Disclaimer

Statement for open documents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780108. Any dissemination of results here presented reflects only the consortium view.

D5.1 Security and Trust Models

Document Identification			
Status	Final	**Due Date**	30/09/2018
Version	1.0	**Submission Date**	28/09/2018

Related WP	WP5	**Document Reference**	D5.1
Related Deliverable(s)	D3.1, D7.1	**Dissemination Level(*)**	PU
Lead Participant	UH	**Lead Author**	Kimmo Järvinen (UH)
Contributors	KU Leuven, XLAB, ATOS, Wallix, KUD	**Reviewers**	Michel Abdalla (ENS) Alvaro Garcia Recuero (ATOS)

Keywords:
Hardware, Adversary Model, Trust, Security, Hardware Support, Requirements

This document is issued within the frame and for the purpose of the FENTEC project. This project has received funding from the European Union's Horizon2020 under Grant Agreement No. 780108. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

(*) Dissemination level.-PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

List of Contributors	
Name	Partner
Kimmo Järvinen	UH
Josep Balasch	KU Leuven
Svetla Nikova	KU Leuven
Sujoy Sinha Roy	KU Leuven
Miha Stopar	XLAB
Alvaro Garcia Recuero	ATOS
Norman Scaife	Wallix
Brecht Wyseur	KUD
Yolan Romailler	KUD

Document History			
Version	Date	Change editors	Changes
0.1	04/06/2018	Svetla Nikova (KU Leuven)	ToC
0.2	29/06/2018	Josep Balasch, Sujoy Sinha Roy, Svetla Nikova (KU Leuven)	Storyline + structure
0.3	25/07/2018	Josep Balasch (KU Leuven)	First draft
0.4	06/09/2018	Kimmo Järvinen (UH), Alvaro Garcia Recuero (ATOS)	Added text, a draft of Atos use case
0.5	12/09/2018	Kimmo Järvinen (UH), Josep Balasch (KU Leuven), Miha Stopar (XLAB), Norman Scaife (Wallix), Alvaro Garcia Recuero (ATOS), Brecht Wyseur (KUD)	Integrated input from partners
0.6	14/09/2018	Kimmo Järvinen (UH), Josep Balasch (KU Leuven), Alvaro Garcia Recuero (ATOS), Yolan Romailler (KUD)	Version for internal review
0.7	25/09/2018	Kimmo Järvinen (UH)	Addressed most of the reviewers' comments
1.0	27/09/2018	Kimmo Järvinen (UH), Josep Balasch (KU Leuven)	Final version

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable Leader	Kimmo Järvinen (UH)	28/09/2018
Technical Manager	Michel Abdalla (ENS)	28/09/2018
Quality Manager	Diego Esteban (ATOS)	28/09/2018
Project Coordinator	Francisco Gala (ATOS)	28/09/2018

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
ABE	Attribute Based Encryption
AES	Advanced Encryption Standard
AWS	Amazon Web Services
CP-ABE	Ciphertext-Policy Attribute Based Encryption
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DDR	Double Data Rate
DES	Data Encryption Standard
DPA	Differential Power Analysis
ECC	Elliptic Curve Cryptography
FA	Fault Attack
FE	Functional Encryption
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
HW	Hardware
I/O	Input/Output
IND-CPA	INDistinguishability under Chosen Plaintext Attack
IoT	Internet of Things
KP-ABE	Key-Policy Attribute Based Encryption
OP-TEE	Open Portable Trusted Execution Environment
PCB	Printed Circuit Board
PKG	Private Key Generator
RLWE	Ring-Learning With Errors
RSA	Rivest-Shamir-Adleman
SCA	Side-Channel Attack
SIMD	Single Instruction Multiple Data
SoC	System-on-Chip
SQL	Structured Query Language
SPA	Simple Power Analysis
SW	Software
TEE	Trusted Execution Environment
WP	Work Package

Executive Summary

In this deliverable D5.1 “Security and Trust Models”, we discuss the hardware-supported FE schemes studied in WP5 of FENTEC and describe the security and trust models for the computing platforms executing computations required by the FE schemes. That is, in this deliverable, we solely focus on the implementation layer of the FE schemes and all discussion on efficiency as well as security and trust models are about the implementations rather than the theoretical schemes or algorithms per se. We define a generic computing platform modeling the large variety of computing platforms covered by FENTEC and its use cases. We employ this generic model to define adversarial, security and trust models that cover different types of adversaries as well as layers of security and trust in various possible deployments of FE. We study multiple proposals for FE and provide high-level discussion about the critical aspects when they are deployed in actual implementations. The general computing platform also allows discussion of how to optimally map different parts of the FE schemes into the computing platform. For example, we discuss which parts of the FE schemes should be delegated to specific HW accelerators (co-processors) or trust anchors for improved performance or security, respectively. We also map the use cases studied in FENTEC to the models described in this deliverable to provide practical examples of how the models fit practice. Finally, we identify specific requirements for hardware development environments (development kits) to be used in implementation work in the three implementation tasks of WP5.

1 Introduction

1.1 Purpose of the document

This deliverable D5.1 "Security and trust models" describes the requirements for hardware-supported functional encryption schemes. It will set the guidelines and requirements for hardware implementations to be developed in FENTEC. This deliverable—and, in particular, the requirements it sets—are based on the analysis performed in Task 5.1 of WP5. These requirements include trust, adversary and security models that are based on the earlier requirement analysis performed in WP3 and their corresponding use case specifications in WP7.

The focus of this deliverable is strictly on the implementation layer and higher abstraction layers are out of the scope this deliverable (e.g., algorithms and system/application layers), as long as they do not have direct implication for the implementations. Consequently, all references to efficiency and security are implied for implementations of functional encryption schemes, rather than for their algorithms per se. For example, this deliverable focuses on adversaries trying to exploit information leakage from implementations via physical side-channels, not on adversaries trying to exploit information leakage from algorithms or protocols.

This deliverable will also describe the level of hardware support that is required for different functional encryption schemes. In particular, it will describe, on the one hand, which parts of implementations must be trusted and which can be assumed susceptible for physical attacks based on the above requirements, and, on the other hand, which parts of the functional encryption schemes should be implemented in hardware and which in software (HW/SW codesign) in order to maximize the efficiency. The focus of the latter is how to optimally map a functional encryption algorithm or a scheme to a specific computing platform and, thus, includes (a) design of hardware coprocessors and accelerators, (b) software optimizations tailored to specific existing architectural features (for example, assembly-optimized code, usage of SIMD extensions, etc.), and (c) their efficient interplay in HW/SW codesigns. The levels of hardware support link directly to the schemes to be developed within Tasks 5.2–5.4.

1.2 Structure of the document

This deliverable is structured as follows. Section 2 defines the security models by describing a generic computing platform and identifying three levels of trust for this platform (full, partial, or no trust). These security models are closely linked to the three tasks of WP5 (hardware-optimized, hardware-assisted, and hardware-operated). Section 3 discusses hardware support for FE implementations developed in the tasks of WP5. We introduce the basics of FE schemes that are of main interest to WP5, explore the security and efficiency trade-offs of FE implementations, identify specific building blocks that are needed in implementations of each FE scheme, and discuss how the FE implementations could be targeted by an adversary using side-channel and/or fault attacks. Section 4 links the discussion of the previous sections to the needs of the three use cases of FENTEC. Specifically, it identifies what are the requirements of the use cases for the implementations to be developed in WP5. Section 5 discusses the hardware requirements for the tasks of WP5 and, based on these requirements, identifies certain development kits that will be used. The deliverable ends with a summary and conclusions in Section 6.

2 Security models

This section introduces different elements that allow defining the requirements for hardware-supported FE schemes. We begin by introducing a generic description of a computing platform and discussing the assumed adversarial models. We define three levels of trust in the platform that give rise to the underlying security models of three different tasks of WP5.

