D3.1

Report on Protocol choice and implementation

Reference number:	317930
Project acronym:	HINT
Project title:	HINT: Holistic Approaches for Integrity of ICT-Systems
Start date of the project:	1st October, 2012
Duration:	36 months
Programme:	FP7/2007-2013

Deliverable type:	Report
Deliverable reference number:	ICT-317930 / D3.1 / 1.0
Work package contributing to the deliverable:	WP 3
Due date:	July 31 - M22
Actual submission date:	4th August, 2014

Responsible organisation:	ARMINES-ENSMSE
Editor:	Jean-Baptiste Rigaud
Dissemination level:	Public
Revision:	1.0 (r-2)

Abstract:	Based on the requirements & constraints defined in WP1, we first describe the different Hardware Trojan (HT) detection schemes that are studied in the HINT project. Then the security risks linked to those schemes are analysed and the corresponding countermeasures are specified. Even though most proposed countermeasures are based on existing proven schemes, to our best knowledge, such a security analysis of HT detection schemes from a protocol level is a first of its kind.
Keywords:	Hardware Trojan detection, Active Checking, Integrity, Protocol

This project has received funding from the European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreement no 317930.

Editor
Jean-Baptiste Rigaud (ARMINES-ENSMSE)

Contributors
Jacques Fournier (CEA)
Jean-Baptiste Rigaud (ARMINES-ENSMSE)
Dave Singelee, Benedikt Gierlichs, Oscar Reparaz(KUL)
Julien Francq (CCS)
Thomas Hübner (MORPHO)

Executive Summary

In this deliverable, we first describe the different Hardware Trojan (HT) detection schemes that are studied in the HINT project: those schemes have been defined based on the requirements & constraints defined in WP1. Then the security risks linked to those schemes are analysed and the corresponding countermeasures are specified. Our analysis shows that for security reasons, when deploying those HT detection schemes, mutual authentication and data signature schemes have to be implemented. For doing so, we chose already existing and proven schemes, as there was no need to devise anything fancier & we describe how those schemes can be implemented. Finally the HT detection schemes are redefined to embed those countermeasures. To our best knowledge, such a security analysis of HT detection schemes from a protocol level is a first of its kind (c.f. the state-of-the-art review provided at the beginning of this document), taking into account the practical security issues that could be raised in the context of the use cases defined in WP1.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

In this document, we focus on protocols used for implementing the HT detection mechanisms studied in WP3. When looking back at the literature on HT detection, one can observe that the work is mostly focussed on the measurement and decision-making techniques used and not on the sequence of operations and procedures (what we refer to in this project as the *protocol*) that would have to be followed in order to put those techniques into practice: the only exception, to our best knowledge, to that is in [19, 20] where the authors propose a sequence for doing things. However they do not go further than that, i.e. look at whether those schemes are applicable in an industrial environment or if they introduce any security vulnerabilities. In the HINT project, our objective being to propose secure, practical and industry-oriented schemes, we ought to look at HT detection both at a "high" protocols level (which is covered in this deliverable) and at the "lower" measurements and decision making levels (which covers techniques detailed in D3.2).

In this document we first classify and review the different HT detection methodologies as given in the literature. This review serves as a basis for understanding how we came to define the four HT detection schemes described in Chapter 2, schemes which are the most relevant to the use cases described in deliverables D1.1 & D1.2 of the HINT project. Then in Chapter 3 each scheme is analysed from a security point of view and adequate countermeasures are defined in Chapter 4. Finally in Chapter 5, we describe the secure versions (integrating the countermeasures) of four detection schemes described in Chapter 2.

Note that at this stage, this document does not focus on what happens in response once the HT detection mechanism has made a decision, i.e. reaction procedures, notification to the user, notification to the supplier etc. In the rest of this chapter, we will look at the HT detection techniques as proposed in the literature. We regrouped those techniques into four classes.

1.1 Glossary of terms

In this document the following terms will be used:

Hardware Trojan (HT): This refers to a malicious, hidden modification of a hardware instantiation of a circuit in order to leak data manipulated by the circuit, or deprecate its quality or its functionalities.
Golden circuit/device: It is a circuit which is known to be Hardware Trojan free and which is used to make reference measurements.
Device under Test (DuT): It is a circuit on which measurements are made with the aim of determining whether it bears a Hardware Trojan or not.

1.2 Hardware Trojan detection based on hardware reverse engineering

A first class of HT detection scheme consists in performing hardware reverse-engineering of the DuT. The process of performing hardware reverse engineering can be summarised into five major steps [25]: decapsulation, delayering, imaging, annotation & schematic creation. This process is an invasive approach, destructive, expensive and time-consuming. In [5], the authors propose a method to alleviate part of the burden of reverse engineering by replacing the last two steps (annotation and schematic creation) by the use of machine learning techniques for the detection of Hardware Trojans. Nevertheless such approaches are still prohibitive for the use cases envisioned in HINT.

1.3 Hardware Trojan detection based on embedded test structures

A second class is based on the use of embedded test structures (the way ATPG[1]-like structures are added for verification). For example [16] proposes a technique to learn about a "trusted region" without the need for trusted golden circuits. This technique combines trusted simulation models and measurements from a piece of circuit called Process Control Monitors (PCMs). The authors assume that PCMs are never infected, and argue why this is typically the case. PCMs are simple circuit structures that serve for measuring fundamental parameters of the fabricated silicon, which indicates the operating point of the fabrication process, and are typically present either on the die or on the wafer kerf. We note that this approach is only suitable for ASICs and requires access to measure the PCM. The authors manufactured chips on TSMC 350nm process. The Trojan ex-filtrates an AES key by modulating a RF transmission. The statistical techniques that the authors use include data from Monte Carlo simulations of a golden device and measured PCM power consumption traces. The authors model the dependency between PCM simulations and PCM measurements and use the same kind of dependency to transform circuit simulations into expected circuit measurements. The classification step compares the expected circuit measurements with the actual circuit measurements. In this step PCA[2] and SVM[3] techniques are involved. The proportions of false positive and false negatives that the authors achieve are very good and very close to the ideal (0% false positive, 7% false negative). But those techniques have the disadvantages of potentially having a high impact on the DuT's design, of not having an exhaustive-enough nor a precise-enough test coverage.

1.4 Hardware Trojan detection based on side channel analysis

A third class is based on detection methodologies using side channels (power or EM[4]) measured on the DuT. The overall principle behind is highlighted in [24] (Chapter "Hardware Trojan De-

[1]Automatic Test Pattern Generation
[2]Principal Component Analysis
[3]Support Vector Machine
[4]EM: ElectroMagnetic

tection"). There the authors acknowledge that any malicious insertion on the hardware (such as a HT) should be reflected in some side channel parameter, such as leakage current or quiescent supply current, dynamic power trace, path-delay characteristic or (explicitly) electromagnetic radiation (EM) due to switching activity (or any combination thereof). This reference also states that EM radiation, due to switching activity, can be used to detect the presence of extra Trojan gates in a non-invasive, non-destructive manner, meaning that the circuit can be used in its nominal way while searching for the HT. In this reference it is also stated that any technique based on measuring the electromagnetic radiation falls in the same category of measuring the transient supply current, but the differences and the strengths of each approach (EM vs. transient supply current) are not further analyzed. Reference [13] follows the same line of research. This reference shows that practical HT detection using EM-based SCA techniques is possible and gives an example. The author successfully identifies HT using a standard laboratory EM setup (consisting of near-field magnetic probes ETS Lindgren 7405 of 1 cm diameter and a preamplifier from 100 KHz to 3 GHz). However, we note that the conditions are very favorable to detect the HT: the HT details are perfectly known, as well as the exact time when the HT is activated. This is usually not the case in a typical HT detection scenario.

In order to increase the sensitivity of the HT detection methodologies using side channels, several approaches have been studied. In [3], the authors propose a general method of modelling the noise for increasing the HT detection sensitivity. In [27] the HT detection problem is formulated as a signature outlier identification problem, and the HT detection problem is solved by comparing each signature with an estimated value of other signatures. The method is agnostic with respect to the specific side-channel used (instantaneous power, electromagnetic radiation). This method has the advantage of being somewhat resistant to process variations, is scalable and does not require a trustworthy golden IC for reference.

In other cases, spatially partition-based approaches are proposed to allow more local approaches either by using "add-on circuitries" like power ports [1] or separate voltage rails [4] or by using chosen input vectors like in [9, 19]. Alternatively the partitioning can be done on the temporal level as proposed in [20] in order to tackle the problem of process variations: the proposed method combines the current signature of a chip at two different time windows to "*completely eliminate the effect of process noise*". The authors argue that this process provides high detection sensitivity for HT of various sizes. The method does not require golden circuits as references.

Most of the drawbacks of the proposed methods for detecting Hardware Trojans using side channels have been summarized in [7]. The main points of this paper are that a) "*methods from previous works are not robust with respect to process and test environment variations and therefore cannot reliably detect very small HTs*", b) almost all previous works lack a thorough experimental section and hence its usefulness is not very clear. They further analyze the experiments carried out in the literature, and identify two main shortcomings: a) non-realistic, abnormally large HTs are used and b) HT are inserted at gate level or even at RTL. Such experiments do not accurately reflect a real situation, where the HT would be put by an untrusted foundry at layout level. This insightful observation shows that previous works use over-complex modifications of the original circuit to include the HT, and thus the detection figures should be taken with due care.

One of the rare cases where a "realistic approach" based on a "realistic scenario" is described

is in [18]: this indeed is quite close to the approaches envisaged in HINT. In this paper, the authors consider an ASIC AES designed using a 180nm ASIC UMC process. The HT occupies a relatively small proportion of the area: 0.5% in area (190 GE). The Trojan causes a Denial of Service and is triggered when a certain 30-bit kill-sequence is transmitted through a serial I/O line. Most of the area of the Trojan accounts for 30 flip-flops to store the input value. The rest of the Trojan is a combinatorial circuit that performs the comparison with a predefined pattern. The modifications were performed at the mask-layout level in the GDSII format. The authors take power consumption measurements of the ASIC at a sampling rate of 1 GS/s using a differential probe. For the analysis, 1 million traces are required per chip. The clock frequency is 10 MHz. The first part of the analysis consists in building power templates by calculating the mean of all power traces from each ASIC. After that, the authors calculate the difference of means (DoM). This serves as a first approximation towards distinguishing Trojans, as the "trojanized" ASICs have different patterns in the DoM trace than the Trojan-free ASICs. The second step in the Trojan classification is applying PCA to the mean traces of each device, to provide less redundant information. The output of this step is fed to a Support Vector Machine (SVM) classifier. This experiment assumes that for training the SVM chips without and with a Hardware Trojan are available. The classification accuracy that the authors achieve is satisfactory.

