Time-Modulated Hardware Trojans: Clock-Based and Interface-Based Examples

Momin Charles
UCLouvain
ICTEAM/ELEN/Crypto Group
Louvain-la-Neuve, Belgium
charles.momin@uclouvain.be

Bronchain Olivier
UCLouvain
ICTEAM/ELEN/Crypto Group
Louvain-la-Neuve, Belgium
olivier.bronchain@uclouvain.be

Standaert François-Xavier
UCLouvain
ICTEAM/ELEN/Crypto Group
Louvain-la-Neuve, Belgium
francois-xavier.standaert@uclouvain.be

Abstract*—Hardware Trojans are an important threat to the security of integrated circuits. They assume a malicious manufacturer able to infect implementations with hard-to-detect circuit modifications that can compromise their security. Hardware Trojans are sometimes classified as digital (if they are triggered and send their payload as regular outputs on a communication interface) or physical (if they are triggered and send their payload via a physical side-channel such as an EM signal). Typical examples of digital hardware Trojans are cheat codes, which are triggered under some rare input conditions, and time bombs, which are triggered when a counter internal to the implementation reaches some value. In this paper, we investigate a class of physical hardware Trojans that can trigger malicious circuitry thanks to a standard communication interface (as a digital hardware Trojan), by exploiting a timing side-channel. We denote these physical hardware Trojans as Time-Modulated, since they exploit the rhythm at which computations are performed, and provide two exemplary instances of such Trojans. The first one is clock-based: it exploits a recent idea of hardware Trojan using a side-channel by Ender et al. at ASIACRYPT 2017, and can inject an exploitable fault that applies to any AES implementation. The second one is interface-based: it exploits the delays between multiple message blocks as proposed by Shield et al. at AISC 2015. We extend this work to describe denial-of-service and key recovery attacks against a Trojan-resilient implementation designed following a recent proposal by Dziembowski et al. at CCS 2016. Despite the latter did only claim security against arbitrary digital hardware Trojans, our results show that limited additional (physical) capabilities allow an adversary to circumvent these formal security guarantees.

I. Introduction

The manufacturing of modern Integrated Circuits (ICs) is a complex and expensive process which has become increasingly globalized over the last 20 years. In this context, parties involved in the IC design and fabrication can be untrusted, which can possibly lead to malicious modifications of the circuit. These modifications, usually denoted as Hardware Trojans (HTs), can lead to devastating attacks against cryptographic and security-related implementations, as surveyed in [1]–[3] and illustrated by various examples [4], [5].

According to the taxonomy considered in [3], HTs can be classified based on their trigger mechanisms. On the one hand, they can lead an infected design to behave maliciously once it reaches a particular digital internal state. In such cases, they are known as being digitally-triggered. Typical instances are cheat codes and time bombs. The first ones refer to HTs triggered once the chip receives a specific value or sequence at its inputs [6]–[11]. The second ones refer to HTs triggered after a specific number of executions, such as exposed in [11]. On the other hand, analog signals (e.g., EM) can be used to implement trigger mechanisms [12]. Besides, HTs may not require to be triggered. Instead, they can operate continuously and are denoted “always-on" [13]–[15] in such cases.

HTs can also be classified according to their malicious behavior, usually called payload. As for the triggering mechanisms, the payload can be classified as digital or analog. Digital ones can for instance modify the content of memories or affect internal states [6]–[8]. Analog ones affect circuit parameters such as the delay, power or noise margin [9], [16].

In [1], the authors additionally use the physical characteristics of the HTs to classify them as functional or parametric. Functional ones require logic modifications by adding or removing logic gates [6]–[8], [17]. Parametric ones are implemented thanks to modifications of the physical properties of the existing logic or wiring [9], [16].

In this paper, we investigate a class of HTs that we denote as Time-Modulated (TM) Hardware Trojans. They are triggered by an analog signal (i.e., a timing side-channel) which is available over digital communication interfaces. This allows a straightforward malicious modification strategy while escaping any countermeasure that would only prevent digitally-triggered HTs. We illustrated this claim by describing two instances of TMHTs, using two different types of time modulation: one based on the clock signal, the other based on the I/O interface. We build on two previous works for this purpose.

Our first instance is based on a proposal published at ASIACRYPT 2017 [16] in which timing violations were used to bias random number generators and therefore break underlying assumption of the hardware protections used (i.e., masking). In this work, we show that the same mechanism can be used to introduce computational errors in implementations of the Advanced Encryption Standard (AES). The faults can then be exploited to recover the full encryption key with an efficient Differential Fault Analysis [18]. While such an exploit can theoretically be detected by an informed evaluator, it illustrates the wide range of mechanisms that are simple to deploy by

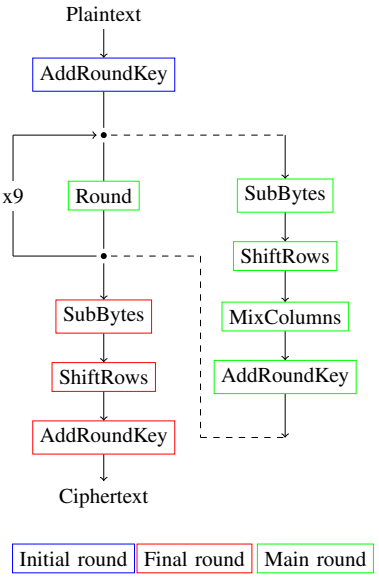

Fig. 1. AES Encryption Process

HT adversaries and therefore multiply the amount of defaults that should be systematically tested by hardware designers).

