Slovak University of Technology
Faculty of Electrical Engineering and
Information Technology
Institute of Computer Science and Mathematics

Reg. No.: FEI-12307-29017

Improving CPA of (EC)DSA and Timing Analysis of McEliecePKC

Dissertation thesis

2015 **Ing. Marek Repka**

Slovak University of Technology
Faculty of Electrical Engineering and
Information Technology
Institute of Computer Science and Mathematics

Reg. No.: FEI-12307-29017

Improving CPA of (EC)DSA and Timing Analysis of McEliecePKC

Dissertation thesis

Study programme:	Applied Informatics
Study field number:	2511
Study field:	9.2.9 Applied Informatics
Supervisor:	prof. RNDr. Otokar Grošek, PhD.

Bratislava 2015 **Ing. Marek Repka**

Slovak University of Technology in Bratislava
Institute of Computer Science and Mathematics

Faculty of Electrical Engineering and Information Technology
2015/2016

DISSERTATION THESIS ASSIGNMENT

Author:	Ing. Marek Repka
Study programme:	Applied Informatics
Study field:	9.2.9. Applied Informatics
Reg. No.:	FEI-12307-29017
Student ID:	29017
Supervisor:	prof. RNDr. Otokar Grošek, PhD.

Thesis title: **Improving CPA of (EC)DSA and Timing Analysis of McEliecePKC**

Assignment: Sensitive information can leak through Side Channels of cryptographic products. This sensitive information can be exploited to radically shrink the set of possible keys of a cryptosystem. Attacks misusing this leaked information are called Side Channel Attacks (SCA).

Main tasks:
1. Make state-of-the-art SCA techniques.
2. Analyze an SCA attack.
3. Propose a countermeasure to the attack.

Literature:

1. Kocher, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In CRYPTO '96: Proceedings, pages 104-113, London, UK. Springer-Verlag.
2. Mangard S., Oswald E., Popp T. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Graz University of Technology, Graz Austria. Springer 2007.
3. Rohatgi P. Chapter 14, Improved Techniques for Side-Channel Analysis. Scienc+Business Media 2009.
4. Schindler W., Lemke K., and Paar C. A Stochastic Model for Differential Side Channel Cryptanalysis. CHES 2005, LNCS 3659, pp. 30-46, 2005.

Assignment date: 29. 09. 2012
Delivery date: 31. 08. 2016

Ing. Marek Repka
author

prof. RNDr. Otokar Grošek, PhD.
head of department

prof. RNDr. Otokar Grošek, PhD.
garant of study programme

Súhrn

Postranné kanály sú tu tak dlho ako kryptografia, no formálne prvá práca zaoberajúca sa postrannými kanálmi bola publikovaná v roku 1996 [36]. Pokrok za posledné desaťročie v tejto oblasti je nesmierny. Niet divu, veď útoky postrannými kanálmi sú jedinou možnosť ako zlomiť dnešnú i budúcu kryptografiu. Táto práca sa zaoberá útokom korelačnou analýzou spotreby na DSA a ECDSA, a implementáciou pôvodného originálneho McEliece kryptosystému s verejným kľúčom (McEliece PKC) obsahujúc nástroj na meranie úniku informácie postranným kanálom. V práci je CPA útok na DSA a ECDSA vylepšený a je tu tiež diskutované účinné a efektívne protiopatrenie voči tomuto útoku. V práci je ďalej prezentovaná implementácia originálneho McEliece PKC s nástrojom na meranie zvolených typov únikov. Možnosti tohto nástroja sú v práci aj demonštrované, ako aj analýza doby výpočtu v spojení s vnášaním chýb, ktorá je tiež vylepšená. Návrh efektívnej metódy pre výpočet p-tej odmocniny v konečných poliach charakteristiky p je uvedený ako posledný výsledok. Práca uvádza súčasný stav problematiky, metodológiu, ktorá mapuje analyzované kryptografické implementácie, meracie nástroje, a metódy útokov, ktoré boli použité na dosiahnutie popisovaných výsledkov. V závere je práca zhrnutá.

Abstract

Side-channel attacks are here as long as cryptography exits. However, formally, the first work was published in 1996 [36]. Tremendous progress has been made in this field in past decade. Indeed, side-channel attacks are the only way how to break today cryptography, and moreover, they are everywhere. This work deals with correlation power analysis of ECDSA and implementation of McEliece Public Key Cryptosystem (McEliece PKC) with embedded leakage measurement tool. The CPA attack against ECDSA has been improved, and an effective and efficient countermeasure is discussed. The original McEliece PKC has been implemented together with the embedded leakage measurement tool. The measurement tool is demonstrated as well as timing fault injection analysis which is also improved. The proposal for efficient computation of p-th root in extended finite fields of characteristic p is mentioned as the last result. State-of-the-art is provided in the first part of the work. Then the methodology maps analyzed cryptographic implementations, used measurement devices and attack methods, which were used to achieve desired results. Finally, the work is concluded.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction & Motivation

Side-channel attacks play role in wars and espionages as long as cryptography exits. Hackers and crackers use side-channel attacks naturally. However, formally, the first work was published in 1996 [36]. Vast progress has been made in this field in past decade. Today cryptography is very robust against linear, differential, and algebraic cryptanalysis. Nevertheless, side-channel attacks still remain big threat. Cryptography resistant to quantum-cryptanalysis is vulnerable as well since it must also be implemented to real devices. Essentially, it is not problem only of the implementation itself. This problem is also at the levels of the implementation description and the cryptographic algorithm design.

