Noname manuscript No.
(will be inserted by the editor)

A Stealthy Hardware Trojan based on a Statistical Fault Attack

Charles Momin · Olivier Bronchain ·
François-Xavier Standaert

Received date / Accepted date

Abstract Integrated Circuits (ICs) are sensible to a wide range of (passive, active, invasive, non-invasive) physical attacks. In this context, Hardware Trojans (HTs), that are malicious modifications of a circuit by an untrusted manufacturer, are one of the most challenging threats to mitigate. HTs aim to alter the functionality of the infected chip in a malicious way, e.g. under specific conditions known by the adversary. Fault attacks are a typical attack vector. However, for a HT to be exploitable by an adversary, it also has to be stealthy. For example, a HT that would directly inject exploitable faults in a block cipher may be spotted by analyzing its functional behavior (i.e. the positions and the distribution of the faulty values appearing). In this paper, we propose a stealthy HT instance leading to successful and hidden Statistical Fault Attacks (SFA). More precisely, the faults are injected when the chip is running under condition for which metastabilty occurs (i.e. with a increased clock frequency), leading to the apparition of faults at random positions within the target implementation. In addition, an internal bit is set to a value known only by the adversary, allowing him to perform efficient SFA. Compared to classical SFA, the HT uses its control on the target to circumvent behavioral detection tests. Indeed, it also adds computation errors in the early rounds of the target cipher which are not exploitable via SFA.

Keywords Stealthy Hardware Trojans, Statistical Fault Attacks.

1 Introduction

Hardware Trojans (HTs) are intentional and malicious alterations of a circuit's specifications that lead to a modification of its functionalities. As for their software counterparts, they can lead to damaging attacks against systems that may be used in sensitive applications (e.g. sensors, encryption engines, ...).

As mentioned in the taxonomies from [9,4,32], HTs usually deliver their malicious functionality (i.e. the payload) either under specific conditions (i.e. the trigger), as shown by various examples in the literature [20,18], or continuously (i.e. always on), as exposed in [23, 22]. They may be implemented by adding or removing logic gates (i.e. functional alterations) or/and by modifying physical parameters of the existing logic (i.e. parametric alterations).

As explained in [16] for Path Delay HTs and generalized here, the design of HTs is driven by two main objectives. First, the triggerability assures that the HT can be launched with a high probability under a secret manipulation (i.e. physical or digital) known only by

UCLouvain, Louvain-la-Neuve, Belgium

E-mail: charles.momin@uclouvain.be
E-mail: olivier.bronchain@uclouvain.be
E-mail: fstandae@uclouvain.be@uclouvain.be

the adversary. Second, the stealthiness assures that the HT is triggered with extremely low probability under honest and/or randomly chosen manipulations of the chip.

Due to their wide range of payload and trigger possibilities, HTs are a serious threat to consider in the design and manufacturing process of Integrated Circuits (ICs). Especially, taking into account that the latter has become increasingly globalized over the last 20 years, more and more insertion vectors are emerging [29]. In this context, HTs force verification processes to perform additional tests in order to guarantee a chip's correct behavior. As surveyed in [30], these tests may range from simple functional specification verification to complex side-channel analysis such as proposed in [33,3,26,1]. Another approach aims to prevent an adversary to properly perform specific types of trigger mechanisms at the cost of additional logic gates, as done in [31,12].

In this paper, we present a new instance of HT that allows an informed adversary to perform a stealthy Statistical Fault Attacks (SFA) against any AES implementation following a pipelined architecture. Based on functional and parametric modifications, our instance of HT exploits timing violations in the early cipher rounds and leverages Boolean addition to hide a (potentially suspicious) fault in the set bit model. Our results based on a proof-of-concept FPGA implementation show that it can be implemented with a limited amount of logic while allowing the HT to escape functional detection mechanisms. To the best of our knowledge, this is the first stealthy HT with digital payload against a block cipher implementation based on metastability.

2 Background

This section begins with an overview of Faults Attacks (FAs) and goes then further into the details of SFAs. It ends with an explanation of the metastability effect that may be observed in synchronous designs.

2.1 Faults Attacks

FAs are active attacks that take advantage of computation errors. First introduced against public key cryptosystems in [8], they have been generalized shortly after against any secret key cryptosystems with the apparition of the Differential Fault Analysis (DFA) proposed in [7]. A fault can be seen as a localised change Δ in the value of an internal state s to a faulty value s_f, such that $\Delta = s \oplus s_f$. When he performs a DFA, an adversary makes some assumptions on the fault model and possesses pairs of correct and faulty ciphertexts. For each key candidate, he decrypts the pairs and checks if the differences between the internal states corresponds to what he expects. Faults can also be used to gain information when they are ineffective (i.e. $\Delta = 0$) leading to so-called Ineffective Fault Attack (IFA)[10]. An adversary able to set the value of a bit then learns the value of the latter during an encryption process if no computation error occurs when he injects the fault. Other techniques expoit the data-dependent nature of the induced faults under specific conditions (i.e. fault sensitivity) [21, 15].