Our second instance exploits the triggering mechanism of [17] to circumvent the security guarantees provided by the Trojan-resilient architecture from [19]. While this last work ensures that the exploitation of any digital HT can only succeed with a small probability, our results show that very limited (admittedly physical) capabilities allow a concrete adversary to beat these security bounds with a limited amount of additional logic.

II. Clock-Based Time-Modulated HTs

In this section, a clock-based TMHT introducing faults within AES hardware implementation is presented. These last ones are intended to allow performing a DFA in order to recover the full encryption key. When trying to perform classical DFAs, the model and the location of the fault have a large impact on the required corrupted executions amount and post-processing complexity. By combining delay insertion and increased clock frequency as in [16], our TMHT can induce errors of a well defined model at a very accurate location and therefore leads to low cost/complexity attacks.

A. Background

Here, the AES specifications are shortly reminded. Then, the concept of DFA is presented. Finally, the timing constraints in standard IC designs are also reminded.

1) The Advanced Encryption Standard: The AES is a block cipher proposed in [20] and adopted by the NIST in 2001. It can operate on blocks of 128 bits with secret key of 128, 192 or 256 bits (this work will focus on the 128-bit version). As seen in Fig. 1, the encryption process involves 10 successive rounds, each performing 4 different operations over the successive encryption states represented by a 4×4 matrix of bytes.

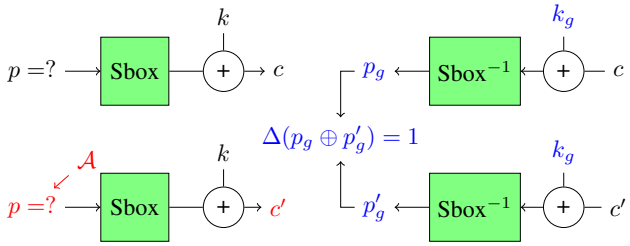

Fig. 2. 1-bit Fault Attack Example

These operations are defined at the byte level, with each byte representing an element of the finite field GF(2^8). As many modern block ciphers, the AES is built following the confusion-diffusion paradigm. In particular: (1) The ByteSub operation is a non-linear byte substitution of each byte of the state. (2) The ShiftRow operation is a cyclic shift of each row of the state, depending on the row position. (3) The MixColumn operation is a multiplication modulo $x^4 + 1$ over GF(2^8) between each column of the state (considered as a polynomial over GF(2^8)) and a fixed polynomial. (4) The AddRoundKey operation is an addition in GF(2^8) (i.e., a XOR operation) used to involve the key dependency.

2) Efficient DFA Against AES: First introduced in [21] in the context of asymmetric cryptosystems, faults attacks have been shown to be a security threat to many encryption or authentication algorithms. In [22], Biham and Shamir then specialized these attacks to almost any secret key encryption cryptosystems. To illustrate the DFA principle they proposed, one may consider a basic encryption scheme (represented in Fig. 2) composed of a layer of Sboxes and a key addition. This scheme takes as input a plaintext p, a key k and outputs the corresponding ciphertext c, where p,k,$c \in [0,1]^b$. This corresponds to the last operations of the last AES round. In these conditions, an adversary $\mathcal{A}$ tries to guess k only based on the value of c. Since $\mathcal{A}$ has no information about p, that leaves him 2^b key candidates. It is then considered that he is able to introduce an error in the flip-bit model (i.e. is able to flip a random bit). For a second encryption with k, he therefore induces a fault in p to produce the faulty plaintext p' before it is encrypted as the faulty ciphertext c'. After that, he makes a guess on the key value (depicted k_g) and decrypts both c and c' to obtain p_g and p'_g. Since he knows that p and p' only differ from one bit, the key k_g is a potential candidate if and only if p_g and p'_g differ from one bit, leaving him exactly b candidates. This attack reduces the size of the key space from 2^b to only b possibilities. The adversary can then conclude the attack by exploiting additional correct/faulty ciphertexts pairs (which reduces the amount of possible key candidates exponentially) and if necessary by testing the remaining candidates exhaustively. Considering that AES Sbox is 8-bit long, using two faulty ciphertexts is on average enough to be left with only the correct key when attacking the bits related to one Sbox.

