D1.2

Report on specifications and overall architecture

Project number:	317930
Project acronym:	HINT
Project title:	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 / D1.2 / 1.0
Work package contributing to the deliverable:	WP 1
Due date:	May 2013 – M08
Actual submission date:	21st June, 2013

Responsible organisation:	CCS
Editor:	CCS (Marc MOUFFRON)
Dissemination level:	Public
Revision:	1.0

Abstract:	In this document, following the definition of the project's use cases in D1.1 and the identification of their corresponding security requirements, we further detail their security analysis. Then we present two architectures that shall constitute the backbone of the HINT technology: one based on PUFs and the second one based on side channel analysis. Finally we provide a first idea of how we intend to demonstrate those architectures through dedicated prototypes.
Keywords:	Smart ID Cards, Professional Mobile Radio, Security Analysis, Physically Unclonable Functions, Side Channel Analysis, Hardware Trojans, Architecture, Application prototypes

Editor

Marc Mouffron (CCS)

Contributors

Thomas Hübner (MOR)

Carsten Rust (MOR)

Holger Bock (IFAT)

Jacques Fournier (CEA)

Julien Francq (CCS)

Verena Brunner (TEC)

Martin Deutschmann (TEC)

Jean-Baptiste Rigaud (ARMINES)

Dave Singelée (KUL)

Jens Hermans (KUL)

Disclaimer

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 317930.

Executive Summary

This report on specification and overall architecture is the second and final deliverable of WP1 of the HINT project.

The role of this deliverable is to define the overall trust architectures and their use in the application prototypes. It includes descriptions of:

- a complement on use cases and security analysis. This analysis focuses on the PUF Authentication and PUF-based signature for ID-cards and on Hardware Trojan detection on PMR handhelds.
- a trust architecture based on the two building blocks: PUF anchors and SCA detection of Hardware Trojan. The PUF-based architecture explains the PUF integration in the ID-card and the required management to achieve authentication and signature capacity. The SCA-based architecture sets up the environment and tool chain for Hardware Trojan inclusion and detection on a FPGA board.
- the overall architecture for the two HINT application prototypes. The Unclonable ID-card application shows the whole lifetime management of a PUF-based ID-card. The PMR prototype evinces the modus vivendi of HT detection on an actual product.

This report reflects contributions from the HINT partners and provides the technical specifications for the study and development work of next WPs.

The two roots of trust in HINT are the PUF-based and SCA-based technologies. They complement each other and could have been combined in a single application. Nevertheless due to the intrinsic novelty and complexity of each technology the HINT partners will demonstrate them separately. Each of the following WPs will be dedicated to one point specifically. The PUF-based architecture intended for use in ID-card set forth the work on WP2. The SCA-based architecture defines the content of WP3 to design and detect the Hardware Trojans. The outputs of WP2 and WP3 provide the basis of the two application prototypes ID-card with PUF-based authentication features and PMR prototype with HT demonstration and detection.

This WP1 has stated the usages and stakes of the overall trust architectures. The preliminary definitions of the PUF-based architecture and the SCA-based architecture are set out. They would be improved through the work on the next WPs as results are obtained. This would be finally demonstrated by the two application prototypes defined in the last chapter.

Contents

List of acronyms

AES	**Advanced Encryption Standard**
ASIC	**Application Specific Integrated Circuit**
BCH	**Bose Ray-Chandhiri Hocquenghen**
CC	**Common Criteria**
CCRA	**Common Criteria Recognition Arrangement**
CMOS	**Complementary Metal Oxide Semiconductor**
COTS	**Commercial Off-The-Shelf**
CPLD	**Complex Programmable Logic Device**
CRP	**Challenge-Response Pair**
CRTM	**Core Root of Trust for Measurement**
CTRL	**Control**
DEMA	**Differential Electromagnetic Analysis**
DES	**Data Encryption Standard**
DH	**Diffie Hellman**
DSA	**Digital Signature Algorithm**
DSP	**Digital Signal Processing**
DUT	**Device Under Test**
EAL	**Evaluation Assurance Level**
ECC	**Error Correcting Code**
ECDH	**Elliptic Curve Diffie-Hellman**
ECDSA	**Elliptic Curve Digital Signature Algorithm**
EEPROM	**Electrically Erasable Programmable Read Only Memory**
EM	**ElectroMagnetic**
FE	**Fuzzy Extractor**
FPGA	**Field Programmable Gate Array**
FSM	**Finite State Machine**
GPU	**Graphics Processing Unit**
HT	**Hardware Trojan**
HTIE	**Holistic Tool for Information Extraction**
HW	**Hardware**
IAS	**Integrity Authentication Scheme**
IC	**Integrated Circuit**
ICT	**Information and Communications Technology**
ID	**Identity**

IP	**Intellectual Property**
IV	**Input Vector**
ML	**Machine Learning**
NVM	**Non-Volatile Memory**
OS	**Operating System**
PIN	**Personal Identification Number**
PMR	**Professional Mobile Radiocommunications**
PUF	**Physically Unclonable Function**
RAM	**Random Access Memory**
RNG	**Random Number Generator**
RSA	**Rivest Shamir Adleman**
RT	**Real-Time**
SCA	**Side Channel Analysis**
SFR	**Security Functional Requirement**
SRAM	**Static Random Access Memory**
SW	**Software**
TC	**Trusted Computing**
TOE	**Target of Evaluation**
TPM	**Trusted Platform Module**
USB	**Universal Serial Bus**
WP	**Work Package**

List of Figures

List of Tables

Chapter 1 Introduction

The aim of the HINT project is the study of novel schemes for checking that a system or a device is genuine.

The counterfeiting or the infection by Hardware Trojans (HT) of Integrated Circuits (IC) can apply to Application Specific Integrated Circuits (ASICs), Commercial Off-The-Shelf (COTS) devices as well as to Field Programmable Gate Arrays (FPGA). In the HINT project, the former ones (ASICs, COTS) are studied through the ID-card use case and the latter one (FPGA) is studied through the PMR use case. Both use cases have already been introduced in the deliverable D1.1 [REF 1].

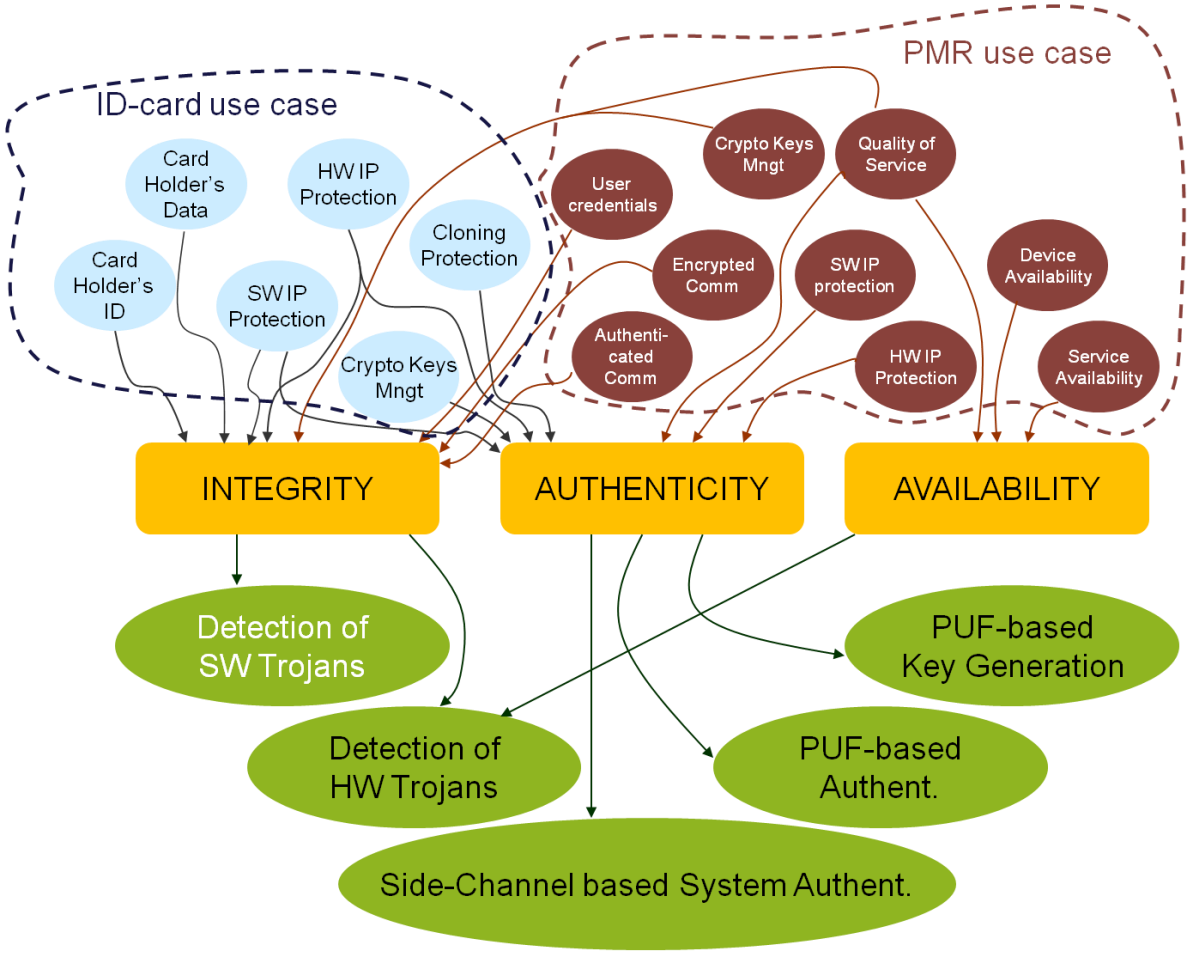

Figure 1: Summary of HINT requirements

The requirements for these two use cases are categorized with regards to the three overarching objectives pursued within HINT technologies:

- Integrity
- Authenticity
- Availability

These requirements allowed us to identify several relevant technologies pertaining to the scheme developed in the HINT project. Part of the scheme was suggested in D1.1 [REF 1]

and we detail here the technical domains that are going to be investigated during the next phases of the HINT project.

The core idea for the HINT project is to efficiently and securely measure some form of “fingerprint” from a given IC, based on two technologies: PUF and SCA measurements. Even if they differ in their nature, when combined, they complement each other to achieve IC assurance and hence devices assurance.

Verifying IC trustworthiness can be done first of all as a post-manufacturing step to validate conformance of the fabricated IC to the original functional and performance specifications but this may not achieve a sufficient coverage. So the HINT project adds capability checking after delivery of the product to the user, which implies “on-the-field” or “at-time-of-use” checking.

Hence the architecture stemming from the requirement of D1.1 needs to prepare this "after delivery checking" capacity. New scenarios demonstrating such capabilities developed in the HINT project will be elaborated, pertaining to the “after delivery” trust in devices.

The two complementary technologies: PUF-based and SCA-based could have been combined in a single application. Nevertheless due to their intrinsic nature, novelty and complexity the HINT partners will demonstrate them separately. Furthermore due to these differences in nature the approaches to build the trust architectures and the application prototype will be quite specific to each of them.

This document is organized as follows. We first focus on refining specific security requirements pertaining to the two use cases defined in D1.1 [REF 1]. This part shall be read as a supplement of the security analysis of D1.1. Then the core of this document will be the trusted HINT architecture definition following the two primary technologies identified for performing authentication and integrity verification an IC. This chapter states the environment needed to build each architecture and specifies the principles of that architecture. An assessment analysis concludes each architecture description. The "HINT Application prototypes" are then exposed with their main characteristics and their main challenges.

Chapter 2 HINT Use Cases Security Analysis

A security analysis of the two main HINT application scenarios “Unclonable ID-cards” and “Professional Mobile Radio” has already been presented in Deliverable D1.1 [REF 1], Sections 3.2 and 3.3 respectively. The scenarios were analysed following the well-established methodology for Common Criteria security evaluations and through the mean of Protection Profiles. The Section 2.1 gives a brief overview of Protection Profiles in the context of HINT project. Accordingly, we described the main assets of an ID-card, as it is deployed in the HINT usage scenarios, and outlined possible attack scenarios. These analysis results can be used as a basis for specifying a Protection Profile for an ID-card with PUF technology. The following Section 2.2 complements the analysis from the previous deliverable. We analyse security aspects related to the integration of PUFs into ID-cards. This includes on the one hand assets enabled by the deployment of PUF technology. On the other hand, we consider security issues and vulnerabilities of the PUF implementation and operation itself. Moreover, we give a general description, how to define a protection profile for an ID-card product with integrated PUF-technology.

For the second HINT application scenario on PMR, the security analysis was also conducted in accordance with the Common Criteria methodology, in this case in an even more rigid way than for the ID-card scenario. As already explained in D1.1, HINT aims at preparing a protection profile covering the new aspects of integrity and authenticity in the PMR context. Therefore, we presented an extensive formal analysis for PMR in Deliverable D1.1. Section 2.2 in this document completes the analysis. We analyse the coverage of threats by objectives as well as of objectives by selected Security Functional Requirements.

2.1 HINT contribution to Protection Profiles

In the context of secure devices development, especially (but not exclusively) for high-security smart cards such as ID-cards, Protection Profiles are well established in the Common Criteria methodology since many years now.

The notion of a “Protection Profile” has been introduced into the Common Criteria approach as a kind of “generic Security Target”, embracing the security architecture of a group of related product families such as smart cards. The developer designing product (a smart card or a PMR handheld) and in charge of writing the corresponding Security Target can then reference the associated Protection Profile from which he derives his own security requirements.

Not only does this approach ease the burden of writing a Security Target for the developer, but it also facilitates comparison of products from different vendors with regard to their security architecture.

Protection Profiles (in contrast to vendor-specific Security Targets) are public documents, and often bear a certificate of their own. Examples for existing and commonly referenced Protection Profiles pertaining to smart cards or PMR handhelds with a high security evaluation assurance level are listed below.