Besides, a Statistical Fault Attacks (SFA) [14] (SIFA for the ineffective case [11]) is a method taking advantage of faulty values that follow a biased distribution (i.e. that is not uniformly distributed) in order to recover the secret key used in symmetric cryptosystems such as the Advanced Encryption Standard (AES)[**?**]. For each key candidate, an adversary decrypts some faulty ciphertexts and checks the distribution bias of the internal states bits. Since the AES is designed following the confusion-diffusion paradigm, the internal states of the final rounds are supposed to be uniformly distributed for different plaintexts. It follows that the key candidate giving the highest bias is assumed to be the correct key. Depending on its power, an adversary may have a perfect control, partial control or no control on the faults he induces. The complexity in term of required faulty ciphertexts depends on the bias of the faults distribution and thus on the ability of the adversary to inject accurate faults.

2.2 Metastability

In synchronous designs, each computation is processed at a given rate driven by the clock frequency. The circuit can be represented by a sequence of combinatorial logic operations separated by registers. At each positive edge of the clock signal, the data at the input of a register is transferred to the output of the latter and feeds the next combinatorial logic block. In a practical implementation, the delay over the path taken by the data between two registers (i.e. the propagation delay) mainly depends on the logical depth of the combinatorial elements composing the path. If the propagation delay between two registers is longer than the clock period T_c, the stability of the data outputted by the register is not assured. Precisely, the setup timing constraint is expressed as follows:

$$T_c \geq t_{pcq} + t_{pcd} + t_{setup},$$

where t_{pcq} is the propagation delay between the input and the output of the register, t_{pcd} is the propagation delay of the combinatorial logic and t_{setup} is a technology-dependent delay that is required by the register to properly handle the data. If this constraint is not met, the data at the output of the register is said to be in a metastable state (i.e. has not defined logical value) for a short period before stabilizing to a uniformly distributed value.

3 Stealthy HW Trojans and Unsuspicious Fault Patterns

FAs are powerful tools for performing key recovery attacks. Yet, they require that the adversary is able to induce faults with a certain level of accuracy. Depending on the targeted error model, this requires an initial phase of reverse engineering and a specific setup. When considering HTs, straightforward faults injection mechanisms can be implemented in the infected chip. However, due to their particular behaviors, these faults can be analyzed by verification algorithms in order to detect the malicious circuit modifications.

In this section, we propose a HT that induces faults and hides them so that they have no specific (i.e. detectable) patterns while still enabling SFA. First, we explain the objectives we are aiming for in order to insert faults in the context of so-called stealthy FA and how we fulfill these with an HT. Then, we describe the architecture of the latter and conclude with implementation results.

3.1 Stealthy FAs: Objectives

As mentioned in the background, FAs are possible to perform if the adversary knows some information about the model of the injected faults. More into the details, computation errors are defined by their locations (i.e. which bits in the state are targeted) and by their distributions. Considering the AES-128 as a target, various divide-and-conquer attacks exploiting different faults models and with different complexities have been proposed, as the DFA surveyed in [2] or the SFAs from [14,11]. Based on the literature, one may observe that fault(s) are usually injected at some point during the three last rounds in order to be exploitable (i.e. to provide easily exploitable information on the secret key). The accuracy of the fault locations depend on the considered attacks: some DFAs require a specific error pattern, such as one faulty bit per byte before the SubBytes of the 9th round in [17]; some are more liberal and can exploit one faulty byte per column before MixColumn of the 9th round or 1 byte before the MixColumn of the 8th round in [28]. The same holds for the faults distributions: these may vary from uniform to fixed-value distributions depending on the attacks.

Based on this state-of-the-art, an adversary may naturally take advantage of HTs in order to inject faults. However, depending on the running conditions of the chip, the latter may be expected to appear of not. As a result, the faults injected by a HT can seem suspicious to an evaluator if they present systematic exploitable patterns under specific environmental conditions. Following, the design goal of the HT adversary is to inject faults that do not imply detectable deterministic behaviors while still allowing to perform a FA.

3.2 Threat Model

We consider as an adversary a malicious manufacturer in charge of producing chips that implement the AES specifications. The latter is able to perform any kind of circuit modifications (i.e. functional and parametric) and aims to induce stealthy exploitable faults in the design. In practice, the proposed Trojan requires adding logic to a given core and to ensure that the critical path of the resulting implementation is the one of his choice. While this is a strong threat model, it can be introduced at many stages of the supply chain such as in malicious and obfuscated third-parties IPs, corrupted design houses or malicious foundries.

Once manufactured, the chips are deployed in some systems to which the adversary has physical access. It is assumed that he is able to tweak the system in order to control the clock frequency of the infected chip. Again, this is admittedly a strong threat model (which is usually the case with hardware Trojans). Yet, we note that is was already used in [13]. Besides, such a fine grain control is the same as when exploiting static power in side-channel analysis: in this case as well, the clock needs to be precisely controlled by replacing the external crystal with another waveform generator, as done in [25, 19, 5]. In general, claims about the practical-relevance of HTs are always difficult since there are limited public reports on actual exploits based on HTs. Yet, our view is that it is the multiplication of attack vectors (more than the concrete relevance of a single HT) that makes this threat critical. Hence, the investigation of such HTs, especially when able to bypass certain types of (here functional) verifications, are important for the understanding of this field of research.