1.5 Hardware Trojan detection based on measurements of data path delays

A forth class is based on the measurement of path delays within a DuT, which can be viewed as another kind of side channel. In this category, one of the approaches is based on the addition of internal structures (often referred to as *shadow registers*) which allow to measure internal timing information [6,14,15]. In [21], the "shadow registers" work with a "shadow clock" which runs at the same frequency as the main clock. An ajustable phase offset permits to measure path delays along an arbitrary large number of paths in the design. In [15], the authors propose "shadow registers" whose sampling times are monitored via a dedicated clock signal and which are used to measure the associated delay path: if a Trojan is present on this particular path, then the measured delay path is expected to be different from the data path without Trojan. By extending this principle to several data paths and by considering that the monitoring of the sampling time of each "shadow register" as a Challenge, then the IC can be authenticated through a PUF-like challenge-response pair (CRP). A similar philosophy is proposed in [14] where existing test structures are modified to introduce "embedded test structures" that detect delay anomalies introduced by a Trojan.

A detection of Trojans based on path delays without additional circuitry is discussed in [11]: the measurement of the path delays corresponding to each of the 64 bits of the DES ciphertext are determined by simulation. Results based on timing simulations are also proposed in [12]): this time the authors investigated about HT inserted at different design levels (gate level and layout level) of an implementation of a Scalable Encryption Algorithm (SEA) in 90nm technology. Yet none of the proposed approaches handle the problem from an industrial perspective where measurements have to be made and evaluated "in the field".

1.6 Hardware Trojan detection for HINT

In order to address the core requirements of integrity, availability and "industrial & *in-the-field* usability" as depicted in D1.1 and D1.2, our initial efforts have focused on non invasive techniques like side channel analysis and measurement of data path delays as depicted in Chapter 2.

Chapter 2 Hardware Trojan detection methodologies

Four schemes are presented to describe the detection procedures studied in the WP03. Those four schemes are organised under two groups: passive and active approaches. In this chapter we recall what we mean by passive and active approaches and give a brief overview, from a protocol level, of the different approaches that are considered in the HINT project. Details about how the detection methods (algorithms, measurement equipment used...) are implemented are given in the HINT deliverable D3.2.

A general testing approach, which makes abstraction of the different approaches considered in the project, is depicted in Figure 2.1. The details specific to the different passive and active approaches can be found in the external inputs. All testing approaches can be easily built upon this basic procedure.

Note that in some cases we may know Trojan details, or even know the Trojan precisely. For instance, reverse engineering may have revealed a Trojan, and in-the-field tests shall now be deployed to gain an overall assessment about the batches affected. In such cases we might even have a Trojan-Model as part of the "golden reference data" used in the Matching Phase.

Each detection procedure consists of two phases: a learning phase and a matching phase. During the learning phase, the chip designer measures a Trojan-free chip (or one having a known Trojan) to obtain reference data. This reference data is then used in the matching phase to determine if a device contains a Hardware-Trojan or is Trojan-free.

Learning phase: The *Learning Phase* is the one where the "reference data" is generated. This reference is built from a golden circuit which is Trojan-free or with a known Trojan. Depending on the exact use-case, this golden circuit could be

- one of the manufactured Integrated Circuits (IC) which is, once the reference data has been measured, reverse-engineered[1] to make sure that the device was Trojan-free.
- if we are targeting FPGA implementations, the golden circuit could be one FPGA implementation realised by the tester.
- a simulation of the original design, in which case this should be an additional parameter to be considered in the Matching Phase.

This Learning Phase is usually done in the **secure & trusted premises** of the designer. This could be in a laboratory environment where all the acquisitions and calculations are done using

[1] This could be a destructive and time-consuming or lengthy process.

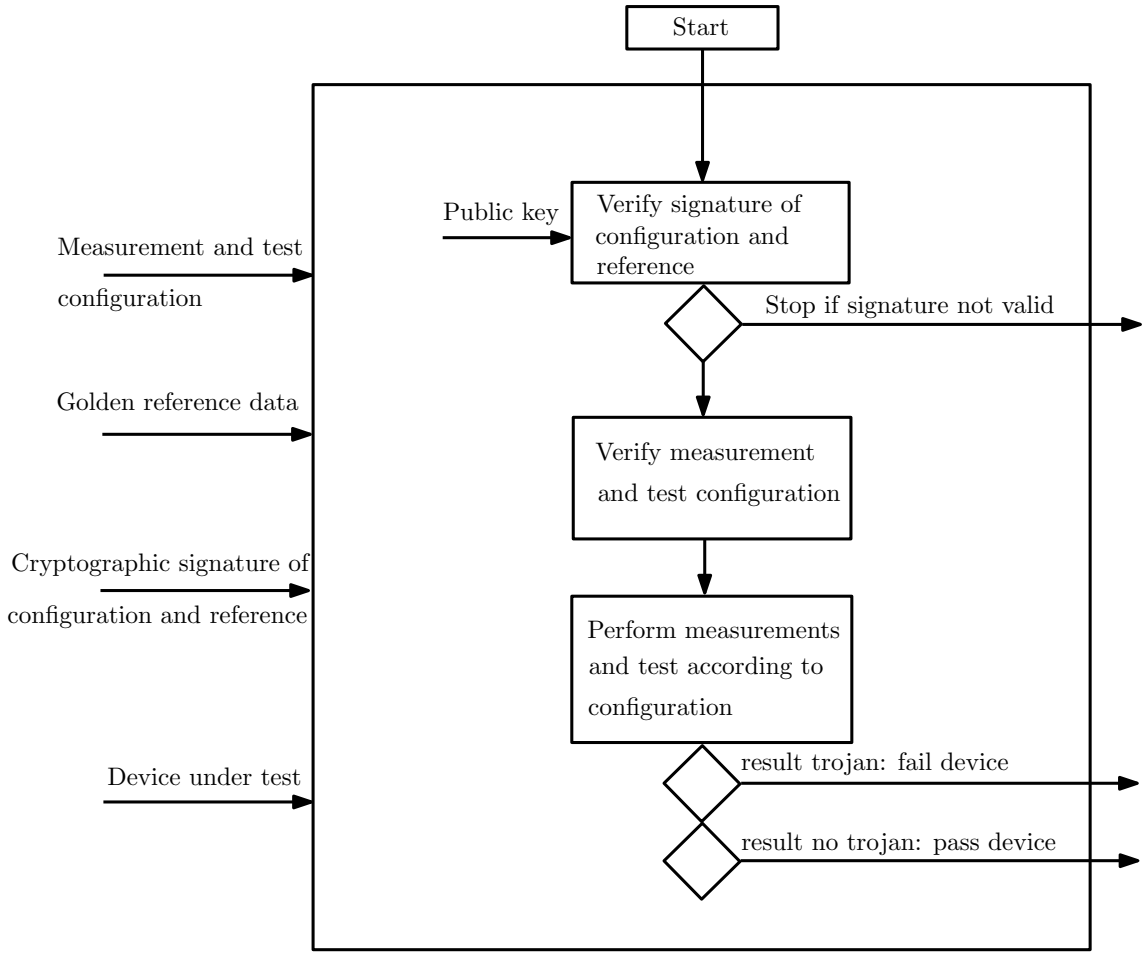

Figure 2.1: General testing approach

the calculation powers of PC-like devices. In principle, there are no particular timing constraints on the learning phase, nor on the amount of collected side channel measurements.

Matching phase: The *Matching Phase* is one where the measurements are done on "some" Device under Test (DuT) and that in the end the measured data is "compared" to the reference one generated during the *Learning Phase*. At the time this report was written, this *Comparison* step was still under deep research & study and several alternatives of pattern matching techniques were investigated depending on the type of HT we would like to detect, the measurement procedure... In practice, we wouldn't have any particular timing constraint on the *Learning Phase* but would like the *Matching Phase* to be quick. Moreover, in practice especially when done "in the field", for the *Matching Phase*, less complex equipment and less secure environments than those of the *Learning Phase* might be used.

In the HINT project, we envision two use cases for the matching phase.

- In the first use case, the chip designer itself will perform the testing during the matching phase within its secure and trusted premises: this particular case takes us back to security scenarios similar to those of the learning phase.
- In the second use case, the matching phase is carried on a device in the field, i.e. the

environment can be considered as being **untrusted & even aggressive**. Depending on the use case scenario, this phase could also be made to run on "embedded" devices with specific constraints in terms of measuring equipment used, performance, memory allocated and power consumption. For example, it can be executed in a docking station while recharging the device.

The second use case is more general than the first one, since the matching is carried out in a non-trusted environment. That is why we will mainly focus on this second use case when performing the risk analysis and proposing countermeasures. Another difference is the use of a less complex test procedures (simpler probe, no lab environment, very few measurements, less complex analysis, restricted time, less accuracy in the outcome of the matching test).

The distinction between passive and active approaches is made from the point of view of the DuT. That is, passive approaches do not require specific action from the DuT beyond normal operation, whereas the DuT has to actively collaborate with the tester in the active approaches.

2.1 Passive Approaches

In this section, we describe those approaches whereby the Device under Test (DuT) is simply observed, i.e. we collect the side channel information (either power consumption or Electromagnetic waves emitted) during a normal calculation done by the DuT and we analyse those side channel data collected to determine whether the DuT has been modified or not. We could simply be observing the device (with 'non-controlled' inputs) or make the device work on specific input data that could allow us to accentuate the presence of a Hardware Trojan (HT).

2.1.1 Passive approach with 'non-controlled' input data

In this approach we measure the side channels of the golden device and the DuT under normal operating conditions. More precisely, we operate the device under normal conditions and provide not specifically chosen inputs.

In the learning phase we want to collect a sufficient amount of measurement data from the golden device to be able to characterize some particular attributes of its distribution. In the matching phase we want to collect side channel measurements from the DuT and decide if the data comes from the same distribution (DuT is trojan-free) or not (DuT is infected) using some sort of "test" for the comparison. This test should be robust, reliable and quick to calculate. In addition, it is desirable that the test somehow quantifies the confidence we can have in the result.

The attributes of interest of the side channel measurement data distribution depend on and vary with the "test" that we will use for the comparison in the matching phase. The attributes and the statistical procedures will be specified and explained in HINT deliverable D3.2.

We now detail the meaning of all external inputs in Figure 2.1 with respect to this approach.

