack on KeySchedule	Type of Attack	Attack in Round
[6]	✔	✘	✔	S.Bit | S.Byte	8 and 9
[7]	✔	∼	✘	M.Bit (Same Byte)	7, 8 and 9
[8]	✔	✔	✘	S.Byte	8
[9]	✔	∼	✘	S.Byte	9
[10]	✘	✘	✔	S.Byte	8 and 9
[11]	✘	✘	✔	S.Byte	8
[12]	✔	✘	✔	S.Byte	7, 8 and 9
[13]	✘	✘	✔	M.Byte	9
[14]	✘	✘	✔	M.Byte	9
[15]	✘	✘	✔	M.Byte	9
[16]	✔	✘	✘	S.Byte | M.Byte	8
[17]	✔	✘	✘	S.Byte | M.Byte	8

✔ = Applicable. ✘ = Not applicable. ∼ = Maybe applicable. S.Bit and M.Bit = Single Bit and Multi Bit, respectively. S.Byte and M.Byte = Single Byte and Multi Byte, respectively.

In the case of a single fault injection into the state matrix, the AES cipher has vulnerabilities if this fault is inserted before the S-box() operation of the ninth round [6]. In this case, only 50 faulty ciphertext are needed to recover the key. This vulnerability was experimentally shown by inserting glitches in the cipher clock signal [5]. In this case, the attacker must be able to repeat the fault in three different positions within the same byte using the same plaintext. For faults inserted in the seventh or eighth rounds, the attacker must be able to inject faults in different positions within the same byte of the state matrix, being their positions known or not [7]. It is also possible to attack the MixColumn() operation in the seventh, eighth, and ninth rounds. For the attack made on the MixColumn() operation in the eighth round, 20 pairs of correct/faulty ciphertext are necessary. For the attack on the MixColumn() operation in the ninth round, 40 to 50 different pairs of correct/faulty ciphertext are needed [7].

In the scenario where an attacker is able to change a whole byte [6,8–12,16,17], the vulnerability of the cipher is extended to the three attack points presented in Table 3. When injecting a full byte, the attacker can recover the key preforming the attack during the generation of the state matrix, or the S-box() process, or the KeySchedule(). These attacks can be carried out from round 7 onwards [6,8,10–12,16,17]. In the case of [9] the attack is carried out on round 9, but the authors claim that their attack could be adapted and implemented in previous rounds.

On the other hand, if the attack scenario considered includes the injection of multiple bytes, it can only be exploited in the state matrix and in the KeySchedule(), as this vulnerability cannot be exploited in the S-box() process [13–17]. These works suggest that this attack can be deployed in the eighth and ninth rounds of the cipher when targeting the state matrix, but only possible in the eighth round when targeting the KeySchedule().

4. Countermeasures against DFA

In order to protect the cryptographic system against the attacks, there are different techniques and schemes that allow detecting if the circuit is being attacked by fault injection. It should be noted that in this paper, we will focus on fault detection schemes and not on correction schemes because in the literature most of the protection solutions are only fault detection oriented. Additionally, fault correction has a higher resource consumption and is out of our scope. These techniques are basically divided into three groups: hardware redundancy, temporal redundancy, information redundancy, or a combination between them. The following describes each type of protection.

1. The hardware redundancy [35,36] countermeasure consists of duplicating the circuit to be protected or part of it in order to compare the result obtained after encryption or decryption to check if there is a difference. Figure 3 depicts the representation of two examples of hardware redundancy. Figure 3a the complete redundancy of the cryptographic circuit, while Figure 3b depicts an example of partial redundancy. This type of countermeasure is the most direct and simplest, but at the same time, it is the one with the highest resource cost. Given that in the AES cipher, the main vulnerability, regarding fault injection, is based on the propagation of faults in the last rounds, the protection against this type of countermeasures would consist of distorting this propagation. To do this, it is possible to use two redundant state matrixes and exchange their values with the aim of both distorting and detecting a possible fault [35]. Another protection solution using redundant logic is to duplicate vulnerable operations such as S-boxes. A scheme of this type allows to execute the same function on multiple data at the same time, i.e., duplicating the S-boxes for the same data and verifying the correct operation in an iterative way [36].

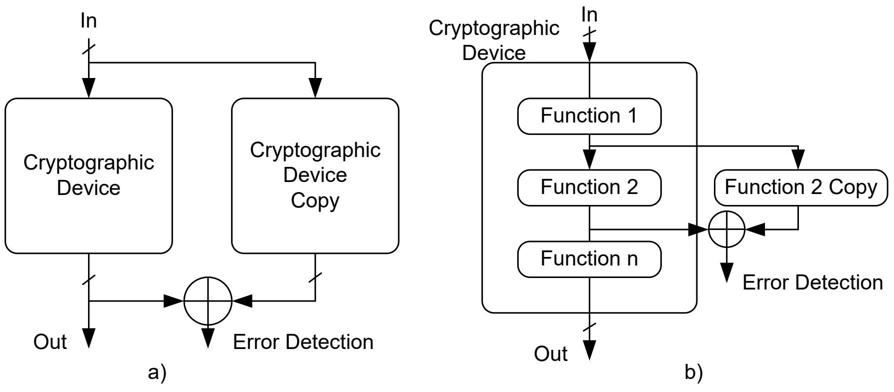

Figure 3. Representation of hardware redundancy, (**a**) Full redundancy, (**b**) partial redundancy.

2. The temporal redundancy [37–39] countermeasure is based on repeating operations of the cipher in a reverse way or by duplication, thus being able to check the result of an operation with its previous value or the one obtained from the second run. Figure 4 depicts two examples of temporal redundancy. Figure 4a) depicts the encryption and decryption, allowing to compare the result, after decryption, with the input value. In the case of Figure 4b), the reverse operation belonging to the decoding of a single function is repeated. This type of countermeasure takes advantage of the fact that most cryptographic circuits implement both encryption and decryption. While having a low resource cost, it significantly increases the time required since it needs to operate twice as long to perform the inverse operations for the verification. On the other hand, depending on the level of security and redundancy, protection can be achieved with greater or lesser resource consumption.
   In [37], the use of pipeline architectures that use both edges of the clock signal is proposed in order to reduce the throughput degradation when duplicating the cipher function. A solution called recomputing with permuted operands (REPO) is presented in [38]. This approach consists of the duplicate computation of two inputs and the permutation between them to perform a new computation and check if the result is the same or if any faults have been introduced. In [39], a solution is presented where the result of each round is stored in a register and is used to repeat the round. In addition, using the multiplexed keys of KeySchedule() allows to perform the rounds alternately and compare the results in search for faults.