Indeed, side-channel attacks are the only way how to break today strong cryptography. A cryptosystem may have got so big vulnerability that its secrets can be revealed using information about computation time or electromagnetic emanation, or by achieving an invalid state, or by fetching in an invalid input. Therefore, it is very important to investigate possibilities of side-channel attacks and how to be effectively protected against them. Designing and implementing an effective, moreover efficient, countermeasure is not trivial because by implementing a countermeasure another leakage can be produced, and one type of countermeasure does not cover all leakages. Furthermore, there are 2nd and 3th order attacks against protected implementations.

State-of-the-art with scale and steps of side-channel attacks are provided in the first part of this work. Discussion trough attacks, which destruct or do not even touch the device, or which observe only or act with the computation, is provided. Data or operation dependent behavior of the cryptographic product is exploited in order to reveal the secret. Countermeasures, masking and hiding of the exploitable behavior, are discussed. Methodology maps analyzed cryptographic implementations, devices, and attack methods, which were used to achieve described results.

The CPA attack against integer multiplier with one constant secret operand was improved. This achievement was demonstrated on the 16-bit integer multiplier in FPGA

in the work (Repka, Varchola, and Drutarovsky [58]). Thanks to this improvement, the CPA attack can be simulated and more blocks of key can be guessed.

Despite the original McEliece PKC [42], is post-quantum cryptosystem, it is not an exception as regards side-channel attacks. The original McEliece PKC providing test vectors and embedding leakage measurement tool was implemented, and the implementation was described in the work (Repka [52]). Using the embedded tool the implementation can be used to analyze various leakages. The tool is presented on several examples this work.

Chapter 2

State of the Art

2.1 Side Channel Kinds

There exist several kinds of side-channels which can be exploited by an adversary. The most common side-channels nowadays are electromagnetic emanation, power consumption, computation time, light (photon emission), sound, residual data, and also nonstandard behavior of a device (fault injections, for instance, to achieve buffer overflow or another invalid state, or flipping bits of an intermediate result in memory using laser beam, or changing clock frequency) or nonstandard behavior of an algorithm (like fetching invalid inputs to achieve buffer overflow or also errors in output of the algorithm independently of the device). The electromagnetic emanation is very common and dangerous side-channel because the electromagnetic emanation can propagate across walls to far distances. For instance, it is possible to reveal which keys were pressed when a password was being typed in. Fault injections attacks are very powerful because they can involve very clear and very sensitive information, but, in many cases, especially in the case

Table 2.1: List of the most common side-channels

Side-channel	**Reference**
Power consumption	[35], [9]
Electromagnetic emanation	[47], [14]
Software defined radio	[23]
Computation time	[36], [11]
Sound	[22]
Photon emission	[30]
Nonstandard behavior	[4],[80]
Residual data	[26]
Social engineering	[25]

of injecting the physical errors, a close access to the sensitive parts of the device and very good knowledge of the algorithm implementation is needed. Even timing attacks are very dangerous because they can be performed remotely without letting any markable footmarks. The most common side-channels are listed in Tab. 2.1.

2.2 Scale of Side Channel Attacks

Side-channel attacks can be scaled regarding the manipulation by computation, regarding the destruction of the device, and regarding the analysis performed.

2.2.1 Scale regarding the manipulation by the computation

Regarding the manipulation by the computation, side-channels attacks are divided to active and passive attacks.

Passive attacks do not interfere with the computation process, they just trace or observer the standard physical behavior. As an example of passive attacks, there is a timing attack focused on CACHE of CPU [8].

Active attacks affect the computation process in order to cause an error (fault injection attacks). This error can be introduced in to the computation via incorrect or invalid input or also by changing a physical property (temperature, voltage, clocking, capacitance) of the device. The supposition for the physical fault injections is that the adversary has access to the device and he/she knows details about the implementation and the technical realization depending on the introducing type of errors. The device independent algorithm based fault injections can be performed remotely. However, in order to know what invalid inputs can provide a sensitive information, cryptanalysis of the algorithm must be made. In the paper [19], there is an example of this kind of attacks entering invalid elliptic curve points, and, in the work [70], many methods of error injections via changing physical properties are listed. For instance, one can use a laser beam or electromagnetic discharger to flip a bit in memory, or one can use a microprobe to manipulate signal in buses. The work [4] explore efficient and practical methods for fault injections.