- Measurement configuration: Oscilloscope settings, clock speed, supply voltage, probe type, power or EM side channel, random inputs, etc.
- Test configuration: Attributes of interest, statistical tool, decision threshold, confidence threshold, etc.

- Golden reference data: Values for the attributes of interest extracted from measurements of the golden device.
- Cryptographic signature of configuration and reference: Cryptographic signature of the measurement and test configuration as well as the golden reference data. This could be for instance an RSA-OAEP or RSA-PSS signature.
- Device under test: The DuT that shall be tested for the presence of HTs.

In the learning phase we will characterize some particular attributes of the golden device's side channel measurements' distribution. This data will serve as golden reference. We will then use cryptographic algorithm XYZ and our private key to generate a cryptographic signature over the measurements and test configurations as well as the golden reference data. The outputs of the learning phase are: configuration, golden reference, signature. They are inputs to the matching phase.

In the matching phase we first load external inputs. Then we check the validity of the cryptographic signature over configuration and reference data using our public key[2]. If the signature is valid we know that the data is authentic and unaltered. If the signature is not valid we abort the test. Next we ensure that the equipment used in the matching phase is properly configured according to the measurement and test configuration. This ensures that the measurement conditions are very similar (ideally identical) to the conditions in the learning phase. Note that in practice the tools used during the matching phase might have to take into account the fact this is not always the case: during learning a much more sensitive probe can be used than during matching, or a higher sampling resolution is used, which is then downscaled through post-processing to fit the matching conditions.

Then we collect side channel measurements from the DuT and apply a statistical test to compare the measurement data with the golden reference. If confidence in the result is higher than the specified threshold, the result is compared with the specified decision threshold to output a binary fail or pass. If confidence in the result or the result itself are close to the thresholds (in a grey zone) we may execute the protocol again from start with fresh and perhaps more measurements from the DuT to obtain a clear result.

2.1.2 Passive Approach with Increased Sensitivity to HT

The efficiency of the previous passive approach can be improved by choosing the input vectors to the DuT. The goal is to increase the sensitivity of the detection scheme to Hardware Trojans.

Context

Existing side-channel approaches for HT detection suffer from different sources of "noise" which mask the end effect on the HT on the measured side-channels:

- measurement noise (it can be decreased with an appropriate side-channel equipment – sensitive probes, amplifiers, oscilloscope, *etc.*),
- process variations noise (which will force us to do multiple measurements on different DuTs to assess the efficiency of our detection methods),

[2]Note that this somehow anticipates upon the security analysis done afterwards in the document but at this stage, this seemed really trivial...

- the global DuT side-channel which is supposed to be greater than those of the inserted HT (this latter is supposed to be stealthy and then be very small). In order to detect small sequential/combinational Trojans in large circuits ($> 10^5$ transistors), we need to improve the SNR (Signal to-Noise Ratio) using appropriate side-channel isolation techniques.

Increase HT Detection Sensitivity

Concerning the last point, we have to increase Trojan detection "**sensitivity**". The sensitivity can be improved by increasing the current contribution of the Trojan circuit relative to that of the original circuit. The extreme solution to increase the ratio "HT power consumption" over "DuT power consumption" is to divide the DuT into several regions using a fine grained partitioning and measure the side-channels of one region when the others are switched off[3].

Depending on the DuT itself, the fine-grained partitioning (or "functional decomposition") can be done very easily. Sometimes, the DuT contains clearly-defined functional blocks which can be selectively activated by using control signals or precise commands. For example, some components dedicated to interfaces, like those used in PMR (Professional Mobile Radio) terminals, can be easily partitioned because the regions have minimal interconnection. In these cases, it is easy to maximally activate one region while minimizing activity in other regions.

Such approaches have been studied in [4], where one can however foresee the technical complexities linked to the way the partitioning is done and to the experimental set up needed. Moreover, some circuits might be available as a flattened gate-level netlist. The AES we use in HINT project is in this form. In this kind of DuT, there are complex interconnections, so partitioning will be more difficult. This is why we propose a region-based vector generation approach, which finds and ranks the test vectors which induce maximum activity in one region, while minimizing the activity in other regions.

Related Limitations of the Method

When applying methods which increase the sensitivity of the detection scheme to HT by a functional decomposition of the DuT, we have to fight against underlying limitations.

1. The granularity of the partitioning of the the DuT must be done carefully. If the regions are too small, there will be too many regions to analyze and so the time for matching phase will be too long and the needed space for storing (golden) side-channel measurements will be too big.

2. When a particular region is being activated, the test vectors should try to create activity for a big number of possible Trojan trigger conditions, possibly connected to many other wires of the region (in real life, we don't know where a HT is inserted). The number of possibilities appears enormous, and so an exhaustive enumeration is not possible. Instead, we will have to choose a small number of input vectors that induces maximum effects.

3. This method will not have effects on all the possible HTs. For example, such a method does not cover the "Always On" HTs (that are continuously activated whatever the input values), time-based HTs (e.g. which counts DuT clock cycles) and those that don't use internal signals to trigger malevolent mechanisms.

[3]This method has also the advantage to localize the HT.

2.2 Active Approaches

In the active approaches, we will be tampering with the DuT to make it work in an 'abormal' way for the time during which we shall try to collect some side channel information and try to make a decision as to whether there is an HT or not.

2.2.1 Active approach using some IC privileged mode

The 'passive' approaches described in the previous sections make use of side channel information, which might conceptually seem to be antagonistic with the large amount of work and effort that has been poured into the design of secure chips during the past decades in order to reduce the side channel leakages (with more or less success)! So we expect that in some cases, we shall be faced with the problem that countermeasures that have been added to reduce side channel information leakages in secure chips might become severe impediments to the approaches described in sections 2.1.1 & 2.1.2. To circumvent this difficulty, we hereby define an "active" approach where the IC will help us entering into some 'privileged' mode where those countermeasures (or some of them) are deactivated as depicted in Figure2.2. The latter figure applies both to the *Learning Phase* where the reference templates are built and the *Matching Phase* where the devices are tested for HTs either within the secure premises of the fabless company or in the field in insecure environments.

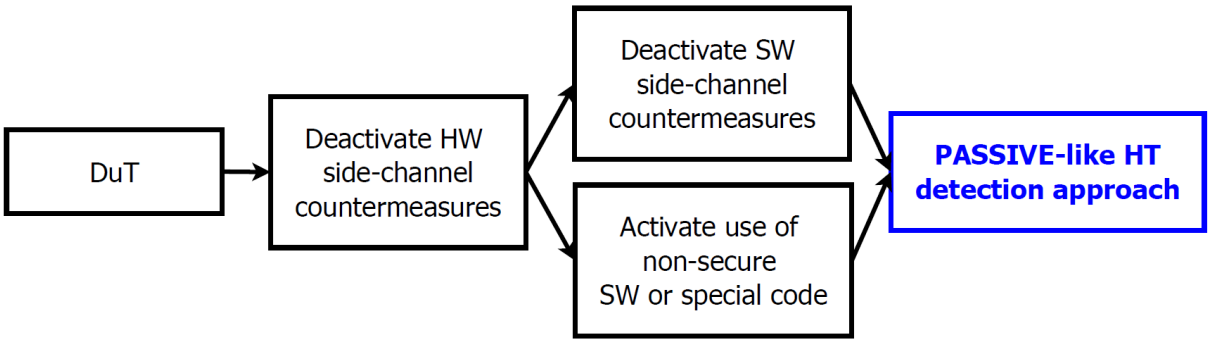

Figure 2.2: Active approach by using some privileged mode

2.2.2 Active approach based on internal data path timing measurement

In the literature, there are hints [6, 11, 12] showing that some Hardware Trojans (HT) can be detected by measuring the effect that have on the timing of the internal data paths of the Device under Test (DuT): those initial results were based on simulation results and did not give any hint of how such an approach could be used in practice.

In our case, we have a tool, based on clock glitching techniques [2], that allows to measure the timing of some data paths of a given circuit: for example, for an AES circuit, this technique has been used to measure timing information associated to the 128 bits of the main data path (for the state matrix of the AES) of a hardware implementation of the AES.

Some initial results have shown that using this 'clock glitching tool', malicious modifications of an AES implemented on an FPGA could be detected [10]. In this section, we provide a first

definition of the protocol that would have to be implemented when detecting the presence of HTs based on data path timing measurements using our clock glitcher. This protocol consists of two major steps:

- The *Learning Phase* is the one where the "Reference distribution curve" is constructed from a golden circuit (or a model of it).
- The *Matching Phase* is the step where distribution curves are constructed from the Device under Test (DuT) and compared with the reference distribution curve to determine whether an HT is present or not.

Both phases implement a core *Measurement* step which is common to both.

Learning phase

The main steps of this phase are illustrated in Figure 2.3. The *Measurement* step is further detailed in Figure 2.5. Note that the *Measurement* step is repeated a predetermined number X times in order to compensate for the noise present in the measurements.

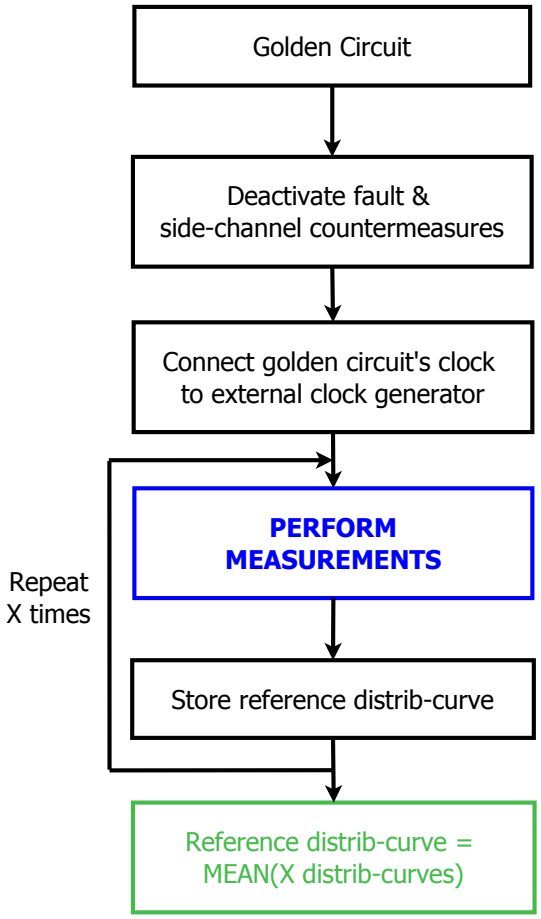

Figure 2.3: Learning phase for active data path measurement approach

Matching phase

The main steps of the *Matching Phase* are depicted in Figure 2.4 whereby the *Measurement* step is further detailed in Figure 2.5.