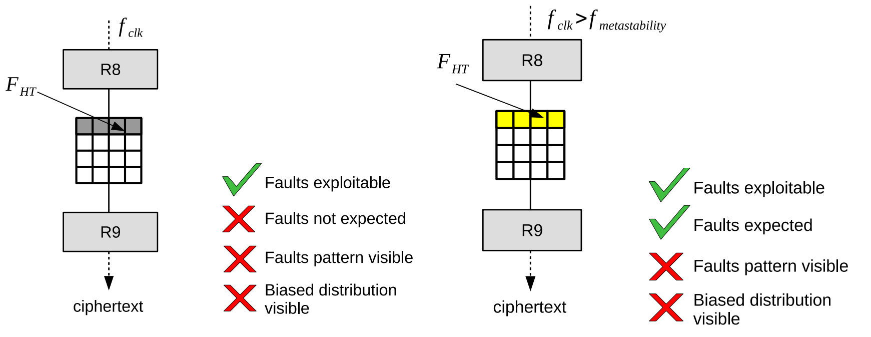
3.3 Trojan Principle

We now describe a new HT that injects both exploitable and unsuspicious faults in an AES implementation, and allows thus to perform a stealthy FA. We first present our main design choices and follow with the HT description.

Computation errors are rare under nominal running conditions. Any fault occurring in this context (as shown in Fig. 1a, faults that are not expected to appear in the state are depicted in grey) can be attributed to two different phenomenons: either because of a defective circuit or because of malicious modifications. In order to avoid trivial detection, the HT could first lead the chip to inject faults only when such non-nominal running conditions are detected, as depicted in Fig. 1b (the yellow depicts faulty states that are expected to appear considering the running conditions) . In this work, the non-nominal conditions are obtained by increasing the clock frequency above the maximum one. Yet, while some faults are expected to appear in such configuration, those that are induced could still be classified as suspicious by an evaluator because of recognizable patterns.

In order to improve the stealthiness of the HT, the faults must therefore appear as being random (i.e., uniformly distributed). For this purpose, the HT is composed of three main mechanisms.

First, uniformly distributed outputs are induced. This can be achieved by injecting random faults at random places of the full targeted internal state, as depicted in Fig. 2a (in orange, the faults are expected and follows a uniform distribution) . More practically, a delay is introduced on the paths of early rounds state computations (i.e. rounds that cannot be targeted with an efficient divide-and-conquer FA) in order to make these the critical path of the system. Because of the metastability effect and the confusion-diffusion paradigm, one may expect to observe uniformly distributed values for the internal states of the final rounds when the clock frequency is sufficiently increased.



(a) Nominal. (b) Non-nominal.

Fig. 1: Detectable fault injection under various environment conditions.

Second, keeping in mind that the errors should be exploitable, additional faults following biased distributions and targeting some specific bits are injected in an exploitable round state. The practical locations of the latter are the first bits of each column of the state before the Mixcolumn operation of the 9th round (1 bit per column, so 4-bits out of 128 in total) such that a divide-and-conquer attack by parts of 32 bits is possible . The stuck-at-0 fault model (i.e. unique value is 0) enjoys the benefits of a significant distribution bias and is thus used in order to allow performing an efficient SFA.

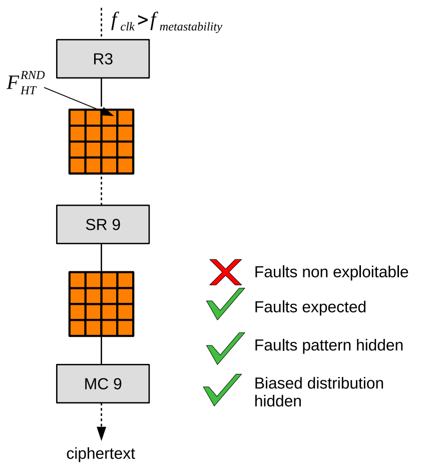
Finally, to ensure the stealthiness of these biased bits, a linear combination of some random bits known by the adversary is added (i.e. XOR-ed) to these in order to keep their distributions uniform. Bits from the previously computed ciphertext are used in practice, as shown in Fig. 2b (in blue, the fault are biased and not expected. In brown, the fault are expected, follows a uniform distribution from a non-advised evaluator point of view, but a biased distribution from the adversary point of view).

These can be assumed being random because of the random behavior of the internal state before the 9th round. Since the outcome of a XOR between 0 and a bit is the bit, we directly use the Boolean hiding bits computed as the faulty values injected.

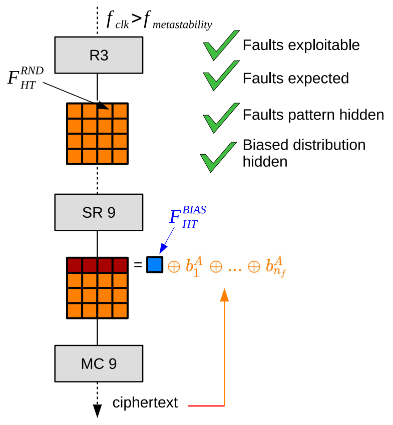
Additionally, these injections are only performed if random faults in the early rounds effectively appear. If it was not the case, an evaluator could recover the specific location of the fault as well as the biased distribution of it. For this purpose, a specific mechanism checking for fault injections in early rounds is used to compute a trigger signal that is fed to the HT logic. In this context, the faulty behavior of the chip seems unsuspicious while it allows an informed adversary to perform a SFA by increasing to clock frequency of the chip and by using different faulty ciphertext computed on-the-fly.