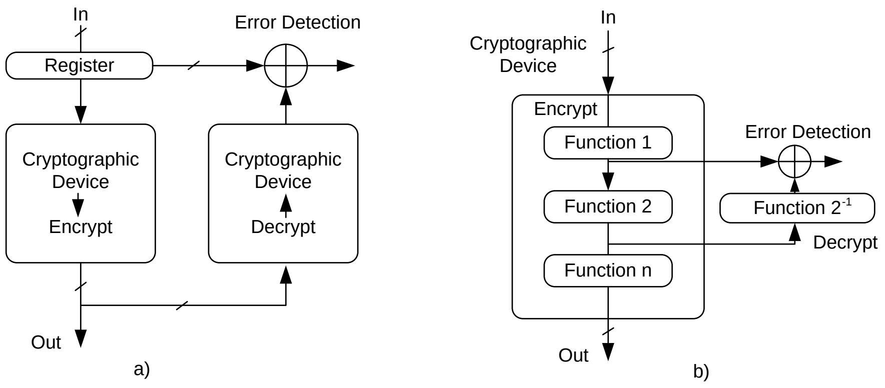

Figure 4. Representation of temporal redundancy, (**a**) Total redundancy, (**b**) partial redundancy.

3. The information redundancy [40–52] countermeasures consists of adding information to the state of the ciphers before the intermediate operations. This information allows to detect if the state was tampered during the process. It is possible, for example, to detect whether faults have been injected during state transformation processes using a parity predictor and a parity checker. A representation of an example of information redundancy can be seen in Figure 5. In this case, it has been chosen to add a parity bit to the state matrix and compare this parity after performing the encryption operation. The main advantage of this type of approach is the low resource overhead. Depending on the type of added information, the cost can be very small. However, the main advantage of this type of countermeasure is the difficulty of relating the parity before and after the encryption operations, since these are non linear. However, different types of faults will not be detected, depending on the number of bits used for fault detection. For example, the use of a single parity bit can detect odd state changes, but if the injected fault changes an even number of bits, it will not detect it.
   When this type of solution is applied to the AES cipher, it is necessary to take into account whether the nonlinear S-box() function is implemented through the use of Lookup Tables (LUTs) or not. For example, if the solution needs to access the intermediate result of the reverse multiplication of the S-box(), the scheme cannot be applied if the S-box() is implemented in a memory since there is no access to that intermediate result. On the contrary, if the scheme involves adding a parity bit in memory, as a signature of the input data, this scheme cannot be applied if the S-box() implementations are not implemented with memories.
   Taking this into account, there are numerous options available by applying this type of scheme to the AES cipher. An option is to calculate the parity of input data of the S-box() and add it to the memory in order to compare the parity at the input and output of the S-box() [41,42]. The same approach is also applicable if the differential result between the input and output data of the (S-box()) is added as a memory output instead of computing the parity [44]. It is possible to detect fault injection by validating the operation $Sbox_{input} \oplus Sbox_{output} = Diff$.
   In [45] a Hamming code signature generator targeting an AES cipher based on T-boxes() is presented. It generates a signature of the data before and after the operations of the T-boxes(). This allows to determine if any fault has been introduced in the S-box() and MixColumn() operations. This approach is applicable to both encryption and decryption and is capable of protecting the state matrix and S-box(), and also is applicable to KeySchedule().
   In addition to the parity prediction added to the S-box(), it is also possible to add backup and multiplexing elements composed of an additional S-box() that allows reconfiguration of a possible injected fault. This allows to drive the data by the correct line in online mode [47]. On the other hand, depending on the type of parity predictor and the types of considered faults covered, different solutions with different resource consumption can be obtained, such as [40,52]. These works attacks both at bit and byte

level, being applicable to the KeySchedule(). In [48,51], the authors present different approaches in which the S-box() is implemented in composite fields and faults can be detected as data transformations result in odd faults being changed to even faults.

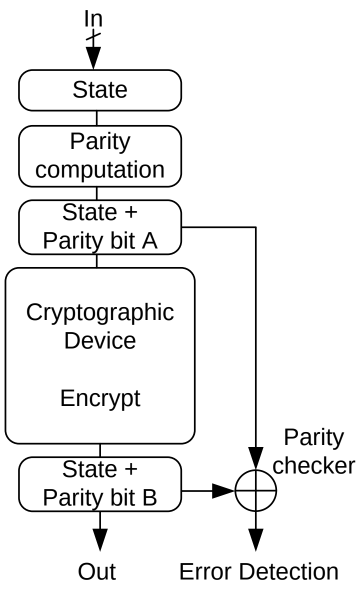

Figure 5. Representation of information redundancy.

Another possible approach is to divide the detection logic between linear and non-linear layers [49]. In the first case, a simple linear code can be used, while in the second case, by computing the inverse operation, the necessary fault detection can be provided. In addition, since the S-boxes are based on the computation of the multiplicative inverse, it is possible to compute the least significant bits of the inverse product as proof of the accuracy of the result. The number of computed bits can be adjusted according to the desired cost/protection ratio. This scheme requires the intermediate result of the inverse multiplication and cannot be applied to memory-based S-box() implementations as the intermediate result is not accessible. To solve this problem, it is possible to develop a new mathematical formulation for the S-box() that considers both the inverse multiplication and the affine transformation and therefore does not matter the way in which the S-box() is implemented [43].
On the other hand, it is possible to develop a fault detection scheme based on a nonlinear $(n,k) - code$ and a robust cubic network [50]. In this solution, the length of the signature used to verify the data is configurable, so that it is possible to achieve the desired coverage/cost ratio, depending on the number of bits used for the signature. A solution based on the use of polynomial residue number systems (PRNS) is presented in [52]. In PRNS, irreducible polynomials over $GF(2^4)$ are used and error detection is achieved by applying redundant modules. Due to the independence between PRNS operations, the cipher has intrinsic resistance to attacks, being able to detect up to 4-bit errors that may occur in a single $GF(2^4)$ AES core.