In [18], Piret and Quisquater proposed an improved DFA

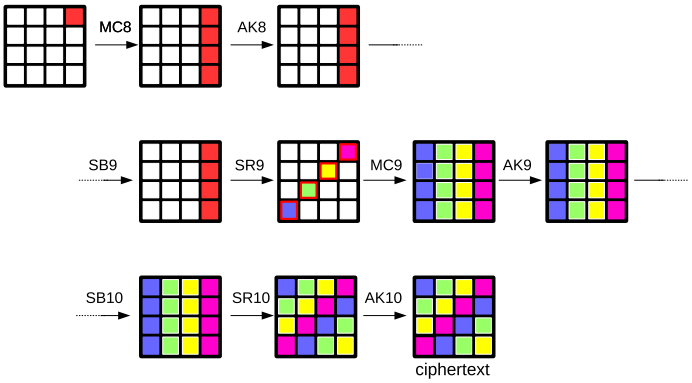

Fig. 3. Error Propagation Path in AES

against the AES, by improving the type and the amount of errors required to perform the attack. They showed that $\mathcal{A}$ can recover the 128-bit key of the AES based on only two correct/faulty ciphertext pairs. Each of them is obtained by introducing a single fault in the random byte error model (i.e., by replacing a byte by a random 8-bit long value). The attack exploits a random byte error appearing in the state before the MIXCOLUMN operation of the 9th round (depicted MC_9) in the AES. This leads to 4 random byte errors in the ciphertext value. By leveraging the linearity of the MIXCOLUMN operation, $\mathcal{A}$ is able to filter the key candidates (i.e., to reduce the key space) for the corresponding faulty bytes. As depicted in Fig. 3, a random byte error appearing before MC_8 then induces a random faulty byte in each column of the state in the 9th round and thus to 16 random byte errors in the ciphertext. By using 2 pairs of faulty/correct ciphertexts and by performing the filtering for both MC_8 and MC_9, $\mathcal{A}$ is on average left with the correct key candidate only. This method requires a time complexity $\approx 4 \times (4 \times \mathcal{O}(2^8)) = \mathcal{O}(2^{12})$ in order to recover the full 128-bit key.

3) Timing Constraints: In IC design, one aims to meet the correct specifications at a targeted clock frequency. This implies timing constraints that, if not fulfilled, may lead to signals' inconsistencies when the clock frequency exceeds some threshold. These constraints imply monitoring two main paths in synchronous designs:

- **The launch path** taken by the data going from the data launching register A to the data capturing register B at the clock edge.
- **The capture path** taken by the control clock signal from its source to the data capturing register B.

The signals' propagation over the launch path must be faster than the time between two positive edges of the clock. This time interval is defined as the association of the clock period T_c and the parasitic propagation delay of the clock signal over the capture path. The propagation over the launch path is composed of three main delays. The first one is the delay needed after the clock edge to guarantee that the output Q of A is stable. It is denoted as t_{pcq} which stands for Propagation from Clock to Q. The second one is the propagation delay of the combinatorial logic. It is represented by t_{pd} which stands for Propagation Delay. The third one is the time during which the data at the input D of B needs to remain stable. This allows registers to charge internal capacitive loads in order to have a stable signal at the clock edge. This delay is called the setup time and is denoted as t_{setup}. If the propagation over the capture path is considered as being ideal (i.e., with no delay) the constraint becomes:

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}, \tag{1}$$

or alternatively by defining the slack time t_{slack}:

$$t_{slack} = T_c - t_{pcq} - t_{pd} - t_{setup} \geq 0 \tag{2}$$

If the setup timing constraint is not fulfilled, the value captured by the data capturing register B is said to be metastable. The value captured by the register in such a case stays in the metastable state during a short random period of time before setting to a random logical value.

B. Threat Model

In the following, the considered adversary $\mathcal{A}$ is the same as [16], who is manufacturing a hardware implementation of the AES. He is only able to proceed to parametric modifications at or after the place-and-route. Therefore, he cannot add or remove logic gates. Once the IC is deployed and loaded with a secret key, he has a physical access to the chip and is able to modify the clock signal of it.

C. Trojan Implementation

When the setup timing constraint expressed in (2) is unmet, the data stability is not assured. This can be used by $\mathcal{A}$ in order to induce faults which correspond to the path delay fault model from [23]. In [9], the authors introduced the Path Delay HT (PDHT) class based on this model. By adding delays over some rarely sensitized paths, they are able to induce a fault by using a cheat code. They propose a method to insert the necessary delays using parametric modifications at the sub-transistor level. This results in the legitimate functionality, excepted for the targeted path that contains faults. Our TMHT follows the methodology proposed in [16], where the authors implement a PDHT which is triggered by increasing the clock frequency above the maximum operation frequency. Thanks to delay addition, the targeted paths become faulty before others. In the previous proposal, this kind of HT is used to induce non-uniform randomness in a masked implementation of the PRESENT block cipher which then becomes weak(er) against side-channel attacks. The following shows it can be used in order to induce faults in any (e.g., unprotected) AES architecture, which are then exploited via a DFA to recover the 128-bit AES key.

1) Basic Principle: In order to recover the key, $\mathcal{A}$ aims to induce a fault in the state processed in the 8th round of the AES. For this purpose, he adds delay over a path in the 7th round and may expect to obtain a fault when the clock frequency f_{clk} reaches a given threshold. At this stage, $\mathcal{A}$ has to obtain two faulty/correct ciphertext pairs to perform a DFA.