3.4 Practical Implementation

The HT architecture presented here holds for an unrolled implementation of the AES. Nevertheless, these results can be easily extended to other architectures (e.g. loop) possibly at the cost of limited additional logic. The HT is divided in four main parts: the injection of the random faults, the trigger mechanism, the injection of the biased faults and the linear addition. As seen on Fig. 3, the injection of the random faults and the trigger signal computation are closely linked. Delay is added over the logical paths at the end of an early (e.g. the 3rd) round and the corresponding delayed signals are fed to the input registers of the next (e.g. 4th) round. The delay introduction is however not restricted to an unique round and could be distributed over different early rounds in order to gain stealthiness. To detect if a random fault occurs, n_t extra registers are tapping the signals without delay. At the clock edge, the values of the initial and the extra registers are mutually compared using a XOR operation and any difference is detected by applying an OR operation on all the outputs of



(a) Not exploitable.



$f_{clk} > f_{metastability}$
R3
F_{HT}^{RND}
SR 9
F_{HT}^{BIAS}
$= \oplus\, b_1^A \oplus \dots \oplus b_{n_f}^A$
MC 9
ciphertext
Faults exploitable
Faults expected
Faults pattern hidden
Biased distribution hidden

(b) Exploitable.

Fig. 2: Hard-to-distinguish malicious faults.

the comparisons. The resulting trigger signal is then transferred to the 9th round using a dedicated register at each pipeline level.

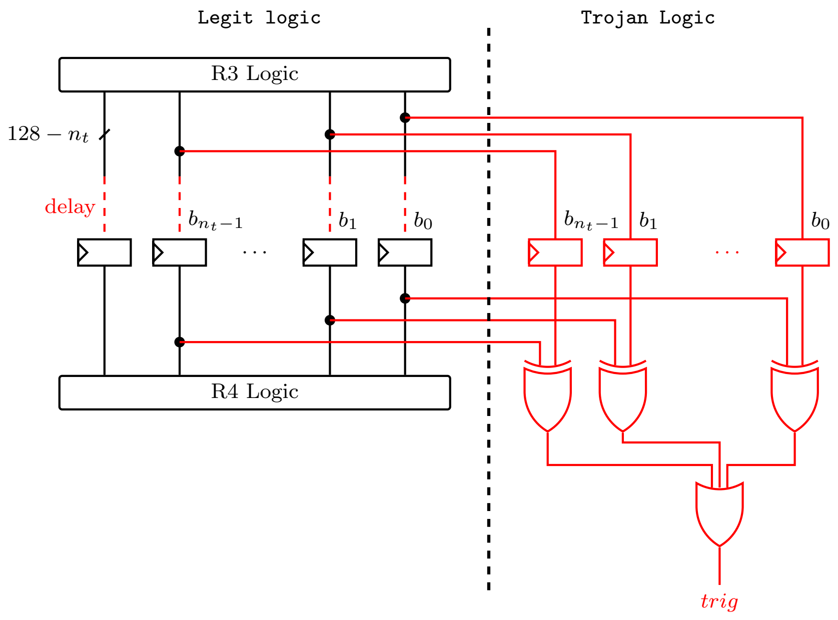


Fig. 3: Trigger and random faults injection mechanisms.

The second fault injection as well as the Boolean addition are controlled by the trigger signal. From an adversarial viewpoint, setting a bit to 0 and XOR-ing it with a value that is known (to the adversary) boils down to set the bit to this value. Taking into account such considerations, the biased fault injections are implemented with muxes controlled by the trigger signal in the 9th round, as depicted in Fig. 4. The faulty values provided to the muxes are finally computed by performing a XOR operation between n_f bits of the ciphertext whose positions are known by the adversary. It should be noted that, to ease the mechanism representation, Fig. 4 illustrates the malicious logic for 1 bit only (out of the 4).

In practice, four different linear combinations and four 1-bit muxes are used to implement the biased faults injections. The signal controlling the muxes is generated at early round and is propagated with 1 register per stage of the architecture pipeline.

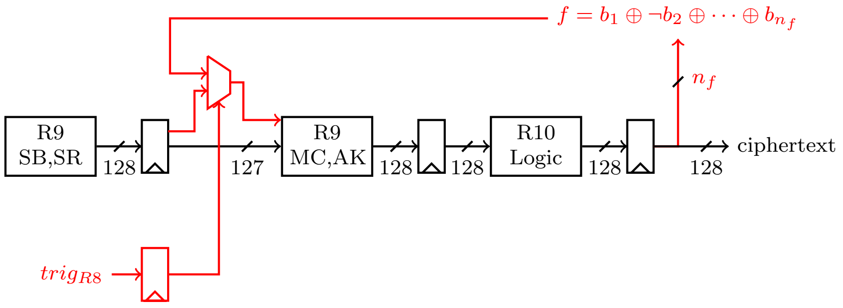


Fig. 4: Biased faults injection and hiding mechanisms. To ease the representation, the injection of only 1 faulty bit (out of 4) is represented.