4. A combination of techniques [53,54] can be used to mitigate the limitations that each one of them presents individually. Figure 6 depicts the combination between hardware redundancy with the use of a parity bit. The combination of different countermeasures depends on the balance between resource consumption and security. Depending on the targeted protection level, different schemes can be obtained. Three types of protection can be designed, being these at the algorithm level, at the round level, or at the operation level. The first consists of performing the complete process of encryption and decryption, the second compares the result of encryption and decryption for each round and the third performs the comparison after each operation of each round [53].

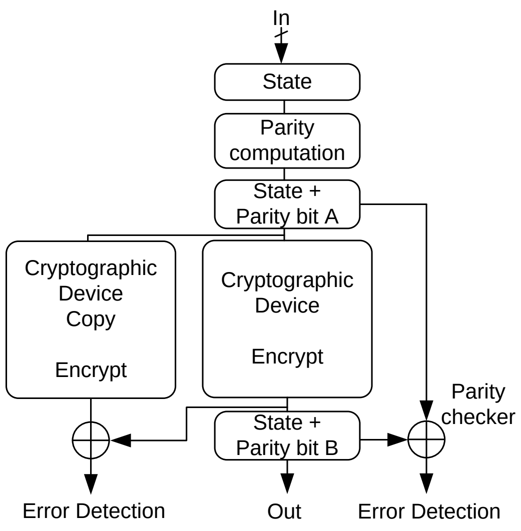

Figure 6. Representation of combined redundancy.

Another possible approach is the combination of hardware and temporal redundancy, i.e., each encryption or decryption operation is duplicated and performed one after the other in each round. Each round is divided into two, one half being used for encryption and the other for decryption, in an alternate mode, thus allowing the check to be carried out. This type of approach doubles the number of cycles required to encrypt the information, but is able to balance it since it is able to operate at a higher frequency [54].

Table 4 lists the different protection proposals and their classification. This table shows the block or function where each countermeasure offers protection (state matrix, S-box() or KeySchedule()), type of fault injection that is able to detect, and, finally, the fault coverage against the type of fault considered. Note that only proposals that have been implemented and tested experimentally are listed, leaving out those that are only described theoretically without test data or implementation costs. Without their data, it is not possible to make a comparison. For example, in the case of [40], the proposed information redundancy can protect against the injection of any type of fault in the state matrix or in the S-boxes, but does not protect the KeySchedule(), with a fault coverage of 80.47%. On the other hand, in the case of [37], the proposed temporal redundancy can protect against D and E type faults, that is, single-byte faults or multi-byte faults and has a 99.82% of fault coverage. However, it leaves the circuit unprotected against other types of faults.

Table 4. Countermeasure classification.

Reference	Type of Countermeasure	Protects the State	Protects the S-Box	Protects the KeySchedule	Type of Fault Detection	Fault Coverage (%)
[35]	HW Red.	✔	✘	∼	All	100
[36]	HW Red.	✘	✔	✘	All	98.22
[37]	Temp. Red.	✔	✔	✘	D and E	99.82
[38]	Temp. Red.	✔	✔	✔	All	99.99
[39]	Temp. Red.	✔	✔	✘	A	100
[40]	Info. Red.	✔	✔	✘	All	80.47
[41]	Info. Red.	✔	✔	✘	B.1 and C.1	75.6
[42]	Info. Red.	✔	✔	✘	B.1, C.1 and F	88
[43]	Info. Red.	✔	✔	✘	All	77.48
[44]	Info. Red.	✔	✔	✘	All	91∼98
[45]	Info. Red.	✔	✔	✔	All	100
[46]	Info. Red.	✘	✔	✔	A, B.1 and C.1	99.12

Table 4. *Cont.*

Reference	Type of Countermeasure	Protects the State	Protects the S-Box	Protects the KeySchedule	Type of Fault Detection	Fault Coverage (%)
[47]	Info. Red.	✘	✔	✘	A, B.1, C.1 and F	97
[48]	Info. Red.	✔	✔	✘	All	90
[49]	Info. Red.	✔	✔	✘	All	100
[50]	Info. Red.	✘	✔	✘	A, B.1, B.2, C.1 and C.2	97
[51]	Info. Red.	✘	✔	✘	A, B.1, B.2, C.1 and C.2	93.75
[52]	Info. Red.	✔	✔	✔	D and E	99.93
[53]	Combined	✔	✔	✘	All	100
[54]	Combined	✔	✘	✘	All	100

✔ = Applicable. ✘ = Not applicable. ∼ = Maybe applicable.

The fault coverage data in Table 4 correspond to those provided by the authors of the countermeasures. It should be noted that in some cases the authors do not give exact values and claim that their countermeasure is able to protect against one type of fault or another completely. In the case of Table 5, the fault coverage data have been obtained by applying Equation (1).

Table 5. Implementation cost, performance and fault coverage corrected of each scheme.

Type of Countermeasure	Scheme	Technology	Area Overhead	Frequency Degradation	Throughput Degradation	Fault Coverage Corrected (%)
None	Original	-	1	1	1	0
HW red.	[35]	Virtex 5	1.8655	1	1	100
	[36]	0.35 µm	1.3573	0.6364	0.6363	98.22
Temp. red.	[37]	0.13 µm	1.36	NA	NA	24.96
	[38].1	Virtex 4	1.159	0.966	0.4838	99.99
	[38].2	Virtex 4	1.131	0.904	0.4546	99.99
	[39].1	Virtex 4	1.043	0.85	0.85	12.5
	[39].2	Virtex 4	1.0758	0.8342	0.8342	12.5
Info. red.	[40]	Virtex 5	1.1566	0.739	0.7389	80.47
	[41]	Virtex 1000	1.0781	0.7088	0.7088	18.9
	[42]	NA [1]	1.731	0.6451	0.6449	33
	[43]	Virtex 4	1.167	0.886	0.4649	77.48
	[44].1	Virtex 5	1.2538	0.8847	0.8848	98.22
	[44].2	Virtex 5	1.2039	0.868	0.8638	91.14
	[45]	Spartan 6	1.16	1	1	100
	[46]	NA	1.44	NA	NA	37.17
	[47]	NA	1.4	NA	NA	48.5
	[48]	NA	1.35	NA	NA	90
	[49]	XCV1000E	1.77	0.8653	0.87	100
	[50].1	Virtex II PRO	1.3145	0.9778	0.9778	60.63
	[50].2	Virtex II PRO	1.2845	0.9354	0.9354	60.63
	[50].3	Virtex II PRO	1.2406	0.9654	0.9654	60.63
	[51].1	Spartan 3	1.8102	1.0016	1.0016	58.59
	[51].2	Spartan 3	1.5844	0.8308	0.8308	58.59
	[52]	0.18 µm	1.18	0.65	0.74	24.98