- *Protection Profile for smart cards with signature creation [REF 2]*. Here, an on-board key generation constitutes one of the most attractive applications, since each card holder can be certain that his or her signing key is never known to any other entity than his /her card (even not known to the card issuer, or to any issuing authority, not even to the holder herself!).

- *Protection Profile for smart card based travel documents [REF 3].*
- *Protection Profile for ICs Platform [REF 4].* This is a Protection Profile for hardware vendors. In light of the PUF technology, such a profile would have to be augmented by security aspects and vulnerability analysis regarding the safe implementation of PUFs. SCA based detection of HT could also be added to such a Protection Profile.
- *Java Card Protection Profile [REF 5].*
- *Cryptographic Modules, Security Level "Enhanced" [REF 6].* This Protection Profile is relevant for a cryptographic module embedded in a PMR handheld and could be augmented by security aspects related to safe implementation of PUFs and the SCA-based detection of HT.

In September 2012 the CC Management Committee published a CCRA (Common Criteria Recognition Arrangement) vision statement embracing the concept of "*Collaborative Protection Profiles*" [REF 7]. The aim of this initiative is to facilitate mutual recognition of Protection Profiles (hitherto written and certified on a national level) and harmonize the development and certification process. Mutual recognition of smart card products (and smart card Protection Profiles) is a well known issue and will become increasingly difficult as the assurance level of mutually recognized products is lowered to EAL (Evaluation Assurance Level) 2. Hence, the concept of Collaborative Protection Profiles paves the way for the development of products with a high security level (like EAL 4 or EAL 5) in the Common Criteria framework of certification.

As a consequence, for anyone intending to write a Protection Profile we recommend to envisage the concept of a Collaborative Protection Profile, especially for ID-card products with integrated PUF technology or PMR handheld with SCA-based HT detection.

2.2 Security Functions for the HINT ID-Card Use Case

2.2.1 Specification of Security Functionality enabled through PUF technology

For both applications envisaged in the HINT application prototype, i.e. PUF authentication and PUF-based signature, the PUF deployment shall enable us to spare a cryptographic key (authentication key or signature key) explicitly stored in a smart cards non-volatile memory. Attacks on such a key which extract it from EEPROM of Flash memory (probing attacks, invasive attacks, etc.) will then no longer apply as long as they rely on the key to be stored in dedicated persistent memory.

2.2.1.1 Solely PUF-based Authentication of an ID-card

Authentication of an ID-card through PUF technology can be based on a challenge-response protocol; with a *strong* PUF implemented in the smart card IC, processing challenges inside the card IC and producing a response to be verified outside the card. An example for such a PUF is an arbiter PUF allowing sufficiently many challenge bits. Authentication may alternatively be achieved with a *weak* PUF by extracting a key and using it for whatever authentication protocol is desired.

For secure authentication, it is state-of-the-art to use a cryptographic protocol which relies on a secret key. Both symmetric and asymmetric authentication protocols exist, but in each case the underlying secret key must be stored on the card in a secure manner. Such a key is usually stored in non-volatile memory like EEPROM or Flash. Using a weak PUF, a cryptographic key may be extracted (or embedded) into a template for subsequent use. In case of a *symmetric* protocol (where both partners in an authentication need to share the

secret key), either an *externally generated* key could be embedded, or an *extracted* key needs to be output by the ID-card in a secure environment during registration.

In case of authentication with *strong* PUFs no dedicated secret key will be embedded or extracted. A simple protocol step would look like the following.

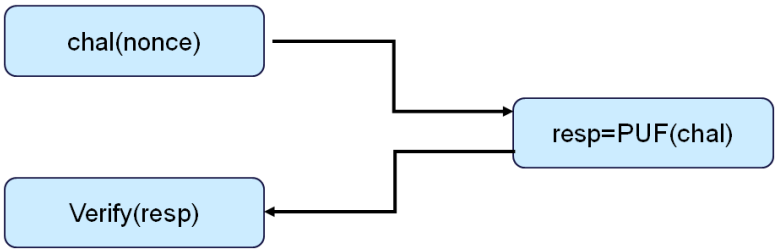

Figure 2: Solely PUF-based challenge-response protocol

Instead of the extraction of a dedicated key, a number of *challenge-response pairs* is computed in advance and safely stored for later access by the verifier. In a certain sense, this list would "replace" the secret authentication key, and thus would certainly constitute a valuable asset. However, such a list would not be stored on the smart card, but maintained on some server's site, so it does not affect a smart card Protection Profile. On the other hand, the robust generation of the response by the PUF constitutes an asset (to be treated in the chapter concerning the security of the PUF itself). Moreover the safe output of the PUF response is a security concern (for instance in terms of attacks aiming at a denial of service through altering the correct PUF response prior to output).

2.2.1.2 PUF-Based Signature

Signature creation is one of the most essential applications for high secure smart cards. Signatures are used for authentication of (signed) "documents" as well as for authentication in the framework of challenge response protocols.

Digital signatures deploy asymmetric cryptography which means that the smart card contains a private signing key which needs to be kept absolutely confidential in order to avoid a forger to generate any signature on behalf of the legitimate key owner. This private signing key up to now has usually been stored in non-volatile memory (EEPROM, Flash), possibly in an encrypted or masked manner. Nonetheless, a variety of attacks exist trying to read out such a key.

The PUF-based signature application envisaged for HINT will make use of extractors which allow to safely "hide" such a key in a protected template, which is stored in the card's persistent memory instead of the signing key. This template is "locked" with a PUF-response in a way that compromise of the template does not allow an attacker to retrieve the embedded key without knowing a corresponding PUF response.

Some new security issues need to be considered in such a situation. To begin with, like in the authentication application the secret signing key is never stored in non-volatile memory, thus thwarting attacks aiming at reading out memory-cells from memory while a card is not active. However, unlike for the authentication based on PUF challenge-response pairs, the signing key does exist *temporarily* whenever it is extracted from the locked template using a PUF response, when it is used in a cryptographic operation, loaded into registers, etc., until the moment of its safe deletion from RAM. Hence, the security requirements of an associated Protection Profile need to be carefully examined with regard to their applicability to the PUF-based signature process.

Another new aspect is the extraction process, which might itself offer further vulnerabilities that need to be treated by an augmentation of security objectives and requirements. This process affects both the PUF operation and the resulting extracted signing key.

Finally, the process of PUF response generation itself (needed to “open” the locked key-template) is covered in the section dealing with PUF security (see section 2.2.2).

2.2.1.3 PUF-Based Authentication Making Use of Signature Schemes

In case all architectural components needed for the key extraction by means of a weak PUF and for signature generation are available, it is also possible to make use of such an extracted key and components for an asymmetric challenge-response protocol, as it has been used in classical crypto systems. The difference compared to classical schemes lies in the fact, that the “secret” does not need to be stored in non-volatile memory as long as the semiconductor device is not powered.

Authentication based on signature generation relying on keys extracted from weak PUFs look like the following.

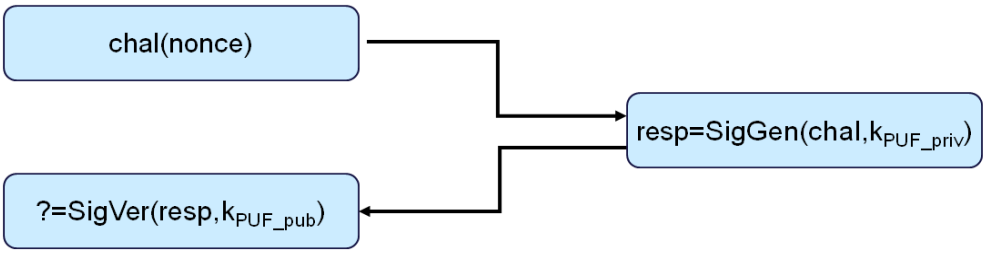

Figure 3: PUF-based challenge-response protocol using signature functions

In general such authentication schemes using signatures may be based on any available signature algorithm, may this be RSA, DSA, ECDSA or any other signature algorithm already implemented in the ID-card software.

Using asymmetric cryptography reduces the effort for handling secrets tremendously, instead of reading out a variety of responses that have to be kept secret over the life cycle of a PUF-based product. The verifier only needs to use an authentic public key corresponding to the PUF-based private key for her verification step of the protocol.

2.2.1.4 PUF-Based Authentication Making Use of Diffie-Hellman Variants

As an optimization of overhead usually present in standard signature algorithms some schemes variants of the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol might be used. Such protocol variants provide extremely efficient prover actions shifting big portions of the calculation effort to the challenger/verifier side.

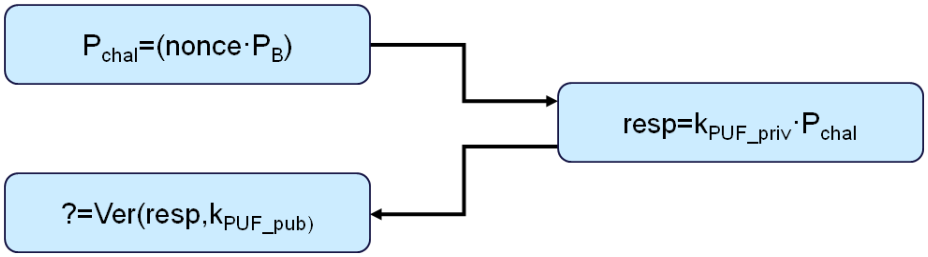

Figure 4: PUF-based challenge-response protocol using Diffie-Hellman variants

Again in such case the verifier only needs an authentic public key corresponding to the private key generated from the PUF response.

2.2.2 Security Features concerning the PUF itself

While the preceding section dealt with security aspects implied by the deployment of PUF technology (replacement of an authentication key by PUF challenge-response pairs, or extraction of a private signing key instead of persistent storage), this section will cover security issues and vulnerabilities of the PUF implementation and operation *itself*. Due to the different nature of PUFs to be implemented for the two selected applications for the ID-card use cases they will be treated separately in two sub-paragraphs.

For typical cryptographic algorithms, analytic attacks will usually not be dealt with in a Protection Profile, since the choice of algorithm and configuration parameters (like key-lengths) is deferred from guidelines or other regulation. The implementation of a dedicated cryptographic hardware supporting computation however is usually covered by IC Protection Profiles.

For a PUF design, many aspects are intrinsically hardware-design dependent, and no standards are known as yet (comparable to cryptographic standards like DES, AES, RSA, ECDSA, etc). Therefore, an associated Protection Profile (and much more so a derived Security Target for a specific product) will need substantial amendments covering hardware security aspects of a PUF implementation. In light of the very different physical nature of PUFs, it appears to be a challenging task to however summarize their properties into a common Protection Profile.

2.2.2.1 PUF-Authentication of an ID-card

A number of attacks on strong challenge-response PUFs are already known in the literature, and research on such attacks is also part of the HINT project.

Most attacks on PUFs aim at extracting the challenge-response behaviour of a PUF, allowing to predict a PUF's response for a hitherto unprocessed challenge, or in the extreme case enabling the attacker to model the complete PUF enabling him to compute a response for many (if not all) arbitrary challenges.

Another motivation for an attacker might be to disturb the PUF with the intention to make it create false (invalid) responses aiming at a denial of service.

Attacks on strong PUFs may be roughly categorized as follows:

- *Cryptographic attacks*. This category of attacks is comparable in spirit to crypto-analytic attacks on cryptographic algorithms. The attacker may use challenge-response pairs (possibly with the option of choosing challenges, or just being able to monitor them), but does not use any hardware fault intrusion or side-channel observation. In other words, he treats the device at hand as a purely logical operator. Examples for such attacks with regard to PUFs are machine-learning attacks.
- *Side-channel observation.* Here, the attacker is allowed to monitor the PUF operation of a specific device during its operation. Such attacks may be both passive (e.g. measuring electromagnetic emission) and invasive (e.g. probing buses) from a physical point of view. Usually, information obtained through side-channel observation will extremely facilitate subsequent analysis (or even make it superfluous).
- *Fault intrusion*. Here the attacker has the ability to introduce intentional misbehaviour into the PUF operation, or manipulate PUF cells. Although a faulty response may be

a direct aim of an attacker, usually such fault intrusion aims at facilitating subsequent analysis benefitting from dedicated ill-formed responses.

In case that post-processing (like error correction) is used prior to release of the response, vulnerabilities of the post-processor also needs to be taken into account. Depending on the type of implementation (dedicated HW or SW) this aspect may need to be deferred to an OS Protection Profile (see Section 3.3.4).

2.2.2.2 PUF-Based Signature

The attack categories “Side-channel observation” and “Fault intrusion” described in the previous paragraph also apply to the PUF-based signature application. However, due to the use of a Fuzzy Extractor a “weak” PUF may be deployed in place of a strong PUF.

On the other hand, in this application post-processing appears to be mandatory in order to enable robust retrieval of an embedded signing key. Useful private signing keys will raise the need for an embedded key of at least few hundred bits length (like for ECDSA). Hence, robustness of the PUF response as well as the capacity of the post-processing step will constitute a particular asset to be covered by a Protection Profile.

In addition to the attacks to be considered in the previous section, the secure operation of the Fuzzy Extractor constitutes an asset, along with the secret key retrieved from it. In particular, the extractor must be implemented in a way not admitting the leakage of information of either the embedded key or the processed PUF response. Unlike for the challenge-response application, here the secrecy of the PUF response itself constitutes an asset, because it may help an attacker to unlock the protected template which is stored in the card’s persistent memory. In order to deal with this situation, one may:

- Consider the PUF response as a *secret* itself. Notice the difference to the application of Fuzzy Extractors for biometric templates, where a biometric feature (like a fingerprint) cannot be considered as secret, whereas a PUF response of an embedded PUF offers at least the chance to keep it confidential.
- Guarantee that only “valid” PUF responses are fed into the extractor. This might be regarded – in a very vague sense – as the analogue of a “liveliness control” in biometric applications. Since biometric features need to be treated as public, it is essential to guarantee in each verification process that a real “living” trait has been processed and not a fraudulent copy (be it physical or even logical). It is rather unclear though, how such a “liveliness control” could be realized.