2.1 Generic computing platform

To focus the discussion and to cover the large scope of different computing platforms considered in WP5 of FENTEC, we begin by describing a generic computing platform that captures the essential features of the different computing platforms. The generic computing platform consists of four main elements: main processor unit(s), main memory, I/O interfaces, and co-processor(s). The following describes these elements in more details:

- **Main processor unit(s).** The main processor unit is a general-purpose processor, that is, a microprocessor or a microcontroller that runs a SW program. The main processor unit is responsible for controlling the operations of the system. The SW program can implement various levels of the systems ranging from a full SW implementation that executes all operations computed in the system (confer WP4 of FENTEC) to a host processor that delegates most of the computation load to co-processors and only takes care of controlling the data flows and interfacing. The main processor units of the generic computing platform may consist of several processor units (processor cores).
- **Main memory.** The main memory is used for storing large amounts of data within the computing platform. This memory can be either volatile (e.g., DDR) or non-volatile (e.g., Flash or hard-drive) memory. This memory does not include local storages within the main processor unit(s) (e.g., registers, cache memories, etc.) or the co-processors (e.g., internal memory blocks of FPGAs). In general, accessing the main memory is slower than using these local storages, but significantly faster than any data storage outside the computing platform (accessed via the I/O interfaces). In most cases, the main memory allows storing relatively large amounts of data (from some MBs up to several GBs).
- **I/O interfaces.** These are the interfaces that the computing platform uses for connecting with the rest of the system and other systems. These interfaces can include a large variety of different types of interfaces, both wired and wireless. Some examples include PCI Express, USB, Ethernet, WiFi (IEEE 802.11), Bluetooth, etc.
- **Co-processor(s).** These are additional HW engines designed for a specific purpose, which are cryptography related functionalities in the context of FENTEC. The reasons for introducing HW co-processors can originate from efficiency and/or security. Co-processors are typically significantly more efficient, e.g., in terms of speed, power and energy consumption compared to SW implementations of the same functionality and allow offloading these expensive operations from SW to HW. From the security perspective, co-processors have an advantage compared to SW implementations because they allow isolation of security-critical computations from other parts of the system and are also easier to protect against side-channel and fault attacks. Consequently, HW co-processors can be used as trust anchors in

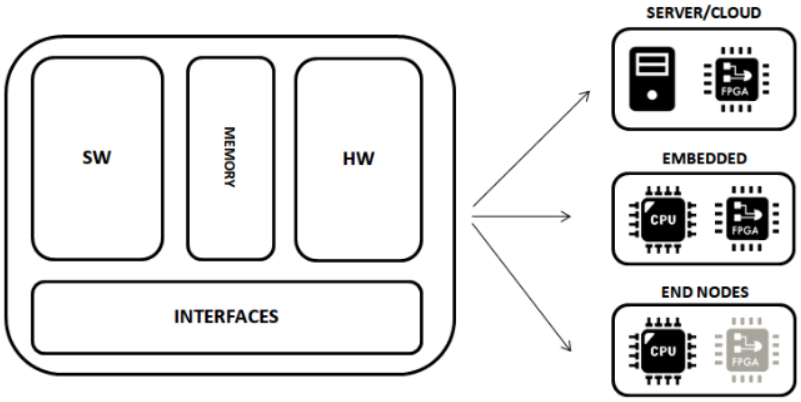

Figure 1: Generic computing platform (left) and different instantiations (right).

the generic computing platform.

The advantage of defining the above generic computing platform is that it covers all ranges of computing platforms that are of interest to FENTEC. As shown in Figure 1, we have identified the three main types of computing platform to be studied in FENTEC: server/cloud platforms (high-end), embedded platforms (mid-end), and end nodes (low-end). All of them are covered by the generic computing platform as discussed in the following.

- **Server/Cloud [high-end].** The main processor units are powerful microprocessors (e.g. x64 architecture) with several parallel cores. Hardware support can be available internally, in the same SoC (e.g. Intel SoC FPGAs [61]), or available externally by interfacing to FPGA platforms (e.g. Amazon EC2 F1 clusters [10] or a development board in PCIExpress). The main memory is large enough to be considered unlimited in most cases relevant for FENTEC. I/O interfaces are fast (up to several Gbps).
- **Embedded [mid-end].** The main processor units are mid-range microprocessors (e.g. ARMv8 [14] and ARMv7 [12] architectures) and there may be either one or several cores available. Hardware support can be available internally, in the same SoC (e.g. Xilinx Zynq SoC with ARM cores [95]), or available externally by interfacing to FPGA platforms (e.g. on the same PCB). Memory is more limited than above, but most probably sufficient for the purposes of FENTEC. The platform may have several I/O interfaces, but they are typically slightly slower than above.
- **End nodes [low-end].** The main processor unit is a constrained processor (e.g. ARMv7-M [13] or AVR architectures [74]). Hardware acceleration could only be available externally, by interfacing to FPGA platforms, but this is subject to strict power and energy constraints. Memory is limited and I/O interfacing is typically slow and also under strict power and energy budgets.

2.2 Adversarial model

The natural adversarial model in WP5 is that of an implementation-level attacker, i.e. one that targets functionalities executed by a computing platform. In particular, we consider an adversary that targets cryptographic implementations, whether in the form of hardware co-processors or

software routines executed in the main CPU. The goal of the adversary consists in extracting the secret keys stored in (and processed by) the computing platform.

It is important to stress that such an adversary exploits characteristics exclusively at the implementation layer. It is therefore conceptually different from adversaries that target vulnerabilities at other abstraction layers, for instance, algorithm-level adversaries (which may use mathematical tools to attack the underlying cryptographic constructions) or system-level adversaries (which may use software exploits to e.g. gain access to privileged information).

Research on implementation attacks has been very active since the late 90s, when physical attacks were brought to the attention of the research community. These attacks exploit the physical nature of implementations running on a computing platform. They allow adversaries to gain information about internal computations through observation and/or manipulation of physical channels. Combining this information with known input or output messages, enables the adversary to retrieve the secret cryptographic keys used by the implementation.

In general, attacks are often categorized in two families depending on the adversary role:

- **Side-channel attacks** (SCA) are passive: the adversary simply monitors and analyses information that inherently leaks through physical channels during normal operation of the computing platform. There exist several side-channel sources which have been shown to be exploitable by adversaries. The most notable examples are: execution time [66], power consumption [67] and electromagnetic emanations [51].
- **Fault attacks** (FA) are active: the adversary intentionally triggers errors in the platform to disrupt the expected control and/or data flows. Many techniques can be used by adversaries to this end: from simple voltage manipulations [16] and clock glitches [15], to more complex injection of electromagnetic radiations [81] or laser pulses [90].

Side-channel and fault attacks are acknowledged as one of the main threats against cryptographic implementations. Their simplicity and devastating consequences (i.e. key extraction) has been demonstrated in multiple works targeting different cryptographic implementations: from public-key cryptosystems (e.g. RSA [66, 67, 28], ECC [48, 21] or pairings [64, 79]) to symmetric-key ciphers (e.g. AES [68, 80] or DES [67, 22]). Quite naturally, the uncovering of physical attacks had a big impact for industry and triggered immediate research on mitigation strategies. Nowadays, testing implementations against physical attacks is part of evaluation processes (e.g. Common Criteria [42] or EMVCo) [46] used for certifying the security of commercial products (particularly those aimed to security-sensitive applications such as banking or government identification).

From an adversarial model perspective, it is often assumed that both side-channel and fault attacks demand the attacker to have physical access to the platform under attack. Such requirement is necessary in order to collect the necessary information (for SCA) or for tampering with the platform (for FA). It therefore applies to many real-world products and applications, particularly in the embedded domain: from low-end platforms (e.g. network nodes, smart card tokens, etc.) to mid-end platforms (e.g. smartphones, routers, etc.). Side-channel and fault attacks against high-end platforms are less common but also exist, e.g. timing side-channel attacks have been shown to be applicable when targeting cryptographic implementations running on remote servers [18]. In addition, recent works have shown that power consumption measurements can be obtained [88, 98] or faults inducted [56, 69] remotely in multi-tenant FPGA platforms, i.e. in which resources of a single FPGA are shared among multiple distrusting clients. Based on these results, it may, in some cases, be necessary to consider physical attacks also for high-end comput-

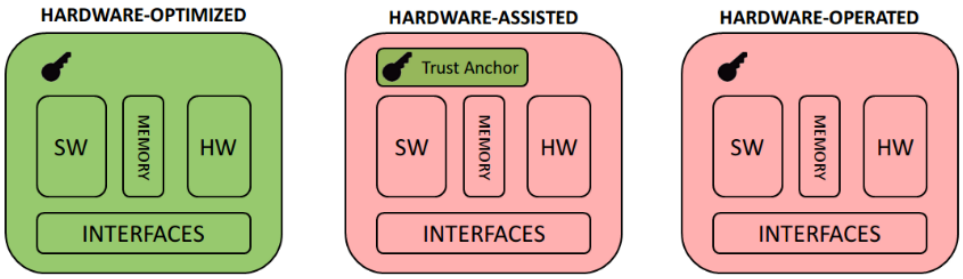

Figure 2: Underlying security models for the different tasks in WP5. Green (trusted environment), red (untrusted environment).

ing platforms in a cloud/server setting, in particular if sharing of resources between distrusted parties is envisioned.

2.3 Trust levels and security models

Determining whether physical attacks pose a realistic threat against a particular implementation depends on two factors: (a) whether the adversarial model applies to the end application (or use case) and (b) whether the platform provides underlying features that can be leveraged on for security.