Table 5. *Cont.*

Type of Countermeasure	Scheme	Technology	Area Overhead	Frequency Degradation	Throughput Degradation	Fault Coverage Corrected (%)
Combined	[53].1	Virtex 6	1.2097	0.7765	0.3825	100
	[53].2	Virtex 6	1.189	0.8012	0.7344	100
	[53].3	Virtex 6	1.3808	0.7818	0.7644	100
	[54].1	90 nm	1.2433	2.801	1.3337	100
	[54].2	90 nm	0.8542	1.5325	0.2702	100
	[54].3	90 nm	0.7857	1.5947	0.2406	100

[1] NA = Not Applicable.

5. Comparison between Different Protection Solutions

Table 5 depicts the cost of each solution analyzed in Section 5. The table considers the different alternatives presented for the same approach, considered technology, area overhead, frequency degradation, throughput degradation, and percentage of Fault Coverage Corrected (FCC). Note that the FFC is a metric used in this paper to compare the approaches with each other. This factor is explained in detail in the next section. The cases where an NA is used are those in which the authors do not provide information on a given characteristic.

5.1. Fault Coverage-Corrected

Since each approach has a different fault coverage and considers the protection against different types of DFA, it is necessary to create this factor in order to compare them. The FCC is obtained after assigning a correction factor value to each countermeasure according to the types of covered faults. This aspect is very important since there are countermeasures where the authors consider only one type of fault and do not take into account that there are DFA analyses that use other types of fault and that also allow the recovery of the secret key. Not considering all types of faults, even if some of them are complex to achieve, can lead to a security leak and offer incorrect fault coverage. Due to the different DFAs, we assume eight different types of faults (Table 2). Given that we consider eight different types of faults and considering that all of them are equally important because, depending on the DFA used, they allow recovering the secret key, all of them have been assigned the same weight and the correction factor is 12.5%. The value of each fault coverage corrected is obtained by (1), where FCC is Fault Coverage Corrected, TFC is Type of Fault Considered and FC is Fault Coverage. TFC represents the number of fault types that the countermeasure is able to detect and this value is multiplied by the weight of each fault type, that is, 12.5%. FC represents the percentage of fault coverage offered by the countermeasure considered.

$$FCC = TFC * 0.125 * FC \tag{1}$$

For example, the solutions that have been designed and tested considering all types of DFA faults have been assigned a correction factor of 100% and therefore their fault coverage value corresponds to those presented in Table 4. On the other hand, if we consider the countermeasure [37] and Table 4, it can be seen that this countermeasure can protect against two types of fault, type D and E and its percentage of coverage is 99.82%. Therefore, considering (1), the FCC = 2 × 0.125 × 99.82 = 24.96%. Despite having 99.82% coverage against E and D in Table 4, it does not offer any protection against the other types of faults and therefore its FCC in Table 5 is 24.96%.

5.2. Protection Solution Comparison

Hardware redundancy—[35,36]: In the first case [35], the fault coverage is 100% and the resource consumption in terms of area is 86%. If the frequency and throughput is considered, this solution does not have any penalty. In the second case [36], the fault coverage is 98.22% and the resource consumption in terms of area is 35%. In this case, the scheme presents a 37% penalty for the frequency and throughput. In both cases, neither of them offers protection for the KeySchedule(), and this, depending on the cipher implementation, can be a point of vulnerability if the attacks presented in [6,10–15] are used.

Temporal redundancy—[37–39]: As seen in Table 5, in [38,39] two implementation versions of this approach are proposed, herein denoted by .1 and .2, respectively. In [38], pipeline and iterative architectures of the protection solution are presented. In [39], two different pipeline architectures of the cipher with this protection approach are proposed. In [37] the area consumption is 36%, while in the cases of [38] are 15.9% and 13.1% and in the cases of [39] are 4.3% and 7.58%, respectively. If the frequency degradation is considered, no data are available in [37] but in [38,39] they reach a penalty of up to 10% and 16.58%, respectively. Only [38] presents a higher degradation if the throughput is considered, reaching a 54.54% penalty. Despite this penalty, it can be seen that only the countermeasure of [38] offers a level of protection greater than 99%, while [37,39], by covering only a few types of fault, their coverage drops drastically to 24.96% and 12.5%. Therefore, of these three, the most interesting is [38] since it is able to protect all blocks of the cipher, with a high level of coverage, and its trade-off between resource consumption and frequency is very close to that of the unprotected cipher but with a strong throughput penalty.

Information redundancy—[40–52]: These solutions present very different area consumptions due to the great variety of possible designs of the parity checker. In [40,41,43], data item .2 in the table [44], and [45,51] the area consumption to implement the protection does not exceed 20%. When they are analyzed, it can be seen that both the operating frequency and the throughput decrease significantly, reaching 35% less, except for the case of [45] where the frequency does not have a penalty. In the case of [40,41,43] it can be seen that their fault coverage is below 85%, reaching 18.9% in the case of [41], since it only detects odd-type faults. On the other hand, the solution presented in [45] is capable of offering a coverage of up to 100% of all types of faults considered without presenting any frequency or throughput penalty, with only a 16% cost in area. As seen in Table 4, this solution is able to protect the state matrix, the S-box() and the KeySchedule().

Combination of techniques—[53,54]: For these proposals, the authors present three different solutions for each countermeasure. In [53], three solutions are presented one at the algorithm level, another at the round level and another one at the operation level. In the case of [54], results are presented for implementations optimized for size, speed and efficiency of throughput per gate. As seen in Table 5, in [53] the area overhead ranges from 18% to 38%, being the round protection the one with less frequency degradation. If throughput is taken into account, it can be seen that the protection at the algorithm level shows a degradation of 62% with respect to the unprotected implementation, with the other two options being similar in terms of frequency and throughput. Considering that the three options present the same level of security, the protection at round level would be the most interesting, since the balance between resource consumption and operating frequency is the most efficient one. In the case of the solution presented in [54], the data presented should not be taken into account, since the authors implement them with optimizations. This means that the data from the protection schemes together with the cipher are lower than the unprotected cipher itself. However, these implementations are considered because they represent a good example of a combination of protection solutions.