2.2.3 HINT contribution to Protection Profiles for unclonable ID-cards

Within the HINT project, it is not intended to write a complete Protection Profile for unclonable ID-cards for a number of reasons. To begin with, there exist already (see the list of examples above in §2.1) a number of established Protection Profiles for ID-cards, so the design of a Protection Profile by itself does not constitute a valuable research topic. Moreover, the introduction of PUF technology into an ID-card will rather augment an existing Protection Profile, whilst it seems more appropriate to consider the amendments instead of writing a complete Protection Profile. Finally, a comprehensive Protection Profile largely depends on the nature of the ID-card (for example a health card, an electronic passport, etc.), which will not be a focus of HINT, anyway.

For the two unclonable ID-cards we have in mind with the HINT project, namely the PUF-Authentication and the PUF-based signature, the security architecture as drafted in an existing Protection Profile (depending on the final product into which the demonstrated

protocols shall be embedded, like for instance an electronic passport) will have to be augmented by two categories of functions, introduced above namely:

1. *Security aspects enabled through PUF technology*. – This category shall describe alternative or additional security functionality enabled through PUF technology.
2. *Security aspects of the PUF implementation itself*. – This category shall describe the secure implementation of the PUF itself, vulnerability of its design, and how to thwart attacks against the PUF implementation/operation.

Since each application involves HW as well as SW development, it is expected that Protection Profiles for both IC products as well as smart card OS need to be augmented accordingly. The security of the PUF implementation clearly focuses on IC-production, though PUF operation (e.g. key extraction) might also be implemented by dedicated SW.

2.3 Security Analysis of the PMR Use Case

2.3.1 Security Analysis

The security analysis was presented in a style of a Protection Profile in D1.1 Section 3.3 [REF 1] to take benefit of its reusability as explained in Section 2.1. We give here a few complement pertaining to PMR use case within HINT scheme. The complete definitions of the following threats, organisational security policies and assumptions can be found in D1.1 Section 3.3 [REF 1].

It is important to notice that the insertion of a HT can have many different impacts so our analysis shall be very general and cannot focus on a specific purpose like authentication and signature similarly to what is done in previous Section 2.2.

We shall also point out that many techniques pertain to HT detection but that in HINT project we only target SCA-based detection of HT. The relevance of SCA for this purpose is explained in Section 3.4.5.

2.3.1.1 The Threats

The previous analysis presents a list of threats:

- T.Denial-of-Service
- T.Information_Leakage
- T.Malfunction
- T.Phys-Tamper

Here we add a new threat to complete the ones already given:

T.Sensitive_Data_Alteration
For instance a HT or other event or action can modify internal nodes or memory contents of the circuit where these data are stored or processed. All the changes of sensitive data such as keys, passwords, PINs or user data have major impacts on the services delivered by the terminal.

2.3.1.2 The Organisational security policies

In D1.1 we already list organisational security policies:

- P.Safe_Programming
- P.Safe_Personalization
- P.Key_Function

Here we add a new organisational security policy to complete the ones already given:

P.Audit_HT

The operational product is submitted to a regular “on-the-field” or “at-time-of-use” check to audit for HT detection.

2.3.1.3 The Assumptions

The assumptions in D1.1 describe the security aspects of the environment in which the PMR products will be used or is intended to be used. They are:

- A.PMR_Manufact
- A.PMR_Delivery
- A.Pers_Agent

2.3.1.4 The Security Objectives

The security objectives for the PMR products were stated in D1.1. They are:

- OT.AC_Perso
- OT.Data_Int
- OT.Data_Conf
- OT.Prot_Abuse_Func
- OT.Prot_Malfunction
- OT.Prot_Availability
- OT.Prot_Inf_Leak
- OT.Prot_Phys_Tamper
- OT.Hardware_Integrity

Here we add a new security objective to complete the ones already given:

OT.Audit_HT

Tools available “on-the-field” or “at-time-of-use” provide detection of Hardware Trojans.

2.3.2 Coverage

According to this description we can define the coverage of threats by objectives and afterward of objectives by selected Security Functional Requirements (SFRs). This is given in next two Tables.

	A. PMR_Manufact	A. PMR_Delivery	A.Perso_Agent	T.Denial-of-Service	T.Information_Leakage	T.Sensitive_Data_Alteration	T.Phys-Tamper	T.Malfunction	P.Safe_Programming	P.Safe_Personalisation	P.Key_Function	P.Audit_HT
OT.AC_Perso		X								X		
OT.Data_Int						X						
OT.Data_Conf					X							
OT.Prot_Abuse_Func				X							X	
OT.Prot_Malfunction								X				
OT.Prot_Availability				X								
OT.Prot_Inf_Leak					X							
OT.Prot_Phys_Tamper						X	X					
OT.Hardware_Integrity						X	X					
OT.Audit_HT												X
OE.PMR_Manufact	X								X			
OE.PMR_Delivery		X								X		
OE.Personalisation			X									

Table 1: Security Problem Coverage

SFR	OT.AC_Perso	OT.Data_Int	OT.Data_Conf	OT.Prot_Abuse_Func	OT.Prot_Malfunction	OT.Prot_Availability	OT.Prot_Inf_Leak	OT.Prot_Phys_Tamper	OT.Hardware_Integrity	OT.Audit_HT
FIA_UAU.2\User User authentication before any action	X	X	X	X						X
FIA_UID.2\User User identification before any action	X	X	X	X						X
FCS_COP.1/Data Cryptographic operation				X			X			
FCS_COP.1/Voice communications Cryptographic operation				X						
FCS_COP.1/Integrity Cryptographic operation			X						X	X
FPT_PHP.2 Notification of physical attack					X			X		
FDP_ACC.1 Subset access control				X						X
FMT_MSA.3 Static attribute initialization									X	X
FMT_MSA.1 Management of security attributes									X	X
FDP_ACF.1 Security attributes based access control	X								X	X
FMT_SMF.1 Specification of management functions									X	X
FMT_SMR.1 Security roles	X			X		X				X
FAU_GEN.1 Audit data generation										X
FAU_GEN.2 User identity association	X									X
FAU_ARP.1 Security alarms						X		X		X
AU_SAA.1 Potential violation analysis						X		X		X

Table 2: Security Objectives for the TOE coverage

Chapter 3 HINT Architecture

In this chapter, we present the architecture of the general system that will be developed in the HINT project. The architecture definition is mainly based on the use case requirements for the HINT application scenarios as well as on the analysis of the core technologies which are within the focus of the project, both described in deliverable D1.1. Before introducing the architecture, we first relate in Section 3.1 the HINT concepts to well-established Trusted Computing principles. We thereby highlight how the project shall enhance the current state-of-the-art in that area. In Section 3.2, we give a general overview of the HINT system that will be developed. We outline the main objectives for this HINT system and clarify several essential terms in the description of objectives and requirements.

As a matter of fact, the HINT architecture relies on the two core technologies considered in the project, Physically Unclonable Functions (PUF) and Side Channel Analysis (SCA).

Section 3.3 describes the architecture of the subsystem based on PUFs. The section first introduces a trust architecture for the application example of an ID-card featuring integrity checks based on PUFs. Trust relationships with regard to PUF technology are described for all involved stakeholders as they were already identified in D1.1. Further subsections of Section 3.3 outline the hardware architecture for a secure device with a PUF module – again along the example of a high-security smart card – and software components required for the post-processing of PUF data. Finally, a first analysis of the specified architecture with regards to reliability and vulnerability is presented.

Section 3.4 describes the SCA-based architecture. We first outline the basic structure of an SCA-based integrity/authenticity check mechanism and consider different possible variants for realizing the main components, e.g. a passive and an active approach for realizing an SCA-based detection component. The following two subsections introduce the general concept and describe the measurement equipment and hardware platform that will most likely be used for prototyping the SCA-based approach as well as the tool chain. Based on these specifications, Subsection 3.4.4 analyses the architectural requirements. Finally, we present a first feasibility analysis for the detection of Hardware Trojans with SCA. The analysis is based on a state-of-the-art review and will be refined during the course of the project, based on the realization and evaluation of the application prototypes.

3.1 HINT & Trusted Computing

The HINT scheme shall offer a possibility to enhance current Trusted Computing (TC) principles. Current TC schemes are based on the presence of a secure element called Trusted Platform Module (TPM) which for example, has the ability to verify the integrity of the software being executed, starting with the boot code that is then used as root of trust for the rest of the software hierarchies. All the TC architecture's trust relies on the hardware used. With the HINT technology, one could for example check the integrity or authenticity of the hardware before executing the rest of the TC scheme for (software) trust establishment.

For example, in the case where a TPM is used to attest the integrity of a system's boot code (and hence for the measurement of the 'core root of trust'), the TC procedure is illustrated in Figure 5.

In this case, an important security assumption is made: the chip/processor doing the Core Root of Trust Measurement (CRTM) can be trusted. However, the CRTM could be corrupted in such a way that even if an authorized boot code is uploaded, the CRTM is made to send to the TPM a pre-stored correct "signature" or generate a signature corresponding to the pre-stored digest of an authorized boot code (Figure 6).

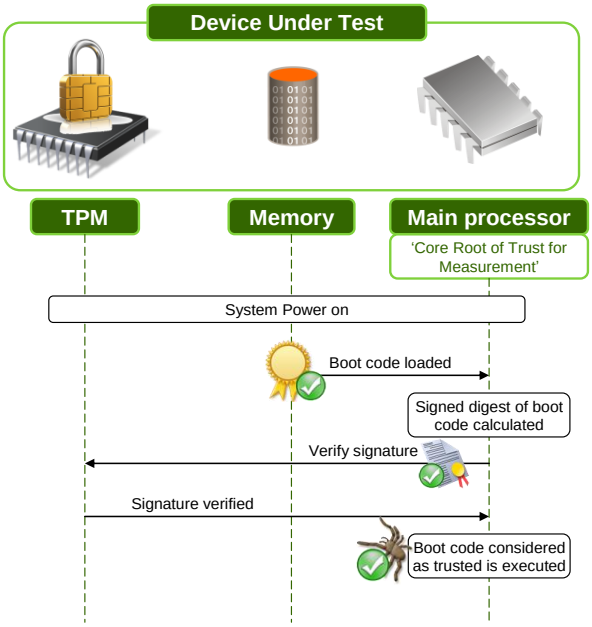

Figure 5: TC scheme for root of trust measurement

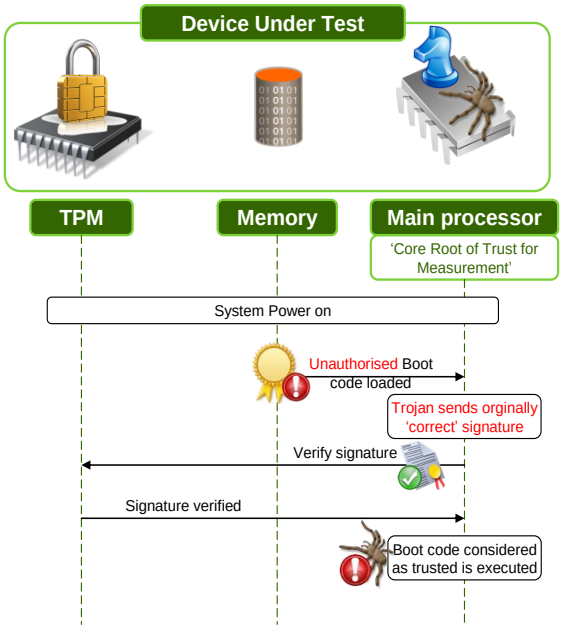

Figure 6: TC scheme for root of trust measurement with infected processor

In this scope, the HINT scheme could be used to verify the integrity/authenticity of the CRTM upon Power On and hence provide the reader / docking station a means to decide whether to proceed with the system's boot or whether it is trusted or not (Figure 7).

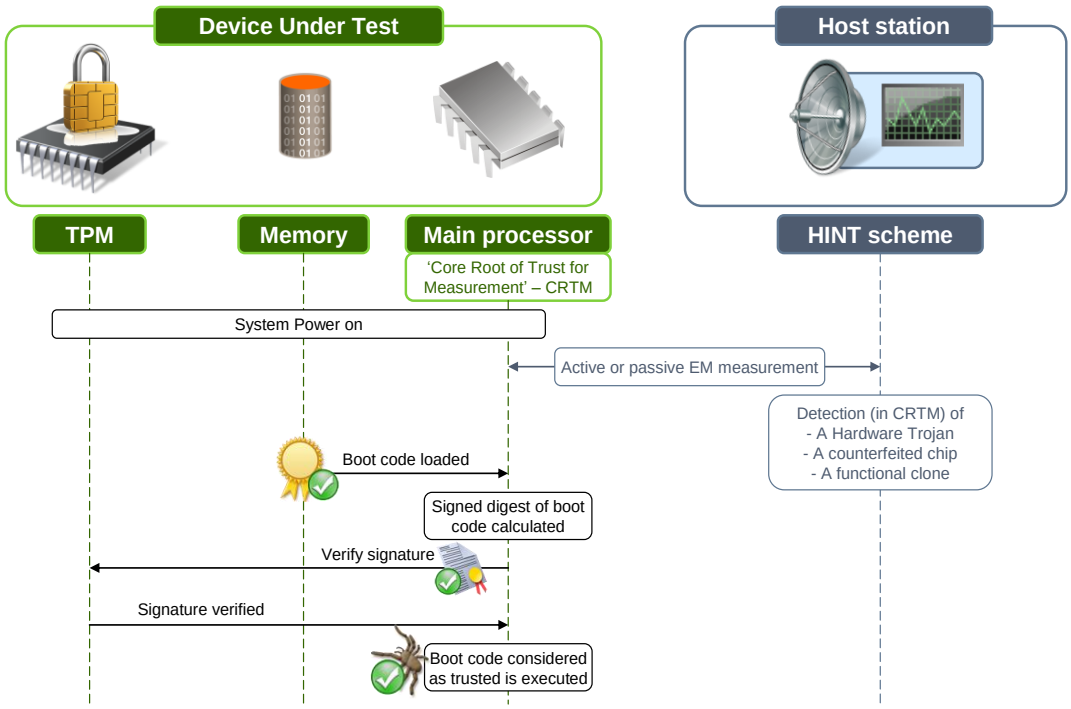

Figure 7: Root of trust measurement with hardware integrity check

3.2 System Overview

The scheme which shall be investigated in the HINT project shall allow “on-the-field” or “at-time-of-use” integrity and authenticity verification of a ‘critical’ device. Actually, several approaches and techniques shall be investigated and proposed resulting to some sort of ‘toolbox’ from which one (or a composition) of those approaches can be integrated by an end user (secure systems integrator).