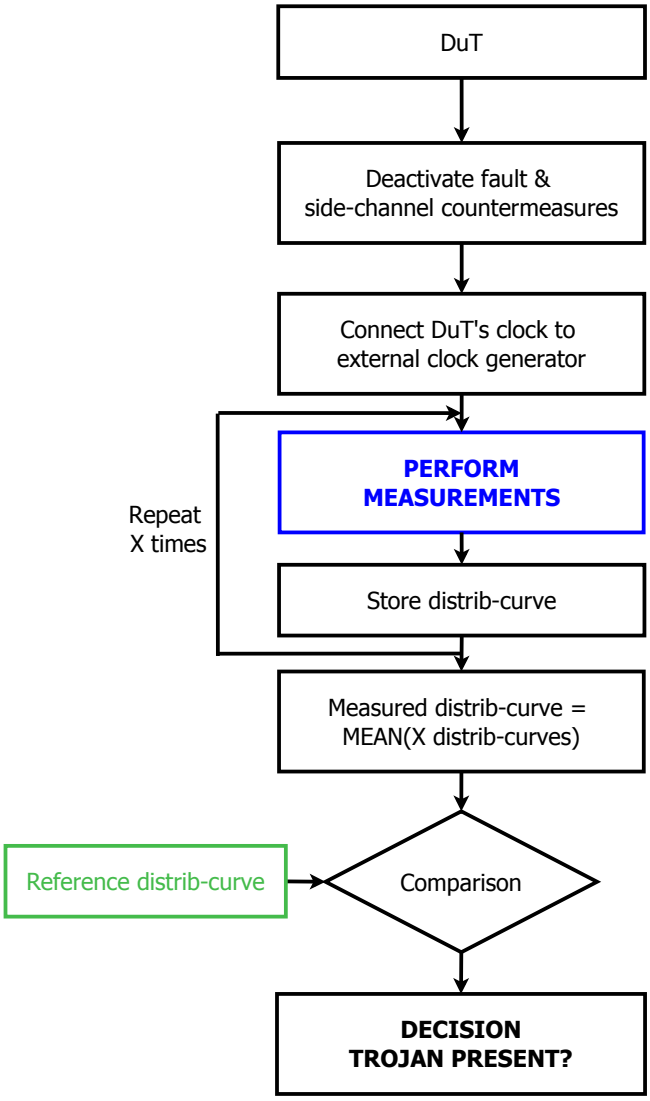

Figure 2.4: Matching phase for active data path measurement approach

The core MEASUREMENT step

The *Measurement* step present in the Learning & Matching phases is detailed in Figure 2.5. Note that

- The instant T when the clock glitch needs to be injected is predetermined and depends on the function that is tested and the way the timing associated to each particular data path bit is calculated[4].
- D is the value of the glitch, i.e. the number of picoseconds by which clock period is reduced at the instant T.
- D_{step} is the elementary step by which D can be increased. This is determined by the clock glitching tool. In our case, $D_{step} = 35ps$.

[4]Note that details about the inner workings of the measurement tools, the principles behind this approach and the study of the efficiency of the proposed method shall appear in **D3.2**.

- D_{max} is the maximum clock glitch that can be inserted if we want to keep the DuT still functional. This is, for most cases we met so far, equal to half of the nominal clock period of the DuT.
- Since our experiments showed that for a particular data path, the timing information measured by our tool depends on the data sent to the DuT, the measurement is done on a large number Y of randomly chosen input patterns[5].

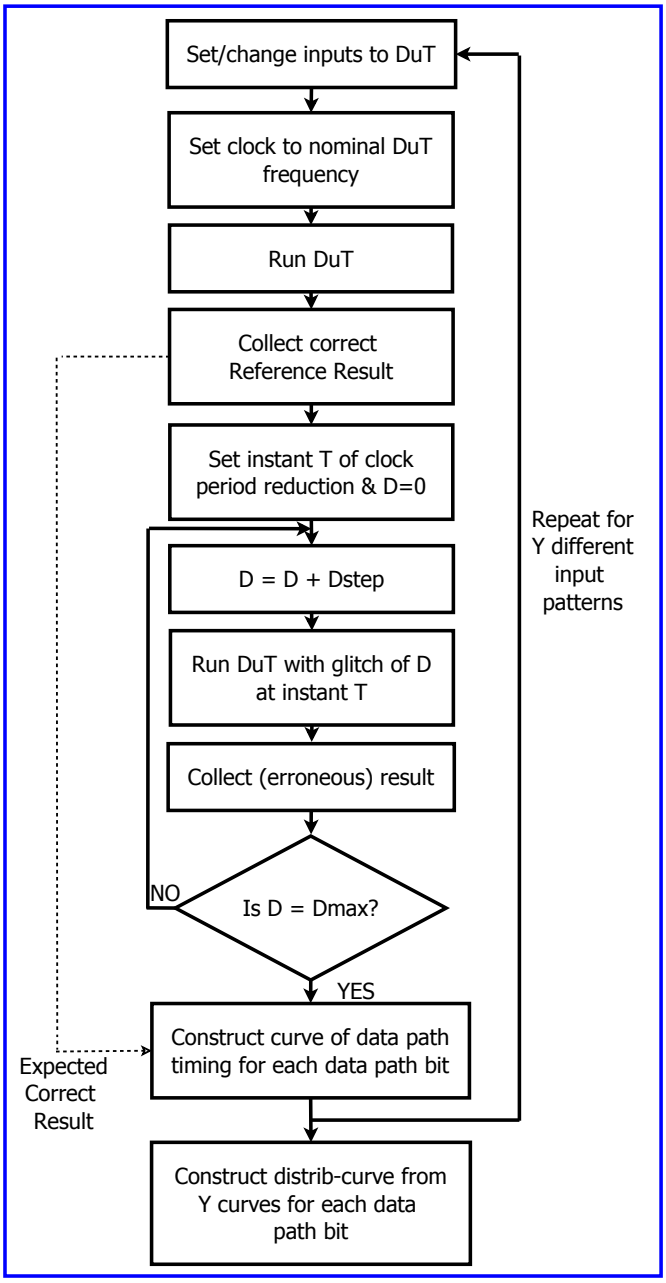

Figure 2.5: Core measurement steps of the learning & matching phases

[5]For example for the AES, the value of the key is kept constant but a thousand random plaintexts are used.

Chapter 3 Security Risk Analysis of HT detection methodologies

In this chapter we analyse the four detection methodologies described in Chapter 2 to outline the underlying security risks. The security analysis is based upon certain security assumptions. These will be discussed first.

3.1 Security assumptions

In this section, we give an overview of the most important security assumptions made during the risk analysis.

Learning phase: As already mentioned before in this deliverable, one assumes that the learning phase is carried out in secure conditions, usually in the secure & trusted premises of the designer. To reliably detect a Hardware Trojan, it is important that the learning phase is performed correctly:

- The golden device, which is used to build the golden reference data actually implements functions similar to those of the DuT and not other designs,
- the golden device does not contain a Hardware-Trojan, and
- all tests are executed correctly (i.e. the correct parameters are used, the procedures are followed as planned, etc.).

The reference data resulting from these tests, is not being modified, and no relevant data is being deleted. All these assumptions are important, since one explicitly relies on the correctness of the golden reference model during the matching phase.

Testing equipment: Another important security assumption is that we trust the testing equipment, both during the learning phase as during the matching phase. Otherwise, the trust in the results of the testing equipment is limited. Indeed, if the testing equipment would be malicious during the learning phase, then it can output incorrect reference data, for example reference data associated with a circuit containing the Hardware-Trojan. If it would be malicious during the matching phase, it could output the incorrect decision regarding the presence of a Hardware-Trojan. Nothing prevents the malicious testing equipment from outputting that all tests were successful and that no Hardware-Trojan is present, without even carrying out a single test. To avoid this, the trust base in our setup will have to contain at least the testing equipment itself. The trust requirements on the testing equipment used during the matching

phase can be relaxed by using multiple testing equipments, and assuming that at least half of them is working correctly. In that case, one could repeat the tests on these different testing equipments during the matching phase, and take a majority vote (i.e. the output chosen by the majority of the testing equipments is selected). However, this is not efficient and very slow, and would significantly increase the cost. It also does not offer protection again a malicious testing equipment attacking the DuT during the tests (e.g. stealing sensitive data such as a secret key). An alternative solution would be to verify each testing equipment manually. If there are only a few testing devices in the field, this approach could be feasible. However, even when this is done, one can argue about the trust one can have into the device validating the testing equipment...

Low false positive and negative rates: All (active and passive) test approaches are based on statistical processes. Therefore, the test results could be incorrect due to false positives or false negatives. This should of course be taken into account. In this deliverable, we assume that the testing approaches developed in the HINT project have sufficiently low false positive or false negative rates, such that this does not pose a security threat. Depending on the use case, also moderate failure rates could still be acceptable, for example when distinguishing infected from non-infected batches in a large-scale test.

Authentication module: As will be discussed later, the active Hardware-Trojan detection methodologies will require the execution of a role-based authentication protocol. If this protocol is carried out successfully, the DuT will enter temporarily into a privileged mode where some countermeasures are switched off and/or where some "dangerous" operations are allowed. The role-based authentication may be even more sophisticate, assigning different modes to different environments. For instance, a certain category of test environments may use DuT commands with disabled countermeasures, while other test environments may even invoke special software commands, etc. These dedicated assignments will be encoded in certificates, as will be discussed later. The role-based authentication protocol will be carried out between the testing equipment and a dedicated circuit in the DuT. The latter is denoted by "authentication module" in the rest of this deliverable and is an important component in the DuT. The setup is depicted in Fig. 3.1. To ensure a correct outcome of the matching phase and not to add security weaknesses to the chip, one has to make the following assumptions on this authentication module:

- When the authentication protocol finishes successfully, the authentication module indeed sends out a "privileged mode command" to the other components of the DuT. If this would not be the case, the authentication module could prevent the DuT being properly tested, even when the testing equipment authenticates itself to the authentication module. This would make it impossible to detect a Hardware-Trojan.
- The authentication module only sends out a "privileged mode command" to the other components of the DuT when the instance of the authentication protocol was carried out successfully. If this would not be the case, one would be able to switch off the side-channel countermeasures on the device, which would be a significant security vulnerability.
- The public keys which are hard-coded in the authentication module cannot be altered, unless under very specific conditions. Furthermore, this authentication module is Hardware-Trojan free.