Admittedly, the localized critical path requirement may be hard to achieve considering designs that have tight slack time regarding the targeted timing constraint. In such cases, ensuring that the critical path is properly located may requires a suspicious reduction of the maximal operating frequency. However, we argue that a malicious manufacturer could hide such frequency diminution behind process variations that are likely to have a significant impact for designs that reach the limits of what can be achieved with a targeted technology. In addition, such a malicious manufacturer could provide gate timing models that are deliberately devalued (i.e., slower than what he can achieve in practice), which would give him a certain degree of freedom to arrange paths delays within the design.

3.5 Trojan Stealthiness Properties and Implementation Results

Regarding the properties stated in the introduction, the following discussion holds for HTs that are based on the architecture described above. First, the adversary may benefit from different stealth-related features. The main one is that the faulty behavior of the chip is unsuspicious. As a reminder, faults are expected to appear under the trigger condition (i.e. a clock frequency increase). Moreover, these faults appear as being non-exploitable when trying to perform FAs. Indeed, non-biased behaviors of the targeted internal state are observed. In particular, faults appear to be uniformly distributed across all bits, as shown in Fig. 5. The same holds for the values of the latter that seem to follow a uniform distribution. It results that, to functionally discover the malicious nature of the computation errors, an informed evaluator has to recover the positions of the bits involved in the computation of the linear combination used to hide the biased faults. Although this can be done by trying all the possible combinations, the complexity of such a strategy equals to $c_{\mathcal{A}} = \binom{128}{n_f}$ and is represented in Fig. 6. An adversary trying to functionally hide the HT aims to maximize the latter. Considering that a complexity of 2^{64} is hard enough, the resulting parameter value $n_f = 16$ is used for the HT implementation. Under this configuration, it appears that the malicious circuitry can only be functionally detected by using time-consuming strategies, which motivates or even make the use of alternative methodologies mandatory. For example, side-channels' based detection mechanisms may be appropriate to get around the problem.

In this context, the footprint of the HT in the design is of paramount importance. We next show that it is more or less negligible depending on the size of the malicious circuitry inserted. First, the delay is free in term of logic gates. So the trigger and the Boolean hiding mechanisms are the most important parts to assess the impact of our HT on the total cost in comparison to a non-infected implementation. As shown in Table 1, the increased logic with $n_f = 16$ only goes up to 2.28%. Whether such a small modification can be detected with high probability based on the physical properties of the chip (i.e. the power consumption, the EM emissions, internal delays, . . .) is an interesting open problem. For that purpose, we note that the proposition [27] is an interesting research direction since its detection mechanism exploits the physical impact of the malicious logic on the internal propagation delays. However, we note that such detection-based approaches rely on the existence of a golden chip while our goal in this paper is primarily to escape Trojan detection based on functional testing and the qualitative guarantees it allows.

Table 1: Logic cost of the HT.

For $n_f = 16$	Honn.	$n_t = 4$	$n_t = 8$	$n_t = 16$	$n_t = 32$
Slice reg.	10370	10383	10387	10395	10411
LUTs	11116	11393	11588	11552	11421
Δ Slice reg.	0.0%	0.12%	0.16%	0.24%	0.39%
Δ LUTS	0.0%	2.49%	4.24%	3.92%	2.74%
Δ glob.	0.0%	1.35%	2.28%	2.14%	1.61%
For $n_f = 6$	$n_t = 1$	$n_t = 4$	$n_t = 8$	$n_t = 16$	$n_t = 32$
Slice reg.	10380	10383	10387	10395	10411
LUTs	11321	11417	11477	11358	11382
Δ Slice reg.	0.1%	0.12%	0.16%	0.24%	0.39%
Δ LUTS	1.84%	2.71%	3.25%	2.18%	2.39%
Δ glob.	1%	1.46%	1.76%	1.24%	1.42%

We note that in practice, unitary chip verifications with a high time complexity are unlikely to be performed due to the cost they generate. As a result, smaller parameters values can be used in order to reduce the HT impact on the side-channel leakages. For instance, using $n_f = 6$ allows to implement the HT using on average 0.37% less additional logic than the cases with $n_f = 16$, and still implies a time complexity of 2^{32} for the verification process.

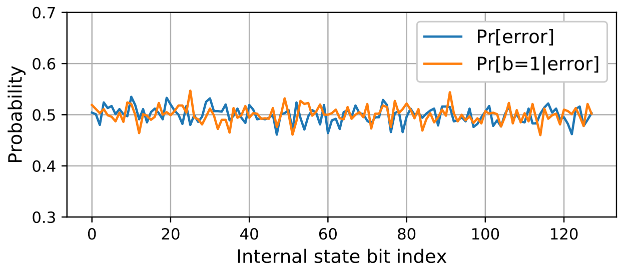


Fig. 5: Behavior of internal bits distributions.