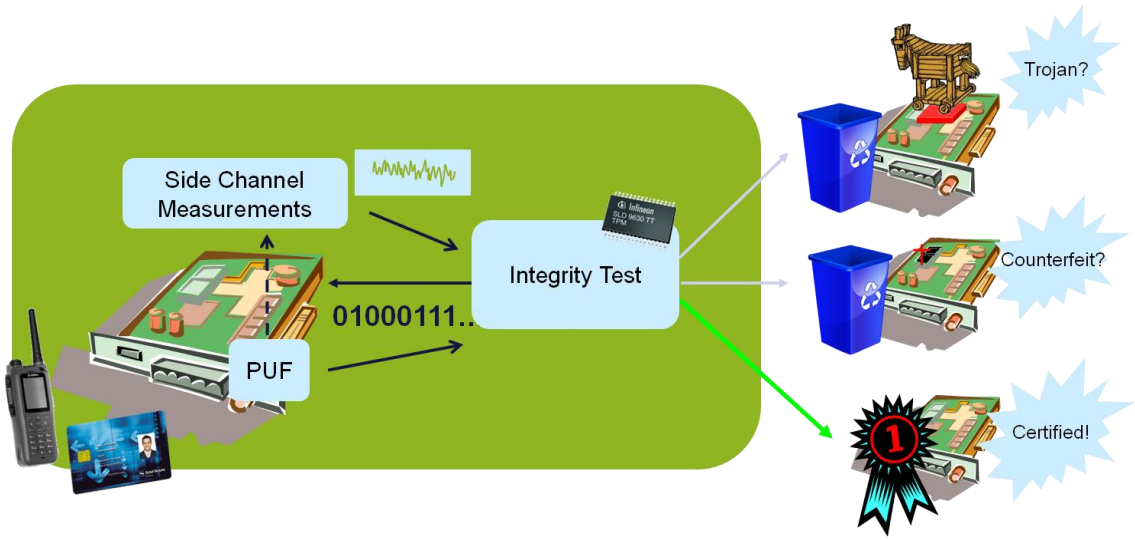

Figure 8: Overview – Holistic integrity checking

From now on, in this chapter, the ‘integrity’ aspect would include the ‘availability’ requirement considering that a system which is no longer ‘available’ can be considered to have been tampered with or to have been modified. This integrity aspect shall be covered by a study on the detection of malicious hardware modifications of the DUT (or typically referred to as ‘Hardware Trojan detection’). The ability to verify the ‘identity’ of a given circuit, i.e. performing hardware authentication, is covered by the study on the capability to reliably and securely use Physically Unclonable Functions (PUFs).

One of the core characteristics of the proposed scheme shall be that it shall be “on-board”, that is, the integrity/authenticity verification mechanism shall be done “at-time-of-use” in an embedded mechanism, i.e. integrated into a reader or docking station with which the targeted security device is communicating with. This concept of “on-board” shall be opposed to evaluation/characterization schemes which are done in laboratory conditions with PCs (having a lot of computing power and memory resources), sensors/probes which have no limits such that cumbersome equipments like oscilloscopes, GPUs… can be used.

The two trust architectures are explained hereafter. They both imply an integrity/authenticity verification “at-time-of-use” but due to the very nature of each technology the processes that allows this checking appear to be very different for each other. This is detailed in each architecture description in the following Sections 3.3 and 3.4.

3.3 PUF-based Architecture

In this section we will first describe the trust architecture of the environment of the PUF-equipped ID-card and its application field. In a second step we will zoom into the hardware architecture of the ID-card’s semiconductor chip. Further, as the PUF technology raises challenges in order to handle noisy PUF responses for certain applications, different

constructions of Fuzzy Extractors – that are needed for post-processing of the raw PUF data – are described. Concluding, a robustness and vulnerability analysis is presented.

3.3.1 Trust Architecture Specification

The trust architecture of a PUF-based system is basically specified by the relationship graph of the stakeholders involved. The most straight forward approach is to focus on the repeatedly performed application of the functionality that is provided by means of a PUF in the field. This application would have two basic flavours, depending on what is performed, whether it is a challenge-response protocol or a signature based protocol. Nevertheless in both flavours the trust relationship – if being built on participation of the PUF – will include two aspects:

a) It includes the trust of the verifier instance in the response/signature giving evidence that the piece of hardware on which the protocol was executed is the very component that it is intended to be and thus genuine.
b) Provided a cryptographic binding of the instance to an issuer – e.g. by means of a certificate – the PUF-based protocol will give evidence to the verifier that the prover is an authentic instance belonging to a cluster or family of instances being issued by this very (eGovernment) institution.

A typical example for a PUF-based trust architecture in accordance with the HINT approach is the following architecture for a PUF-enabled ID-card:

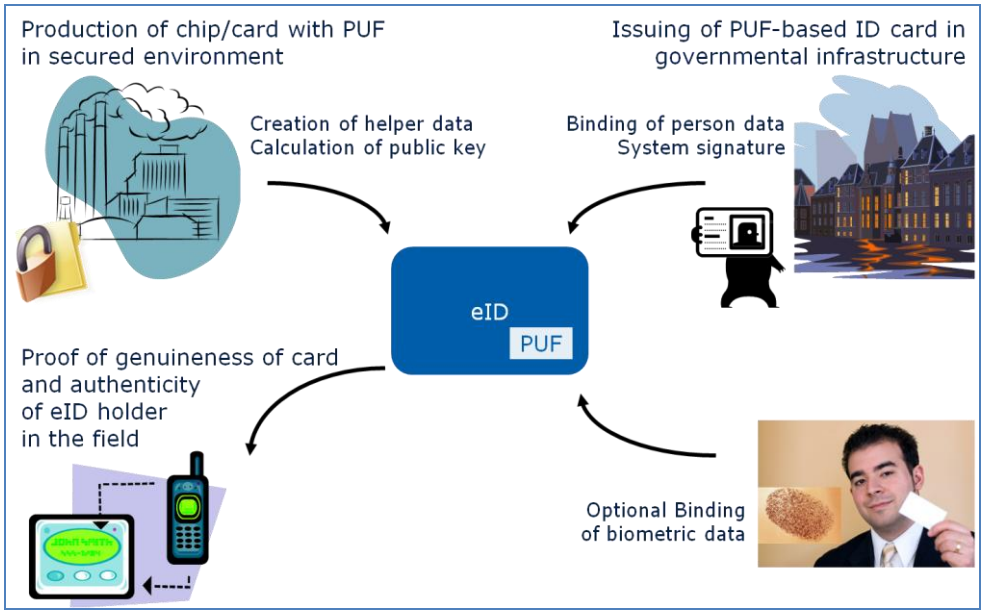

Figure 9: PUF-based trust architecture and ecosystem

Although the final goal in the use case environment is to be able to prove the integrity – in this case genuineness – of the hardware on which an authentication or signature protocol is based, the trust architecture of the overall system is more complex.

There are several involved entities that either have to be trusted or have to be able to prove their own authenticity while processing the various steps needed over the life cycle of a PUF-based secure device. In the following, we describe these entities and the trust relationships along the example of a PUF-based ID-card.

IC Manufacturer

The semiconductor developer and manufacturer is the first coming into place when we follow the live cycle of a PUF-based product. During manufacturing and testing, the characteristics of the PUF have to be tested and helper data for later post-processing of PUF data at the run-time of a PUF-based protocol have to be stored. Depending on the protocol and the type of product, even key extraction (for symmetric systems) or public key generation (for asymmetric protocols) might be necessary and proper handling of all these data and keys have to be ensured by the hardware manufacturer. This also includes the protection of PUF data against side-channel attacks later in the field by appropriate design measures. For the trust relationship in the overall life cycle of the PUF-based product this usually means, that a *certified* hardware manufacturer has to be chosen. For products certified according to the Common Criteria scheme the evaluation process includes sites audits of the manufacturing plant. Nevertheless, in the case of hardware genuineness to be proven it's in the hardware manufacturers own interest to perform proper handling of PUF-based data before delivery.

OS and Application Software Provider

Drivers and application software have to be able to process PUF-based protocols in a secure way. For the trust relationships during the life-cycle, this means, that the software has to be able to trust in the hardware and firmware routines provided by the hardware manufacturer and to provide trusted routines and protocols for the issuing and personalization steps. Trust in the software itself is supported by evaluation and certification performed by accredited independent third parties. The software provider must be able to trust, that the PUF functionality and helper data for protocols are authentic and for this purpose has to be provided authentic data and methods to verify this authenticity, e.g. MAC or signature values over public keys generated at the hardware manufacturer's production site.

ID-card issuer

To provide flexible integration in existing systems including backward compatibility and to allow involvement of secrets not depending on the hardware or software manufacturer it is mandatory to use independent key sets for further steps in the life cycle. At personalization stage the individual ID-card holder data is stored in the card IC's non-volatile memory. Such data may include name and address of the card holder for identification purposes, but also properties and rights granted to the card holder. At this stage it is necessary to rely on the fact that the hardware and software are trustworthy. Also the environment in which card holder's data is handled and pre-processed has to be trustworthy. In this step, a binding of card holder's data to the chip may be performed by means of a MAC or signature using the PUF-based secret/private key. So it can be assured that same card holder's data on any IC other than the genuine one, may be detected and authentication based on identical card holder data must fail if not performed on the IC bound to this card holder during the issuing step.

At this stage, also binding of biometric data to the IC may also be performed. Not only logical and administrative personal data, but also biometric data of the ID-card holder can be stored in the ID-card's IC. Authenticity of such biometric data may be provided by a certificate of the issuing instance, usually including a signature over the biometric reference data for the person holding the ID-card.

By this all means can be provided to later allow in-the-field verification of both the *identity* of the card holder as well as the *genuineness* of the ID-card used for proving this very identity.

Verification in the field

Any verifier in the field is running a PUF-based protocol to examine whether the identity of a given person is authentic and the means of verification – i.e. the ID-card itself – is

untampered with and genuine. After such verification, the verifying instance or service can trust that the card holder is exactly the one he or she claims to be and that the ID-card is the one representing this card holder with his or her associated properties and rights. If asymmetric cryptography is implemented throughout the various steps of the protocol, the verifying instance itself does not need to store a secret. Thus any machine being able to communicate to the ID-card may verify the identity of the card holder, as long as it has access to authentic public keys and the possibility to verify this authenticity. This is usually handled by using a certificate infrastructure, wherein each certificate corresponding to a private key of a card contains the appropriate public key and a system signature over that card public key.

3.3.2 Hardware Architecture Specification of the ID-Card Semiconductor Device

A possible schematic of a PUF-based secure device – again using the example of an ID-card IC – is shown in Figure 10. Besides the classical components of state-of-the-art ID-card ICs, like micro controller unit for program code execution, volatile and non-volatile memory to store code, data and peripherals, crypto-modules for symmetric and/or asymmetric cryptographic functions as well as interface modules for contact-based and contactless communication, a dedicated HW-PUF-module is depicted. To protect the PUF responses and especially private keys derived from the PUF during execution from side-channel or fault attacks, the PUF module has to be integrated in the overall security concept of the ID-card IC and all measures like bus encryption and fault detection have to be in place and activated.

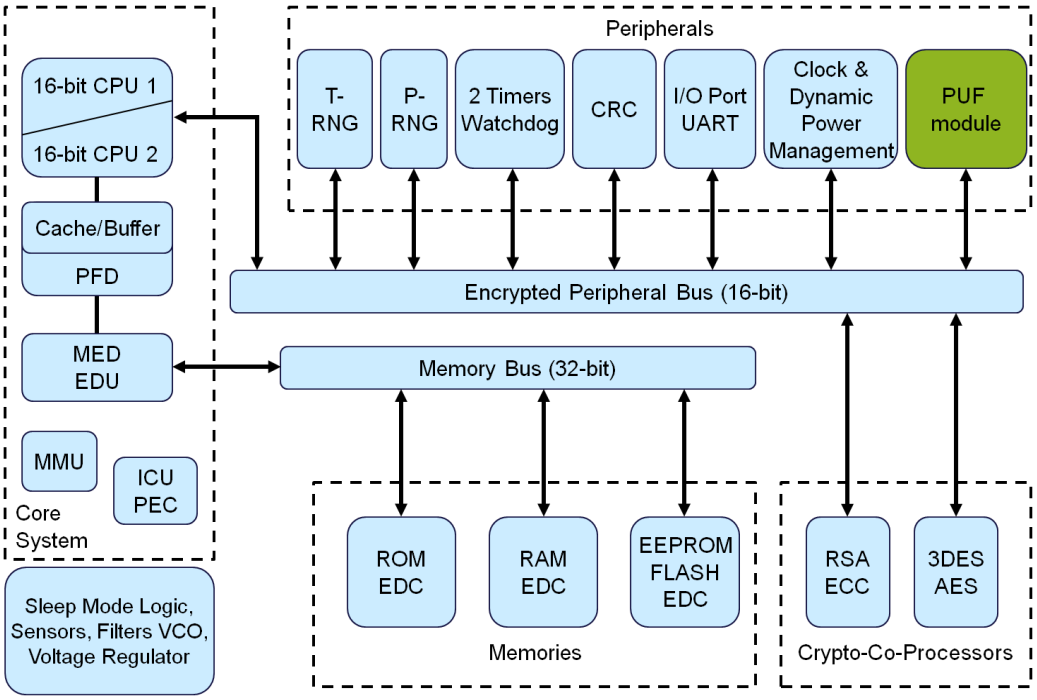

Figure 10: ID-Card IC Top level schematic including PUF module

If we zoom further into the PUF module on the ID-card chip we will specify its necessary sub-components:

- The actual PUF cell array
- The error correction circuit
- Dedicated non-volatile memory (NVM) for helper data to be used during error correction

- The key extraction component
- The PUF module control finite state machine (FSM)
- The bus interface to the peripheral bus

Figure 11 shows an overview over the PUF module's internal sub-components.