When attacking a true random generator, the adversary intends to bias statistic of the generator. This can be achieved, for an example, by activity of electromagnetic field, or by changing voltage or clocking [41], [10].

2.2.2 Scale regarding the destruction of the device

The scale regarding the destruction of the device is divided to noninvasive (the device is not destructed at all), semi-invasive (the device is damaged, but it is still bale to operate), and invasive (the device is destructed, it is not able to perform its function at

all). The subjects of the tampering are commonly ASIC, FPGA, μController, CPU, and their embedded flues, memories, buses, multiplexers, and registers.

As an example for the noninvasive attacks, there is a timing analysis of CACHE. In the papers [82], [8], there is CACHE timing model described. This model has been validate on different implementations of AES. Also electromagnetic emanation analysis attacks can belong to noninvasive attacks. Computation time analysis of Diffie-Hellman, RSA, DSS can be found in [36]. The most dangerous remote attack is based on electromagnetic emanation side channels [47], especially in the case of smart cards (ECDSA [32]).

The work [70] deals with invasive and particularly by semi-invasive side-channel attacks. In this type of attacks, the cover of chip is removed usually by an acid, scalpel, or a laser cutter. Afterwards, buses, or embedded memories can be analyzed. There are also backtracking imagining techniques which use a light of different wavelength (UV, X-Ray, IR, sono, laser). Also power consumption side-channel, like correlation power analysis ([9], [38]) or many fault injection attacks, like differential fault injection analysis ([61]) are very often semi-invasive.

2.2.3 Scale regarding the statistical analysis

The statistical analysis play a big role in side-channel attacks. The statistical analysis is used to distinguish between possible key hypotheses, and thus reduce the size of the set of possible keys to a size that can be searched through in a feasible time. Note that also non-probabilistic analyses have been tried to use, such as, for instance, SVM [27].

2.2.3.1 Simple analysis

In some cases the cryptosystem's physical behavior leaks so much that the secret can be guessed using only few signal records (traces). This case happens often for operational dependences, for example, if the operation and data flow is driven by values of bits of the secret. Conditional branching and loops provide the sensitive leakage. Work [19] contains some examples on simple analyses. Also the cold boot attack [26] is instance of the simple analysis.

2.2.3.2 Univariate and multivariate analysis without templates

In this case the adversary analyses big number of signal traces. If he/she is guessing one variable, then it is univariate analysis, and if he/she guesses two or more variables, it is multivariate analysis. Special instances of multivariate analyses are High-Order analyses [44] which are used against cryptosystems protected by masking. As example [40], there are analyses like difference of means, distance of means, various correlation coefficients, conditional entropies [6], and also another non-probabilistic methods [27].

2.2.3.3 Univariate and multivariate analysis with templates

If the adversary can moreover make statistical profile of the device performance, we are talking about profiling or template analyses. These attacks are very powerful attacks [43]. Such attacks use more sophisticated description of the sensitive leakage [71], like stochastic methods [64], multivariate Gaussian distribution [62], multivariate regression, and conditional entropy (Mutual Information Analysis – MIA [6]). These attacks, however, need to have access to the same device (or another instance of the device) before the attacks are performed, in order to make the statistical profile of the leakage (the templates).

2.3 Steps of Side Channel Attacks

Generic steps of side-channel attacks are expressed in this section. There are four main steps, namely side-channel analysis, leakage modeling, measurements, and distinguishing respectively.

2.3.1 Side-channel analysis

The first step in side-channel attacks is to find the exploitable vulnerability of the cryptosystem. In this step, an adversary collect the most information possible to get, like information about the cryptographic algorithm, its implementation, the platform it is implemented on, and also about the environment. This information is then analyzed in order to find weak points which can be misused. Very often it is enough to know the architecture of the cryptographic algorithm and the architecture of the cryptographic device. From the knowledge of the architecture, the adversary can find registers, which register a sensitive intermediate result, whose power consumption or electromagnetic emanation contain a sensitive leakage, or find multiplexers which root a sensitive intermediate result which can be after fault injection to the multiplexers routed to the output, or possible vulnerabilities for buffer overflow.

The output of this side-channel analysis are some knowledges about what side-channels provide signal which contains a sensitive leakage, which method can be used to measure the signals, whether there are methods to effectively model the leakage or exploit the vulnerabilities, what statistical methods can be used for distinguished between hypotheses.

2.3.2 Leakage modeling

When the leakage point is known. The adversary looks for possibilities how to model the leakage, and how to reveal the most clear information of the leakage. The model is

then used in distinguishing between possible hypotheses about the secret the adversary wants to reveal.

There are two different approaches to model the leakage signal. The adversary must make the model of the signal assuming all the possible data that can be processed or all the possible operations that can be performed. The adversary makes the model for data processing performance if the adversary desires to exploit data dependency, while the adversary makes a mode for operation performance if the adversary wants to exploit operational dependency.