Besides, the triggerability is straightforward: the adversary only has to increase the clock frequency until he observes computation errors appearing. The complexity of the resulting SFA depends on two things: the bias of the faults injected in the 9th round and the effective probability that the trigger signal is set to a high logical level in the presence of metastability. As shown in [24], the number of ciphertexts required to perform the attack is approximated by $1/\epsilon^2$ where ϵ is the bias from 1/2. The practical value of the latter depends on the true sampled distribution. Considering $n_t = 1$ register for the trigger detection mechanism, the trigger signal is set when a difference appears between the sampled signal value and its delayed version (which propagates to the rest of the logic, cfr Fig. 3). Under the non-nominal running condition (i.e., $f_{clk} > f_{metastability}$), any transition of the bit value will

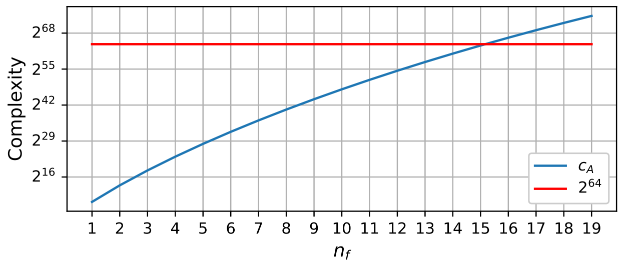


Fig. 6: Exhaustive detection complexity.

induce metastability, which has the effect of flipping the bit value with a probability of 0.5. Considering random inputs (and so a probability of transisiton equals to 0.5), the probability of setting the trigger signal equals 0.25. Considering n_t registers, the HT is not triggered when no difference are detected, which occurs with a probabilty equals to

$$\overline{\sigma_{n_t}} = (1 - 0.25)^{n_t} = 0.75^{n_t}.$$

Based on that, the probability to have the trigger signal set is given by:

$$\sigma_{n_t} = 1 - 0.75^{n_t}.$$

In this context, after the linear combination has been removed, the actual sampled probability distribution of each faulty value x injected at the 9th round is given by:

$$p(x) = \sigma_{n_t}\theta_x + (1 - \sigma_{n_t}) \cdot 0.5,$$

where θ_x is the biased distribution of x. Taking into account that $x = 0$ for the considered HT, the biases expected for different values of n_t are in Fig. 7.

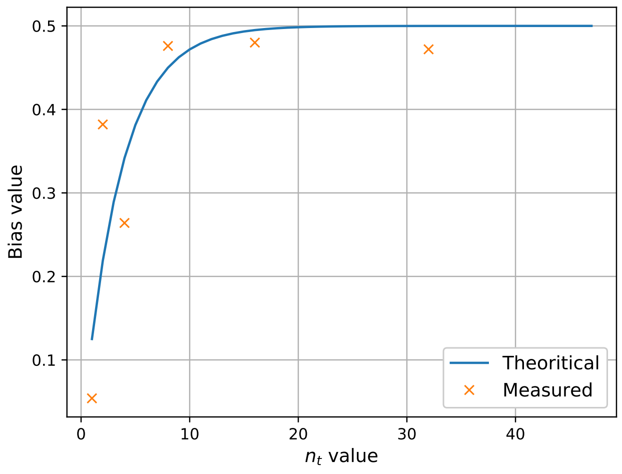


Fig. 7: Observed bias in term of n_t for stuck-at-0 faults injections.

3.6 Extenstion to Modes of Operations and Protected Hardware

The presented HT could be used against SPN-based block cipher other than the AES. Due to the confusion-diffusion paradigm, the random fault injection in an early round would remain useful to randomize the computed states. Regarding the exploitable fault injection, the practical amount and locations of such faults would depend on the diffusion properties of the targeted block cipher.

If used in the context of AEAD algorithms, an infected block cipher implementation can be used to perform a key recovery attack under the condition that the adversary has access to several ciphertext values computed by the infected core, with the same key. As explained above, it is not necessary for an attacker to know the value of the plaintext processed by the core. Such conditions are met for modes that do not rely on key dependant whitening (at the output only) and that keep a constant key value for all the block cipher executions. Practical examples of such modes are GCM, SUNDAE, CBC, CFB or OFB. On the contrary, it is intersesting to note that modes relying on re-keying mechanism (as adopted by leakage-resilient modes [6]) or that apply a key dependent whitening before releasing the encrypted data (e.g., OCB) are protected against the presented HT.

Hardware Trojans being implementation attacks, it is also interesting to note that our proposed Trojan could still apply to implementations embedding countermeasures against standard implementation attacks such as masking and double-execution.

As far as masking is considered, the only difference is that the biased fault injection mechanism has to be implemented at the sharing level instead of the bit level. In particular, the injection of the biased value known by the adversary (i.e., the signal f in Fig. 4) can be replicated to all the shares of the targeted bit. Considering d shares, this is performed with an additional cost of $(d-1)$ muxes compared to the unmasked implementation. As shown in the literature, the masking cost of linear operations (i.e., XORs) is linear with the amount of shares d while the non-linear operations (i.e., ANDs) have a cost proportional to d^2. These consideration taken into account, we can even argue that the relative impact of the malicious logic on the total cost will be reduced in the context of a masked implementation.