2) Delay Insertion: The adversary is restricted to layout modifications. As proof of concept, the proposed TMHT is implemented on a FPGA with which parametric modifications are impossible to perform. Instead, the delay is introduced using routing modifications over the path of the 28th bit of the state in the 7th round of the AES architecture. This is done by manually rerouting the signal through switch boxes using the FPGA editor tool from Xilinx. As the induced fault should allow to perform a DFA, it must be precisely localized. Attention must thus be paid by $\mathcal{A}$ when he adds delay over the targeted path in order to obtain that the latter is the only one resulting in a fault.

By attacking one bit, it is expected to observe an error with a probability of 50%. Since $\mathcal{A}$ has a physical access to the design, he can proceed to multiple encryptions in order to collect the 2 required faulty ciphertexts.

3) Implementation Results: The setup used for the practical implementation of the TMHT is an AES core implemented in a Virtex6 FPGA on a ML605 Evaluation Board. The clock is generated with a controllable external signal generator and fed to the AES core through a SMA connector. The synthesis results of the unaltered AES core shows that the delay over the targeted path is ≈ 1.8ns. Moreover, the latter is not involved in the critical path, over which the propagation delay is ≈ 7.25ns, resulting in a slack time of 42.75ns for a targeted clock frequency of 20MHz. The impact of the HT insertion is shown in Fig. 4 and results in a significant increase of the propagation delay over the targeted path. More into the details, an increase of 38.13ns is obtained. It results that the target becomes the critical path with a total propagation delay of 39.96ns. The corresponding slack time is 10.04ns which is significantly smaller than the slack time of the second critical path which is equals 42.75ns. Considering this new configuration, one may expect to observe faults for $f_{clk} > 25$MHz.

In practice, the trigger frequency was measured at 56.25MHz, which is higher than the predictions. We assume that this variation comes from the timing models of Xilinx. These may include error margins which can distort the delays approximations. As shown in Fig. 5, the error probability converges to a uniform distribution for $f_{clk} \geq 58$MHz, which is in line with theoretical expectations.

By exploiting two pairs of faulty/correct ciphertexts, the full encryption key is recovered in less than 1 second on a desktop computer. This makes the attack very efficient once the TMHT has been introduced compared to usual DFA setups. Indeed, the faulty ciphertexts are obtained trivially by increasing the clock frequency during all the encryptions processes. In contrasts, classical DFA setups (e.g., with laser fault injection) usually require long reverse engineering phase

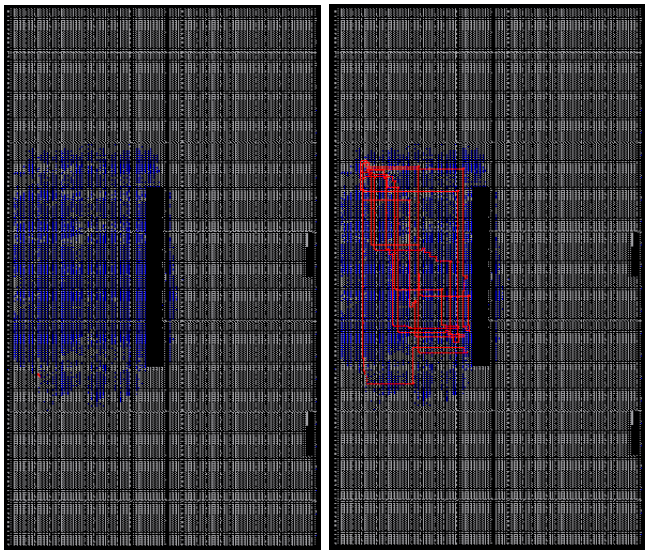

Fig. 4. Routing Modification (from FPGA editor)

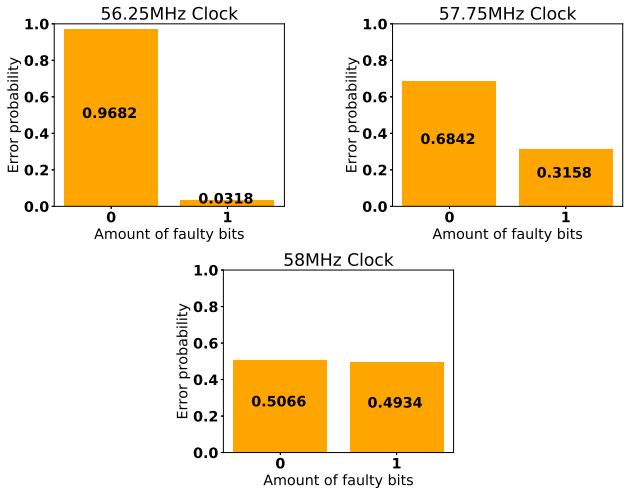

Fig. 5. Observed Error Probability

to introduced the desired faults. This has the advantage of drastically reducing the time required for the adversary to access the target device. Extension to ASIC requires other mechanisms to properly embed this kind of TMHT and is left as an interesting open problem.