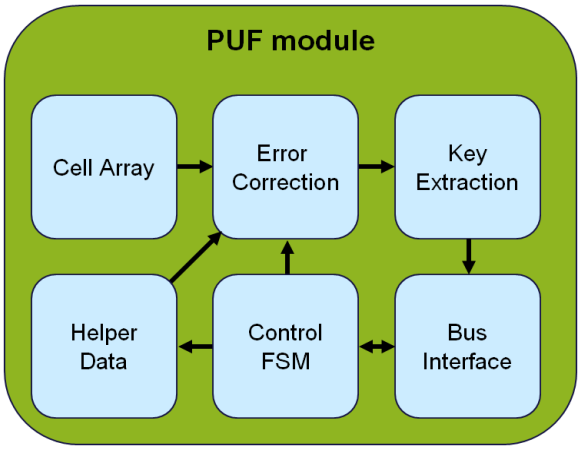

Figure 11: HW Architecture of a PUF-Peripheral for an ID-card chip

As one of the focus topics inside the HINT project is – besides the novel cell-concepts which will be worked on in WP2 and described in Deliverable D2.1 – defined by research on the post-processing of the raw PUF data, i.e. error correction and key extraction, we next elaborate on the error correction and key extraction part of the architecture.

3.3.3 Helper data schemes needed for Post-Processing of PUF Raw Data

One of the key challenges of applications using PUFs is to handle the existing noise caused by physical process variations. Further it has to be assured that for applications where a key is derived from the PUF, the generated key fulfils the required properties for the underlying application (e.g. length, distribution, entropy, etc.). Therefore a scheme has to be implemented which on the one hand addresses the need of error correction and on the other hand supplies suitable extraction. As described already in D1.1 [REF 1], the so-called Fuzzy Extractors (FE) exactly satisfies this need. A Fuzzy Extractor is a primitive that allows the extraction of uniformly random strings from a non-uniform source in a noise-tolerant way [REF 31]. We will use the Fuzzy Extractor construction in two ways: *embedding* externally generated keys, and *deriving* keys from the PUF response itself. For the second application, a noticeable simplification exists, known as the syndrome construction [REF 8].

Code-Offset Construction

The code-offset construction can be applied in two ways, depending on whether an external key is embedded or if the key is derived from the PUF response itself. In both cases helper data *HD* is created by calculating the *XOR* combination of a reference response *R* and a codeword *C*, which is needed later in the recovery procedure to re-calculate the key. The figures below illustrate the two cases; on the left with the key derived from the PUF response directly and on the right with an external embedded key.

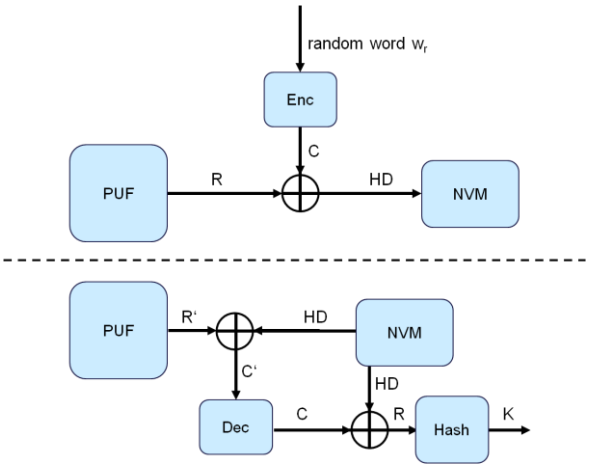

Figure 12: FE with PUF-based key generation

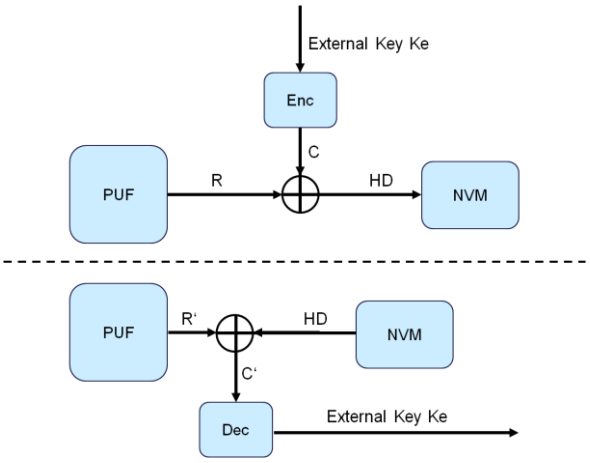

Figure 13: FE with external injected key

Note that embedding an external key has the advantage of being able to use any key structure (this is of particular relevance in case a key is calculated in a specific manner and not merely a random bitstring). Also, the distribution of the key space is entirely independent from any distribution of the PUF responses (which might not be uniformly distributed, for instance).

On the other hand, extracting a key from a reference PUF response *R* allows to use the Syndrome Construction, a simplification which allows storing only the syndrome of *R* as Helper Data instead of storing the entire *XOR* sum of *R* with a codeword *C*.

Syndrome Construction

In this construction, the PUF response *R* is basically multiplied by the parity check matrix of a previously defined error-correcting code. The output is defined as helper string *HD*. To recover the PUF response *R*, a noisy response *R'* is multiplied by the parity check matrix to obtain the syndrome of *R'*. Since no codeword *C* is retrieved in this case, the syndrome construction cannot be used for the key embedding application.

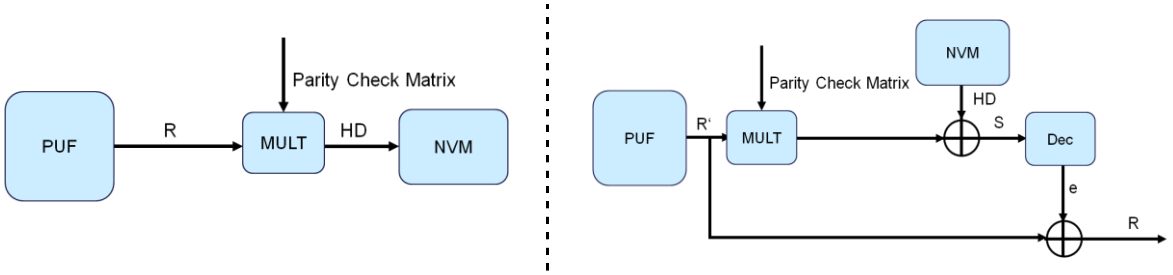

Figure 14: Syndrome Construction

Depending on the error rate of the PUF responses, a suitable Error Correcting Code can be chosen. One approach is to choose a combination of two codes, as proposed in [REF 30].

3.3.4 Assessment through robustness and vulnerability analysis

Reliability

We make a distinction between long-term and short-term reliability, referred to as aging and repeatability respectively. For the former category, degradation of electrical component changes the PUF behaviour over time. Electromigration in interconnects and negative/positive bias temperature instabilities and hot carrier injections in CMOS transistors

are responsible for this. Careful circuit design might help: it is recommended to avoid high electric fields. Also one can introduce an 'idle' state, disconnecting PUF and power supply most of the time. Additional chip area is required for this solution however.

Repeatability imperfections are caused by internal device noises. The distinction between variability and noise is essential. Both cause deviations with respect to the nominal behaviour. Measurements of structural variability, originating from manufacturing processes, are reproducible. One can state that they are defined by spatial distributions (and orientations) of individual molecules of the solid materials. Noise however is a non-reproducible temporal phenomenon. Generally speaking, in electronic circuits, both variability and noise are undesired. PUF circuits measure variability, but are bothered by noise as well, as it reduces the repeatability.

Both variability and noise are technology dependent. They remain major design and manufacturing challenges, especially while downscaling dimensions according to Moore's law. Random Dopant Fluctuation and Line-Edge/Width Roughness are important sources of variability for CMOS devices [REF 12]. White thermal noise and 1/f noise affect the CMOS channel current [REF 11]. Interconnects are affected by Line-Edge/Width Roughness and white thermal noise too.

Environmental deviations worsen the repeatability problem. The PUF chip environment is mostly defined as (but not limited to) the outside temperature and the supply voltage. In the ideal case, PUFs operate in a constant nominal environment. Note that internal device noise is not avoidable as opposed to environmental deviations. Thermal effects are always present, as PUFs typically operate at room temperature and not at absolute zero (−273.15°C).

Repeatability issues are challenge-response pair (CRP) dependent. In every PUF, analogue information is digitalised in one way or another, always including some form of thresholding. Consider for instance the arbiter circuit in an arbiter PUF, outputting a '0' or '1' response depending on which of its two inputs detects a rising edge first. CRPs with a small nominal time difference between the rising edges, are easily affected by noise [REF 10]. CRPs corresponding to large time differences on the other hand, tend to be very repeatable. Similar mechanisms cause the CRP dependency for other PUFs.

For strong PUFs, employed in a CRP-based authentication application, repeatability is probably less of an issue. When comparing an (e.g. 128 bits) response with its enrolled database equivalent, one can accept a specified amount of erroneous bits. For weak PUFs however, there is a zero tolerance as secret keys are supposed to be perfectly reproducible. Error correcting codes (ECC) like BCH are typically employed, as part of a Fuzzy Extractor [REF 8].

An alternative for ECCs has been proposed in [REF 14]. The conventional paradigm of using public challenges to generate secret PUF responses is reversed. The scheme exposes response patterns and keeps secret the particular challenges that generate response patterns. The key is assembled from a series of small (initially chosen or random), secret integers, each being an index into a string. A newly repeated PUF output string is searched for highest-probability matches with the stored patterns, re-generating every index.

The less repeatable a PUF is, the larger the hardware overhead of its accompanying ECC shall be. Two fundamentally different approaches are used to minimize the ECC overhead, although they both introduce hardware overhead somewhere else. Finding the global optimum is a difficult exercise. First, one can indicate non-repeatable CRPs as invalid. Consider for instance an SRAM PUF, where the local mismatch of each memory cell generates a nominal '0' or '1'. As a post-manufacturing step, one could mark all unstable cells (small mismatch so that thermal noise impact is relatively large). Additional cell testing and bookkeeping circuitry is required.

A second approach stabilizes CRPs to a 'golden' reference. Consider for instance a ring oscillator PUF where each column of inverters (covering all oscillators) can be powered by one out of two supply voltage levels [REF 12]. By selecting an appropriate voltage configuration for each CRP, the response is guaranteed to be repeatable, even for changes in outside temperature.

Vulnerability

The security requirements differ depending on whether we need a ‘weak’ or ‘strong’ PUF. For weak PUFs, it is imperative to keep the responses on chip, just as the secret keys to which they are post-processed. Hardware attacks (invasive, through side channels or via fault injection) should be taken into account. PUFs are often assumed to be resistant against the first category. One can argue that invasion damages the physical structure and hence also the PUF. Experimental evidence is generally lacking however, except for the coating PUF [REF 19]. Electromagnetic radiation is an exploitable side channel for ring oscillator (RO) PUFs [REF 15]: with an antenna one can perform semi-invasive measurements of individual ring oscillator frequencies. In addition to PUF circuits, Fuzzy Extractors might leak side-channel information as well [REF 16].

For strong PUFs, CRPs can be obtained by anyone. The security relies on the unpredictability of the CRP’s behaviour. It should be infeasible to construct a clone via a mathematical model. Modelling through Machine Learning (ML) algorithms, given a training set of CRPs, is a major threat. High modelling accuracies might be obtainable through ML techniques like support vector machines and artificial neural networks. Algorithms automatically learn the input-output behaviour by trying to generalize the underlying interactions. The more linear a system is, the easier it is to learn its behaviour.

The arbiter PUF, which quantifies the variability of gate delays, shows additive linear behaviour making it vulnerable to modelling attacks. In the paper proposing arbiter PUFs as a security primitive, ML was already identified as a threat [REF 13]. In the latter paper, the authors reported a modelling accuracy of 97% for their 64-stage 0.18um CMOS implementation. Similar results were obtained for a more recent 65nm implementation, also having 64-bit challenges [REF 32]. Variants of the arbiter PUF which introduce additional non-linearity (XOR, feed-forward, lightweight secure) provide more resistance but can still be modelled [REF 17]. Glitch PUFs, which are delay-based too, contain weak challenges leading to almost no glitches [REF 18]. As a consequence, those CRPs tend to be rather predictable.

Hardware attacks on strong PUFs should be considered too, as they can facilitate modelling. Recently, repeatability imperfections of PUF responses have been identified as a threat [REF 10]. CMOS device noise renders a significant fraction of the CRPs unstable, hereby providing a side channel for modelling attacks. As a proof of concept, 64-stage 65nm arbiter PUFs have been modelled as such, without utilizing any ML technique, with accuracies exceeding 97%. The same principles might be recyclable for attacking other strong PUFs.

So-called controlled PUFs enhance the security of strong PUFs via additional hardware [REF 9]. In this case, response bits are post-processed using a Fuzzy Extractor. Cryptographic hash functions are non-invertible and modelling vulnerabilities of the CRP behaviour are hence hidden. Also, one can pre-process the challenges with a cryptographic hash function to counteract chosen-challenge attacks. However, the use of controlled PUFs poses two major problems. First, the increased hardware footprint undermines the potential of strong PUFs for resource constrained applications. Second, the additional building blocks might be vulnerable to hardware attacks.