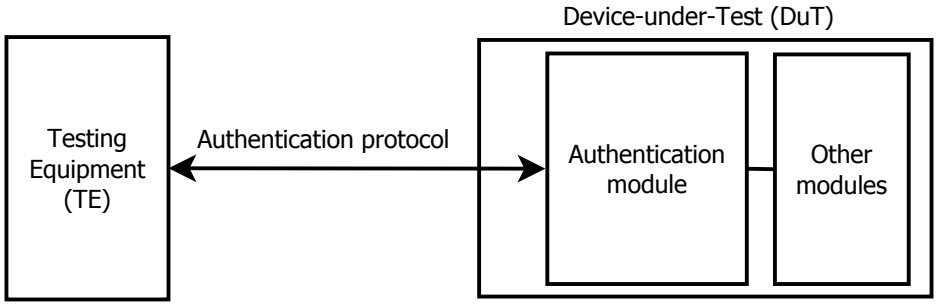

Figure 3.1: Authentication module

DuT: Furthermore, we also have to assume that there is a secure connection between the authentication module and the parts of the DuT that control the side-channel countermeasures/clock signal/.... Only the authentication module should be able to output a valid command to enter privileged mode. Neither external nor internal (i.e. other components on the DuT) parties should be able to send these commands. Even more, it should also be impossible to spoof this command. Besides the authenticity of the "privileged mode command", also the freshness of this command is important. It should be impossible to replay the command. Furthermore, besides the exact timing, the parameters related to the privileged mode (i.e. what operations are allowed, under which constraints) should be part of the authenticated "privileged mode command" and preferably be encoded in a certificate. This way, the DuT knows exactly which security sensitive operations it should temporarily allow.

3.2 Threat modelling approaches

Risk analysis is based on the notion that any system has assets of value worth protecting. These assets have certain vulnerabilities, which can be exploited by internal or external threats to cause damage to these assets. Since one can have different views of a system, there is no single method to model the system's threats. More in particular, there are at least two common approaches to risk analysis:

Attack-Centric: Attack-centric risk analysis starts with an attacker, and evaluates his/her goals, and how he/she might achieve them. So one views the system from the viewpoint of an attacker to determine the main risks. This approach usually starts from either entry points or assets.

Software-Centric: This approach starts from the design of the system and attempts to step through a model of the system, looking for types of attacks against each element of the model. Therefore, this approach requires that a system model is constructed in advance, where all the system's components can be identified.

Besides these two common methodologies, there are also hybrid threat modeling techniques which combine various aspects of attack centric and software centric and threat modeling. There is no default threat modeling technique which is the de facto choice. If the system can be easily described by means of data-flow or component diagrams, then software-centric threat modeling is often used. If the threat model of the system can be easily described by means of

assets that need to be protected or by a set of attack motivations that need to be considered, then attack-centric threat modeling is often the best choice.

3.3 Attack-centric risk analysis

As discussed in the previous section, various threat modeling techniques exist, each with their own viewpoint on the system. In this deliverable, we will use the attack-centric risk analysis approach since the assets that need to be protected, can be easily defined. This can be immediately translated into a set of possible attacks. More in particular, the following relevant attack goals can be identified:

HT not detected: During the matching phase, the testing equipment will perform tests on a device in the field. This is done in an untrusted environment. To ensure that the testing equipment outputs reliable results, the measurement conditions should be aligned with the conditions in the learning phase. Therefore, the testing equipment needs to use the correct measurement and test configurations, and the correct golden reference data. Otherwise, the test results become unreliable and the Hardware-Trojan might not be detected. If an attacker succeeds in modifying the measurement and test configuration data or the golden reference data, or replacing it by the data of another type of HT detection methodology, then the presence of the HT might not be detected.

Sensitive data recovery: A second class of attacks aims at recovering the sensitive data used in the DuT. This sensitive data could be a secret key (e.g., when an encryption algorithm is implemented in the DuT), confidential code, etc. In other words, the Hardware-Trojan detection method is being misused to carry out attacks on the DuT and break its security. This type of attack becomes particularly important when the DuT turns off certain security features (i.e. when entering a privileged mode) or when it performs certain operations which would help an attacker to recover the secret data/key/code. Therefore, this type of attack is particularly relevant when active HT testing approaches are used.

Denial of Service: A third category of attacks are the Denial-of-Service attacks. In this scenario, the Hardware-Trojan detection method is being misused to damage the DuT, or to make a specific service of the device temporary unavailable.

We will now give an overview of how these attack goals can be achieved (i.e. the risks associated to each of the attack goals). We will first present the risks which are applicable on all testing approaches considered in the HINT project. Next, we describe security risks which are only applicable for one specific passive or active approach. Some of these risks are similar to the general risks described in Section 3.3.1, but contain more details specifically related to the particular testing approach. It is important to point out that we implicitly took into account the security assumptions described above. Therefore, all risks which would break these security assumptions are not shown in the overview below.

3.3.1 General testing approach

Table 3.1 summarises the analysis of the general HT detection scheme of Figure 2.1, which are valid for all testing approaches discussed in Section 2.

Identified Risk		Attack goal		
#	Description	HT not detected	Secret data recovered	Denial of service
RGEA_01	Incorrect golden reference model is used.	X		
RGEA_02	Incorrect measurement and configuration data is used.	X		
RGEA_03	The input data is chosen or modified in such a way that the HT is more difficult to detect.	X		
RGEA_04	The effects of the Hardware-Trojan are not visible in the measured data during the matching phase.	X		
RGEA_05	HT is not active or switched off during the matching phase.	X		
RGEA_06	The side-channel measurements during the matching phase leak information.		X	
RGEA_07	Incorrect configuration data is used, resulting in dangerous testing operations that could damage the DuT.			X

Table 3.1: Security analysis of general HT detection scheme

3.3.2 Passive approach with uncontrolled input data

The attack-centric risk analysis of the passive approach with uncontrolled input data, is identical to the risk analysis of general HT detection scheme.

3.3.3 Passive approach using increased sensitivity to HT

In Table 3.2 below we analyse the particularities linked to the passive approach based on chosen input vectors for increasing sensitivity to HTs (Section 2.1.2).

Identified Risk		Attack goal		
#	Description	HT not detected	Secret data recovered	Denial of service
RPAC_01	The regions of the DuT that are not tested, are not properly switched off.	X		
RPAC_02	The input data is sent to the wrong part of the DuT.	X		

Identified Risk		Attack goal		
#	Description	HT not detected	Secret data recovered	Denial of service
RPAC_03	Regions of the DuT remain shut off, even after the matching phase.			X
RPAC_04	The input data is chosen in such a way that the side-channel leakage of the sensitive data (such as secret keys) is optimized.		X	
RPAC_05	Side-channel countermeasures are switched off when testing other region of the DuT.		X	

Table 3.2: Security analysis of passive approach with increased HT sensitivity

3.3.4 Active approaches

In Table 3.3 below, an analysis of the active approach of Figure 2.2 is given. The active approach based on internal path timing measurements, described in Figures 2.4 & 2.5, is a special case of this active approach. Therefore, the risks shown below apply to both types of active approaches.

Identified Risk		Attack goal		
#	Description	HT not detected	Secret data recovered	Denial of service
RAAP_01	In the matching phase, the sensitive data (such as secret keys) is leaked during tests (in privileged mode).		X	
RAAP_02	An unauthorized entity turns off the security mechanisms by forcing it to enter in privileged mode (while in fact the DuT is not being tested by the testing equipment at all).		X	X
RAAP_03	The DuT is forced to stay into privileged mode (longer than needed). Similarly, the DuT can be blocked to re-enter normal mode after finishing the tests in privileged mode.		X	X
RAAP_04	Other security-sensitive operations, outside the intended scope of the HT-detection, are being performed during this privileged mode in order to get information about sensitive data (such as secret keys).		X	

Identified Risk		Attack goal		
#	**Description**	HT not detected	Secret data recovered	Denial of service
RAAP_05	During privileged mode, dangerous operations are performed that damage the device or make it (temporarily or permanently) unavailable.			X
RAAP_06	The DuT does not enter privileged mode during the matching phase, making it more difficult (or impossible) to detect the HT.	X		
RAAP_07	During execution of the tests during privileged mode, the input data gets modified.	X		
RAAP_08	Incorrect operations/tests are being executed during privileged mode (for example, during a man-in-the-middle attack where commands are changed in transit from the testing equipment to the DuT).	X	X	X

Table 3.3: Security analysis of active approaches

Chapter 4 Definitions of Countermeasures

From the risks and attack goals identified in the previous chapter, we now propose some countermeasures which take into account those threats. In a nutshell, to circumvent the threat that the HT is not detected, the golden reference and measurement and configuration data must somehow be authenticated (using cryptographic techniques). This way, the probability of detecting a HT is maximized. To prevent secret/sensitive data from being leaked during the matching phase, the testing equipment should authenticate itself to the DuT, such that the latter knows that proper tests in the context of Hardware-Trojan detection are effectively being carried out, and no security attacks. Even more, also the exact context and configuration of these tests (which operations should be allowed, for how long, under which circumstances, etc.) should be sent in an authenticated way to the DuT, encoded in a certificate. By carefully specifying the testing configuration data and the exact testing procedure, it becomes harder for an attacker to perform a denial-of-service attack. Therefore, the countermeasures proposed to cover the first two types of attacks will be the main focus of this section, as these also help to protect against denial-of-service attacks.

In the rest of this section, we will now specify

- how the testing equipment can check the authenticity of the measurement and test configuration as well as the golden reference data.
- how the (authentication module of the) DuT can verify the authenticity and freshness of the commands sent to the DuT during the matching phase in active checking.

The solutions proposed in this section make use of cryptographic techniques and algorithms. A good overview and introduction to applied cryptography can be found in the "Handbook of Applied Cryptography" [17].

4.1 Authentic configuration and reference data

During the learning phase, all the necessary data is compiled such that a testing equipment can repeat the tests in (nearly) ideal circumstances. As already mentioned in Section 2.1.1, the following data needs to be specified:

- Measurement configuration: Oscilloscope settings, clock speed, supply voltage, probe type, power or EM side channel, random inputs, etc.
- Test configuration: Attributes of interest, statistical tool, decision threshold, confidence threshold, etc.

- Golden reference data: Values for the attributes of interest extracted from measurements of the golden device.