5.3. Overall Comparison

The trade-offs and relative comparison for each of the presented solutions are depicted in Figures 7 and 8. Figure 7 trade-off is considered between the degradation of the operating frequency versus the cost in terms of area. Figure 8 presents the trade-off between the F.C.C. and the Area-Delay-Product (ADP). The reference identified in both figures as [0] corresponds to the unprotected AES cipher and is used as a reference point. The green, yellow and red stripes in Figure 8 denote the ranges of highest to lowest level of protection, respectively, considering the fault coverage.

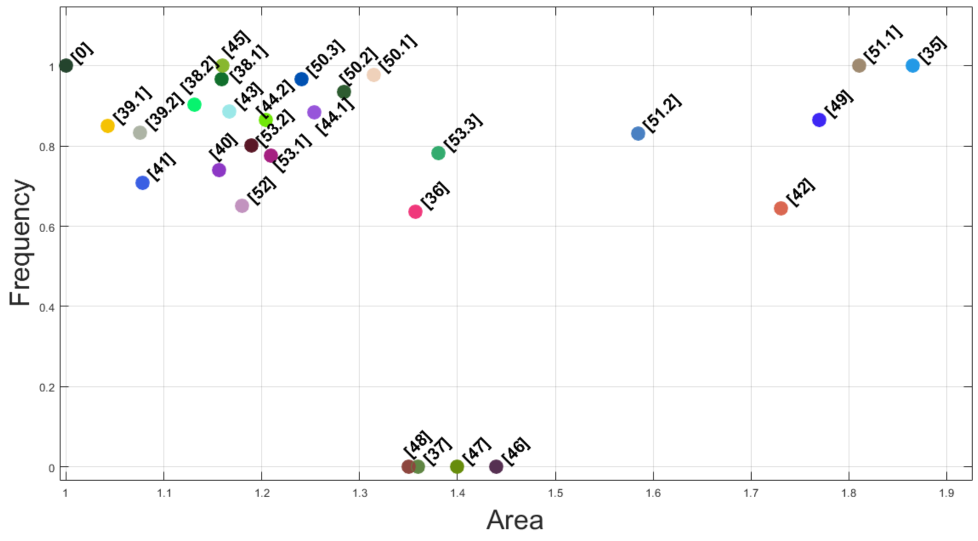

Figure 7. Frequency degradation VS Area overhead of each countermeasure.

In the analysis of frequency overhead vs. area cost (Figure 7) it is necessary to note that the solutions in Refs. [37,46–48] by not presenting data on frequency degradation, they appear at the bottom. These data have been included in that figure because, although it is not possible to analyze them together with other solutions, it allows us to visualize the cost in area with respect to other countermeasures. For the analysis of fault coverage against the Area-Delay-Product (Figure 8) these solutions do not appear, since it is not possible to determine this factor.

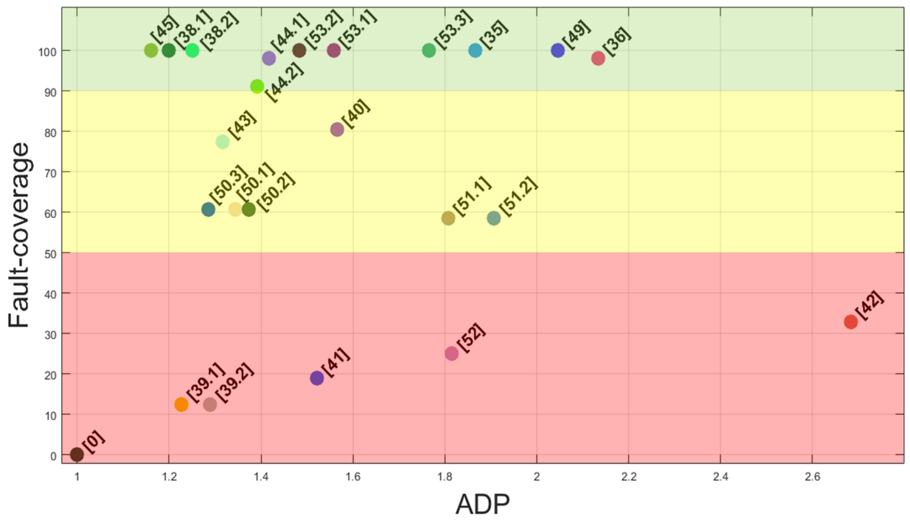

Figure 8. Fault Coverage vs. Area and Frequency overhead.

According to Figure 7 and focusing on the area consumption, it can be seen that the highest consumption countermeasures are those located above 30%, including hardware redundancy [35,36], with [35] presenting the highest resource consumption. A second group of countermeasures between 10% and 30% is where most solutions are located. Finally, the solutions in [39,41] that are below 10%. Regarding the operating frequency after adding the countermeasures, it can be seen that all are between 60% and 100%.

Of these, the ones that offer the best performance, with 99.99% or 100% fault coverage, are [35,38,45,49,53]. Both [35,45] offer the same operating frequency as the unprotected cipher, but with different implementation costs. In general, among all the countermeasures, the best trade-offs in terms of area and frequency are [38,45].

Taking into account the values depicted in Figure 8, it can be seen that the countermeasures studied can be divided mainly into three groups. On the one hand, there are the countermeasures that offer a high level of security, between 90% and 100% fault coverage. Between 50% and 90% are the solutions that offer an intermediate level of security. Finally, there is the group of countermeasures with a low level of security, less than 50%. The countermeasures that are below 90%, medium and low level of protection, are due to the penalty that they suffer by not offering protection against all types of considered possible faults. For example, in the previous analysis it was possible to see how [39,41] were the best proposals in term of costs, but in this analysis they fall significantly below 20% since they only consider one or two types of faults. Among the different proposals, in this case, it can be seen that the best options are [38,45] because they offer 100% coverage and their ADP is less than 30%. By analysing [38,45] in Figure 7, it can be seen that they present a very efficient trade-off, where the solution presented in [45] has an area cost bellow 20% and no frequency degradation. On the other hand, the [38] solutions present a small penalty in terms of frequency, but their area resources in the case of data item .2 in the table [38] is even lower than [45].