III. Interface-Based Time-Modulated HTs

In this section, a second TMHT is proposed. It differs from the one described in Section II both by the interface used for the trigger mechanism and by its insertion method. First, it is triggered through the I/O of the chip, while the previous is triggered thanks to a modulation of the clock signal. Second, it is functional (i.e., requires logic insertion) while the clock-based one is implemented using parametric modifications. In the following, the implementation of the Trojans-resilient circuits of Dziembowski et al. is recalled and a description of a TMHT against the latter is proposed.

A. Background

Here are summed up the theoretical foundations of the HT resilient architecture proposed in [19], as well as the description of a practical prototype [24].

1) CCS compiler: In [19], the authors propose a generic compiler TR that maps any deterministic specification Γ of a circuit to an architecture resilient to any digital HT built by a PPT adversary $\mathcal{A}$. As depicted in Fig. 6, the architecture is composed by untrusted devices, produced by a single or multiple colluding manufacturers, and by a unique trusted device that is supposed to be small. The untrusted devices are first tested up to t times. Afterwards, these are used n times by a user during a so-called online phase. While these online runs are performed, the probability that the overall construction deviates from the correct specifications (i.e., the robustness) is bounded. More in details, the architecture is split in three components:

- **The Sub-circuits** denoted as D^j implement λ independent executions of a passively secure three-party computation protocol. On input x, a sub-circuit computes $\Gamma(x)$ based on shares of x where each of these shares do not reveal individually any information about x.
- **The Mini-circuits** denoted as D_i^j are independent untrusted circuits such that (D_0^j, D_1^j, D_2^j) correspond to the sub-circuit D^j.
- **The Master** next denoted as M is the only trusted part of the entire design. It is responsible for sharing the input x, routing information between the mini-circuits and performing a majority vote among the λ sub-circuits. In order to easily assure that M can be trusted, it carries out only simple operations and its size is independent of the specifications Γ.

Once the mini-circuits are built, a testing phase is performed. The latter consists in the verification of the input/output behavior of the untrusted mini-circuits. Each of these are checked a random number of time $t' \leftarrow 0, \ldots, t$ where t is maximum number of tests. Thanks to these, the probability that a tested mini-circuit interacting with random inputs differs from its specifications during the n next executions (for $n \leq t$) is bounded by:

$$\Pr[D_i^j(x) \neq \Gamma_i^j(x)] \leq \frac{n}{t}. \tag{3}$$

In order to avoid any direct interaction between the mini-circuits and $\mathcal{A}$, the data they deal with are randomized. More into the details, each of these receives a share of the adversary's inputs x. Considering the sharing mechanism, these cannot recover x and HTs based on cheat codes are avoided. Additionally, thanks to the random number of tests performed on each sub-circuits, the untrusted circuits cannot synchronize in order to differ simultaneously from their specifications on the n^{th} run. The trigger mechanisms based on time bombs are thus also prevented.

The protection scheme exploits test amplification. Thanks to the redundancy of λ sub-circuits, the output of the HT

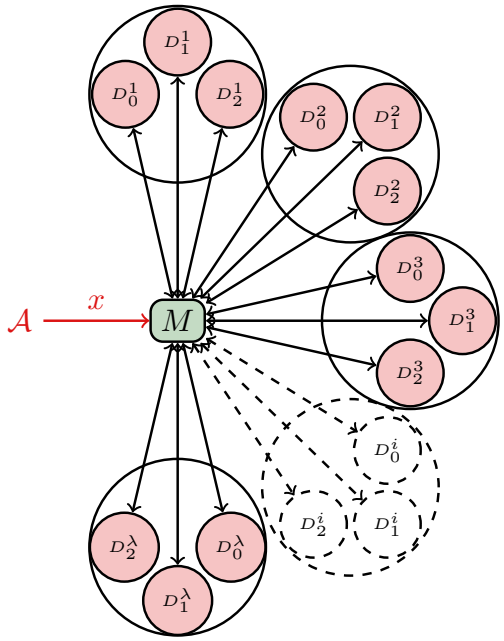

Fig. 6. HT Resilient Framework Architecture. Green denotes the trusted component while red the untrusted ones.

resilient design outputs incorrect values if at least $\lambda/2$ of the sub-circuits deviate from their specifications. In [19], Dziembowski et al. proved that the robustness bound of the complete architecture is given by

$$\Pr[Rob = 1] \leq \left(\frac{n}{t}\right)^{\lambda/2} \tag{4}$$

This solution relies on a three-party computation protocol secure against passive adversary since the parties are ensured to follow their specifications for n online executions. This is a clear improvement compared to other countermeasures based on multi-party computation protocol that require actively secure computations and non-colluding manufacturers [25]. Indeed, ensuring security against active adversaries implies a higher communication cost, which is a critical factor for implementation cost.

2) Block cipher implementation: In [24], the authors described an implementation of the protection scheme of Dziembowski et al. dedicated to a block-cipher case-study. They proposed to use the passive three-party computation protocol from [26] which allows to reach a 55Mbps encryption throughput for the AES with one order of magnitude smaller trusted area than an unprotected implementation.