Based on this, we can distinguish between the following trust levels:

- **Full trust** corresponds to a computing platform that is not susceptible to physical attacks. This can be simply because it is out of reach of adversaries (e.g. in a trusted environment) or because the complete platform is certified to provide protection against physical attacks (e.g. a smart card or secure element).
- **Partial trust** corresponds to a platform which is partially susceptible to physical attacks. In general, this implies the existence of two domains: one that includes components and/or architectural features to enhance security and one that does not feature any countermeasure.
- **No trust** corresponds to a platform which provides no guarantee against physical attacks. Examples are general-purpose computing platforms (e.g. processors or FPGAs) which are aimed to general applications and therefore do not consider security against physical attacks.

These levels give rise to the security models assumed in the different tasks in WP5 (see illustration in Figure 2 for mapping to the platform):

- **Hardware-optimized** (T5.2) corresponds to a full trust scenario. The goal is to develop highly efficient FE implementations by optimizing the reference implementations of WP6.
- **Hardware-assisted** (T5.3) corresponds to a partial trust scenario. The goal is to find a suitable split of operations in the FE implementation such that critical computations are managed by a trust anchor.
- **Hardware-operated** (T5.4) corresponds to a no trust scenario. The goal is to incorporate countermeasures against SCA/FA in the critical building blocks of the FE implementation.

Analysing the security of the implementations developed in T5.4 is envisioned in T5.5.

3 Hardware support for functional encryption

Functional encryption [31, 77] overcomes all-or-nothing limitations of classical encryption where the relationship $D(E(x)) = x$ holds for encryption $E(\cdot)$ and decryption $D(\cdot)$. In functional encryption, decryption does not output x but $f(x)$ for some function f. To be precise, for each function f a key f_{key} is needed such that $D(E(x), f_{\text{key}}) = f(x)$.

Functional encryption thus allows fine-grained access control over encrypted data. The basic functional encryption schemes provide functionality for decrypting inputs provided and encrypted by a single party. Lately, functional encryption schemes for decrypting inputs provided and encrypted by different parties have been designed [4, 37]. The first schemes are known as single-input schemes, while the former as multi-input.

Functional encryption schemes involve three parties:

- Trusted entity that generates cryptographic keys for encryption (master keys) and decryption (FE keys)
- Encryptor
- Decryptor

In both, single-input and multi-input schemes multiple encryptors can be involved but, in the former, there is a stronger security property: the decryptor does not obtain any knowledge on particular plaintexts but only on a group of plaintexts.

Besides the number of parties involved in the encryption phase, the schemes differ whether they are selectively or adaptively secure and whether they hide or not the function f. The following three subsections discuss different types of schemes and attack scenarios for each of the types.

It has to be noted that in the last years the research on functional encryption can be divided into two categories:

- Concrete, efficient schemes for restricted functionalities (like inner-products)
- General functionality schemes based on indistinguishability obfuscation or multilinear maps

The schemes of the second type rely on non-standard and ill-understood assumptions. Furthermore, these schemes are in many cases extremely time-consuming. FENTEC is committed to the design and implementation of efficient schemes of the restricted functionality (but still of practical interest).

3.1 Single-input and multi-input schemes

Multi-input schemes are generalization of single-input schemes and has been introduced by Goldwasser *et al.* [57]. In multi-input schemes n encryption slots are explicitly given - a user assigned to the i-th slot can create a ciphertext $E(x_i)$ from his own plaintext x_i. In multi-input schemes, a key corresponding to function f enables a user holding encryptions of $x_1, \ldots, x_n$ to compute $f(x_1, \ldots, x_n)$ without learning any additional information on the x_i's. Functional encryption enables computing on encrypted data and its multi-input flavor allows for mining aggregate information from several different data sources.

Note that single-input scheme could be used for multiple users but in this case not only $f(x_1, \ldots, x_n)$ would be revealed but also $f(x_1), \ldots, f(x_n)$.

The security requirement for single-input and multi-input schemes is the same - decryption keys should be collusion resistant. That means that a group of decryption keys should not enable retrieving any information about the encrypted messages, beyond the union of what can be individually learnt by each of the decryption keys. This security notion is named indistinguishability and, informally, states that an adversary that possesses the secret keys corresponding to functions $f_1, \ldots, f_t$ should not be able to decide which of the two challenge messages x_0 or x_1 was encrypted, as long as $f_1(x_0) = f_1(x_1), \ldots, f_t(x_0) = f_t(x_1)$.

The attack vectors are thus the same in single-input and multi-input schemes. While hardware optimizations might help speeding up the computation in all three phases (generation of keys, encryption, decryption), in most cases dedicated hardware accelerators might be most needed for decryption - for example, the most time-consuming operation in many schemes is the decrypt operation due to the discrete logarithm computation (not needed in all schemes).

The decrypt operation presents a bottleneck in many multi-input functional encryption use cases also due to the fact that a decryptor executes computations on much bigger data than each particular encryptor - the data increases with the number of encryptors. Let us have a look at some multi-input functional encryption use cases:

- **SQL queries on encrypted database.** SQL queries could be done with ordinary (single-input) functional encryption, but then a separate key for each SQL query would be needed; to avoid this encrypted database could present ciphertext c_0 and encrypted users query could be denoted as c_1, then $f(x_0, x_1)$ could be computed where x_0 and x_1 are plaintexts corresponding to the ciphertexts c_0 and c_1 respectively; if database is dynamic rather than static, each entry could be separately encrypted to allow faster updates - in this case function f would provide $f(x_0, x_1, \ldots, x_n)$ where $(x_0, \ldots, x_n)$ presents a database.
- **Computing over encrypted data stream.** Law enforcement agencies may require to be able to check for suspicious activities in a stream of encrypted phone calls or video frames; this could be achieved by computing only $f(p_0, \ldots, p_n)$ where p_i is the i-th phone call.
- **Non-interactive differentially private data.** Different stakeholders have different data collections (for example, hospitals holding individual blood samples); data collections can be encrypted with functional encrypted scheme which allows an analysis over all collections without releasing the data in the clear; note that data holders can prepare data collections non-interactively which is not the case when using non-cryptographic methods to achieve differential privacy.
- **Multi-client delegation of computation.** Multiple (weak) clients wish to jointly delegate the computation of function f on their inputs $x_0, \ldots, x_n$ to a computationally powerful server.

3.2 Selective and adaptive security

At a high level, selective security guarantees security only for messages that are fixed ahead of time (even before the adversary interacts with the system). Such level of security may be justified in some cases, however adaptive or full security is more realistic option which guarantees security

even for messages that can be adaptively chosen at any point in time. In selectively secure schemes (selective security under chosen-plaintext attacks denoted as s-IND-CPA), the adversary must begin by declaring the challenge vectors before the keys are generated. That means the adversary declares the challenge messages x_0 and x_1 upfront, before seeing the public keys.

In adaptively secure (IND-CPA) schemes the adversary can declare messages after seeing the public keys. Selective security is a weaker notion and, in most cases, serves only as a stepping stone to proving adaptive security or leading to a new scheme which is fully secure. An example of such steps is functional encryption scheme for inner products—first, Abdalla *et al.* [4] designed selectively secure scheme, later Agrawal *et al.* [7] designed fully secure scheme for inner products which build on top of [4]—it uses the same ElGamal-like system but with an additional group element.

Whenever available, a fully secure scheme should be used and FENTEC is thus focusing on the implementation of fully-secure schemes (although the hardware optimizations for both types would usually be the same and applicable for both types).

3.3 Function-hiding and non-function hiding schemes

In many applications, it is important to consider privacy of the function being computed. For example, if a hospital uses cloud storage to store medical records of its patients, the privacy of data is protected by data being encrypted. However, when later computing a function f directly at the cloud storage provider's premises, the hospital might not want to reveal anything about f (to the cloud storage provider).

Many existing functional encryption schemes do not guarantee any hiding of the function. Informally, function privacy requires that given a decryption key f_{key} for a function f, one should not be able to learn any information about f, except $f(x, f_{key})$. For challenge functions f_0 and f_1, an adversary should not be able to decide for which it is holding secret keys, as long as $f_0(x_0) = f_1(x_1)$.

Regarding the attack vectors and hardware optimizations for low-level primitives, there are no differences in non-function-hiding and function-hiding schemes. Although obviously, in function-hiding schemes an adversary has less information at his disposal. Also, function-hiding schemes tend to be more computationally expensive because of use of extra computational layers [4].

3.4 Hardware optimizations

Existing constructions of practical functional encryption schemes can be divided into three groups based on either:

- Modular arithmetic,
- Pairings, or
- Lattices.

The next subsections discuss performance bottlenecks and some of the existing hardware optimizations for each of the three groups.