6. Discussion

The following discusses the considerations that must be taken into account when selecting the most adequate type of countermeasure solution and the targeted fault coverage.

Taking into account Table 3, DFA analyses focus on attacking three different points, the state matrix, the S-box() or the KeySchedule(). This is very important since many countermeasures do not protect these three points at the same time, possibly leading to DFA vulnerabilities. For example, in the analysis presented in Section 6, the solutions in [35,49] are able to offer 100% fault coverage, but they are not able to protect the S-box() in the case of [35] and the KeySchedule() in the case of [49]. It should be noted that, depending on the selected protection solution, the cipher remains equally vulnerable to attack. Therefore, if complete protection is desired, only the options [38,45] are useful. In case there is no need for a KeySchedule() protection because it is generated and stored before performing all the rounds of the cipher, refs. [37,39–44,48,49,53] would also be useful.

Figure 8 can be used to guide the selection of the approach to use. For example, if the restrictions are very high and the fault coverage must be moderate, the proposals [43,50] may be enough. On the other hand, if the application does not have a high level of time or area restrictions, but, on the contrary, needs a maximum level of protection, valid options may be [35,44,54]. Furthermore, depending on the type of implementation that is performed, it may be possible to assume a lower level of protection. For example, if the application uses rolled implementations instead of unrolled ones, it becomes difficult to insert faults between operations of the rounds, since everything is done in the same clock cycle.

Regarding the trade-off between each type of countermeasure, on the one hand, hardware redundancy countermeasures are those that can offer the highest level of security at a higher cost of resources. As a consequence, these solutions must be oriented to environments where designs require a high level of security, but where resource limitations are not a strong constraint. On the other hand, temporal redundancy countermeasures allow for an improvement in the cost of resources compared to hardware redundancy, but on the contrary, they will have a high cost in operation frequency. In the case of information redundancy countermeasures, they have the great advantage of not presenting a high cost of resources and operating frequency, but on the other hand, the difficulty in determining parity in non-linear functions brings with it an increase in resources and execution time. Furthermore, depending on the solution, they will not be able to detect even-type faults. On the other hand, the use of information redundancy allows to determine faults inside the operations, such as S-boxes(), but if the fault is injected in the transition values (between the different functions), the countermeasures will not be able to detect them. Therefore, one of the most interesting alternatives is the use of combined countermeasures that allow us to find a trade-off between the advantages of each one of the types of countermeasures and to minimize the disadvantages that each one of them presents by itself.

From the designer point of view, access to the codes or designs that allow the implementation of the schemes is another problem to take into account. Most of the solutions reported in the literature are described theoretically. For example, information redundancy countermeasures can be very complex from the mathematical point of view that they require, making it very difficult to implement in order to prove their viability in different environments or applications.

7. Conclusions and Future Works

In this paper, a comprehensive study of the fault injection, DFA and protection solutions for the AES cipher is presented. It shows that there are a large number of vulnerabilities reported in the literature for the AES cipher when attacked by active attacks called fault injection. At the same time, there are a large number of proposed protection solutions that try to detect possible faults injected into the circuit. The three groups of protection schemes involving fault injection protections have been studied, which are hardware redundancy, temporal redundancy, information redundancy, or a combination of them. Each has different advantages and disadvantages. Herein, a complete study of the main characteristics of the protection approaches has been carried out, considering area overhead, frequency degradation, throughput degradation, and fault coverage. In order to be able to compare all of them, a metric called Fault Coverage Corrected (FCC) was introduced to recalculate the fault coverage according to the types of faults they are able to detect. As a result of this review, it has been possible to compare the most important protections, comparing the trade-off between area consumption versus frequency degradation and the trade-off between fault coverage and area delay product. This has allowed to group the different protection solutions that can be used depending on the level of security and constraints.

As future work, different directions are proposed. The need to improve the transfer between theoretical studies and the availability of open access implementations. Adequately test the design of countermeasures that consider protection against all possible faults used by DFA in different technologies. Finally, it should not be left out that all countermeasures used must be tested equally against Side Channel Attacks such as Differential Power Analysis or Correlation Power Analysis. The use of countermeasures against fault injection can increase the leakage of information useful for this type of analysis since the greater the data redundancy, the greater the leakage of information. Therefore, a joint countermeasure analysis must be a priority element if maximum protection of this type of ciphers will be achieved.

Author Contributions: Conceptualization, F.E.P.-O., E.T.-S., A.J.A.-J., C.J.J.-F. and R.C.; methodology, F.E.P.-O., E.T.-S. and R.C.; formal analysis, F.E.P.-O., E.T.-S., A.J.A.-J., C.J.J.-F. and R.C.; investigation, F.E.P.-O. and E.T.-S.; resources, A.J.A.-J., C.J.J.-F. and R.C.; data curation, F.E.P.-O. and E.T.-S.; writing—original draft preparation, F.E.P.-O., E.T.-S. and R.C.; writing—review and editing, F.E.P.-O., E.T.-S., A.J.A.-J., C.J.J.-F. and R.C.; visualization, F.E.P.-O., E.T.-S., A.J.A.-J., C.J.J.-F. and R.C.; supervision, A.J.A.-J., C.J.J.-F. and R.C.; project administration, A.J.A.-J., C.J.J.-F. and R.C.; funding acquisition A.J.A.-J., C.J.J.-F. and R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by Grant PID2020-116664RB-I00 funded by MCIN/AEI/10.13039/501100011033.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: Authors want to acknowledge to projects Programa Operativo FEDER 2014-2020 and Consejería de Economía, Conocimiento, Empresas y Universidad de la Junta de Andalucía under Project US-1380823, to the SPIRS (Secure Platform for ICT Systems Rooted at the Silicon Manfacturing Process) Project with Grant Agreement No. 952622 under the European Union's Horizon 2020 research and innovation programme, to Erasmus+, to COST Action CRYPTACUS, to FCT (Fundação para a Ciência e a Tecnologia, Portugal), to the ERDF (European Regional Development Fund, EU) through the projects FCT: UIDB/50021/2020 and LISBOA-01-0145-FEDER-031901 (PTDC/CCI-COM/31901/2017, HiPErBio) and to Disruptive Sdn seCuRE communicaTIons for eurOpean defeNse, EDIDP-CSAMN-SDN-202-74-DISCRETION.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DFA	Differential Fault Analysis
IoT	Internet of Things
AES	Advanced Encryption Standard
RSA	Rives-Shamir-Adleman
FIA	Fault Injection Attack
NIST	National Institute of Standards and Technology
REPO	Recomputing with permuted operands
LUT	Lookup Tables
PRNS	Polynomial residue number systems
FCC	Fault coverage corrected
ADP	Area-Delay-Product