This data will be used as input by the testing equipment. In the design of our solution, we do not make any assumptions on where this (chip dependent) data is stored or is downloaded. It could be downloaded from a server, stored locally in the testing equipment, sent by the DuT, etc. Irrespective of the data storage or communication, this data needs to be authentic to ensure reliable HT detection. Therefore, during the learning phase, the chip designer signs the measurement and test configuration as well as the golden reference data with its private key. The resulting signature is also used as input by the testing equipment, together with a certificate containing the corresponding public key. The certificate may also contain attributes governing specific privileged mode conditions relevant during the matching phase. This certificate is signed by the private key of an entity, which can be traced back to the root (private) key of a trusted Certification Authority (CA). This CA can be operated by the chip designer itself, or by a trusted third party.

Each testing equipment contains the public key of the CA, which is installed during manufacturing and stored securely in a trustworthy environment. The authenticity of this public key is important, as it should be altered by unauthorized parties.

In the matching phase, the testing equipment first loads all external inputs: measurement and test configuration, golden reference data, signature on this data, and the certificate of the public key of the chip designer. Next, the following steps are carried out by the testing equipment, as shown in Fig. 4.1:

1. Check the validity of the certificate (chain) using the public key of the CA. If the certificate is not valid, abort the test. Otherwise, retrieve the public key of the chip designer from the certificate.

2. Use the public key of the chip designer to check the validity of the cryptographic signature over the measurement and test configuration and golden reference data. If the signature is not valid, abort the test.

3. If the signature is valid, the testing equipment knows that the measurement and test configuration and golden reference data are authentic and unaltered. It can now use this data to properly configure itself to this measurement and test configuration. Before the test operations can start, the testing equipment will first authenticate itself to the DuT (see Section 4.2 for more details). During or after the matching phase, the measurements collected during the tests are then compared with the authentic golden reference data.

The cryptographic signature algorithm could be RSA [22] or ECDSA [26] (Elliptic Curve Digital Signature Algorithm). The latter is recommended due to its smaller key size. For example, one could use a 256-bit ECDSA key, which gives medium to long-term security.

4.2 Authentication to the DuT

The matching phase is carried out on a DuT in the field, in a hostile environment. Therefore, the DuT will only allow "dangerous operations" – from a security point of view – when requests for such operations are authentic and fresh. Therefore, an authentication protocol needs to be carried out, which involves two main parties: the testing equipment (e.g., the charger of the

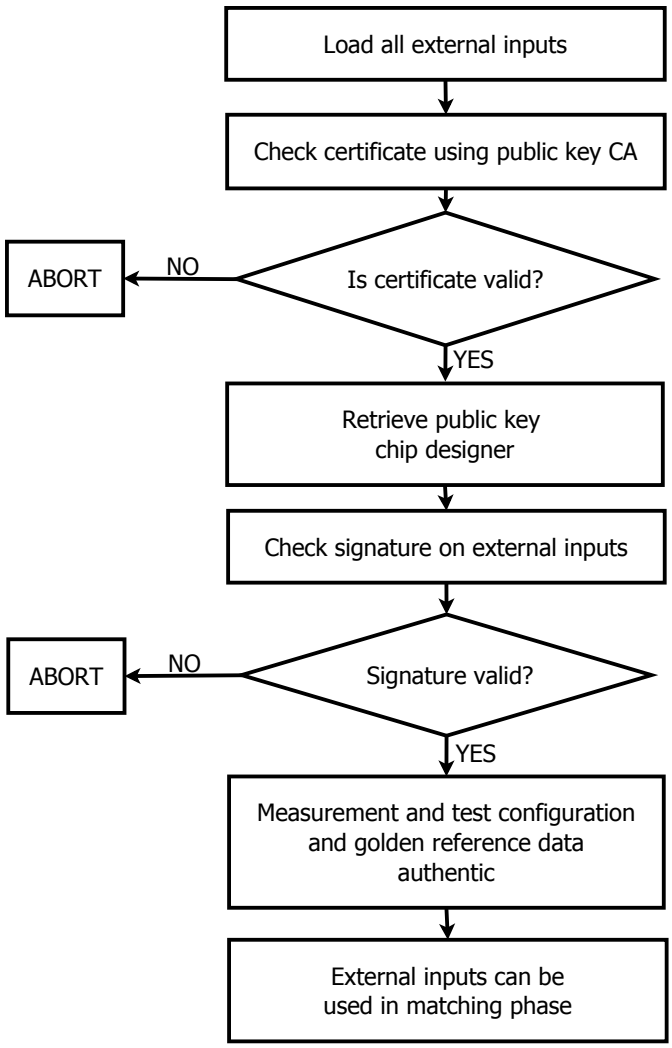

Figure 4.1: Verification procedure of external inputs during matching phase

PMR) and the authentication module of the DuT. As discussed before, we assume that there is a secure connection between the authentication module and the control part of the DuT (if these would not be integrated). So in our setup, it is sufficient that the authentication module can authenticate the messages/requests from the testing equipment.

Two protocols will be discussed. The first one (basic authentication protocol) is more efficient, but offers slightly less security, and is mainly intended for the use case where there is only 1 command that can be sent to the DuT (e.g., switch to privileged mode with pre-defined parameters x and y). If there are different types of commands which can be sent to the DuT during the matching phase (e.g., some testing parameters need to be specified), then the second protocol is the preferred choice.

4.2.1 Basic authentication protocol

The use case for this basic authentication protocol is the scenario where a command is being sent to the DuT to start the test, including a list of static, pre-defined parameters (typically put in a sort of certificate). When this command is authentic, the DuT enters a privileged mode where it allows the operations that correspond to the test parameters included in the certificate. The DuT remains in this privileged mode, until a counter or timer expires (period also included in the certificate), or when an authenticated command is sent to the DuT to switch back to the normal mode of operation. Note that only the testing equipment needs to authenticate itself to the DuT.

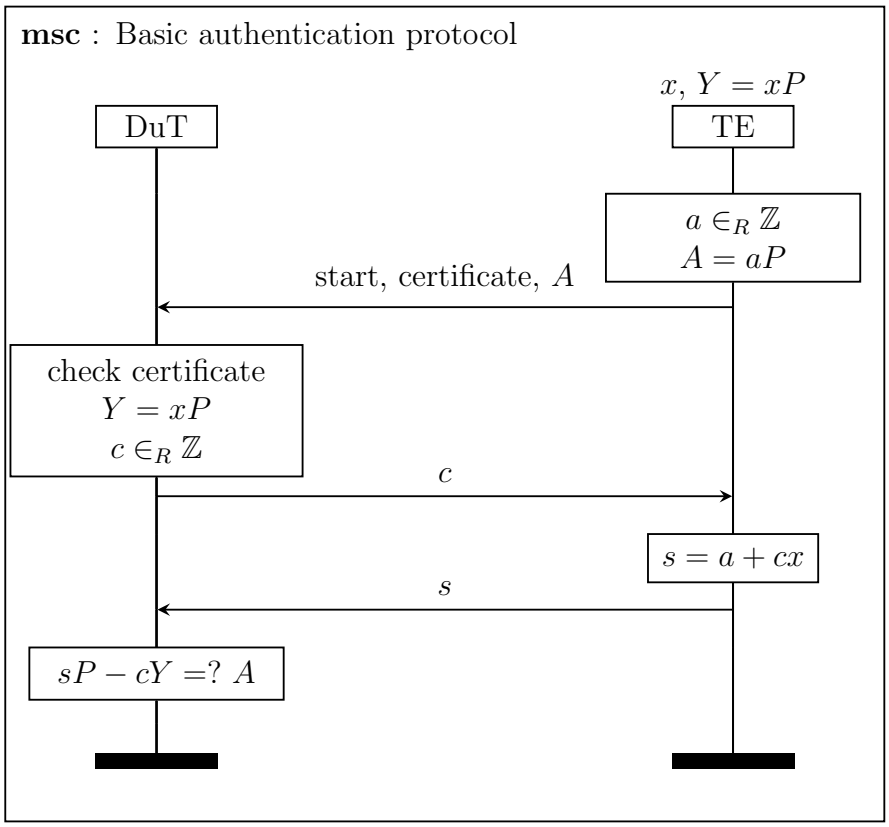

Figure 4.2: ECC-based Schnorr protocol to start privileged mode

The basic authentication protocol is based on the Schnorr zero-knowledge proof [23] and is shown in Fig. 4.2.1. For efficiency reasons, the ECC-based (Elliptic Curve Cryptography) version of the protocol will be used. Both devices share an elliptic curve, and a point P on the curve. The testing equipment (denoted by TE in the rest of this section) starts the authentication protocol by generating a random nonce a and computing the point $A = aP$ on the curve. Next, it send the respective start command (e.g., "start privileged mode"), a certificate which contains all relevant testing parameters, and the point A to the DuT. This certificate could contain the operations that are allowed, other configuration data, a timer indicating the end of the privileged mode (when a secure clock is present, which cannot be altered during the privileged mode), a counter (indicating the number of operations that are allowed), etc. In any case, it will contain the public key $Y = xP$ of the testing equipment. The certificate is signed by a trusted authority (e.g., chip designer), and the corresponding public

key is hard-coded in the authentication module of the DuT. When receiving this message, the authentication module of the DuT first checks the validity of the certificate. Next, it retrieves the public key Y and generates a random response c, which is sent to the testing equipment. The protocol ends by the testing equipment computing the response $s = a + cx$ and sending this to the authentication module of the DuT. This module can check the response by computing $sP - cY$. If this value corresponds to A, then the protocol ends successfully and the device can enter privileged mode, using the testing parameters in the certificate.

The privileged mode ends when the session is terminated. This could take place when a timer expires, when a counter equals to zero (if included in the certificate) or when the session ends improperly (due to an error). It is strongly recommended have the DuT automatically terminating the privileged mode, based on these criteria. Otherwise, the testing equipment would have to end the privileged mode by sending a stop command, and this would impose a security risk. Indeed, using the basic authentication protocol, the testing equipment has no means to verify that the DuT actually received such a stop message and has quit privileged mode. An attacker can block this message (depending on the assumptions regarding the connection between the authentication module of the DuT and the testing equipment), and force the DuT to stay in privileged mode without the testing equipment being able to notice this. If there is no option to end the privileged mode automatically, one could also use more advanced authentication techniques to authenticate the data from the DuT (see section 4.2.2). However, this comes at an extra cost.