More into the details, an online run (i.e., an encryption) considering this architecture starts on the reception of a plaintext p. The latter is shared by the master M and sent to the the mini-circuits D_i^j. Each of these then holds one share of p and has thus no information on its value. In a similar way, mini-circuits also hold shares of the secret key k. Then, the λ sub-circuits run the block cipher specifications Γ by following the three-party computation protocol and obtain the shares of the ciphertext $\Gamma_k(p) = c$. The encryption ends with

the transfer of the ciphertext shares to the master M. The later reconstructs c based on these and performs the majority vote amongst the reconstructed values of each sub-circuits. In such a case, the outputted value is the expected ciphertext c with a probability given in (4).

B. Threat Model

In the following, a PPT adversary $\mathcal{A}$ compliant with the specifications of the previously described HT protection scheme is considered. As allowed in [19], the adversary controls a single foundry which is responsible for building all the untrusted mini-circuits of any system based on the described architecture. So he can insert any digital logic inside these circuits with no restriction.

After the testing phase, $\mathcal{A}$ has only a remote access to the architecture, and can request n outputs corresponding to inputs of his choice.

C. Trojan Implementation

Here is described a TMHT architecture against the previously described HT-resilient protection scheme. As proof of concept, the HT-resilient AES implementation in [24] is considered and a key extraction mechanism based on the TMHT is presented. The later could however be used with multiple different aims since the real challenge when considering the generic compiler of [19] is to efficiently trigger an inserted HT. Once done, various malicious behaviors can be implemented in any protected system. Due to its analog triggering mechanism, the proposed TMHT is not formally considered in the threat model of [27] (on which [19] is based), where the authors focus on cheat codes and time-bombs. We show however that a TMHT should be considered as a threat, since the trigger mechanism can be implemented with a limited amount of resources while being exponentially difficult to detect functionally. Additionally, the insertion mechanism and the interface used by potential attacks are the same as the one considered in [27] and [19].

1) Principle: The robustness bounds claimed in (4) hold n parts for HTs exploiting digital inputs thanks to secret sharing. Hereunder, the proposed TMHT relies on sensing an analog signal (i.e., a time interval) with malicious digital logic inside the sub-circuits. In short, the adversary can directly communicate with the D^j's by using a modulation of the interface usage: the time interval t_i between successive executions of the three-party computation protocol is chosen by the adversary and can be measured by the D^j's to retrieve information from $\mathcal{A}$.

To do so, the adversary selects a random secret value $s \leftarrow \{0,1\}^b$ and a public value T. These two values are secretly hidden within all the mini-circuits. In s, each bit corresponds to a timing difference between two encryptions. More precisely, a low bit corresponds to an expected $t_i \geq T$ and high bit to $t_i < T$. By inserting the secret logic able to compare t_i to T, the adversary builds a half duplex communication channel between itself and all the mini-circuits. Next, two so-called modulation levels are therefore considered: slow if $t_i \geq T$ and fast if $t_i < T$.

Based on this communication channel, $\mathcal{A}$ is able to trigger malicious behaviors in all the mini-circuits, by modulating the encryption requests according to the sequence s known by the mini-circuits. Afterwards, all the λ sub-circuits can deviate from their specifications and so force the majority vote to exfiltrate a secret shared key.

In the previous work [17], authors proposed a similar triggering mechanism in order to perform a denial-of-service on a communication network thanks to an infected Ethernet controller chip. In our case, we generalize the threat by making remote attack against the HT-resilient generic compiler possible, potentially leading to denial-of-service as well as key recovery or any kind of misbehavior.

2) Trigger insertion: The trigger mechanism described next is inserted in each mini-circuits. It is split in 3 parts:

- **Encryption request detection**: The TMHT has to detect each new encryption request. For this purpose, active control signals such as an input validity signal (denoted as v_{in}) can be used.
- **Modulation level detection**: In addition, it has to detect the modulation level. For this purpose, a counter is used to keep traces of the clock cycles amount that occurred between two requests (as shown in Fig. 7). The modulation level (depicted as m_l) is computed by comparing the counter value to a hard-coded threshold. To avoid substantial comparison logic, the chosen threshold value T_h is a power of two: in allows to manage the comparison process by checking the value of a single bit. Moreover, a feedback loop is used to manage counter overflows.
- **Modulation sequence detection**: Based on the two first mechanisms, it finally needs to detect a sequence of modulation levels. For this purpose, s is hard-coded in a shift register (as seen in Fig. 8). Each time a new plaintext arrival is detected, the value m_l is compared with the MSB of the shift register. Depending on the similarities between both, s is shifted or reset to its initial value. If the register value reaches the value 0, the trigger signal F_{trig} is set and the HT delivers its malicious payload.

Even if the adversary considered in [19] has no logic restriction, the TMHT should be implemented with a minimal number of gates in the mini-circuits. This low number of inserted logic elements makes the detection of malicious behavior increasingly difficult with the size of the infected circuit, as discussed in Section IV.