3.4.1 Modular arithmetic schemes

Modular arithmetic is a system of arithmetic for integers where values reset to zero and begin to increase again after reaching a certain value (modulo). Most traditional public-key cryptography has been based on modular arithmetic (RSA, Diffie-Hellman, ElGamal). It also enables finite fields which underlie elliptic curves. Numerous schemes are still being designed in modular arithmetic due to the well-understood cryptographic assumptions. Functional encryption schemes designed in modular arithmetic are, for example, schemes based on Decisional Diffie-Hellman assumption and Paillier's encryption in [4, 7, 3].

The most expensive modular operation is exponentiation which makes use of repeated modular multiplications. Thus, the efficiency of modular arithmetic schemes depends on the implementation of modular exponentiation. Optimizations to reduce the frequency of modular multiplications and the time for each multiplication are needed.

Modular multiplication is a complicated operation which require expensive trial division operations and cannot be implemented efficiently in hardware. Montgomery introduced [75] in 1985 an algorithm for fast modular multiplication without trial divisions which is well-suited for hardware implementation. In Montgomery modular multiplication, all multiplications and division operations are substituted with shift operations.

Since then many optimizations have been made for modular multiplication using Montgomery method. The many works in this domain include for example the following ones. Daly *et al.* [43] presented a pipeline architecture for implementing modular multiplications and exponentiations used in RSA. Lim *et al.* [70] devised an efficient multiplier using signed digit representation. Bernard [17] presented a scalable pipeline architecture. Lin *et al.* [71] proposed a new technique to realize Montgomery multiplication in high-radix form. Vollala *et al.* [92] proposed bit forwarding techniques for efficient implementation of modular exponential algorithms suitable for hardware implementation.

3.4.2 Pairing schemes

The unique properties of pairings have enabled many new cryptographic protocols that had not previously been feasible. Pairings were first used in 1991 to attack cryptosystems using supersingular elliptic curves. Ten years later, three seminal papers appeared where pairings were used to design novel (or improved) protocols: identity-based encryption [29], short signature scheme [30], and one round tripartite key exchange [62]. Since then numerous pairing-based protocols have been designed for identity-based schemes, key establishment schemes, privacy-enhancing techniques such as anonymous credentials, and functional encryption schemes.

In many cases a particular functionality is known to be constructed only with the usage of pairings. For example in [4] function-hiding scheme is based on pairings while non-function-hiding multi-input inner product scheme is constructed without pairings (an advancement of the previous scheme [5] which used pairings for the same functionality).

A pairing is a function $e : G_1 \times G_2 \rightarrow G_T$ that maps two elements of an elliptic curve to an element of a finite field G_T. Function e is non-degenerate ($e(g_1, g_2) \neq 1$) and bilinear. Computation of pairings involves finite field arithmetic operations and is highly time-consuming. Finite field arithmetic requires different instructions than conventional arithmetic and general purpose

multiprocessors do provide support for it. For this reason software implementations are overcome by specialized implementations in hardware.

For example the works [52, 6, 20] proposed dedicated architectures for computing the eta pairing over binary fields. Cheung *et al.* [36] proposed an architecture for the ate and optimal pairings over prime fields and ordinary curves. Estibals [47] proposed a novel method for efficient hardware implementations of the Tate pairing using tower field arithmetic. None of the previous works is flexible in terms of which elliptic curve to use or the tower field. Ghosh *et al.* [53] proposed a programmable architecture for prime fields, while Beuchat *et al.* [19] designed a programmable architecture for ternary fields. Brinci *et al.* [33] proposed a design of a 258-bit multiplier for computing pairings over Barreto-Naehrig curves—the proposed design is optimized for Xilinx FPGA. Argenziano [11] presented an architecture and implementation of a highly customized FPGA design targeting the most time consuming encryption operation and exploiting the properties of the Discrete Fourier Transform over finite fields.

3.4.3 Lattice schemes

As we get closer to the realization of quantum computers, the resistance of cryptographic protocols to the post-quantum attacks is becoming highly important. Lattice-based cryptography is believed to be secure against quantum computers. Lattice-based cryptographic constructions are based on the presumed hardness of lattice problems for example shortest vector problem. Currently the most efficient construction admitting a known theoretical proof of security is Regev cryptosystem [82] based on Learning With Errors (LWE) problem. Its efficiency is approaching the level that is reasonable to be used in real-world applications. Regev system and its improvements are commonly used in functional encryption schemes [4, 7, 3].

There are two main bottlenecks appearing in LWE-based schemes. These are: matrix-vector and matrix-matrix modular multiplications, and sampling random values distributed according to discrete Gaussian distribution and uniform distribution. Both could be solved by a standalone hardware implementation solving this particular task on arbitrary inputs, or as part of a complete hardware implementation of a scheme.

The discrete Gaussian sampling is a problem of sampling values distributed according to Gaussian distribution but limited only to discrete values. There has been a lot of effort made to present efficient algorithms for the task, see [65, 45], and creating hardware implementations [44, 58]. Differently than in the standard use of (ring-)LWE schemes, functional encryption based on LWE demands sampling relatively big discrete numbers. This presents a challenge since existing hardware implementations are based on precomputed lookup tables, non-usable in the functional encryption case. The current FENTEC software implementation is based on the algorithm from [45] and is open for hardware optimization. One of the bases needed to sample fast is also a fast hardware implementation of cryptographically secure sampler of random uniformly distributed integers. The matrix-vector and matrix-matrix modular multiplications are standard computationally demanding operations which can also be made faster using hardware implementation. Importantly, one way to avoid such costly operations is to replace LWE schemes with ring-LWE schemes. This replacement needs to be first proved theoretically to be secure. If this is done, then the above multiplications can be replaced by polynomial multiplications that can be also further improved by a hardware implementation of algorithm such as FFT or similar, see [86].

3.4.4 Functional encryption and machine learning classifiers over encrypted data

Functional encryption is one of the techniques that enable computation on encrypted data - besides homomorphic encryption, secret sharing and zero-knowledge-proofs. While currently using functional encryption the training phase in machine learning process cannot be done over encrypted data, it is possible to build efficient classifiers (see FENTEC machine learning demo [1] for more) once the model is trained. The classifier does not need any access to the plaintext, it works on ciphertexts alone.

However, functional encryption for evaluating a machine learning function on encrypted data is limited in the following way. Machine learning models are based on optimizing parameters that are considered to be real numbers, represented in computers as floats. Since functional encryption schemes are created around the assumption that the input values are integers, the parameters need to be discretized. If the machine learning function is relatively complex, the usual result of this procedure is that the function must operate with relatively big numbers. This presents a challenge, since in many schemes a computation of the discrete logarithm must be performed. This is extremely costly in terms of computation if the result is relatively large. It might be worth considering possible hardware implementations to solve this problem.

3.5 Mitigations against implementation attacks

In this section we review mitigation strategies that have been proposed in the literature to prevent implementation attacks. We highlight specific techniques which may be relevant to the three types of practical FE constructions considered in FENTEC: modular arithmetic, pairing, and lattice schemes. In general, mitigation strategies can be instantiated at many layers, from circuit to algorithmic level. The latter have the advantage of being more generic, and therefore are more interesting to our envisioned research activities in WP5.

3.5.1 Countermeasures against side channel attacks

Countermeasures against SCA aim to break the link between side-channel leakage and sensitive values processed by the implementation. For instance, a natural protection against timing side-channel attacks is to ensure that implementations run in constant time (that is, independent of the secret values they process). While this may seem conceptually easy, i.e. writing software code without conditional execution branches, it has been noted that microarchitectural features of processor units (e.g. cache memories or early-abort multipliers) can enable new timing leakages, see e.g. [18, 59]. Therefore, validating whether a code is or not susceptible to timing attacks can only be guaranteed by profiling its execution on the target platform, for instance using the tool presented in [83].

In general, countermeasures to mitigate side-channel leakages in power consumption and EM emanations are typically divided in two categories [73]: *hiding* and *masking*. Hiding techniques rely on randomizing the leakage behaviour of the device, for instance, by altering the temporal occurances of leakages within an interval. This can be done by incorporating generic features such as random delays or shuffling to the implementation. Masking techniques on the other hand rely on randomizing the data processed in the implementation. This is typically done by splitting

the sensitive variable(s) in multiple shares, and performing the operations directly on the shares. In practice, both strategies can be combined to enhance the security of an implementation.

The protection of traditional public-key cryptosystem implementations against SCA has received significant attention in the research community. It is well-known that square-and-multiply algorithms (for e.g. RSA modular exponentation) or double-and-add algorithms (for e.g. elliptic-curve scalar multiplication) are susceptible to basic *single-trace* SCA attacks. This is simply because their sequence of operations directly depends on the secret being processed, which can be captured in power/EM measurements and exploited by Simple Power Analysis (SPA) [67]. Alternative algorithms such as the square-and-multiply-always (resp. double-and-add-always) [41] break this dependency at the cost of adding dummy operations. A more compact technique is to use the Montgomery power ladder [63], which achieves SPA-resistance without need of dummy operations. Protection against *multi-trace* SCA attacks, e.g. Differential Power Analysis (DPA) [67], can be achieved by applying masking techniques such as message blinding [66], exponent blinding [41] or exponent splitting [40]. An additional technique for ECC-based constructions is to use randomized projective coordinates [41]. Further security enhancements can be achieved by implementing hiding techniques in lower layers, such as modular multiplications. Randomizing the order in which partial products are computed maybe required to prevent certain types of single-trace *horizontal* attacks [39].