3.4 SCA-based architecture

3.4.1 Trust Architecture specification

The basic structure of the SCA-based integrity/authenticity check mechanism is illustrated in the following figure:

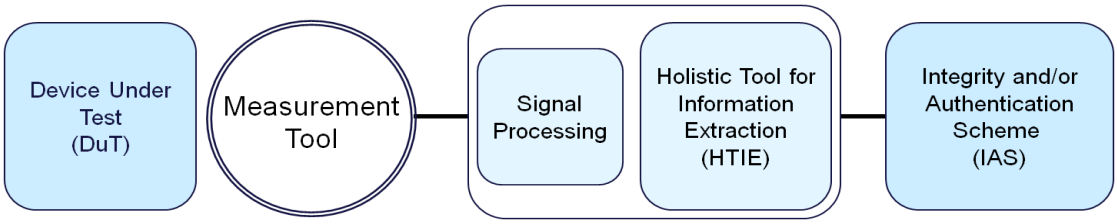

Figure 15: SCA-based architecture

The **Measurement Tool** shall consist of equipment for measuring the side channels from the Device Under Test (DUT). Note that the side channel under investigation shall be either the power consumed by the DUT or the ElectroMagnetic (EM) radiation which is measured by adapted probes. In order of importance, EM measurements shall be our primary focus but if for some reasons, for example when trying to "embed" the whole scheme or if for example the DUT emits radio signals which might render EM measurements impractical, we find out that EM measurement is impractical or ineffective, the power consumption side channel shall be thoroughly investigated in order to have more efficient measurement tools.

The **Holistic Tool for Information Extraction** (HTIE) shall incorporate signal processing and storage tools. The information extraction procedure could include approaches like template based procedures, statistical correlation-based calculations, code execution profiling… The extracted information shall allow the IAS to detect a pre-defined set of hardware Trojans (*not all hardware Trojans would be detectable*). We may also try to see to what extent the SCA-based approach can help to identify or detect a counterfeited/cloned product.

The stability of the HTIE with respect to environmental changes or depending on the use cases shall also be investigated.

The **Integrity / Authentication Scheme** (IAS) should primarily implement integrity checking through the detection of HW Trojans.

The *first approach* for the SCA-based detection shall be **passive,** i.e. the system shall only observe what the device under test is doing under normal operating conditions and shall have no means of communicating with the DUT.

A *second approach* for the SCA-based detection shall be an **active** one where the DUT is asked/made to perform a particular operation or to emit a particular signal that shall allow the detection scheme to "make its decision". It seems easier to detect HT when mastering the operation of DUT than if it operates freely without any control. For example

- The DUT could be reset for the test.
- The environmental operating conditions of the DUT could be brought to particular ranges.
- The DUT could be asked to execute a particular/characteristic set of instructions or operations.

- The DUT could be asked to process particular data.
- Some hardware parts of the DUT could be activated or deactivated on purpose, like for example a PUF or a sensitive part of the DUT.

Adapted '**protocols**' shall be studied for the active SCA-based detection when relevant. For example, such protocols shall implement ways to prevent say replay attacks, man-in-the-middle attacks… between the detection scheme and the DUT. The protocols shall also implement reaction strategies in case Trojans, counterfeits or clones are detected or "suspected". Such strategies shall provide a way to manage the false-positives/negatives based on the 'limits' of the detection scheme or the detection scheme's stability with respect to environmental or use-cases' conditions while assuring high degrees of trust and usability of the DUT.

The **on-board/'embedded'** factor: Note that all the technical choices that shall be made for the measurement tool, HTIE or the IAS shall take into account the fact that the ultimate goal is to have this scheme 'on-board' or 'embedded' to allow "on-the-field" or "at-time-of-use" checking as explained in Section 3.2. A two-step approach shall be used to investigate about the numerous aspects of the SCA-based integrity/authentication scheme:

- First, the tools and concepts shall be implemented for, tested and validated in laboratory conditions (while still having in mind that we will later 'embed' everything).
- Then, we shall study if and how the implemented schemes can be tailored to fit the requirements for being 'on-board', some of which would be:
    - Signal (EM or Power) acquisitions shall be done by on-board analog-to-digital converters and signal processing by on-board Digital Signal Processors (DSP). Several such tools are available off-the-shelf from providers like Tektronix for example with different sampling rates, bandwidths etc. The best suited tool(s) shall be investigated.
    - The HTIE shall be run on an embedded chip with limited computing power, but might also take advantage of the fast DSPs often present in such systems.
    - Embedded systems usually have limited memories whether it is for handling data during computations (RAM) or for storing templates, measured curves etc (NVM – although here this should not be a blocking point with the increasing density of NVMs such as Flash memories).
    - The 'time' available for doing the integrity/authentication checking shall also be an important parameter to take into account. In laboratory conditions, we could have hours/days for performing the verification. In the case of a PMR, the latter may be connected to the docking station for only a few minutes/hours. For a smart-card we are more on the second time-scale.

3.4.2 *Technical specification of the platform*

The architecture that shall be studied for the SCA-based solution is based on the SASEBO board environment actually designed for Side-Channel Attacks but with the right features for HT insertion and detection. The measurement tools are here standard laboratory tools (see section 3.4.3). A high-level view is illustrated in Figure 16.

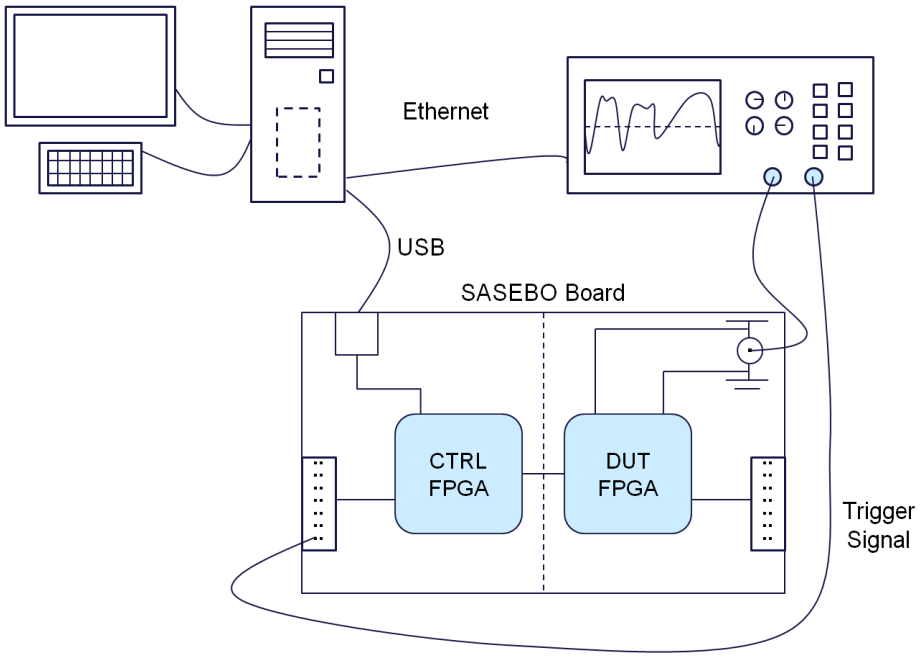

Figure 16: Overview Test Environment Setup

The SASEBO development board embedded two XILINX FPGAs: a Virtex-5 for the DUT and a Spartan-3 for the control interface (CTRL-FPGA). Both FPGAs are connected with a 38-bit wide bidirectional bus. The CTRL-FPGA is connected to an USB interface and therefore able to communicate with a PC. The design of the CTRL-FPGA is a central part of this test environment development.

In order to measure the Side Channels, an oscilloscope is connected to the power supply of the DUT-FPGA. An additional link between the oscilloscope and the SASEBO board is a trigger signal, which is controlled by the CTRL-FPGA.

A certain part of the CTRL-FPGA, the DUT and the oscilloscope form the real-time (RT) domain of the system. The other part of the CTRL-FPGA, the non-RT part of the oscilloscope, and the PC do not work in RT, but perform more complex and abstract tasks.

The PC is responsible for multiple tasks:

- Supply data to and control the CTRL-FPGA;
- Control the oscilloscope in terms of parameters and options;
- Collect the data from the SASEBO board and the measurements from the oscilloscope;
- Perform the Side-Channel Analysis.

The main components will be described in further detail in the next sections:

- Measurement Equipment
- FPGA Development
- System Synchronization and Timing requirements
- Development approach

Measurement Equipment

A digital oscilloscope is used for the data acquisition, which is capable of sampling 5 Giga Samples per second. Traces with up to 10000 measurements points can be sampled. The Oscilloscope supports low-level averaging with up to 512 samples.

It is equipped with a network interface, which can be used to communicate with a PC. This communication channel can be used for data transfer as well as for the control of the Oscilloscope. Two channels are used:

- Channel 1 is used to measure the power consumption of the SASEBO Board.
- Channel 2 is used as a trigger for the measurements.

FPGA Development

A central component of the Test Environment is the Control FPGA implemented on the Spartan 3 FPGA on the SASEBO Board. A main task is to supply a synchronous input profile to the DUT. Since the systems are working in the MHz range, the Input Vectors (IV) must be available to the CTRL-FPGA in advance. In order to make the design more flexible and the amount of input data smaller, there are three different possibilities to progress from one IV to the next: One IV per clock cycle, IV update after a given amount of cycles, and reaction on the outputs of the DUT. These ways are supported concurrently, so a data structure is developed to describe such an input profile. A profile needs to be loaded to the memory of the control FPGA in advance.

In order to keep the system as flexible as possible, the 38 I/O channels are not predefined, but dynamically configurable. Depending on the DUT, each channel could either be configured as an input, an output or be ignored.

To control the system, different commands are available to start, stop and reset the DUT.

The output of the DUT is sampled synchronously with the IV. As additional information, the cycle number since the start of the execution is stored as well. With this approach, the outputs in between two IVs are not sampled. This is a trade-off since the amount of memory would grow substantially and no external memory is available on the SASEBO board. Intelligent ways to handle all intermediate results (for example, by setting assert conditions) could be implemented later in the project if needed.

System Synchronization and Timing requirements

The whole design of the CTRL-FPGA is divided into three main functional blocks as shown in Figure 17:

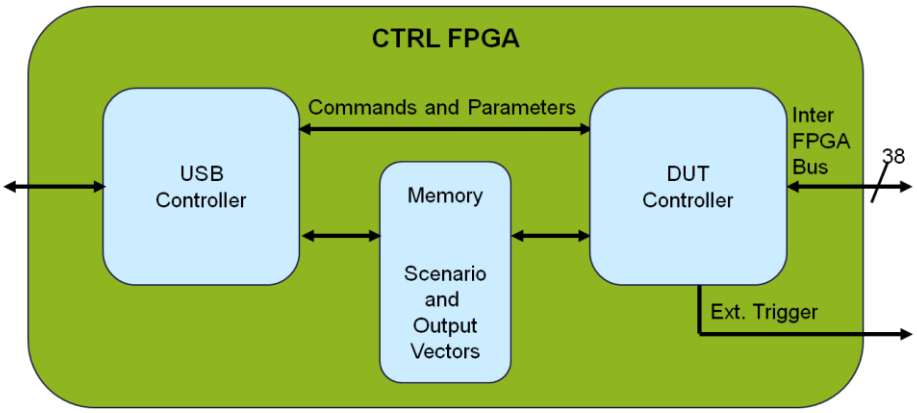

Figure 17: Complete CTRL-FPGA design

- The USB connection module handles the communication with the PC.
- The memory is designed to store the IP as well as the outputs. The memory bus size is scaled to the requirements so that per address one is respectively one output sample can be stored.
- The DUT-Controller is responsible for the actual running of the IP and the communication with the DUT. In order to react immediately to output events and to reduce the delays on the critical path between the two FPGAs, the controller is implemented using a pipeline structure with forward logic.

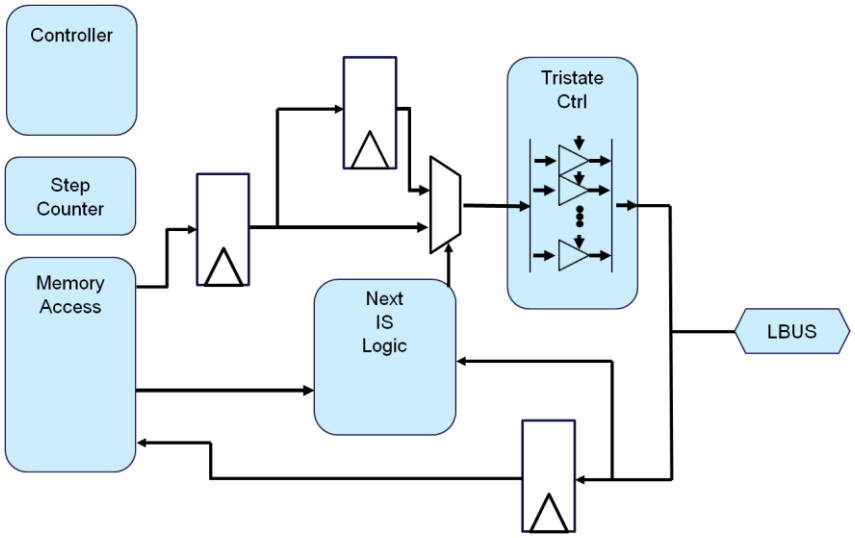

Figure 18: DUT Controller

Development approach

The implementation is handled in a module by module approach, beginning with the DUT-Controller. For verification purposes, the remaining parts of the system are replaced by simulation models. The whole system will be tested with a simulation model of the USB-controller, which will be created based on the available information in the datasheets.

3.4.3 Tool chain and automation

In order to perform an SCA according to the presented workflow, a tool-chain is defined. The following diagram (Figure 19) depicts the tools, equipments and interfaces involved.