References

1. Dutta, I.K.; Ghosh, B.; Bayoumi, M. Lightweight Cryptography for Internet of Insecure Things: A Survey. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 475–481.
2. Kazemi, Z.; Fazeli, M.; Hely, D.; Beroulle, V. Hardware Security Vulnerability Assessment to Identify the Potential Risks in a Critical Embedded Application. In Proceedings of the 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), Napoli, Italy, 13–15 July 2020; pp. 1–6.
3. Boneh, D.; DeMillo, R.A.; Lipton, R.J. On the importance of checking cryptographic protocols for faults. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'97)*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 37–51.
4. Pub, N.F. 197: Advanced encryption standard (AES). *Fed. Inf. Process. Stand. Publ.* **2001**, *197*, 0311.
5. Agoyan, M.; Dutertre, J.M.; Naccache, D.; Robisson, B.; Tria, A. When Clocks Fail: On Critical Paths and Clock Faults. In Proceedings of the LNCS International Conference on Smart Card Research and Advanced Applications (CARDIS'10), Passau, Germany, 14–16 April 2010; pp. 182–193.
6. Giraud, C. DFA on AES. *LNCS* **2005**, *3373*, 27–41.
7. Dusart, P.; Letourneux, G.; Vivolo, O. Differential Fault Analysis on A.E.S. *LNCS* **2003**, *2846*, 293–306.
8. Piret, G.; Quisquater, J.J. A Differential Fault Attack Technique against SPN Structures, with Application to the AES. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03), Cologne, Germany, 8–10 September 2003; pp. 77–88.
9. Moradi, A.; Manzuri Shalmani, M.T.; Salmasizadeh, M. A Generalized Method of Differential Fault Attack against AES Cryptosystem. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES'06), Yokohama, Japan, 10–13 October 2006; pp. 91–100.
10. Chen, C.N.; Yen, S.M. Differential Fault Analysis on AES Key Schedule and Some Countermeasures. In *Information Security and Privacy (ACISP'03), Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2727, pp. 118–129.
11. Ali, S.S.; Mukhopadhyay, D. A Differential Fault Analysis on AES Key Schedule using Single Fault. In Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2011), Nara, Japan, 28 September 2011; pp. 35–42. [CrossRef]
12. Subidh Ali, S.; Mukhopadhyay, D.; Tunstall, M.; Ali, S.S.; Mukhopadhyay, D.; Tunstall, M. Differential fault analysis of AES: Towards reaching its limits. *J. Cryptogr. Eng.* **2013**, *3*, 73–97. [CrossRef]
13. Takahashi, J.; Fukunaga, T.; Yamakoshi, K. DFA mechanism on the AES key schedule. In Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'07), Vienna, Austria, 10 September 2007; pp. 62–72. [CrossRef]
14. Kim, C.H.; Quisquater, J.J. New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough. In Proceedings of the Smart Card Research and Advanced Applications, 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, 8–11 September 2008; Volume 5189, pp. 48–60.
15. Subidh Ali, S.; Mukhopadhyay, D. Differential Fault Analysis of AES-128 Key Schedule Using a Single Multi-byte Fault. In Proceedings of the Smart Card Research and Advanced Applications (CARDIS'11), LNCS 7079, Leuven, Belgium, 14–16 September 2011; pp. 50–64.
16. Saha, D.; Mukhopadhyay, D.; Roychowdhury, D. A Diagonal Fault Attack on the Advanced Encryption Standard. *IACR Cryptol. Eprint Arch.* **2009**, *581*, 1–17.