Pairings typically utilize elliptic-curve scalar multiplications with certain additional operations for computing the pairing value. Consequently, they share many SCA features with ECC but also have some important differences that must be addressed [79]. While SCA on pairings has received significantly less attention than on ECC, several works are available concerning attacks and countermeasures [25, 64, 79, 93]. Proposed countermeasures include point blinding [79, 25], randomized intermediate variables [89, 93], and projective coordinate randomization [64]. A survey of SCA on pairings is available in [26].

The protection of lattice-based cryptographic schemes against SCA has gained attention in recent years. In particular, several recent works have proposed techniques to protect implementations of Ring-LWE using masking techniques, see for instance [85, 84, 76].

3.5.2 Countermeasures against fault attacks

Countermeasures against FA are often classified in two main groups: *error detection* and *infective computation*. The former adds redundancy mechanisms to the implementation in order to detect the occurrence of faults. The most straightforward instance is to run the implementation (or its inverse) multiple times and compare their results. The latter aims to spread the effects of potential faults within the algorithm, such that the output value is not exploitable by the adversary. Despite its simplicity, many proposals on this direction have been broken in the last years, see e.g. [27, 96] for seminal works.

Cuntermeasures against FA have been widely investigated for traditional public-key constructions. Specially for RSA, several designs have been put forward to protect implementations using the Chinese Remainder Theorem, see for instance [55, 50, 32]. Methods to detect faults on ECC-based constructions include point validation [21] or curve integrity check [38]. An extensive review of countermeasures for ECC can be found in [49].

FA against pairings is studied in [24, 35, 54, 79] with focuses on both the scalar multiplication and additional operations. Countermeasures are based on similar ideas used to countermeasure

SCA as well as on blinding loop boundaries in additional operations [54, 79]. A survey of FA on pairings is available in [26].

Lastly, recent works have investigated the susceptibility of lattice-based constructions to FA. For instance, in [23] the authors enumerate several countermeasures (based on correctness checks and comparisons) to mitigate potential attack paths against implementations of lattice-based signature schemes. More recently in [34], the authors propose countermeasures against Differential Fault Attacks of the lattice-based signature schemes Dilithium and qTESLA. Their solutions are based on double computation and verification-after-sign techniques, as well as re-randomizing the deterministic sampling of the noise.

4 Mapping to FENTEC use cases

The section discusses the use cases from the point of view of computing platforms and maps them to the security and trust models from the previous section. The following discussion surveys the use cases only very briefly and, therefore, readers interested about the specifics of the use cases should consult D7.1 for details.

4.1 Use case #1: Digital currency scenario

The digital currency use case has many parties with different kinds of computing platforms and trust models. In the following paragraph, we shortly outline the main role of each party and discuss what kind of computations are expected by each of them and what are their applicable security and trust models.

- **Government.** This party is the master authority or Private Key Generator (PKG) in the system who is responsible of generating and delivering private keys to other parties involved in the payment system and/or functional encryption protocols. Private keys are generated once for each specific participant if their attributes in Attribute Based Encryption (ABE) do not change often (e.g., attribute revocations) and if the access policy is fixed (no updates to it). Hence, the computational load of this party is not excessive although there are likely several merchant entities involved in the protocol and a very large number of customers in the system. This component or party holds the master secret keys, which already implies a very high security level and additionally, due to its functions, it is also a threat for performance and security into the system if it must be contact frequently by other parties in the protocol. The computing platform here must be placed in a secure and trusted environment (no physical access for potential adversaries), is expected to be dedicated hardware, and is assumed to be a fully trusted component.
- **Exchange.** This party is the core of the payment system, whose main functions are to issue new digital coins, to deliver coins, to check double-spending, and to store encrypted invoices of customers. It also holds significant security-critical information such as keys to issue digital coins, hash codes and denomination of each created/spent coins and encrypted invoices. This party is expected to have a significant processing load, so hardware acceleration is needed. We assume that the computing platform is placed in a secure and fully trusted environment (no physical access for potential external adversaries). Secondly, it is expected to use dedicated hardware that is also trusted. In despite of any relation to hardware needs, it is important to note that internally the Exchange party is honest-but-curious regarding customers invoices.
- **Customer's wallet.** This party's main functions are to receive and spend digital coins, namely performing payments to merchants and together they encrypt invoices. Customers wallet is a third-party mobile app running in the customers smartphone. The computing platform performance depends on the type of deployment so while computational power of modern mobile devices is increasing, the computing platform here is not trusted. Also, adversaries can perform attacks to the device by means of physical or remote attacks [8] to the computing platform which implies the need for countermeasures against side-channel and fault attacks here.

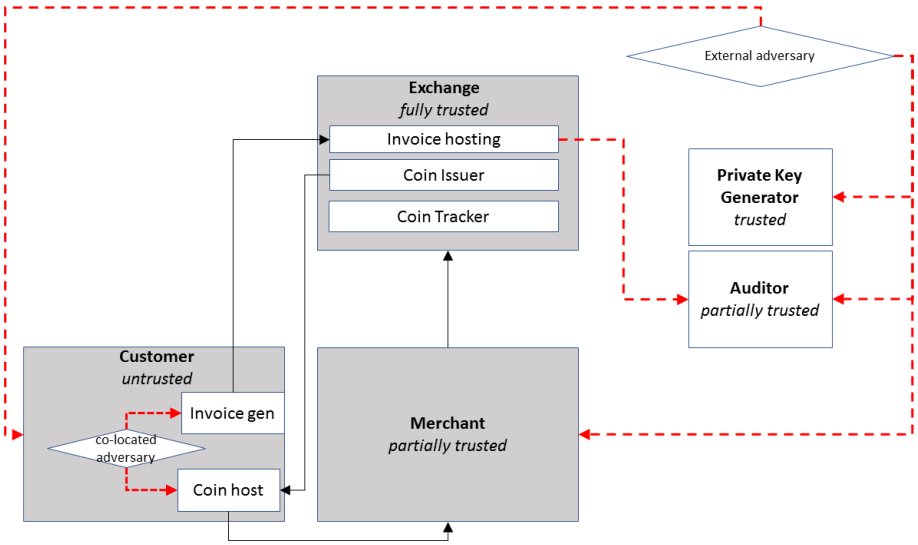

Figure 3: Use case #1.

- **Merchant's e-commerce store.** This party is a web application running in server/cloud environment whose main functions are to process payments, to receive encrypted digital coins, to check validity of coins (FE decryption), to request double-spending checks, and to request change/refunds for payments. This party may have a higher processing load and hardware acceleration may be needed. Also, it is sensitive point if it holds secret keys for digital coins and it may need to be deployed in a server/cloud environment contracted by a merchant (outsourcing decryption in FE is well known [72]) or in merchants own servers. Hence, it may also be subject to side-channel and fault attacks.
- **Auditor.** This party is a web application. It requests invoices in order to learn the data (via FE decryptions) required to perform audits. This party should run in a dedicated hardware or with a hardware trust anchor and, therefore, can be considered a partially trusted component.

In Figure 3, we show the adversaries and possible attacks that can take place in the system marked with red dashed arrows. To start with, we distinguish among external adversaries (e.g., auditors) and internal attacks that try to subvert the normal operation of the underlying payment system or cheat it. The latter is generally out of the scope of the FE security requirements but has an impact on those when the players of the FE scheme are also actors in the underlying payment system itself. Figure 3 shows all the components or parties and their corresponding trust level. Notice that the Auditor and the Private Key Generator (a.k.a Issuer of Trust in deliverable D7.1) is a new actor in the architecture, separate from those components of the payment system. Therefore, we consider security from the perspective of just the components belonging to the FE schemes with ABE that we employ in our protocol, namely KP-ABE and CP-ABE. The goal of our protocol is to perform audits of subsidized digital ticket restaurants or over the actual purchases with some sort of loyalty program that the digital currency will adhere to.