The shown components are described in the following sections. These are:

- The Scenario File
- The Output File
- The simulator
- The Control Tool
- The automation

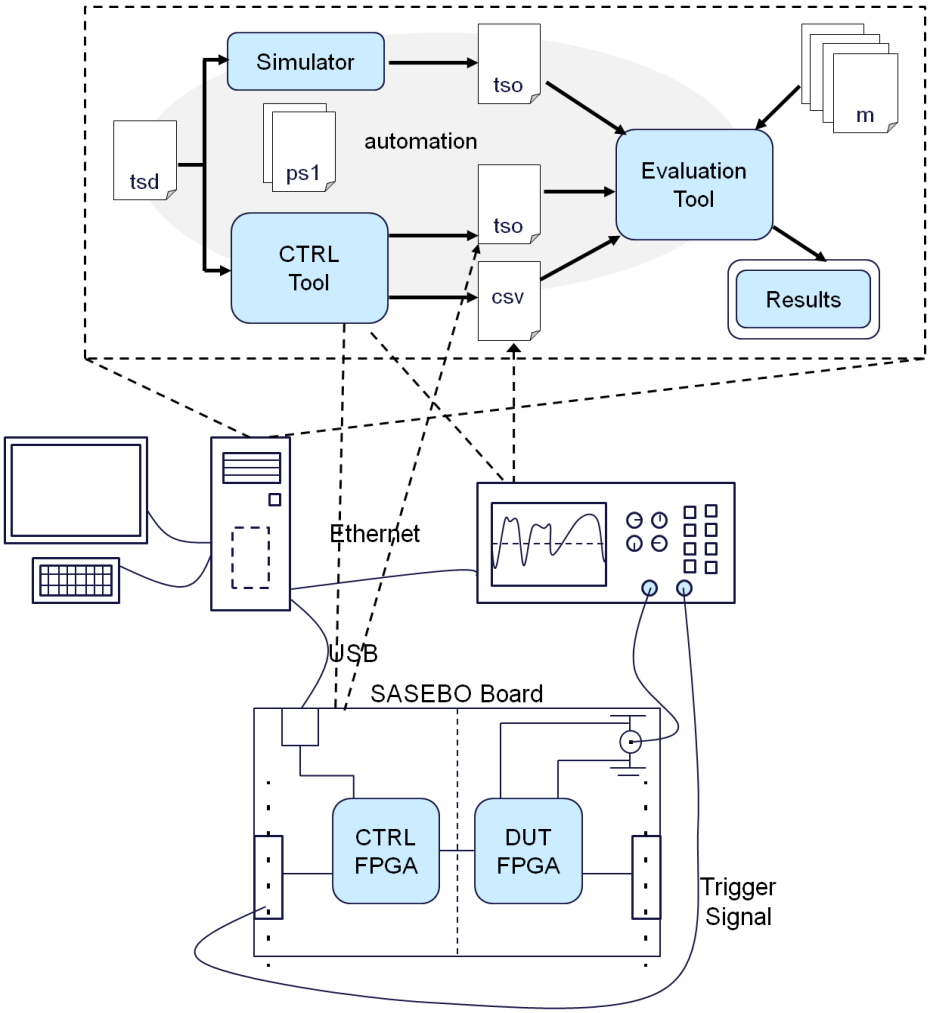

Figure 19: Test Environment and Tool-Chain

Scenario File

To run a test, a scenario needs to be loaded on the CTRL-FPGA. A dedicated file format has been developed to change easily from a scenario to another.

Scenario files are organized into 4 main sections:

- General Information: Basic information about the scenario is stored in this section.
- Parameters: In this section the parameters mentioned above are defined.
- Input Profile: The Input steps are stored in this section.
- Commands: A sequence of the commands stated above.

Output File

The results of a test are stored in a file. This could be used for further analysis tools.

Simulator

To determine the genuine behaviour, the system needs to be simulated. Therefore, the DUT is mounted onto a Test Bench that is based on the CTRL-FPGA design.

This Test Bench then reads the Scenario file, simulates the system (consisting of CTRL and DUT) and writes the outputs to an output file.

Control Tool

The communication with the CTRL-FPGA is performed with a small tool. It receives a scenario file in input, parses it, and configures the CTRL System accordingly. After the execution of all defined commands, the outputs are read back and saved into an output file.

Automation

The connection between all components is made by scripts. Run on the PC, they allow us to repeat the scenarios and to improve the analysis and decision process.

3.4.4 Requirements

In the SCA-based approach the purpose is to defined solutions to detect the HT first in Lab conditions and secondly in a regular “on-the-field” or “at-time-of-use” situation.

In order to perform experiments with Hardware Trojans and Side Channels, an adequate Test Environment is necessary. It includes on the one hand a DUT (either stand-alone or as part of a product) and on the other hand some tools that process tests on the DUT.

Since many different tests should be run and many approaches should be tested, the system must be designed in such a way that all of the different ideas are supported. The Test Environment shall accept extensions if possible. On the other hand this Test Environment shall prepare the work for the prototype so this architecture needs some relevant FPGA/CPLD where the HT are inserted and detected. It shall also be ready to significantly simulate the regular “on-the-field” or “at-time-of-use” testing.

Flexibility and adjustability are two crucial requirements for the Test Environment so the following features must be supported:

- Input vectors need to be supplied in real time.
- Outputs need to be tracked throughout the execution.
- It could be necessary to go through many system states before the applicable one is reached.
- Measurement equipment shall support many different approaches. It shall be capable of sampling at least 5 Giga Samples per second to achieve sufficient sensitivity.
- It should be possible to automate tests as much as possible on the one hand to detect HT more efficiently and on the other hand to get ready for “on-the-field” or “at-time-of-use” tests.
- In order to support different ideas, technical points like clock frequency, number of I/O and other low-level features should kept as flexible as possible.

3.4.5 Assessment of the relevance of SCA for HT detection

There is a rich body of literature describing how hardware Trojans may be implemented (see the overview in D1.1 [REF 1]). A HT may be small or large, complex or simple, spread over the layout or localized in one position. These are key factors that will influence the detectability of the HT and hence determine requirements for our SCA based detection approach.

Here we study the feasibility of detecting HTs with side channel analysis. We provide an overview of the state-of-the-art in the literature with particular attention to limitations. Some of these techniques shall serve as starting points for the development of our SCA based architecture in lab conditions. Then we identify and briefly outline potential issues that may arise when porting the architecture to an embedded implementation.

We first begin with an overview of works that deal with HT detection using electromagnetic radiation-based SCA. [REF 21] acknowledges 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 paper 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 analysed.

[REF 22] 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 favourable to detect the HT: the HT's details are perfectly known, as well as the exact time when the HT is activated. This is usually not the case in a practical HT detection scenario. This reference identifies some other constructive applications for HT detection, such as watermarking, that may be of interest in other sections of this project.

There is an important body of research that deals with the HT detection problem not explicitly measuring the electromagnetic radiation. The approaches followed could be extrapolated to the requirements of the project. The seminal paper [REF 20] gives an overview of the possible methods available to detect HT. [REF 23] formulates some critics on side-channel analysis techniques used for detecting HTs in some previous works. 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 analyse the experiments carried out in the literature and identify two main shortcomings: non-realistic, abnormally large HTs are used and HTs 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.

[REF 24] provides a new side-channel based method for HT detection. It formulates the HT detection problem as a signature outlier identification problem, and solves the HT detection

problem 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. The experiments are executed on HSPICE simulations on post-layout designs. Process variations were simulated using Monte-Carlo techniques. The HT size (in gates) ranges from 0.1% to 0.5% of the total number of gates in the circuit, and the method is shown to detect the HT more efficiently in comparison with previously published methods. This indicates how small can be the smallest detectable circuit modification (namely, a HT) in ratio to the overall circuit size.

[REF 25] tackles the issue of process variation in HT detection. It proposes a new HT detection method that 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 chip instances as references. The experimental part of the paper deals with HSPICE simulations and experiments on an FPGA.

Electromagnetic analysis has been extensively used in the SCA area. In what follows, we analyse the key requirements for EM analysis and the possible application of such techniques to the problem of HT detection. [REF 26] identifies shortcomings and limitations of electromagnetic based SCA, namely, that the measurement setup is more complex compared to traditional (non-electromagnetic) power analysis. Although the paper is written in the context of SCA and not in the context of HT detection, some interesting conclusions can be extrapolated. The authors identify 5 shortcomings:

a) Finding the relevant spatial positioning over the device under test can be a time consuming process.

b) The measured signal has to be properly amplified before being exploited.

c) In some cases, the chip has to be de-capsulated using chemical means.

d) There is the risk of pollution by environmental noise or by the “unintentional emanations”.

e) The efficiency of the Differential ElectroMagnetic Analysis (DEMA) is highly dependent on the probe used.

We note that a), b) and c) and d) are directly extrapolated to the context of EM SCA-based HT detection, and these points describe the limits and limitations of such EM SCA-based HT detection approach. We also note that b) and d) are issues that could severely affect the reliability of the HT detection when ported to an embedded, off-lab application. The paper carries out an interesting analysis of different state-of-the-art EM probes and concludes that there are numerous factors, such as the shape of the probes, the DUT technology, the spectral characteristics and the layout that influence the final efficiency of the EM SCA attack. The paper uses specific-purpose equipment for the execution of the experiments in a laboratory setting.

Some details about high-resolution EM measurement setups are discussed in [REF 27], along with an interesting discussion of side-channel cartography. [REF 29] also deals in more details with the EM measurement setup. The inductive near-field EM probe used has a resolution of 100 micrometer, and is used in conjunction with a 30dB amplifier, an X-Y table of a step size of 50 micrometer and an oscilloscope with a sampling rate of 5 GS/s. The main contribution of the paper is that localized attacks are possible. We note that the fact that localized measurements are possible does not necessarily imply that EM-based HT detection is easier – the HT could be spread over the original circuit rather than concentrated in one particular spot.

Chapter 4 HINT Application Prototypes

The HINT application prototypes are tailored according to the two use cases studied in the document D1.1 [REF 1] (the ID-card use case and the PMR use case) and the two security requirements of hardware authenticity and hardware integrity. In this chapter, we provide a high level description of the prototypes that shall be demonstrated in WP4. A first prototype shall be relevant to the ID-card use case where hardware authentication, based on PUF-based key generation, shall be illustrated. The second prototype shall be based on the PMR use case where hardware integrity verification, based on HT detection, shall be illustrated.

4.1 Unclonable ID Cards Prototype

For the HINT application prototype related to Unclonable ID-Cards, we plan to demonstrate one out of two features, an authentication through a PUF-based Challenge-Response Protocol or a digital signature with private key extracted from a PUF instead of stored in persistent memory. These two options are described in the following.

4.1.1 PUF-based authentication

In the PUF-based authentication application, an ID-card shall authenticate itself by answering to a challenge (i.e. a random bit-string of a specified format prompted to it) with a properly generated response. The response shall be generated by key extracted from a HW-based PUF.

4.1.1.1 Enrolment / Registration

Each ID-card to be registered will be equipped with a unique identifier. This may for instance be a number written into the card's NVM.

During the enrolment phase, for each ID-card to be registered a sufficient number of authentication keys will be generated by querying the embedded PUF. By its very nature, each card's *genuine* PUF (being unclonable) needs to be used for authentication key generation, so the process cannot be simulated. Therefore, this needs to be performed for each individual ID-card to be registered, in a secure environment; moreover for symmetric authentication systems the resulting keys need to be safely stored into a database along with the corresponding ID-card's identifier, and kept secret. In case of asymmetric authentication a public key has to be generated either on card or off card within a secured environment.

4.1.1.2 Identification / Authentication in the field

As the first step within an authentication protocol, the ID-card to be authenticated sends its ID-number to the server for identification. Note that *identification* here means the association of an identifier with an entity and does *not* include the *verification* of this association. The verification is achieved through the subsequent *authentication* procedure.

The card terminal is either trusted and in possession of the symmetric key (or a derivative from the one, depending on a key management hierarchy, or in the asymmetric case requests a public key of that very card from the server database; the card terminal then transmits a challenge to the ID-card. The ID-card's generates a response based on the key extracted from the PUF and that challenge.

4.1.2 Extracting a Signature-Key from a PUF

4.1.2.1 PUF-based Signature-Key recovery

The basic security feature to be demonstrated with the PUF-based signature application is a digital signature generated by an ID-card, *whereby the private signing key is retrieved through an internal PUF and is never stored in any non-volatile memory of the card*. So apart from the very short time-intervals where it is residing in RAM or CMOS flip-flops inside the cryptographic engine during signature generation, the private key does not exist in a readable form accessible for an eavesdropper. The appealing property lies in the fact that a vast number of increasingly successful attacks aiming at reading out a smart card's Flash or EEPROM memory will no longer be applicable at all.

The envisaged application is based on the ECDSA signature scheme, since it provides private keys of comparatively short bit-length while at the same time offering a high security level. For each unique private signature key for an ECDSA signature helper data shall be calculated from the PUF and stored on the ID-card's chip. Depending on the chosen application, this means that one can either (see Section 3.3.3)

1. Derive a private signature key from a reference PUF response
2. Embed an externally generated secret signature key into a codeword

Subsequently such private signature key shall be recovered from the helper data and a fresh PUF response on demand.

It is usually not possible to use a PUF output directly as a private signing key. The main reason for this is the "noise" to be considered for each PUF-response: any two PUF responses will show at least slight differences between their resulting bit strings. However, in order to be useful for conventional cryptographic schemes, the encapsulated key needs to be recovered *exactly*, i.e. without any bit errors. Therefore, a helper data scheme is used which first maps the key to be encapsulated to a code word, and then merges this code word with a PUF response. The difference between the PUF response used for enrolment and a PUF response used for key recovery is to be regarded as *noise*, the influence of which needs to be compensated.

As a consequence, the PUF response must be long enough to allow for redundancy of the error correction applied during post-processing, which usually means the PUF response must be as long as a valid code word. The PUF response should be kept *secret*, since (along with the helper data, which themselves usually are not considered secret) it allows key recovery for an adversary. This implies that also the PUF response should never be stored in NVM at any time.

We chose to configure ECDSA signatures with key length of approximately 230-250 bits. Depending on the capacity of the error-correction code, a PUF response needs to be *significantly* longer (usually a *multiple* of the key-length!). The choice of error correcting code and its configuration depends on a couple of factors, like the error-rate of the PUF used and hence the resulting number of bits which need to be corrected, the failure rate to be accepted within the system, and also the computational capacity available by the ID-cards chip (which might be a limiting factor).