Double-execution would have a more positive impact since it may be complicated for an adversary to exploit the infected core in this case. Because of the random fault inserted at the early round of the target block cipher, it is unlikely that two executions result in the same output. In such a case, the countermeasure would prevent the release of the faulty ciphertexts, which would also prevent an informed adversary to perform a key recovery. We note that in such a situation, the manufacturer in charge assumed in our (admittedly strong) threat model could also add limited malicious logic in the countermeasure mechanism in order to circumvent it when needed. However, the countermeasure would at least avoid the exploit of the HT if another non-colluding manufacturer is in charge of its production.

4 Conclusion

In this paper, a new instance of HT against symmetric cryptosystems is presented. It allows performing SFAs against SPN-like block ciphers by stealthily injecting stuck-at faults (i.e. setting bits to a constant value) in the late rounds of the latter, with stealthiness is guaranteed in front of behavioral detection mechanisms. First, the faults are injected under specific non-nominal running conditions for which errors are expected to appear (i.e. after having increased the clock frequency sufficiently to observe metastability). Second, the exploitable faults (i.e. belonging to the localized stuck-at model) are hidden which makes these appear as unsuspicious (i.e. corresponding to random faults for each bit). This is achieved by injecting random faults in the early cipher rounds and by setting some bits of the last rounds to random values known by the adversary (i.e. Boolean linear combinations of some ciphertexts bits). The so induced random behavior of the computation errors and the wide range of possible linear combinations makes the detection of the malicious circuitry by an informed evaluator unlikely. Using combinations of 16 bits, the HT induces a detection complexity equals to $\approx 2^{64}$ for an informed evaluator, while allowing to recover a 128-bit key with a complexity of $\approx 2^{34}$. The 2.28% of additional logic need to implement this Trojan may still be detected by using accurate side-channel detection mechanisms. However, depending on the verification performed in the ICs manufacturing process, these impacts can be compensated by considering less aggressive design parameters. Besides, and overall, our main conclusion is that functional detection cannot be sufficient to ensure a Trojan-free implementation for symmetric cryptographic algorithms.

Acknowledgments

François-Xavier Standaert is Senior Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by EU and the Walloon Region through the ERC Project 724725 (SWORD) and the Wallinov TRUSTEYE project.

References

1. Aarestad, J., Acharyya, D., Rad, R.M., Plusquellic, J.: Detecting trojans through leakage current analysis using multiple supply pad $\mathrm{i_{ddq}}$ s. IEEE Trans. Information Forensics and Security **5**(4), 893–904 (2010). https://doi.org/10.1109/TIFS.2010.2061228, `https://doi.org/10.1109/TIFS.2010.2061228`
2. Ali, S., Mukhopadhyay, D., Tunstall, M.: Differential fault analysis of AES: towards reaching its limits. J. Cryptographic Engineering **3**(2), 73–97 (2013). https://doi.org/10.1007/s13389-012-0046-y, `https://doi.org/10.1007/s13389-012-0046-y`
3. Balasch, J., Gierlichs, B., Verbauwhede, I.: Electromagnetic circuit fingerprints for hardware trojan detection. pp. 246–251 (08 2015). https://doi.org/10.1109/ISEMC.2015.7256167
4. Beaumont, M.R., Hopkins, B.D., Newby, T.: Hardware trojans - prevention, detection, countermeasures (a literature review) (2011)
5. Bellizia, D., Bongiovanni, S., Monsurrò, P., Scotti, G., Trifiletti, A.: Univariate power analysis attacks exploiting static dissipation of nanometer CMOS VLSI circuits for cryptographic applications. IEEE Trans. Emerging Topics Comput. **5**(3), 329–339 (2017). https://doi.org/10.1109/TETC.2016.2563322, `https://doi.org/10.1109/TETC.2016.2563322`
6. Bellizia, D., Bronchain, O., Cassiers, G., Grosso, V., Guo, C., Momin, C., Pereira, O., Peters, T., Standaert, F.: Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 369–400. Springer (2020). https://doi.org/10.1007/978-3-030-56784-2_13, `https://doi.org/10.1007/978-3-030-56784-2_13`
7. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: CRYPTO (1997)
8. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: EUROCRYPT (1997)
9. Chakraborty, R.S., Narasimhan, S., Bhunia, S.: Hardware trojan: Threats and emerging solutions. In: IEEE International High Level Design Validation and Test Workshop, HLDVT 2009, San Francisco, CA, USA, 4-6 November 2009. pp. 166–171 (2009). https://doi.org/10.1109/HLDVT.2009.5340158, `https://doi.org/10.1109/HLDVT.2009.5340158`
10. Clavier, C.: Secret external encodings do not prevent transient fault analysis. In: CHES (2007)
11. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: Sifa: Exploiting ineffective fault inductions on symmetric cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**, 547–572 (2018)
12. Dziembowski, S., Faust, S., Standaert, F.: Private circuits III: hardware trojan-resilience via testing amplification. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 142–153 (2016). https://doi.org/10.1145/2976749.2978419, `https://doi.org/10.1145/2976749.2978419`
13. Ender, M., Ghandali, S., Moradi, A., Paar, C.: The first thorough side-channel hardware trojan. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 10624, pp. 755–780. Springer (2017)
14. Fuhr, T., Jaulmes, É., Lomné, V., Thillard, A.: Fault attacks on aes with faulty ciphertexts only. 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography pp. 108–118 (2013)
15. Ghalaty, N.F., Yuce, B., Taha, M.M.I., Schaumont, P.: Differential fault intensity analysis. In: Tria, A., Choi, D. (eds.) 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea, September 23, 2014. pp. 49–58. IEEE Computer Society (2014). https://doi.org/10.1109/FDTC.2014.15, `https://doi.org/10.1109/FDTC.2014.15`
16. Ghandali, S., Becker, G.T., Holcomb, D., Paar, C.: A design methodology for stealthy parametric trojans and its application to bug attacks. In: Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings. pp. 625–647 (2016). https://doi.org/10.1007/978-3-662-53140-2_30, `https://doi.org/10.1007/978-3-662-53140-2_30`