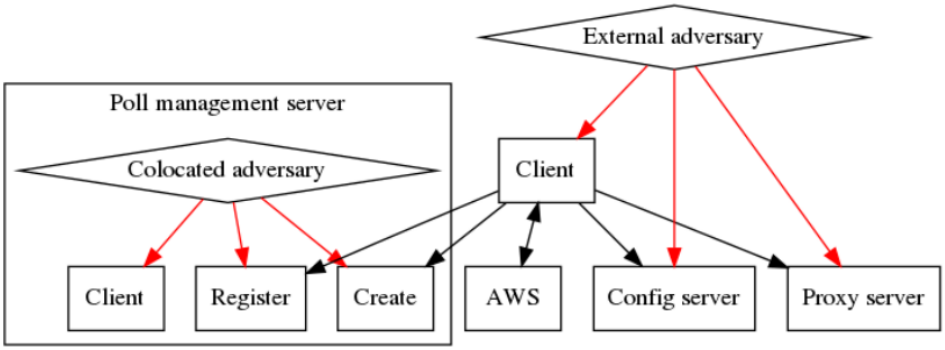

Figure 4: Use case #2.

4.2 Use case #2: Web analytics scenario

The web analytics use-case has a relatively simple architecture. The whole process is initiated and managed on a single server/cloud platform which hosts several different components. The client code only exists within the Awless executable code. The only other possible location involved may be the required proxy server which is used to enable many-to-many redistribution between clients. This may or may not also be hosted on the server/cloud platform.

Note that the intention is to ensure that all cryptography is performed on the clients but that to initialize the cryptography, keys may need to be created and distributed from the server/cloud platform. Certainly, the data path for the main FE data involves the clients encrypting their data and only encrypted data is ever transmitted, in fact to the proxy server where the encrypted computation takes place. Encrypted results are then returned to the clients for decryption along with the combined distributed keys.

The implementation may have some additional built-in security to limit access to the decrypted results to selected clients. This may be as simple as configuring the proxy server to only send encrypted results to authorized clients or may involve additional symmetric cryptography. This will be determined during development.

Recent work into side-channel attacks within cloud servers has shown that such attacks are possible [60]. There are also presumably other methods of implementing side-channel attacks upon cloud servers although countermeasures are available [97]. There is little work upon fault attacks associated with AWS systems despite considerable effort on other types of attack, for example DDoS attacks [9].

Figure 4 provides a very crude separation of potential malicious attackers into those co-located on cloud servers which have access to internal patterns of communication (from which, for example RSA keys can be deduced) and attacks external to the cloud which affect the Awless clients and also any other servers situated outside of the cloud (this may be either the proxy server and/or the configuration server).

Since the use case is using a distributed key method, there may be potential attacks which involve collusion between clients which should also be considered. Also, since the method has probably not been implemented in a live context before it should also be checked whether there are any vulnerabilities associated with the process of initiating the web analytics. For example, a configuration file for a particular web analytics session will be created. This will contain, among

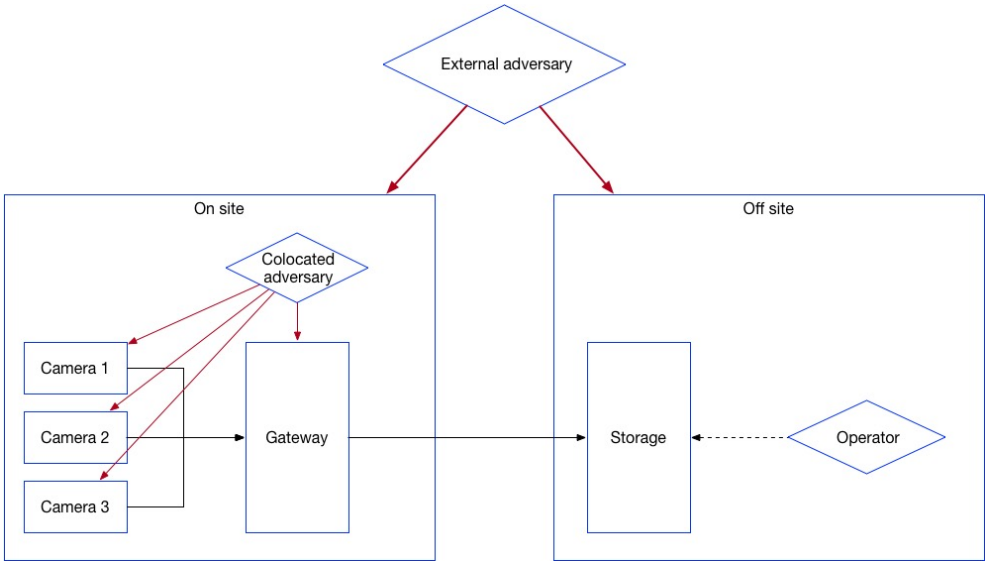

Figure 5: Use case #3.

other values, the address of the proxy server, definitions of which AWS endpoints to monitor and possibly some parameters for the cryptography. It will need to be checked that sensitive information is not leaked while downloading this information from the management platform to the clients.

Based on the above, the computing platforms in a practical deployment of this use case are most probably all falling into the server/cloud category. The computing platforms are all such that adversaries are unlikely to have physical access to them and, consequently, they can be considered as fully trusted.

4.3 Use case #3: IoT scenario

The IoT use case has three types of parties: (1) the IoT edge device (a camera recording live video), (2) an IoT gateway that collects video streams from one or more cameras connected to its local network and streams this to the backend; and (3) a backend that stores and processes this video stream. This setup is shown in Figure 5. It could be used for example for surveillance purposes or for detection of patterns. The whole point of this use case is to put intelligence into the Gateway so that certain properties of the video stream can be pre-processed, without leaking other properties. The first property studied in FENTEC was selected to be "is there movement?" which should be feasible with FE.

Security-critical information is the video content itself and keys used for encrypting/decrypting these streams. In addition, the algorithm that is used for doing the pre-processing on the gateway should be kept private.

Computationally, the video encryption and processing itself is the most demanding operation. Cameras are embedded devices that usually have dedicated hardware chips for video encoding. The IoT gateways we can consider to be "high end" machines, falling into the "server/cloud" category. In practice, high-end industrial gateways such as for example the HPE GL20 is used [91]. It is assumed that adversaries may have physical access to the gateway and the cameras and,

consequently, side-channel attacks and fault attacks are in-scope and protections needed. The back-end is a server operated by a trusted party in protected premises (e.g., not outsourced to the cloud). Adversaries are unlikely to have either unauthorized remote or physical access to the back-end and, consequently, it can be considered as fully trusted.

5 Platform selection

In this section we describe a set of computing platforms that have been identified as suitable targets to perform the different implementation tasks in WP5. The selection of such platforms has been done according to the requirements discussed in this deliverable. These include: (a) requirements for developing efficient FE schemes (efficiency), (b)) requirements to enable hardware trust anchors (security), and (c) requirements to ease security analyses (evaluation).

In the following we briefly describe the baseline platforms we plan to use in our implementations, while motivating our choices. Note however that this list is not exclusive, i.e. we naturally leave the door open to using other platforms based on similar architectures but featuring less resources or manufactured by different suppliers. This is necessary to develop solutions that cover a large spectrum of design choices, e.g. for devices constrained in memory, area, or power/energy.

- **Zynq UltraScale ZCU102 [94].** The core element in this platform is a Xilinx Zynq UltraScale+ multiprocessor system-on-chip. The main reasons for choosing this development platform are twofold. First, the system-on-chip features a powerful processor (64-bit quad-core ARM Cortex-A53) internally interfaced to FPGA logic (Xilinx-7 series). It is therefore a natural platform for quick prototyping of FE implementations based on hardware/software co-design, i.e. where computational bottlenecks can be efficiently handed over to custom-designed co-processors. Second, both the ARM processor as well as the FPGA logic are attached to independent DDR4 memory components. This characteristic is particularly interesting to enable efficient FE implementations. Recent works dealing with fast implementations of lattice-based schemes, e.g. [86], have shown that the main bottleneck in hardware co-processors are dominated by the memory accesses needed for multiplication of polynomials. The reason for this is that, in most platforms, the main processor handles the memory accesses of the co-processor. The Zynq UltraScale ZCU102 overcomes this issue by attaching an independent DDR4 Component (512 MB) to the programmable logic, and therefore has the potential of leading to highly-efficient implementations. We envision to use this platform mainly in T5.2, but also in T5.3 for the cases in which we want to develop our own trusted anchors in FPGA logic.
- **Hikey960 [2].** This development board features a Huawei Kirin 960 octa-core ARM processor (four ARM Cortex-A73 and four Cortex-A53 cores). The main reason for choosing this platform is the availability of open-source libraries to develop solutions based on ARM Trustzone Trusted Execution Environment (TEE). More specifically, the Hikey960 is one of few boards supported by the OP-TEE (Open Portable Trusted Execution Environment) framework [78] which provides the necessary tools to make up a complete TEE. We envision to use this platform in T5.3, particularly for implementations that require hardware-enforced isolation. In addition, the platform can also be used in T5.2 to develop efficient implementations optimized for ARM achitectures.
- **Sakura-X [87].** This platform features a modern Xilinx Kintex-7 FPGA device. The reason for choosing this platform is that it is specifically designed for testing the security of FPGA implementations against implementation attacks. It is therefore a perfect candidate to perform the research activities in both T5.4 and T5.5, which aim to develop and test cryptographic building blocks to be deployed in non-trusted platforms.