4.1.2.2 Enrolment / Key-generation / Helper Data generation

As outlined above, we intend to use the ECDSA signature scheme, with a chosen bit-length of approximately 230-250 bits. Each ID-card shall be endowed with an individual private scalar which constitutes the *private signing key* – basically one large number (e.g. 230 bit) – while system-wide parameters (i.e. curve parameters) are generated outside the cards and shared among all participants of the application by storing them in their cards' NVM.

Key generation and key encapsulation shall be completely performed on-card. Since an ECDSA capable ID-card must provide a true random number generator (RNG) anyhow, this shall be used to generate the private signing key (from which the card also derives the corresponding public key). The challenging part of key-generation lies in encapsulation of the private key, since this comprises the generation of the helper data. To this end, the private key is encoded to a code word under the chosen error correcting code, and a PUF response is generated and merged with the resulting code word to generate the helper data. Subsequently, the helper data are stored to the ID-card's NVM, and the private key is safely deleted from RAM.

It is prudent to store some check-value derived from the key (e.g. a hash) along with the helper data to allow a self-check after each key recovery. This check value must be generated such that no information about the key can be derived from it.

Note that all steps involved are processed inside the ID-card's chip. Moreover, the private key (or any PUF response) is never stored in NVM during the entire process, but only in RAM. The helper data being stored in the ID-cards NVM may be extracted by an attacker with high enough attack potential, but without a corresponding PUF response the attacker should not be able to retrieve the encapsulated private key from them.

The process of key-generation may furthermore be tight to a specific "role" and requiring "role authentication", depending on the application. This means that key generation may only be invoked if specific conditions are satisfied (for instance may only be initiated by an authorized entity authenticating itself prior to key-generation), and/or may only be performed within a secure environment. Also, output of the ID-card's public key and subsequent verification of proper generation of the private key by the outside world (e.g. by signing a test-challenge, or running a DH-key agreement) shall be an integral and mandatory part of key generation in order to declare the enrolment process as complete. Optionally the enrolment body may issue a certificate comprising the card holder's credentials and his public key, and store this certificate in a database and/or in the ID-card itself.

4.1.2.3 Key-recovery / Signature generation

Signature generation is considered to be frequently performed in-the-field, so potentially within a non-secure environment.

In order to initiate each signing process and in particular each PUF-response generation, the legitimate holder of the card first needs to authenticate himself towards his ID-card; this is usually done by a PIN-request. Subsequently for each signing process, a fresh PUF-response is generated inside the card, and the signing key shall be extracted from the helper data using this PUF response. Prior to signature generation, a self check can be performed on the recovered key provided such a check value has been generated during key generation, and only if this check is passed (i.e. if it is confirmed that the private key has indeed been correctly recovered) that key is actually used for the requested signature generation. Following the signing process, the private key shall finally be safely deleted from RAM, likewise the PUF response. Note that the private signing key (or any PUF response) is *never* stored in any non-volatile memory of the card at any point in time.

The most challenging part within the key recovery procedure consists in decoding the result from the merge of the PUF response with the helper data. Unfortunately, for most error correcting codes, *decoding* is far more expensive than *encoding* in terms of computation complexity. So the time complexity introduced by the key recovery is expected to contribute significantly to the overall processing time of the entire signature generation process, although decoding is expected to be cheaper in terms of time complexity than the cryptographic signing operation itself.

4.1.3 ID-Card prototype specifications

A first version of the micro-architecture specification for the PUF module of the PUF-based eID-Card IC is available at IFAT. The module will be embedded in an innovation chip, which is an instance of a "close-to-product" implementation with all necessary components used in state-of-the-art eGovernment micro controllers. This specification cannot be made publicly available since there are product features reflected, which are not subject to the development of the HINT project.

The strategy to embed the novel PUF module in a fully equipped eGovernment micro controller setup is the ease of use when implementing novel protocols making use of the PUF functionality. For example it is possible to implement an asymmetric protocol making use of the elliptic curve hardware accelerator for cryptographic operations and finite field arithmetic. Also interfacing to terminal maybe performed by standard communication means such as ISO 7816 contact based or ISO 144443 contactless protocols. The alternative way of implementing the novel PUF module micro-architecture as a stand-alone module would provoke extremely high effort in interfacing the module by proprietary protocols based on manually established connections.

4.2 PMR Prototype

For HINT the aim of the PMR prototypes is to demonstrate the trusted features of some PMR handhelds stemming from the after-delivery detection of HT.

4.2.1 Overview of the prototypes

The PMR prototype will consist of 3 PMR handhelds embedding a CPLD and some test environment. One handheld Hone will be genuine (without HT). The second and third handhelds HTwo and HThree will be modified to include some examples of HT. The same HT will be embedded in HTwo and HThree but the third handheld may be available as an open board to ease measurements and to better show the means and results of HT detection process.

The HTwo and HThree can show that due to the HT presence some behaviour changes by comparing to Hone. An important aspect is the intended effects of the HT. Possible effects of the HT might include reduction in system reliability, the implementation of a backdoor in order to leak some secret information, a change of the functionality, or even complete destruction of the system (we will avoid the latter). Those modified behaviours need to be further defined after working on WP3 and WP4.

The actual scenarios that can be demonstrated could be for example:

- freezing the communication buses between the processors and show disruption on communications
- showing some information leakage that can result from a HT insertion

We shall also define where and how this modified behaviour can be implemented on the handheld. The possible HT can for instance infect a FPGA or CPLD connected to an internal bus that could be abused. Also, such components which might appear to be harmless and non-critical can be used for malicious purposes. For example, a power source which introduces glitches sporadically could be used to cause errors and therefore enable a fault attacks.

In addition, the test environment can be used to show the detection functioning by applying it to Hone (which shall show no detection) or HTwo and more completely to HThree (which both shall show detection of HT).

4.2.2 The "on-the-field" or "at-time-of-use" checking

The "on-the-field" or "at-time-of-use" checking means that regularly the user submits his handheld to a test environment. This could occur when charging the device on a docking station for instance. It will tell him whether his handheld is still genuine (it was on the delivery time as the same kind of verification was done at the end of manufacturing process to warranty the delivery). In case a HT is detected, or if there is any doubt there might be one, the handheld can be sent back to maintenance or to manufacturer for further investigation.

First the test environment shall show that handheld Hone is genuine, e.g. without HT. Then the test environment could show detection of HTwo's trojan on HTwo itself or on HThree with enhanced capacity detection.

The specification and implementation of those prototypes shall be further studied in WP4, based on the outcomes from WP3.

The challenges in these prototypes are to implement in a real operating device a representative HT whose behaviour can be visible. This is mandatory to actually show the demonstration. The next big challenge is to show detection of HT on this real device like HTwo or even as an open board of a real device like HThree.

Chapter 5 Conclusion

The HINT project addresses the trustworthiness of critical devices and their components using a novel scheme relying on two main technologies, PUF-based authentication and Side-Channel based integrity verification.

In the current document, the use case based security analysis done in D1.1 [REF 1] is translated into technical terms to highlight the technological bricks that shall be studied in HINT.

Then details of the architectures that exploit these two technologies (PUF-based architecture and SCA-based architecture) are given. Those trust architectures are detailed in chapter 3 "HINT Architectures" including the requirements to implement and test those features.

Even if they differ in their nature, when combined, they complement each other to achieve genuine complementary anchors in order to enable a trusted computing architecture for critical devices. Furthermore the suggested architectures shown in the HINT project on smart cards and handhelds are expected to be applicable to many different platform types such as smart cards or SD cards, smart phones or mobile devices, even workstations and servers.

The most challenging part is to achieve regular “on-the-field” or “at-time-of-use” checking. This will be demonstrated in the two HINT application prototypes that we describe in last chapter, "HINT Application prototypes".

Finally this WP1 has stated the usages and security issues of the overall trust architectures. The preliminary definitions of the PUF-based architecture and the SCA-based architecture are laid out. This document has defined the scope of work of the next work packages: WP2, WP3 and WP4. Current findings will be supplemented and improved throughout the project during those next WPs as results become available.

Bibliography

[REF 1] HINT D1.1 “Report on use case and architecture requirements”, February 2013

[REF 2] “Protection Profiles for secure signature creation device – Part 4: Extensions for device with key generation and trusted communication with certificate generation application”

[REF 3] “Machine readable travel document with ICAO application, extended access control with PACE (EAC PP)”, BSI-CC-PP-0056-V2-2012

[REF 4] “Security IC Platform Protection Profile”, Version 1.0, BSI-CC-PP-0035-2007

[REF 5] “Java Card Protection Profile – Open Configuration”, Version 3.0, 2012, ANSSI-CC-PP-2010/03-M01

[REF 6] Cryptographic Modules, Security Level "Enhanced", Version 1.01, BSI-CC-PP-0045-2009

[REF 7] “Common Criteria Management Committee – Vision Statement for the future direction of the application of the CC and the CCRA”

[REF 8] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data,” SIAM J. Comput., vol. 38, no. 1, pp. 97-139, Mar. 2008.

[REF 9] B. Gassend, M. van Dijk, D. E. Clarke, E. Torlak, S. Devadas and P. Tuyls, “Controlled physical random functions and applications,” ACM Trans. Inf. Syst. Secur., vol. 10, no. 4, pp. 1-22, Jan. 2008.

[REF 10] J. Delvaux and I. Verbauwhede, “Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise,” in IEEE Int. Symposium on Hardware-Oriented Security and Trust, HOST 2013, Jun. 2013.

[REF 11] A. Konczakowska and B. M. Wilamowski, “Noise in Semiconductor Devices,” Industrial Electronics Handbook, vol. 1 Fundamentals of Industrial Electronics, 2nd Edition, chapter 11, CRC Press 2011.

[REF 12] S.S. Mansouri, and E. Dubrova, "Ring oscillator physical unclonable function with multi level supply voltages," Computer Design (ICCD), 2012 IEEE 30th International Conference on Computer Design, pp.520-521, Sept. 30 2012 - Oct. 3 2012

[REF 13] J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas, “A technique to build a secret key in integrated circuits for identification and authentication applications,” in IEEE Symposium on VLSI Circuits, VLSIC 2004, pp. 176-179, Jun. 2004.

[REF 14] Z. Paral, and S. Devadas, “Reliable and efficient PUF-based key generation using pattern matching,” International Workshop on Hardware-Oriented Security and Trust - HOST , 2011.

[REF 15] D. Merli, D. Schuster, F. Stumpf and G. Sigl, “Semi-invasive EM attack on FPGA RO PUFs and countermeasures,” Proceedings of the Workshop on Embedded Systems Security (WESS), 2011.

[REF 16] D. Merli, D. Schuster, F. Stumpf and G. Sigl, “Side-channel analysis of PUFs and fuzzy extractors,” Proceeding TRUST’11 Proceedings of the 4th international

conference on Trust and trustworthy computing, pp. 33-47, 2011.

[REF 17] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas and J. Schmidhuber, “Modeling attacks on physical unclonable functions,” In proceedings of the 17th ACM conference on Computer and communications security, 2010.

[REF 18] D. Yamamoto, G. Hospodar, R. Maes and I. Verbauwhede, “Performance and Security Evaluation of AES S-Box-Based Glitch PUFs on FPGAs,” SPACE 2012.

[REF 19] P. Tuyls, G.J. Schrijen, B. Skoric, J.V. Geloven, N. Verhaegh and R. Wolters, “Read-Proof Hardware from Protective Coatings,” In proceedings of CHES, 2006.

[REF 20] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, B. Sunar. “Trojan Detection using IC Fingerprinting”. In IEEE Symposium on Security and Privacy, 2007, IEEE Computer Society.

[REF 21] S. Narasimhan, S. Bhunia. “Hardware Trojan Detection”, in M. Tehranipoor and C. Wang (eds.), Introduction to Hardware Security and Trust, Springer 2012.

[REF 22] A. Lakshminarasimhan. “Electromagnetic side-channel analysis for hardware and software watermarking”. MSc Thesis, University of Massachusetts Amherst, September 2011.

[REF 23] S. Dupuis, G. di Natale, B. Rouzeyre. “Is side-channel analysis really reliable for detecting hardware trojans? ”, In proceeding of Conference on Design of Circuits and Integrated Systems, 2012.

[REF 24] J. Zhang, H. Yu, Q. Xu. “HTOutlier: Hardware trojan detection with side-channel signature outlier identification”. In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2012.

[REF 25] S. Narasimhan, X. Wang, D. Du, R. Subhra Chakraborty, S. Bhunia. “TeSR: A robust temporal self-referencing approach for Hardware Trojan Detection”. In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2011.

[REF 26] B. Mounier, A-L. Ribotta, J. Fournier, M. Agoyan, A. Tria. “EM Probes Characterization for Security Analysis”. In D. Naccache (Ed.): Quisquater Festschrift, LNCS 6805, pp. 248-264, 2012, Springer.

[REF 27] L. Sauvage, S. Guilley, Y. Mathieu. “ElectroMagnetic Radiations of FPGAs: High Spatial Resolution Cartography and Attack of a Cryptographic Module”, in ACM TRETS, Volume 2 Issue 1, March 2009, ACM

[REF 28] J. Heyszl, D. Merli, B. Heinz, F. De Santis, G. Sigl. “Strengths and Limitations of High-Resolution Electromagnetic Field Measurements for Side-Channel Analysis”, in proceedings of Cardis 2012, LNCS 7771, pp. 248-262, Springer

[REF 29] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, G. Sigl “Localized Electromagnetic Analysis of Cryptographic Implementations”, in proceedings of CT-RSA 2012, LNCS 7178, pp. 231-244, Springer

[REF 30] C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, P. Tuyls. “Efficient Helper Data Key Extractor on FPGAs”, in Proceedings of CHES, 2008

[REF 31] Ilan Shomorony. “Authentication schemes based on physically unclonable functions”. Master’s thesis. Worcester Polytechnic Institute, 2009.

[REF 32] G. Hospodar, R. Maes, and I. Verbauwhede, “Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling poses strict Bounds on Usability,” in IEEE International Workshop on Information Forensics and Security, WIFS 2012, pp. 37-42, Dec. 2012.