17. Giraud, C.: DFA on AES. In: Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers. pp. 27–41 (2004). https://doi.org/10.1007/11506447_4, https://doi.org/10.1007/11506447_4
18. Jin, Y., Kupp, N., Makris, Y.: Experiences in hardware trojan design and implementation. In: Tehranipoor, M., Plusquellic, J. (eds.) IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, San Francisco, CA, USA, July 27, 2009. Proceedings. pp. 50–57. IEEE Computer Society (2009). https://doi.org/10.1109/HST.2009.5224971, https://doi.org/10.1109/HST.2009.5224971
19. Karimi, N., Moos, T., Moradi, A.: Exploring the effect of device aging on static power analysis attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(3), 233–256 (2019). https://doi.org/10.13154/tches.v2019.i3.233-256, https://doi.org/10.13154/tches.v2019.i3.233-256
20. King, S.T., Tucek, J., Cozzie, A., Grier, C., Jiang, W., Zhou, Y.: Designing and implementing malicious hardware. In: First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings (2008), http://www.usenix.org/events/leet08/tech/full_papers/king/king.pdf
21. Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., Ohta, K.: Fault sensitivity analysis. In: Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings. pp. 320–334 (2010). https://doi.org/10.1007/978-3-642-15031-9_22, https://doi.org/10.1007/978-3-642-15031-9_22
22. Lin, L., Burleson, W.P., Paar, C.: MOLES: malicious off-chip leakage enabled by side-channels. In: Roychowdhury, J.S. (ed.) 2009 International Conference on Computer-Aided Design, ICCAD 2009, San Jose, CA, USA, November 2-5, 2009. pp. 117–122. ACM (2009). https://doi.org/10.1145/1687399.1687425, https://doi.org/10.1145/1687399.1687425
23. Lin, L., Kasper, M., Güneysu, T., Paar, C., Burleson, W.: Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In: Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings. pp. 382–395 (2009). https://doi.org/10.1007/978-3-642-04138-9_27, https://doi.org/10.1007/978-3-642-04138-9_27
24. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. pp. 386–397 (1993). https://doi.org/10.1007/3-540-48285-7_33, https://doi.org/10.1007/3-540-48285-7_33
25. Moos, T.: Static power SCA of sub-100 nm CMOS asics and the insecurity of masking schemes in low-noise environments. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(3), 202–232 (2019). https://doi.org/10.13154/tches.v2019.i3.202-232, https://doi.org/10.13154/tches.v2019.i3.202-232
26. Narasimhan, S., Du, D., Chakraborty, R., Paul, S., Wolff, F., Papachristou, C., Roy, K., Bhunia, S.: Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach. pp. 13 – 18 (07 2010). https://doi.org/10.1109/HST.2010.5513122
27. Ngo, X.T., Exurville, I., Bhasin, S., Danger, J., Guilley, S., Najm, Z., Rigaud, J., Robisson, B.: Hardware trojan detection by delay and electromagnetic measurements. In: Nebel, W., Atienza, D. (eds.) Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015. pp. 782–787. ACM (2015), http://dl.acm.org/citation.cfm?id=2755931
28. Piret, G., Quisquater, J.: 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. pp. 77–88 (2003). https://doi.org/10.1007/978-3-540-45238-6_7, https://doi.org/10.1007/978-3-540-45238-6_7
29. Robertson, J., Riley, M.: The big hack: How china used a tiny chip to infiltrate u.s. companies. Bloomberg (Oct 2018)
30. Tehranipoor, M., Koushanfar, F.: A survey of hardware trojan taxonomy and detection. Design and Test of Computers, IEEE **27**, 10 – 25 (03 2010). https://doi.org/10.1109/MDT.2010.7
31. Waksman, A., Sethumadhavan, S.: Silencing hardware backdoors. In: IEEE Symposium on Security and Privacy. pp. 49–63. IEEE Computer Society (2011)
32. Wang, X., Tehranipoor, M., Plusquellic, J.: Detecting malicious inclusions in secure hardware: Challenges and solutions. In: IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2008, Anaheim, CA, USA, June 9, 2008. Proceedings. pp. 15–19 (2008). https://doi.org/10.1109/HST.2008.4559039, https://doi.org/10.1109/HST.2008.4559039
33. Xiao, K., Zhang, X., Tehranipoor, M.M.: A clock sweeping technique for detecting hardware trojans impacting circuits delay. IEEE Design and Test **30**, 26–34 (2013)