4.2.2 Advanced authentication protocol using key agreement

When there are different sets of testing parameters or commands, or when these items have to be updated during the privileged mode, more advanced authentication techniques should be used. One solution could be to sign all these commands by the private key of the testing equipment. These signed messages would include a counter which is increased for every new message. However, this problem would not solve the security issue where the DuT is forced to stay into privileged mode. Therefore, we propose a key agreement protocol that both allows to authenticate multiple commands or parameters, and solve the aforementioned security problem. By using key agreement protocols, a secure channel is created between the testing equipment and the authentication module of the DuT, and man-in-the-middle attacks[1] are prevented. Below, we will describe two variants of a key agreement protocol: (1) two-sided ECC-based Diffie-Hellman which gives the highest security guarantees, but requires extra security assumptions, or (2) one-sided ECC-based Diffie-Hellman which does not require extra security assumptions, but is vulnerable to a specific Denial-of-Service attack.

Two-sided Diffie-Hellman

In this case, the authentication module also needs to have a public/private key pair (respectively denoted by PK and SK). The certificate of the corresponding public key needs to be signed with the private key of the chip designer. Both parties can carry out an ECC-based Diffie-Hellman [8] key exchange protocol, to derive a shared, temporary key. This key is then

[1] A man-in-the-middle attack can be informally defined as a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker is able to intercept and modify all messages going between the two victims and inject new ones.

used to authenticate the commands sent to the authentication module, via a Message Authentication Code (MAC). Only commands that contain a correct MAC, will be accepted by the authentication module. To protect replay attacks, the commands should contain a sequence counter, which is increased in every packet[2]. Only when the sequence counter is higher than the last value that was used, the message gets accepted. The sequence counter in the authenticated message cannot be modified by an attacker, since it is also protected by the MAC. The length of the counter should be sufficiently high to avoid overflows. Since the temporary MAC key will only be valid during the matching phase, the number of packets exchanged will be rather limited. Therefore, also the bit length of the counter will be rather small.

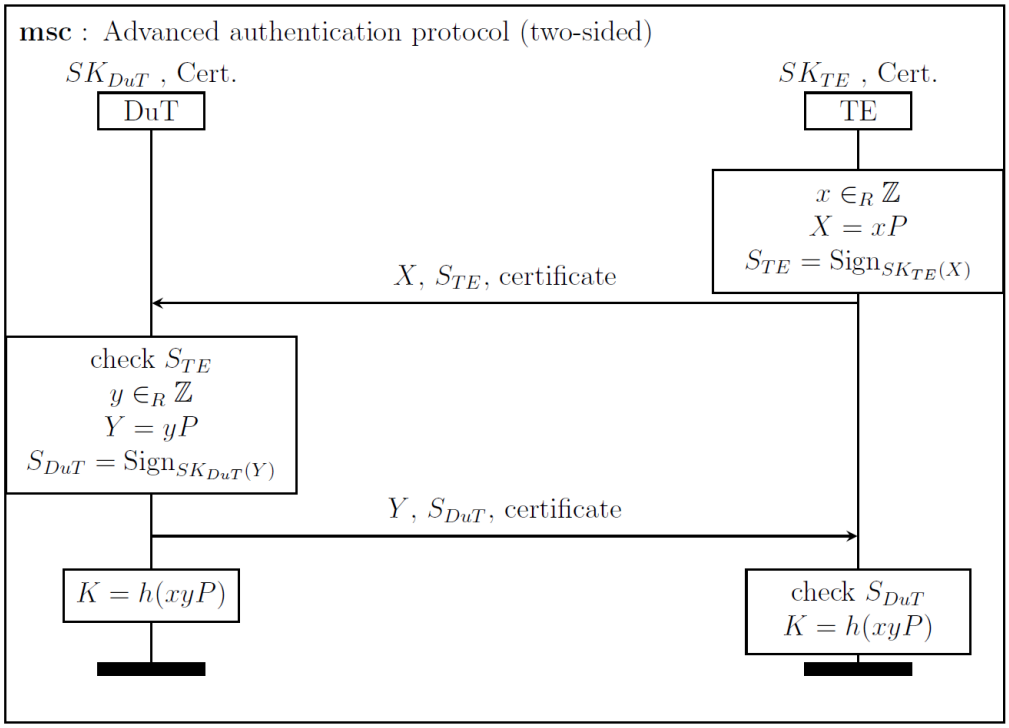

Figure 4.3: Two-sided ECC-based Diffie-Hellman key agreement protocol

It is important to note that the Diffie-Hellman protocol discussed below just serves as an example. One could substitute it by other key agreement protocols, a variety of which have been standardized in the smart card environment. The two-sided ECC-based Diffie-Hellman protocol which will be described more in detail in this deliverable, and which is used to agree on a shared temporary MAC key, is shown in Fig. 4.3 and carried out as follows. The testing equipment (TE) generates a random secret x, computes the corresponding Diffie-Hellman (DH) public key $X = xP$, and signs this DH public key X with its private signing key SK_{TE}. The DH public key X, the signature and the certificate on the public key of the testing equipment are all sent to the DuT. After receiving this value and checking its validity, the DuT performs similar operations: it generates a random secret y, computes the corresponding Diffie-Hellman (DH) public key $Y = yP$, and signs this DH public key Y with its private signing key SK_{DuT}. The DH public key Y, the signature and the certificate on the public key of the DuT are all

[2]Instead of using a sequence counter, one could also deploy message chaining to ensure the freshness of the messages.

sent back to the testing equipment, which then verifies the validity of the received value. Both devices now compute xyP and use a key derivation function h (e.g., based on a cryptographic hash function) to compute the shared, temporary key K. This key K is the temporary MAC key, used to authenticate commands.

To end the privileged mode, the testing equipment sends an authenticated "stop command" (protected via the counter and the MAC) to the authentication module. When this command is authentic and fresh, the DuT will leave privileged mode. Next (only after having ended privileged mode), it will send an authenticated acknowledgement (also including the counter, and protected with the MAC) back to the testing equipment. This way, the testing equipment knows that the DuT has successfully stopped working in privileged mode. After having sent the authenticated acknowledgement, the DuT will change the state of the temporary MAC key to 'expired'. From this moment on, the temporary MAC key can only be used to create an authenticated acknowledgement to prove it has quit privileged mode (see later). From the moment a new temporary MAC key is agreed on, by executing the Diffie-Hellman protocol, the 'expired' MAC key is overwritten and the state of this new key is changed to 'valid'. If the testing equipment does not receive such an authenticated acknowledgement, it will resend (using a fresh counter) the "stop command" to the authentication module of the DuT. If the latter receives such as "stop command" while it has already quit this mode, it will reply with an authenticated acknowledgement, containing a fresh value of the sequence counter. If the testing equipment does not receive an authenticated acknowledgement after several attempts, the testing equipment can conclude that the DuT is under attack or not working properly.

One-sided Diffie-Hellman

The disadvantage of using the two-sided Diffie-Hellman protocol, is that the authentication module should store a private key in secure memory. As will be discussed briefly in Sect. 4.3, a Physically Unclonable Function (PUF) could be used to store this key. Another approach is to use the one-sided Diffie-Hellman protocol, where the authentication module does not need to have a private key. Only a public key PK_{TE} needs to be stored securely (i.e. it cannot be altered) within this module. The protocol is shown in Fig. 4.4 and works as follows. The testing equipment generates a random secret x, computes the corresponding Diffie-Hellman public key $X = xP$, and signs this public key with its private signing key SK_{TE}. Note that this private key corresponds to the public key PK_{TE} which is hard-coded in the authentication module. Both the public key X and the signature on this key are sent to the authentication module. When receiving this value, the authentication module first checks the authenticity of X by verifying the signature, using the hard-coded public key PK_{TE} in its memory. If this verification succeeds, the authentication module will generate a random secret y, compute the corresponding Diffie-Hellman public key $Y = yP$ and send this public key back to the testing equipment. Both devices now compute xyP and use a key derivation function h (e.g., based on a cryptographic hash function) to compute the shared, temporary key K. As before, this key K is used to authenticate the commands sent to the authentication module, via a MAC.

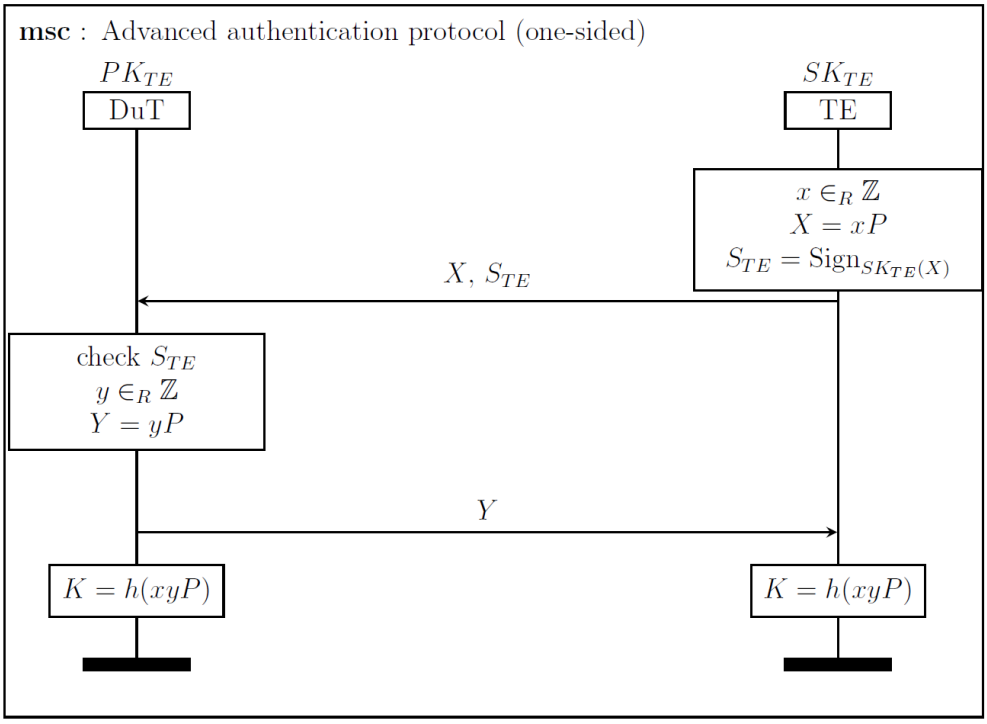

Figure 4.4: One-sided ECC-based Diffie-Hellman key agreement protocol

By using this protocol, only the authentication module can check that the Diffie-Hellman public key X its receives, indeed was sent by the testing equipment. As a result, only the testing equipment can agree on a shared key with the authentication module. Messages from other parties will be ignored. Therefore, only the testing equipment will be able to send authenticated commands to the authentication module (such as the command to enter privileged mode). However, the testing equipment has no assurance that it effectively shares a key with the authentication module. There are two main scenarios:

- It shares a key with a man-in-the-middle attacker. In this case, the testing equipment knows that this man-in-the-middle attacker cannot activate privileged mode on the DuT. So in all circumstances, the DuT will be in normal, secure mode (since it was impossible to activate privileged mode). However, the HT detection approach cannot be applied. This Denial-of-Service attack cannot be avoided. However, note that this attack is also possible when using one of the other protocols described in this section (by blocking all commands that are sent to the DuT).