The first approach is used in the case the adversary has access to the device and the adversary can control the device so he/she can chose inputs, keys, and output of the cryptographic device. In this case the adversary will construct statistical profile, or templates, for all the possible data that can be processed or all the possible operations regarding what he/she wants to exploit. These are template attacks, and they are very powerful because the model is very precise. Essentially, it neglect nothing.

The second approach is to use a top level common models like Hamming Weight or Hamming Distance Power Models (HWMP, HDPM). When HWPM is used, the power consumption is assumed to be proportional to the number of bits set to logic '1' of the processed sensitive variable. However, the HWPM is not sufficiently accurate because, in reality, the power consumption depends rather on the occurrence of bit transitions. Therefore, an adversary will probably use the HWPM only if one of the two consecutive states of the sensitive variable is not known. Since the power consumption depends mostly on occurrence of transitions at the output of the logical gates, the HDPM is more accurate.

Table 2.2: Dynamic Power Consumption for Hamming Weight and Hamming Distance Power Models which are commonly used

	$HD = 0$	$HD = 1$
$HW = 0$	$0 \rightarrow 0$	$1 \rightarrow 0$
$HW = 1$	$1 \rightarrow 1$	$\mathbf{0 \rightarrow 1}$
Dynamic Power Consumption:	No	Yes

Such models are less precise, because they neglect many factors, like glitches, length of buses and wires and another kind of parasitic capacitance, combination logic around, and other noise. The more fitting the model is, the better success rate and errors achieves the attack. Therefore, if the adversary would have got the hardware description sources or configuration files, EDA tools can be used for modeling at a lower level with much better fitting. The more the adversary knows about the implementation, the more fitting model the adversary can produce. In the following text below, we discuss the most common power models used by adversaries.

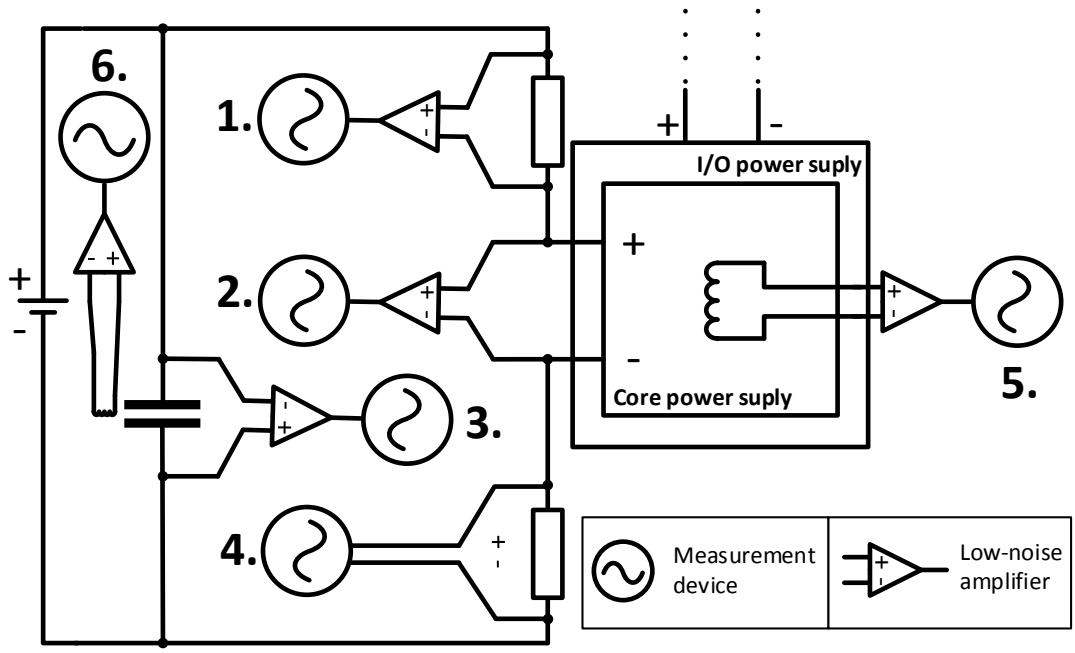

Figure 2.1: Various measurement points for power analyses.

Some works claim that $1 \rightarrow 0$ transition can be neglected, since the corresponding dynamic power consumption is less significant than in the case of $0 \rightarrow 1$. Sometimes the power model is made so those transitions are weighted. But if the power model is weight than the attack is going to be considered as a template attack, so maybe rather than weight the power model, make statistical templates of signal while prepossessing know data. Power consumption can be inversely proportional to the HW and HD in many cases, such as in our case of CMOS devices.

Outcome of this step are model or templates for data, which are used to distinguishing between possible hypotheses about what data was processed in reality.

2.3.3 Measurements

The next task in a side-channel attack is to measure the signal that contains the sensitive information. This work is focused on power analysis, thus we are talking about various measurement points for power consumption measurements. The most common methods are depicted in Fig 2.1.

It holds that the closer to the leakage source the measurements are, the more qualitative information is recorded. When measuring the signal containing the sensitive leakage, it is very crucial to measure the leakage as clear as possible. The quality of the information involved in the measured signal is expressed by signal to noise ratio

(SNR). More about SNR is in [40].