17. Mukhopadhyay, D. An Improved Fault Based Attack of the Advanced Encryption Standard. In Proceedings of the AFRICACRYPT'09, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, 21–25 June 2009; pp. 421–434.
18. Pawar, A.B.; Ghumbre, S. A survey on IoT applications, security challenges and counter measures. In Proceedings of the IEEE International Conference on Computing, Analytics and Security Trends (CAST'16), Pune, India, 19–21 December 2016; pp. 294–299.
19. Ali, S.; Guo X.; Karri, R.; Mukhopadhyay, D. *Fault Attacks on AES and Their Countermeasures, Secure System Design and Trustable Computing*; Springer: Cham, Switzerland, 2016; pp. 163–208.
20. Malkinm, T.G.; Standaert, F.X.; Yung, M. A comparative cost/security analysis of fault attack countermeasures. In Proceedings of the International Workshop on Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, 10 October 2006; pp. 159–172.
21. Baremghi, A.; Breveglieri, L.; Koren, I.; Pelosi, G.; Regazzoni, F. Countermeasures against fault attacks on software implemented AES: Effectiveness and cost. In Proceedings of the 5th Workshop on Embedded Systems Security (WESS'10), Scottsdale, Arizona, 24 October 2010; pp. 1–10.
22. Peterson, I. Chinks in digital armor: Exploiting faults to break smart-card cryptosystems. *Sci. News* **1997**, *151*, 78–79. [CrossRef]
23. Skorobogatov, S. *Low Temperature Data Remanence in Static RAM*; No. UCAM-CL-TR-536; University of Cambridge, Computer Laboratory: Cambridge, UK, 2002.
24. Skorobogatov, S.P.; Anderson, R.J. Optical Fault Induction Attacks. In *Cryptographic Hardware and Embedded Systems-CHES'02*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2523, pp. 2–12.
25. Govindavajhala, S.; Appel, A.W. Using Memory Errors to Attack a Virtual Machine. In Proceedings of the IEEE Symp. Security and Privacy (SP), Berkeley, CA, USA, 11–14 May 2003; pp. 154–156.
26. Quisquater, J.J. Eddy current for magnetic analysis with active sensor. In *Proceedings of the 3rd International Conference on Research in SmartCards (E-Smart'02);* UCL: Nice, France, 2002; pp. 185–194.
27. Barenghi, A.; Bertoni, G.; Breveglieri, L.; Pellicioli, M.; Pelosi, G. Low voltage fault attacks to AES. In Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'10), Anaheim, CA, USA, 13–14 June 2010; pp. 7–12.
28. Anderson, R.; Kuhn, M. Low cost attacks on tamper resistant devices. In *International Workshop on Security Protocols*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 125–136.
29. Fukunaga, T.; Takahashi, J. Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 Block Ciphers. In Proceedings of the Workshop on Fault Diagnosis and Tolerance in Criptography, Lusanne, Switzerland, 6 September 2009; pp. 84–92.
30. Ren, Y.; Wang, A.; Wu, L. Transient-Steady Effect Attack on Block Ciphers. In *Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 2, pp. 433–450.
31. Potestad-Ordóñez, F.E.; Jiménez-Fernández, C.; Valencia-Barrero, M. Vulnerability Analysis of Trivium FPGA Implementations. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 3380–3389. [CrossRef]
32. Potestad-Ordóñez, F.E.; Jiménez-Fernández, C.J.; Valencia-Barrero, M. Fault Attack on FPGA implementations of Trivium Stream Cipher. In Proceedings of the International Symposium on Circuits and Systems (ISCAS'16), Montreal, QC, Canada, 22–25 May 2016; pp. 562–565.
33. Ghalaty, N.F.; Yuce, B.; Taha, M.; Schaumont, P.; Tech, V. Differential Fault Intensity Analysis. In Proceedings of the Workshop on Fault Diagnosis and Tolerance in Criptography, Busan, Korea, 23 September 2014; pp. 49–58.
34. Ghalaty, N.F.; Yuce, B.; Schaumont, P. Differential fault intensity analysis on PRESENT and LED block ciphers. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 174–188.
35. Joye, M.; Manet, P.; Rigaud, J.B. Strengthening Hardware AES Implementations against Fault Attacks. *IET Inf. Secur.* **2007**, *1*, 106–110. [CrossRef]
36. Natale, G.D.; Flottes, M.L.; Rouzeyre, B. On-Line Self-Test of AES Hardware Implementations. In Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Edinburgh, UK, 25–28 June 2007; pp. 1–6.
37. Maistri, P.; Vanhauwaert, P.; Leveugle, R. A novel double-data-rate AES architecture resistant against fault injection. In Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'07), Vienna, Austria, 10 September 2007; pp. 54–61. [CrossRef]
38. Guo, X.; Karri, R. Recomputing with Permuted Operands: A Concurrent Error Detection Approach. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2013**, *32*, 1595–1608. [CrossRef]
39. Rajendran, J.; Borad, H.; Mantravadi, S.; Karri, R. SLICED: Slide-based concurrent error detection technique for symmetric block ciphers. In Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST'10), Anaheim, CA, USA, 13–14 June 2010; pp. 70–75. [CrossRef]
40. Mestiri, H.; Kahri, F.; Bouallegue, B.; Machhout, M. A hardware FPGA implementation of fault attack countermeasure. In Proceedings of the International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA'15), Hammamet, Tunisia, 21–23 December 2014; pp. 178–183. [CrossRef]
41. Wu, K.; Goessel, M. Low Cost Concurrent Error Detection for the Advanced Encryption Standard. In Proceedings of the Test Conference (ITC'04), Charlotte, NC, USA, 26–28 October 2004; pp. 1242–1248.
42. Yen, C.H.; Wu, B.F.; Member, S. Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard. *IEEE Trans. Comput.* **2006**, *55*, 720–731.

43. Mozaffari-Kermani, M.; Reyhani-Masoleh, A. Concurrent structure-independent fault detection schemes for the advanced encryption standard. *IEEE Trans. Comput.* **2010**, *59*, 608–622. [CrossRef]
44. Mestiri, H.; Benhadjyoussef, N.; Machhout, M.; Tourki, R. High Performance and Reliable Fault Detection Scheme for the Advanced Encryption Standard Hassen. *Int. J. Comput. Netw. Inf. Secur.* **2013**, *5*, 49–55. [CrossRef]
45. Potestad-Ordóñez, F.E.; Tena-Sánchez, E.; Chaves, R.; Valencia-Barrero, M.; Acosta-Jiménez, A.J.; Jiménez-Fernández, C.J. Hamming-Code based fault detection design methodology for block ciphers. In Proceedings of the International Symposium on Circuits and Systems (ISCAS'20), Seville, Spain, 12–14 October 2020; pp. 1–10.
46. Breveglieri, L.; Koren, I.; Maistri, P.; Milano, P. Incorporating Error Detection and Online Reconfiguration into a a Regular Architecture for the Advanced Encryption Standard. In Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05) 0-7695-2464-8/05, Monterey, CA, USA, 3–5 October 2005; pp. 72–80.
47. Kermani, M.M.; Reyhani-Masoleh, A. Parity-based fault detection architecture of S-box for advanced encryption standard. In Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems, Arlington, VA, USA, 4–6 October 2006; pp. 572–580. [CrossRef]
48. Karpovsky, M.; Kulikowski, K.J.; Taubin, A. Differential Fault Analysis Attack resistant architectures for the Advanced Encryption Standard. In *Smart Card Research and Advanced Applications VI*; Springer: Boston, MA, USA, 2004; pp. 117–192.
49. Karpovsky, M.; Kulikowski, K.J.; Taubin, A.; Member, S. Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard. In Proceedings of the International Conference on Dependable Systems and Networks, Florence, Italy, 28 June–1 July 2004; pp. 93–101.
50. Mozaffari-Kermani, M.; Reyhani-Masoleh, A. A lightweight high-performance fault detection scheme for the advanced encryption standard using composite fields. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2011**, *19*, 85–91. [CrossRef]
51. Chu, J.; Benaissa, M. Error detecting AES using polynomial residue number systems. *Microprocess. Microsyst.* **2013**, *37*, 228–234. [CrossRef]
52. Bertoni, G.; Breveglieri, L.; Koren, I.; Maistri, P. An Efficient Hardware-Based Fault Diagnosis Scheme for AES: Performances and Cost Department of Electronics and Information Technology. In Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04), Cannes, France, 10–13 October 2004; pp. 130–138.
53. Karri, R.; Wu, K.; Mishra, P.; Kim, Y. Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2002**, *21*, 1509–1517. [CrossRef]
54. Satoh, A.; Sugawara, T.; Homma, N.; Aoki, T. High-Performance Concurrent Error Detection Scheme for AES Hardware. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Washington, DC, USA, 10–13 August 2008; pp. 100–112.