3) Implementation results: The setup used for the practical implementation of the TMHT is the HT-resilient AES from [24] implemented in a Virtex6 FPGA on a ML605 Evaluation Board. A clock signal of 66MHz is provided by an oscillator embedded on the board and is fed to the AES core. For an arbitrarily chosen value T, the corresponding value T_h is the nearest power of two that can be reached taking into account the clock frequency. For the case considered $T = 0.5$s

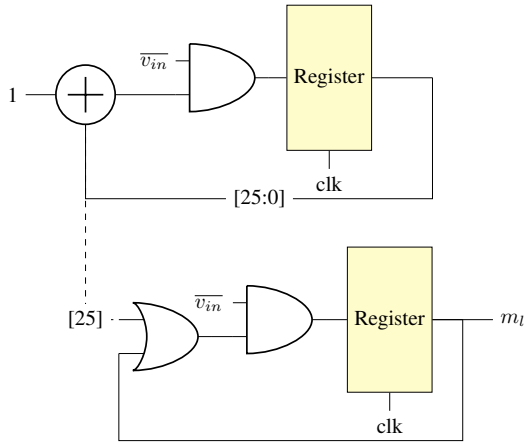

Fig. 7. Modulation Level Detection Mechanism

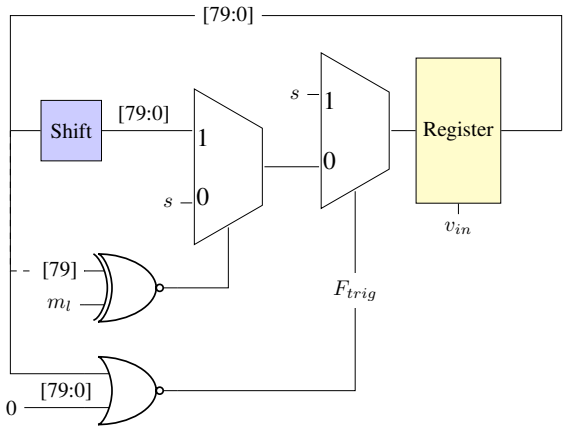

Fig. 8. Modulation Sequence Detection Mechanism

and $T_h = 2^{25}$, which corresponds to a practical value for T of 0.508s. The synthesis report of a mini-circuit shows that the trigger mechanism is inserted with 131 additional slice units, representing 3.8% of the honest architecture. The cost in term of gates would remain identical for bigger architecture, which would make the proportion of malicious design area negligible when the circuit size increases, while keeping the malicious trigger functionality.

Since all mini-circuits are infected, the trigger signal F_{trig} is raised synchronously in each of these. Mounting an attack then consists in choosing in what way they will act after being triggered. In the following, two malicious behaviors breaking the robustness guarantees are proposed.

First, we show that the proposed TMHT allows to output an incorrect value instead of the expected output $\Gamma(p) = c$. More precisely, we highlight the possibility of performing a key extraction attack against the AES Trojan-resilient implementation from [24]. To do so, once F_{trig} is set, the mini-circuits continue to work properly for the entire encryption and so compute the correct value for the ciphertext shares. However, instead of outputting these shares, they output the shares of the key. By doing so, the trusted master M will reconstruct the secret key value k instead of the correct ciphertext c. In addition, the majority among all the sub-circuits will operate in the same way and M will thus output the value k as ciphertext.

TABLE I
HT IMPLEMENTATION IMPACT

	Honest Design	Malicious Design	HT Cost
Slice Register	803	879	9.4%
Slice LUTs	2609	2664	2.1%
Global	3412	3543	3.8%

Secondly, one of the claims in [19] is that this hardware HT-resilient solution is the only solution providing protection against a denial-of-service attack. However, by exploiting this new communication channel (which admittedly deviates from the assumptions of Dziemboswki et al.), the mini-circuits can deviate from their specifications by not responding. In this case, the majority vote does not have any input leading to a denial of service.

IV. CONCLUDING REMARKS

The previous sections showed that TMHTs can be easily instantiated by exploiting manipulations of the clock or of the communication interface. In this section, we conclude these investigations by discussing the stealthiness of these proposals in front of an evaluator, their consequences regarding the need of countermeasures, and some open challenges raised by our results. For this purpose, we consider that the evaluator tries to detect the malicious functionality before the device is deployed. To do so, he can either rely on input/output verification or using empirical inspection [28]–[31].

As far as our clock-based TMHT from Section II is concerned, it does not relies on additional logic gates compared to an honest implementation. Therefore, empirical inspection would hardly distinguish between an honest and a malicious IC, as pointed out in [9], [16]. By contrast, by increasing the clock frequency, the evaluator can detect the fault and recover its location. In this respect, we note that since the fault is under control of the malicious manufacturer, any fault model could be exploited (and some unusual fault models may look less suspicious to uninformed evaluators). Yet, it remains that the faults we exploit have a specific location (i.e., always on the same bit at the 8th round of the AES) which is suspicious enough to consider a risk of malicious circuitry. On the one hand, as mentioned in the introduction, this implies the TMHT is detectable. On the other hand, this verification implies an additional test phase that may impact the time for testing ICs (in particular because this is just one more example among a plethora of side-effects that can lead to malicious behaviors). Overall, it suggests the investigation of clock-based TMHTs with better stealthiness as an interesting open problem.