What makes an attack semi-invasive is the method of the measurements. If the electromagnetic emanation can not be used or the signal is not clear enough, the adversary must get closer to the source. That means that he/she must open the device. It is possible if the device is not protected by detection of opening and manipulation, like filing and signeting of screw holes, flaws, or capacitance and temperature or electromagnetic sensitivity. However, if the electromagnetic emanation is still not possible to capture even afterwards the device was opened, for instance, because of the FPGA shielding, and also when the capacitors (Items 3 and 6 in Fig. 2.1) are shielded, then the adversary still can decapsulate the cover and use laser cutter to remove the shield, or flaws, or cut the power supply of the core and connect it to its own spatial circuit to power supply the core. The special circuit for power supply is designed by the adversary for the special purpose of measuring the power consumption or electromagnetic emanation of the core. It contains filters and another magic to measure as clear leakage as possible. Especially, charging of the certain parts of the circuit of the core is interesting and brings the most clear information. For instance, when sensitive registers change their state from 0 to 1 on their outputs, the interconnected part of the circuit, for which this change applies, must be charged. Because the charging must be very fast, capacitors are used to charge the circuit in the time. Indeed, measurement point 3 and 6 in Fig. 2.1 should are the most information bringing measurement point of the all measurement points depicted. And that is true.

Output of this step are records of the signal that contain the leakage. The records should be processed to make them ready for distinguishing. Here comes traces aligning and noise removing. Then a post-processing method is used to compress traces. The goal is to have as low number of final records as is possible and to gain the most clear leakage it is possible. Te trace aligning is the most critical. It must be known exactly where the sensitive operation is performed or data are processed, and according this point all traces must be aligned. Standardly, trigger signal is used, but the trigger signal must be also implemented what means the attack active. Further for traces aligning, detection of certain operations which can slave as a start point, and various signal processing methods, like Phase-Only Correlation [31] can be used. The most usual methods for compressing are averaging or computing median on intervals of traces. From compression of the traces also Principal Component, Cluster, and Discriminant analyses can be used. Next usual option is to use SNR to identify which parts of traces can be neglected.

2.3.4 Distinguishing

The final step in side-channel attacks is to distinguish between possible hypotheses about the secret. The hypotheses were made in the step leakage modeling. The distinguishing is conducted comparing the measured data with the leakage models or

templates. The comparison is realized using the chosen analysis (simple, without templates, with templates). When an attack without templates is performed, the leakage model is not very fitting, therefore also the comparison will not provide perfect answer. Hence, we rather talk about ordering the hypotheses according the results of the comparison. The most fitting hypothesis to the measured signal will have the first position, and the worst fitting hypothesis will have the last position. Indeed, the most fitting hypothesis is the most portable, and the adversary will try the hypothesis as the first. Therefore, we can talk about a complexity of the attack.

The complexity is computed as the number of remaining hypotheses taken to the account after the side-channel attack distinguishing step. The hypotheses that are taken to the account are then tried, and it is assumed that in the considered hypotheses, there is the one that is correct. How many ordered hypotheses to consider is a next question we deal with in Chapter 5. The next factor for the side-channel attacks is also success rate.

The success rate means the probability the side-channel analysis provide good answer regarding the complexity. Of course, the more complex the attack is the more it is successful. Theses two factors, the complexity and the success rate, depend on all the steps, the chosen side-channel, the leakage model, the measurements, and also on distinguishing method used.

The outcome of this step is vector of the chosen number of the first ordered hypotheses. The number is chosen maximizing success rate, but it is limited by available computation power. The boundary is around 2^{60} nowadays.

2.4 Countermeasures

The goal of each countermeasure is to make the physical behavior independent of the data processing or operation performing, or to detect a tempering with the device in order to delete all the sensitive material.

The detection can be made by checking voltage, capacitance, and electromagnetic field, and when a change is detected the reaction is to delete the sensitive data, like key material. Those are active detections. Passive detection is when the device is protected by filing and signeting of screw holes. In this case the attack can be still performed, but it can be detected.

There are two methods to make the device behavior, like power consumption, independent of the data processing and operation performing, and those are masking and hiding.

2.4.1 Hiding

The goal of hiding is to make an additional noise or make the power consumption constant. If the intend is to hide the exploitable signal in noise, then a noise generator can be added, or an additional logic is added in order to produce noise. Another technique is based on random dummy cycles, or random independent operations performance. If there is the intend to make the power consumption constant, the Dual Rail Logic (DRL) [13] can be used. The advantage is that the power consumption is really independent of data and operations. The disadvantage is that the power consumption is constantly maximal. If we would talking about making constant computation time, the ''for'' loops must not be interrupted if the results is computed, and the conditional branching must be made in the way that every branch must take the same time or power consumption depending on the side-channel. For instance, if all the operations are performed on all the branches but only the desired output of the certain branches are taken. More about all the techniques can be found in [40].