6 Conclusions

In this deliverable, we discussed security and trust models of FE schemes from the implementation point-of-view and identified several requirements that such FE implementations must meet in different cases. We described a generic computing platform and showed how it can be used for analyzing the requirements of FE implementations in the large variety of cases that are relevant in FENTEC. We reviewed existing works on FE and cryptographic primitives needed to compute them from hardware implementation point-of-view considering both hardware optimizations and mitigations against side-channel and fault attacks. We mapped the findings of the deliverable on the three use cases of FENTEC and identified critical components in them regarding both performance and implementation security. Finally, we provided requirements for development kits that are to be used in the implementation research of WP5 of FENTEC and recommended three specific development kits that fulfill all the different requiments: Zynq UltraScale ZKU102, Hikey960, and Sakura-X.

The contributions of this deliverable allow setting the focuses for the research work of Tasks 5.2–5.5 and serve as a valuable reference for implementation work of FENTEC. In particular, the results of this deliverable justify the three main scenarios that are studied in these tasks: (a) the performance focus in a setting where performance-critical parts of FE schemes need to be accelerated in a fully trusted environments, (b) the partitioning focus in a setting where security-critical parts of FE schemes can be delegated to HW trust anchors providing trust to these parts while other parts are executed in non-trusted environments, and (c) the implementation security focus in a setting where the whole computing platform is untrusted requiring strong countermeasures against adversaries having physical access to the computing device. The deliverable also pinpointed the relevant starting points for the research work by reviewing the relevant literature of FE schemes from hardware implementation point-of-view as well as by mapping the security and trust models to the use cases and identifying the possible performance bottlenecks of the use cases.

References

[1] https://github.com/fentec-project/fe-ml-example. (Page 17)

[2] 96boards. Hikey 960 development platform. https://www.96boards.org/product/hikey960/. (Page 25)

[3] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015. (Pages 15 and 16)

[4] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 597–627. Springer, 2018. (Pages 12, 14, 15, and 16)

[5] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 601–626. Springer, 2017. (Page 15)

[6] Jithra Adikari, M. Anwar Hasan, and Christophe Nègre. Towards faster and greener cryptoprocessor for eta pairing on supersingular elliptic curve over $\mathbb{F}_{2^{1223}}$. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography - SAC 2012*, volume 7707 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2012. (Page 16)

[7] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, volume 9816 of *Lecture Notes in Computer Science*, pages 333–362. Springer, 2016. (Pages 14, 15, and 16)

[8] A. Akhunzada, M. Sookhak, N. B. Anuar, A. Gani, E. Ahmed, M. Shiraz, S. Furnell, A. Hayat, and M. K. Khan. Man-at-the-end attacks. *Journal of Network and Computer Applications*, 48(48):44–57, 2015. (Page 20)

[9] Amazon. Denial of service attack mitigation on AWS. https://aws.amazon.com/answers/networking/aws-ddos-attack-mitigation/. (Page 22)

[10] Amazon. Amazon EC2 F1 instances: Run customizable FPGAs in the AWS cloud. https://aws.amazon.com/ec2/instance-types/f1/, 2018. (Page 9)

[11] Domenico Argenziano. Implementation of an FFT hardware accelerator for security applications. In Fatos Xhafa, Leonard Barolli, Fabrizio Messina, and Marek R. Ogiela, editors, *Parallel, Grid, Cloud and Internet Computing - 3PGCIC 2015*, pages 256–259. IEEE Computer Society, 2015. (Page 16)

[12] ARM. ARM architecture reference manual: ARMv7-A and ARMv7-R edition. Reference Manual, 2014. (Page 9)

[13] ARM. ARMv7-M architecture reference manual. Reference Manual, 2014. (Page 9)

[14] ARM. ARM architecture reference manual: ARMv8, for ARMv8-A architecture profile. Reference Manual, 2017. (Page 9)

[15] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs. In Luca Breveglieri, Sylvain Guilley, Israel Koren, David Naccache, and Junko Takahashi, editors, *Fault Diagnosis and Tolerance in Cryptography - FDTC 2011*, pages 105–114. IEEE Computer Society, 2011. (Page 10)

[16] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006. (Page 10)

[17] Florent Bernard. Scalable hardware implementing high-radix Montgomery multiplication algorithm. *Journal of Systems Architecture*, 53(2-3):117–126, 2007. (Page 15)

[18] Daniel J. Bernstein. Cache-timing attacks on AES. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf, April 2005. (Pages 10 and 17)

[19] Jean-Luc Beuchat, Nicolas Brisebarre, Jérémie Detrey, Eiji Okamoto, Masaaki Shirase, and Tsuyoshi Takagi. Algorithms and arithmetic operators for computing the η_T pairing in characteristic three. *IEEE Trans. Computers*, 57(11):1454–1468, 2008. (Page 16)

[20] Jean-Luc Beuchat, Jérémie Detrey, Nicolas Estibals, Eiji Okamoto, and Francisco Rodríguez-Henríquez. Fast architectures for the η_T pairing over small-characteristic supersingular elliptic curves. *IEEE Trans. Computers*, 60(2):266–281, 2011. (Page 16)

[21] Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential fault attacks on elliptic curve cryptosystems. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2000. (Pages 10 and 18)

[22] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. (Page 10)

[23] Nina Bindel, Johannes A. Buchmann, and Juliane Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. In *Fault Diagnosis and Tolerance in Cryptography - FDTC 2016*, pages 63–77. IEEE Computer Society, 2016. (Page 19)

[24] Johannes Blömer, Ricardo Gomes Da Silva, Peter Günther, Juliane Krämer, and Jean-Pierre Seifert. A practical second-order fault attack against a real-world pairing implementation. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2014)*, pages 123–136. IEEE, 2014. (Page 18)

[25] Johannes Blömer, Peter Günther, and Gennadij Liske. Improved side channel attacks on pairing based cryptography. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design — COSADE 2013*, volume 7864 of *Lecture Notes in Computer Science*, pages 154–168. Springer, 2013. (Page 18)

[26] Johannes Blömer, Peter Günther, and Gennadij Liske. Tampering attacks in pairing-based cryptography. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2014)*, pages 1–7. IEEE, 2014. (Pages 18 and 19)

[27] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. A new CRT-RSA algorithm secure against bellcore attacks. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *Conference on Computer and Communications Security, CCS 2003*, pages 311–320. ACM, 2003. (Page 18)

[28] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997. (Page 10)

[29] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. (Page 15)

[30] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001. (Page 15)

[31] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011. (Page 12)

[32] Arnaud Boscher, Robert Naciri, and Emmanuel Prouff. CRT RSA algorithm protected against fault attacks. In Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices -WISTP 2007*, volume 4462 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2007. (Page 18)

[33] Riadh Brinci, Walid Khmiri, Mefteh Mbarek, Abdellatif Ben Rabaa, and Ammar Bouallègue. Efficient hardware design for computing pairings using few FPGA in-built DSPs. *IACR Cryptology ePrint Archive*, 2015:116, 2015. (Page 16)

[34] Leon Groot Bruinderink and Peter Pessl. Differential fault attacks on deterministic lattice signatures. *IACR Cryptology ePrint Archive*, 2018:355, 2018. (Page 19)

[35] Sanjit Chatterjee, Koray Karabina, and Alfred Menezes. Fault attacks on pairing-based protocols revisited. *IEEE Transactions on Computers*, 64(6):1707–1714, 2015. (Page 18)

[36] Ray C. C. Cheung, Sylvain Duquesne, Junfeng Fan, Nicolas Guillermin, Ingrid Verbauwhede, and Gavin Xiaoxu Yao. FPGA implementation of pairings using residue number system and lazy reduction. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 421–441. Springer, 2011. (Page 16)

[37] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. *IACR Cryptology ePrint Archive*, 2017:989, 2017. (Page 12)

[38] Mathieu Ciet and Marc Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Des. Codes Cryptography*, 36(1):33–43, 2005. (Page 18)

[39] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security - ICICS 2010*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2010. (Page 18)

[40] Christophe Clavier and Marc Joye. Universal exponentiation algorithm. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 300–308. Springer, 2001. (Page 18)

[41] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999. (Page 18)

[42] Common Criteria. Common criteria for information technology security evaluation, part i: Introduction and general model, CC v3.1. CCMB-2017-04-001, 2017. (Page 10)

[43] Alan Daly and William P. Marnane. Efficient architectures for implementing Montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic. In Martine D. F. Schlag and Steve Trimberger, editors, *International Symposium on Field Programmable Gate Arrays - FPGA 2002*, pages 40–49. ACM, 2002. (Page 15)