As far as the interface-based TMHT from Section III is concerned, it brings a complementary conclusion. In this case, only empirical inspection can be used (since the exponential

number of trigger sequences to test makes functional verification unlikely to succeed). In this respect, our main conclusion is that this TMHT only requires 3.8% of additional logic gates compared to an honest implementation. Therefore, such an empirical inspection may turn out to be challenging in practice (despite not impossible [30]). So this second instance of TMHT suggests the improvement of the Trojan-resilient compiler of Dziembowski et al. in order to consider the risk of TMHTs as another interesting scope for further research.

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] X. Wang, M. Tehranipoor, and J. Plusquellic, "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.

[2] F. G. Wolff, C. A. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008*, pp. 1362–1365, 2008.

[3] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "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.

[4] S. Adee, "The hunt for the kill switch," *iEEE SpEctrum*, vol. 45, no. 5, p. 32, 2008.

[5] E. Biham, Y. Carmeli, and A. Shamir, "Bug attacks," in *CRYPTO*, vol. 5157 of *Lecture Notes in Computer Science*, pp. 221–240, Springer, 2008.

[6] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "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.

[7] Y. Jin, M. Maniatakos, and Y. Makris, "Exposing vulnerabilities of untrusted computing platforms," in *30th International IEEE Conference on Computer Design, ICCD 2012, Montreal, QC, Canada, September 30 - Oct. 3, 2012*, pp. 131–134, 2012.

[8] B. D. Hopkins, J. Shield, and C. J. North, "Redirecting DRAM memory pages: Examining the threat of system memory hardware trojans," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, May 3-5, 2016*, pp. 197–202, 2016.

[9] S. Ghandali, G. T. Becker, D. Holcomb, and C. Paar, "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.

[10] Y. Jin, N. Kupp, and Y. Makris, "Experiences in hardware trojan design and implementation," in *IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, San Francisco, CA, USA, July 27, 2009. Proceedings*, pp. 50–57, 2009.

[11] M. Jalalitabar, M. Valero, and A. G. Bourgeois, "Demonstrating the threat of hardware trojans in wireless sensor networks," in *24th International Conference on Computer Communication and Networks, ICCCN 2015, Las Vegas, NV, USA, August 3-6, 2015*, pp. 1–8, 2015.

[12] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *High Level Design Validation and Test Workshop, 2009. HLDVT 2009. IEEE International*, pp. 166–171, IEEE, 2009.

[13] L. Lin, W. Burleson, and C. Paar, "MOLES: malicious off-chip leakage enabled by side-channels," in *2009 International Conference on Computer-Aided Design, ICCAD 2009, San Jose, CA, USA, November 2-5, 2009*, pp. 117–122, 2009.

[14] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson, "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.

[15] A. Baumgarten, M. Steffen, M. Clausman, and J. Zambreno, "A case study in hardware trojan design and implementation," *Int. J. Inf. Sec.*, vol. 10, no. 1, pp. 1–14, 2011.

[16] M. Ender, S. Ghandali, A. Moradi, and C. Paar, "The first thorough side-channel hardware trojan," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pp. 755–780, 2017.

[17] J. Shield, B. D. Hopkins, M. R. Beaumont, and C. J. North, "Hardware trojans - A systemic threat," in *13th Australasian Information Security Conference, AISC 2015, Sydney, Australia, January 2015*, pp. 45–51, 2015.

[18] G. Piret and J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," in *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, pp. 77–88, 2003.

[19] S. Dziembowski, S. Faust, and F. Standaert, "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* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 142–153, ACM, 2016.

[20] J. Daemen and V. Rijmen, "The block cipher rijndael," in *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, pp. 277–284, 1998.

[21] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults (extended abstract)," in *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pp. 37–51, 1997.

[22] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pp. 513–525, 1997.

[23] G. L. Smith, "Model for delay faults based upon paths," in *Proceedings International Test Conference 1985, Philadelphia, PA, USA, November 1985*, pp. 342–351, 1985.

[24] O. Bronchain, L. Dassy, S. Faust, and F. Standaert, "Implementing trojan-resilient hardware from (mostly) untrusted components designed by colluding manufacturers," in *ASHES@CCS*, pp. 1–10, ACM, 2018.

[25] V. Mavroudis, A. Cerulli, P. Svenda, D. Cvrcek, D. Klinec, and G. Danezis, "A touch of evil: High-assurance cryptographic hardware from untrusted components," in *CCS*, pp. 1583–1600, ACM, 2017.

[26] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016* (E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds.), pp. 805–817, ACM, 2016.

[27] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," in *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pp. 49–63, 2011.

[28] O. Soll, T. Korak, M. Muehlberghuber, and M. Hutter, "Em-based detection of hardware trojans on fpgas," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014*, pp. 84–87, 2014.

[29] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection," in *2015 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, IEEE, aug 2015.

[30] S. Narasimhan, R. S. Chakraborty, D. Du, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia, "Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach," in *HOST*, pp. 13–18, IEEE Computer Society, 2010.

[31] J. Aarestad, D. Acharyya, R. M. Rad, and J. Plusquellic, "Detecting trojans through leakage current analysis using multiple supply pad i_{ddq} s," *IEEE Trans. Information Forensics and Security*, vol. 5, no. 4, pp. 893–904, 2010.