2.4.2 Masking

Masking [5] is based on randomizing being processed data in order to randomize physical behavior of the device (power consumption) and make so the behavior independent of the data. Random secret mask is applied subsequently step by step to the intermediate results of the algorithm. Data are masked at the input and demasked at the output. The output mask is computed from masks which were applied during the algorithm performance. Secure masking should meet these rules:

1. Mask is secret value that is generated within the device. The value of the demasking mask, and the masked values are computed inside the device securely considering side-channels. All masks, even the demasking one, are secret.

2. A new random mask must be applied at the input to the device, and new random mask must be applied to the each intermediate result before each next operation. That means, before registering new state, mask is applied. Data are demasked only at the output of the device. Thus,the data are demasked only after the final registration, when data are read of the output.

3. Each mask must be independent of the other masks. Every possible input to a function must be possible to obtain by masking of any possible input of that function. All the possible masks, and thus also possible masked input values, must be used with the same probability. Masks must be uniformly generated by a cryptographic generator.

The most common methods are additive masking, boolean masking [39], and multiplicative masking [3]. If the masking is applied to data which enters into a linear

operation, the masking is easy to implement. However, if the operation is nonlinear, the masking is not so easily applicable. We will demonstrate this on AES S-box.

$$y = S(x). \tag{2.1}$$

If the S-box is implemented as a table-look-up, then 256 different (masked) S-boxes must be stored in memory, or computed on-the-fly. The mask m will then address the masked S-box so that

$$y_m = S_m(x \oplus m). \tag{2.2}$$

The masked S-box is constructed as

$$S_m(x \oplus m) = S(x) \oplus m. \tag{2.3}$$

Therefore,

$$y = y_m \oplus m. \tag{2.4}$$

This approach is not perfect since the Eq. 2.3. Better approach is to use two masks.

$$y_{m_2} = S_{m_1,m_2}(x \oplus m_1). \tag{2.5}$$

$$S_{m_1,m_2}(x \oplus m_1) = S_{m_1}(x \oplus m_1) \oplus m_2 = S(x) \oplus m_1 \oplus m_2. \tag{2.6}$$

$$y = y_{m_2} \oplus m_1 \oplus m_2. \tag{2.7}$$

The multiplicative masking can be demonstrated on the case the S-box is implemented in the logic. The S-box is expressed as $y = Ax^{-1} \oplus b \in GF(2^8)$. Using the multiplicative approach, the S-box can be masked as

$$y_m = A(x \cdot m)^{-1} \oplus b \cdot m^{-1}, \tag{2.8}$$

$$y = y_m \cdot m_1. \tag{2.9}$$

An example of provably secure masking of AES can be found in [63]. In the work [3], an effective method for switching between boolean and multiplicative masking can be found. Multiplicative masking can be also demonstrated on RSA. At the input of the decryption algorithm, new mask m is applied on the input message x,

$$y_m = (xm^e)^d \bmod n, \tag{2.10}$$

$$y = y_m m^{-1} \bmod n, \tag{2.11}$$

where e is public and d is private, and $m \in Z_n^*$. Exponent d can also be masked.

For masking and hiding elliptic curve cryptography, see comprehensive state-of-the-art [19], and for a survey of methods for protecting McEliece PKC, consult works (Repka and Cayrel [54]) or (Repka and Zajac [59]).

Chapter 3

Goals

3.1 Improve CPA of 16-bit Integer Multiplier in FPGA

The goal is to look at a possibility to guess a constant operand of a 16-bit integer multiplier in FPGA from generic point of view. This constant operand has been multiplied by known ordered set of second operands. In order to distinguish between possible hypotheses about the value of the constant operand, correlation coefficient should be used. That means, there are not special analyses or preprocessing techniques, nor special side-channel-leakage models used. There is only the classical correlation power analysis employed. The goal is not to adjust the analysis of the multiplier implementation to gain the best success rate, and make it appropriate for the one implementation instance, but rather see such generic attack possibilities.

We can expect more than one HWPM/HDPM key hypothesis remaining after CPA, even using simulated leakage, since multiplication by a constant is a linear function, while for an example, if attacking AES S-Box, there should be identified only one key hypothesis because it is nonlinear function at all. This goal is to look at possibilities how to improve this kind of attacks to reduce the linear impact by only using the same traces and the same generic method.

In this linear case, CPA using measured power traces achieves better success rate than CPA using simulated power traces. Often vulnerability of cryptosystems to this kind of attacks is approximated by using only simulated power traces. Therefore, it is valuable to *analyze possible errors of approximations based only on simulated data.*

Implement an application for measurements & analyses in order to perform CPA attacks and investigate its properties. This application should be able to control a measurement device, and a cryptographic device remotely, and it should be able to automatically perform measurements and analyses. Finally, *Propose a countermeasure against the CPA attack to (EC)DSA.*

3.2 Timing Fault Injection Analysis of chosen steps of McEliece PKC

There exits only a few implementations of McEleliece PKC. Many of them bypass the original proposal of McEliece particularly because of the size of the key pairs. First, implement the original McEliece PKC [42] with unfixed parameters, which will *produce test vectors for all the important intermediate results* and *implement a tool for chosen leakage measurements*. Using the implementation of the original McEliece PKC and the application for leakage measurements & analyses *perform timing fault injection analysis* of chosen steps of McEliece PKC.