[44] Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient software implementation of ring-LWE encryption. In Wolfgang Nebel and David Atienza, editors, *Design, Automation & Test in Europe- DATE 2015*, pages 339–344. ACM, 2015. (Page 16)

[45] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013. (Page 16)

[46] EMVCo. Certification Processes. https://www.emvco.com/processes-forms/certification/, 2018. (Page 10)

[47] Nicolas Estibals. Compact hardware for computing the tate pairing over 128-bit-security supersingular curves. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing-Based Cryptography - Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 397–416. Springer, 2010. (Page 16)

[48] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede. State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 76–87, 2010. (Page 10)

[49] Junfeng Fan and Ingrid Verbauwhede. An updated survey on secure ECC implementations: Attacks, countermeasures and cost. In David Naccache, editor, *Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, volume 6805 of *Lecture Notes in Computer Science*, pages 265–282. Springer, 2012. (Page 18)

[50] Guillaume Fumaroli and David Vigilant. Blinded fault resistant exponentiation. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography - FDTC 2006*, volume 4236 of *Lecture Notes in Computer Science*, pages 62–70. Springer, 2006. (Page 18)

[51] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001. (Page 10)

[52] Santosh Ghosh, Dipanwita Roy Chowdhury, and Abhijit Das. High speed cryptoprocessor for η_T pairing on 128-bit secure supersingular elliptic curves over characteristic two fields. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 442–458. Springer, 2011. (Page 16)

[53] Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. High speed flexible pairing cryptoprocessor on FPGA platform. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *Pairing-Based Cryptography - Pairing 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2010. (Page 16)

[54] Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. Fault attack and countermeasures on pairing based cryptography. *International Journal of Network Security*, 12(1):26–33, 2011. (Pages 18 and 19)

[55] Christophe Giraud. An RSA implementation resistant to fault attacks and to simple power analysis. *IEEE Trans. Computers*, 55(9):1116–1120, 2006. (Page 18)

[56] D. R. E. Gnad, F. Oboril, and M. B. Tahoori. Voltage drop-based fault attacks on FPGAs using valid bitstreams. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7, September 2017. (Page 10)

[57] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014. (Page 12)

[58] Norman Göttert, Thomas Feller, Michael Schneider, Johannes A. Buchmann, and Sorin A. Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 512–529. Springer, 2012. (Page 16)

[59] Johann Großschädl, Elisabeth Oswald, Dan Page, and Michael Tunstall. Side-channel analysis of cryptographic software via early-terminating multiplications. In Dong Hoon Lee and Seokhie Hong, editors, *Information, Security and Cryptology - ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2009. (Page 17)

[60] Mehmet Sinan Inci, Berk Gülmezoglu, Gorka Irazoqui Apecechea, Thomas Eisenbarth, and Berk Sunar. Seriously, get off my cloud! cross-vm RSA key recovery in a public cloud. *IACR Cryptology ePrint Archive*, 2015:898, 2015. (Page 22)

[61] Intel. Intel user-customizable SoC FPGAs. Product Brochure, 2017. (Page 9)

[62] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptology*, 17(4):263–276, 2004. (Page 15)

[63] Marc Joye and Sung-Ming Yen. The Montgomery powering ladder. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002. (Page 18)

[64] Tae Hyun Kim, Tsuyoshi Takagi, Dong-Guk Han, Ho Won Kim, and Jongin Lim. Side channel attacks and countermeasures on pairing based cryptosystems over binary fields. In David Pointcheval, Yi Mu, and Kefei Chen, editors, *Cryptology and Network Security (CANS 2006)*, volume 4301 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2006. (Pages 10 and 18)

[65] Donald Knuth and Andrew Yao. *Algorithms and Complexity: New Directions and Recent Results*, chapter The complexity of nonuniform random number generation. Academic Press, 1976. (Page 16)

[66] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. (Pages 10 and 18)

[67] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. (Pages 10 and 18)

[68] Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *J. Cryptographic Engineering*, 1(1):5–27, 2011. (Page 10)

[69] Jonas Krautter, Dennis Gnad, and Mehdi Tahoori. FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):44–68, August 2018. (Page 10)

[70] Daesung Lim, Nam Su Chang, Sung Yeon Ji, Chang Han Kim, Sangjin Lee, and Young-Ho Park. An efficient signed digit Montgomery multiplication for RSA. *Journal of Systems Architecture - Embedded Systems Design*, 55(7-9):355–362, 2009. (Page 15)

[71] Wen-Ching Lin, Jheng-Hao Ye, and Ming-Der Shieh. Scalable Montgomery modular multiplication architecture with low-latency and low-memory bandwidth requirement. *IEEE Trans. Computers*, 63(2):475–483, 2014. (Page 15)

[72] Zechao Liu, Zoe L. Jiang, Xuan Wang, and S.M. Yiu. Practical attribute-based encryption: Outsourcing decryption, attribute revocation and policy updating. *Journal of Network and Computer Applications*, 108:112–123, April 2018. (Page 21)

[73] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag, Berlin, Heidelberg, 2007. (Page 17)

[74] Microchop. AVR Microcontrollers. http://www.microchip.com/design-centers/8-bit/avr-mcus, 2018. (Page 9)

[75] Peter L. Montgomery. Modular multiplication without trial division. *Math. Comp.*, 44(170):519–521, 1985. (Page 15)

[76] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical CCA2-secure and masked ring-LWE implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):142–174, 2018. (Page 18)

[77] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. https://eprint.iacr.org/2010/556. (Page 12)

[78] OP-TEE. Open portable trusted execution environment. https://www.op-tee.org/. (Page 25)

[79] Dan Page and Frederik Vercauteren. A fault attack on pairing-based cryptography. *IEEE Transactions on Computers*, 55(9):1075–1080, 2006. (Pages 10, 18, and 19)

[80] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003. (Page 10)

[81] Jean-Jacques Quisquater and David Samyde. Eddy current for magnetic analysis with active sensor. In *Esmart 2002*, 2002. (Page 10)

[82] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Theory of Computing - STOC 2005*, pages 84–93. ACM, 2005. (Page 16)

[83] Oscar Reparaz, Josep Balasch, and Ingrid Verbauwhede. Dude, is my code constant time? In David Atienza and Giorgio Di Natale, editors, *Design, Automation & Test in Europe Conference & Exhibition - DATE 2017*, pages 1697–1702. IEEE, 2017. (Page 17)

[84] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptographic Engineering*, 6(2):139–153, 2016. (Page 18)

[85] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-LWE implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015. (Page 18)

[86] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact ring-LWE cryptoprocessor. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 371–391. Springer, 2014. (Pages 16 and 25)

[87] Sakura. Hardware evaluation boards. http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html. (Page 25)

[88] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on FPGAs. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2018*, 2018. (Page 10)

[89] Michael Scott. Computing the Tate pairing. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005. (Page 18)

[90] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2002. (Page 10)

[91] Edgeline Systems. HPE GL20 IoT gateway. https://www.hpe.com/us/en/product-catalog/servers/edgeline-systems/pip.hpe-edgeline-el20-intelligent-gateway.1008670391.html. (Page 23)

[92] Satyanarayana Vollala, Krishnan Geetha, and Natarajan Ramasubramanian. Efficient modular exponential algorithms compatible with hardware implementation of public-key cryptography. *Security and Communication Networks*, 9(16):3105–3115, 2016. (Page 15)

[93] Claire Whelan and Mike Scott. Side channel analysis of practical pairing implementations: Which path is more secure? In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2006. (Page 18)

[94] Xilinx. Zynq UltraScale+ MPSoC ZCU102 evaluation kit. https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html. (Page 25)

[95] Xilinx. Zynq-7000 all programmable SoC. Product Brief, https://www.xilinx.com/support/documentation/product-briefs/zynq-7000-product-brief.pdf, 2018. (Page 9)

[96] Sung-Ming Yen, Dongryeol Kim, and Sang-Jae Moon. Cryptanalysis of two protocols for RSA with CRT based on fault infection. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography - FDTC 2006*, volume 4236 of *Lecture Notes in Computer Science*, pages 53–61. Springer, 2006. (Page 18)

[97] Tianwei Zhang, Yinqian Zhang, and Ruby B. Lee. Cloudradar: A real-time side-channel attack detection system in clouds. In Fabian Monrose, Marc Dacier, Gregory Blanc, and Joaquín García-Alfaro, editors, *Research in Attacks, Intrusions, and Defenses - RAID 2016*, volume 9854 of *Lecture Notes in Computer Science*, pages 118–140. Springer, 2016. (Page 22)

[98] M. Zhao and G. E. Suh. FPGA-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (S&P)*, pages 229–244, May 2018. (Page 10)