- It shares a key with the authentication module of the DuT. Only in this case, the DuT can enter privileged mode. When it receives a secure acknowledgement, it knows that this message can only originate from the authentication module of the DuT, and therefore, it knows that the DuT has quit privileged mode.

Since it is impossible to activate the privileged mode on the DuT and in the same time fool the testing equipment that the device has quite this mode while it has not, the security issue of the basic authentication protocol is mitigated.

4.3 Chip authentication

In Section 4.2, we discussed how the testing equipment could authenticate itself to the DuT during the matching phase. This way, unauthorized parties are not able to force the DuT to enter or stay in privileged mode. Although not the main focus of this work package, it might be interesting to briefly discuss the use case where the DuT authenticates itself to the testing equipment. This use case is particularly relevant when one wants to detect fake chips.

Chip authentication can be achieved by using the two-sided Diffie-Hellman protocol, as discussed in Section 4.2.2. After executing this protocol, the testing equipment and the DuT share a secret key K. Since both parties sign their ephemeral Diffie-Hellman (DH) public key using their private signing key, man-in-the-middle attacks are prevented and both parties know they share a key with the other parties. An impersonator will not have a valid certificate for its private signing key, and as a result, will not be able to agree on a shared key K with the testing equipment. Note that another mutual authentication protocol, besides the two-sided Diffie-Hellman protocol, would also work.

Of course, the two-sided Diffie-Hellman protocol only provides protection against fake chips if one can prevent the private keys (of a genuine DuT) being copied to a fake device, for example via hardware attacks. Fortunately, this copying of keys can be prevented using Physically Unclonable Functions (PUFs), which are studied in WP2 of the HINT project. The private key used to sign the ephemeral DH public keys is protected using memory encryption. Memory encryption is necessary since the non-cryptographic countermeasures are not sufficient to resist current attack potential required for Common Criteria certification for a high security level. With PUF technology, this memory encryption key could be "encapsulated" into a PUF construction (e.g., based on memory cells), retrieved whenever the authentication module of the DuT is initialized, and stored in volatile memory only. Attacks on memory encryption targeting the encryption key (i.e. trying to retrieve it from physical memory) would then not have to be considered (unless attacking the PUF itself).

By combining the PUF technology of WP2, and the authentication protocols and Hardware-Trojan detection methodologies of WP3, a holistic approach for integrity verification can be achieved. This would offer protection against both the fake-chip scenario as against Hardware-Trojans inserted in a device in the field.

4.4 Evaluation

To conclude this section, we will now give an overview of how the countermeasures described above cover the risks listed in Section 3.3.

Risk #	**Mitigation technique(s)**
RGEA_01	Authentic configuration and reference data
RGEA_02	Authentic configuration and reference data
RGEA_03	Authentic configuration and reference data
RGEA_04	Risk is minimized by proper design of testing method
RGEA_05	Risk is minimized by proper design of testing method
RGEA_06	Authentic configuration and reference data Side-channel security mechanisms in DuT (*Passive approaches only*)

Risk #	**Mitigation technique(s)**
RGEA_07	Authentic configuration and reference data
RPAC_01	Authentic configuration and reference data
RPAC_02	Authentic configuration and reference data
RPAC_03	Authentic configuration and reference data
RPAC_04	Authentic configuration and reference data Side-channel security mechanisms in DuT
RPAC_05	Authentic configuration and reference data
RAAP_01	Advanced authentication protocol
RAAP_02	(Basic or) advanced authentication protocol
RAAP_03	Advanced authentication protocol
RAAP_04	(Basic or) advanced authentication protocol
RAAP_05	(Basic or) advanced authentication protocol
RAAP_06	Advanced authentication protocol
RAAP_07	Advanced authentication protocol
RAAP_08	Advanced authentication protocol

Table 4.1: Overview of mitigation techniques of risks identified in Section 3

Chapter 5 Conclusion

In this document we analysed the risks linked to the HT detection schemes described in Chapter 2 and from this analysis we described adequate countermeasures to mitigate those risks. We now summarize how to implement the passive and active HT detection approaches integrating those countermeasures (as summarized in Section 4.4). We hence realize that deploying HT detection schemes in highly security sensitive use cases is not as simple and straight-forward as suggested by most (if not all) research works on the subject and that the impact of deploying such schemes onto the certification processes might be substantial (This shall be addressed in WP5).

5.1 'Secure' Passive Approach

- Run 'Authentication of configuration & reference data' scheme as illustrated in Figure 4.1.
- Run 'Passive HT detection scheme' as described in Sections 2.1.1 & 2.1.2.

5.2 'Secure' Active Approach

- Run 'Authentication of configuration & reference data' scheme as illustrated in Figure 4.1.
- Authenticate the testing equipment using the 'Advanced authentication protocol: one-sided DH' described in Section 4.2.2.
- (*OPTIONAL*) Authenticate the DuT using the embedded PUF technology (Section 4.3).
- Run the 'Active detection scheme' as described in Sections 2.2.1 & 2.2.2 in authenticated mode.
- End advanced authenticated mode by sending a 'stop privileged command' (Section 4.2.2).

Bibliography

[1] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic. Detecting trojans through leakage current analysis using multiple supply pad I_{ddq} s. Information Forensics and Security, IEEE Transactions on, 5(4):893–904, Dec 2010.

[2] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When clocks fail: On critical paths and clock faults. In Proceedings of the 9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application, CARDIS'10, pages 182–193, Berlin, Heidelberg, 2010. Springer-Verlag.

[3] Dakshi Agrawal, Selçuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan detection using ic fingerprinting. In IEEE Symposium on Security and Privacy, pages 296–310, 2007.

[4] Mainak Banga and Michael S. Hsiao. A region based approach for the identification of hardware trojans. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST '08, pages 40–47, Washington, DC, USA, 2008. IEEE Computer Society.

[5] Chongxi Bao, D. Forte, and A. Srivastava. On application of one-class svm to reverse engineering-based hardware trojan detection. In Quality Electronic Design (ISQED), 2014 15th International Symposium on, pages 47–54, March 2014.

[6] Byeongju Cha and Sandeep K. Gupta. Trojan detection via delay measurements: A new approach to select paths and vectors to maximize effectiveness and minimize cost. In Proceedings of the Conference on Design, Automation and Test in Europe, DATE '13, pages 1265–1270, San Jose, CA, USA, 2013. EDA Consortium.

[7] Giorgio Di Natale, Sophie Dupuis, and Bruno Rouzeyre. Is Side-Channel Analysis really reliable for detecting Hardware Trojans? In DCIS'2012: XVII Conference on Design of Circuits and Integrated Systems, pages 238–242, Avignon, France, November 2012.

[8] Whitfield Diffie and Martin Hellman. New directions in cryptography. In IEEE Transactions on Information Theory, volume 22, pages 644–654. IEEE, 1976.

[9] Dongdong Du, Seetharam Narasimhan, Rajat Subhra Chakraborty, and Swarup Bhunia. Self-referencing: A scalable side-channel approach for hardware trojan detection. In Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems, CHES'10, pages 173–187, Berlin, Heidelberg, 2010. Springer-Verlag.

[10] Ingrid Exurville, Jacques Fournier, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Practical measurements of data path delays for ip authentication & integrity verification. In Proceedings of ReCoSoC, pages 1–6, 2013.

[11] Yier Jin and Yiorgos Makris. Hardware trojan detection using path delay fingerprint. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST '08, pages 51–57, Washington, DC, USA, 2008. IEEE Computer Society.

[12] P. Kumar and R. Srinivasan. Detection of hardware trojan in sea using path delay. In Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students' Conference on, pages 1–6, March 2014.

[13] A. Lakshminarasimhan. Electromagnetic side-channel analysis for hardware and software watermarking. Master's thesis, University of Massachusetts Amherst, 2011.

[14] C. Lamech, R.M. Rad, M Tehranipoor, and J. Plusquellic. An experimental analysis of power and delay signal-to-noise requirements for detecting trojans and methods for achieving the required detection sensitivities. Information Forensics and Security, IEEE Transactions on, 6(3):1170–1179, Sept 2011.

[15] Jie Li and John Lach. At-speed delay characterization for ic authentication and trojan horse detection. In Mohammad Tehranipoor and Jim Plusquellic, editors, HOST, pages 8–14. IEEE Computer Society, 2008.

[16] Yu Liu, Ke Huang, and Yiorgos Makris. Hardware trojan detection through golden chip-free statistical side-channel fingerprinting. In DAC, pages 1–6, 2014.

[17] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

[18] Michael Muehlberghuber, Frank K. Gürkaynak, Thomas Korak, Philipp Dunst, and Michael Hutter. Red team vs. blue team hardware trojan analysis: Detection of a hardware trojan on an actual asic. In Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy, HASP '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM.

[19] S. Narasimhan, Dongdong Du, R.S. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia. Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach. In Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on, pages 13–18, June 2010.

[20] Seetharam Narasimhan, Xinmu Wang, Dongdong Du, Rajat Subhra Chakraborty, and Swarup Bhunia. Tesr: A robust temporal self-referencing approach for hardware trojan detection. In HOST, pages 71–74, 2011.

[21] Devendra Rai and John Lach. Performance of delay-based trojan detection techniques under parameter variations. In Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust, HST '09, pages 58–65, Washington, DC, USA, 2009. IEEE Computer Society.

[22] Ron Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM,, 21(2):120–126, February 1978.

[23] Claus Peter Schnorr. Efficient identification and signatures for smart cards. In Advances in Cryptology, CRYPTO'89, pages 239–252, Berlin, Heidelberg, 1989. Springer-Verlag.

[24] M. Tehranipoor and C. Wang. Introduction to Hardware Security and Trust. SpringerLink : Bücher. Springer, 2011.

[25] R. Torrance and D. James. The state-of-the-art in semiconductor reverse engineering. In Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE, pages 333–338, June 2011.

[26] American National Standard X9.62-2005. Public key cryptography for the financial services industry, the elliptic curve digital signature algorithm (ecdsa), November 2005. Accredited Standards Committee X9.

[27] Jie Zhang, Haile Yu, and Qiang Xu. Htoutlier: Hardware trojan detection with side-channel signature outlier identification. In HOST, pages 55–58, 2012.