Chapter 4

Methodology

4.1 Analyzed Implementations

Generally, attacking ASICs is harder than attacking FPGAs, and attacking FGPAs is harder than attacking μControllers or processors. Processors and μControllers are easier attackable as far as they have symmetric buses of constant length and width.

4.1.1 DSA & ECDSA: 16-bit integer multiplier in FPGA

The sensitive integer multiplication is the multiplication kr, where r is known and k is the private key. In this work, we call k as the constant operand or key, and we call r

Algorithm 4.1 Digital Signature by ECDSA

Require: Private key k, Message m, Domain parameters $(\times, G, q)$.
Ensure: Digital signature (r, s) of the message.
1: Generate randomly and uniformly nonce n, $0 < n < q$.
2: Calculate the curve point $(x_1, y_1) = n \times G$
3: $r = (x_1 \bmod q)$.
4: **if** r is 0 **then**
5: GOTO Step 1.
6: **end if**
7: $s = n^{-1} (\mathrm{Hash}(m) + kr) \bmod q$.
8: **if** s is 0 **then**
9: GOTO Step 1.
10: **end if**
11: **return** (r, s).

as the second operand of the multiplication. This sensitive multiplication is performed in the DSA as well as in Step 7 in the ECDSA (Alg. 4.1).

A 16-bit integer multiplier is implemented in FPGA. The FPGA has further implemented only the necessary functionality for our experiments. The power consumption has been measured during multiplication of k by known ordered set of second operands. The CPA analysis targets power consumption caused by registers that register results of multiplications. It is generally accepted that the power consumption of registers is linearly dependent on number of $1 \rightarrow 0$ and $0 \rightarrow 1$ transitions. Thus, the power consumption can be simulated by the HDPM which is better fitting than the HWPM. However, measured power consumption will be noised by other functionality of the FPGA, which runs parallel, and also by the environment. Consider now Signal to Noise Ratio. In our case of analysis, signal consists of dynamic power consumption caused by the 32-bit registers for multiplication results. The noise signal consists of dynamic power consumption caused by LFSR (used to generate the known ordered set of second operands), state machine (used to control dataflow), UART (for communication), and signal added by environment and measurement.

This implementation was made at the Department of Electronics and Multimedia Communications, FEI-TUKE, Košice in Bratislava by Michal Varchola et al., and it was used to achieve desired results which were presented in the work (Repka and Varchola [57]) ,wherein it is shown that CPA using measured power traces is better than CPA using simulated traces, the errors of approximation of CPA success rate and complexity are investigated in Sec. 5.1.4, and finally the CPA of an integer multiplier was improved eliminating the error of the approximation based on simulated power traces. Thanks to the imporvement the CPA is more successful and more blocks of the key can be revealed. The improvement is demonstrated in the work (Repka, Varchola, and Drutarovsky [58]) on the 16-bit integer multiplier in FPGA.

4.1.2 McElice PKC in 64-bit CPU

The original McEliece PKC proposal is interesting thanks to its resistance against all known attacks, even using quantum cryptanalysis, of course, in an IND-CCA2 secure conversion. We made a generic implementation of the original McEliece PKC proposal (Sec. 5.4), which provides test vectors (for all important intermediate results), and also in which a measurement tool for side-channel analysis is employed (Sec. 4.4.2). To our best knowledge, this is the first such an implementation. This Calculator is valuable in implementation optimization, in further McEliece/Niederreiter like PKCs properties investigations, and also in teaching. Thanks to that, one can, for example, examine side-channel vulnerability of a certain implementation, or one can find out and test particular parameters of the cryptosystem in order to make them appropriate for an efficient hardware implementation. This implementation is available [1] in executable binary format, and as a static C++ library, as well as in form of source codes, for Linux

and Windows operating systems.

Since we have Post-Quantum PKC in the secure cryptosystem property setup [18], the only possibility how to break this PQ-PKC is via side-channel attacks. Recently, several side-channel attacks have been published [75, 66, 74, 76, 77]. It is possible to attack key generator, decryptor, and also encryptor. We stressed only the Patterson's algebraic decoding algorithm used in the decryption process. By the tool, secret error vector, secret permutation, and secret goppa polynomial can be guessed, and the success rate of the guessing can be evaluated.

This implementation was published in the article (Repka [52]). More about code-based cryptography and post-quantum McElice PKC can by found in our work (Repka and Cayrel [54]). For McEliece PKC like cryptosystems, we summarize security in work (Repka and Zajac [59]).

4.2 Analyzed Devices

4.2.1 Altera DISIPA FPGA board

For the DSA and ECDSA implementation (Sec. 4.1.1) analyses, we used our novel experimental platform (Fig. 4.1) for measuring power consumption of FPGAs (namely the Altera Cyclone III). The system provides the following features:

1. Measurement points (Fig. 4.1);
2. EMI shield which protects against electromagnetic pollution;
3. Strong Murata filters are assembled on a power line in order to minimize noise from the power supply.

The FPGA and measurement points circuitry have their own chamber in the shield. All: linear regulators + filters, configuration circuitry, input/output circuitry, and the main Murata filter have separate chambers as well. Described improvements enhance signal-to-noise ratio of the leakage, or in other words will reduce the number of traces needed for a successful DPA attack. We want to get as clean leakage signal as possible in order to assess the strength of particular countermeasures. We are curious, if simple (but efficient) EMI shielding, or the usage of another measurement point causes otherwise secure DPA countermeasure to be inadequate.

Up to now, we have found that the selection of measurement points matters. The voltage drop on a series measurement resistor is definitely not the best choice. We found out that the voltage on the decoupling capacitor (Fig. 4.1.c) gives us the best results.

This implementation was made at the Department of Electronics and Multimedia Communications, FEI-TUKE, Košice in Bratislava by Michal Varchola et al. Performed analyses using this device was published in (Repka and Varchola [57], and Repka, Varchola, and Drutarovsky [58]).

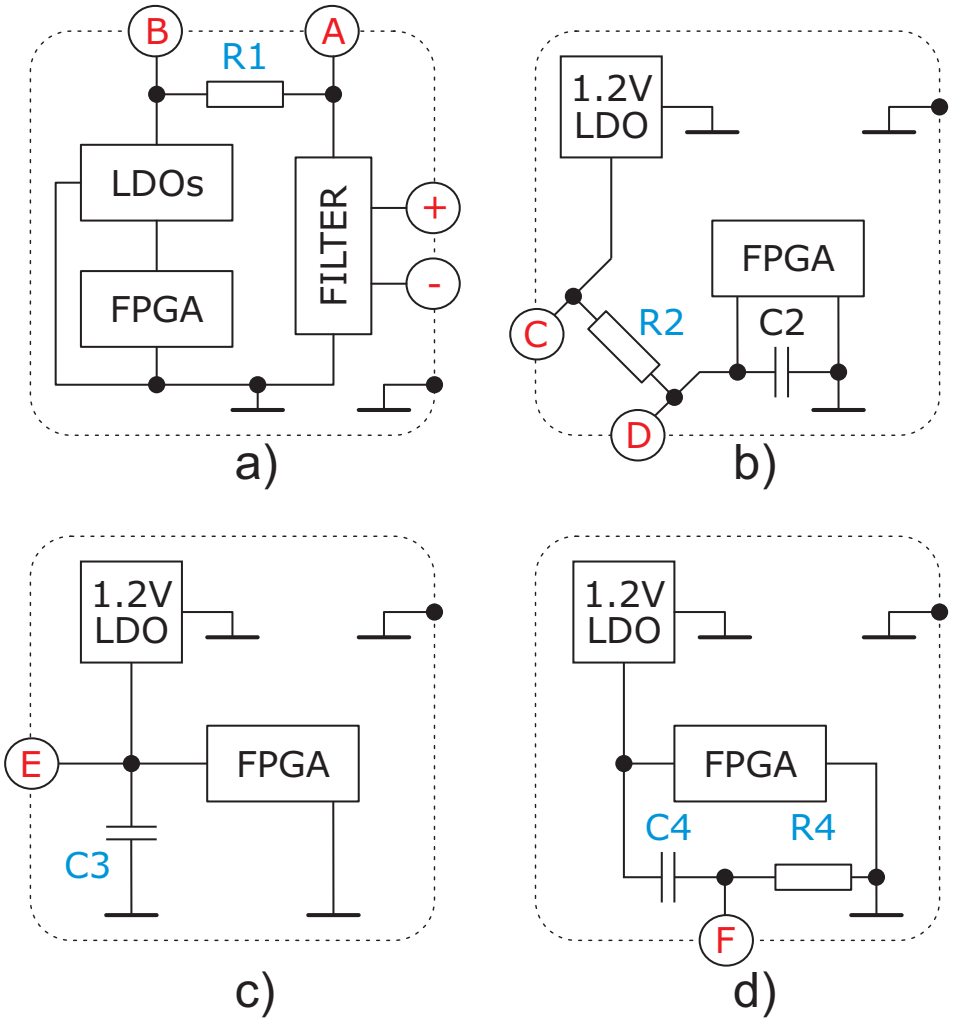

Figure 4.1: Schematic diagrams of measurement points in the DISIPA FPGA board. a) current flow from a linear regulator to the FPGA; b) current flow from the power supply to a linear regulator; c) the voltage on the decoupling capacitor; d) current flow from a decoupling capacitor to the FPGA.

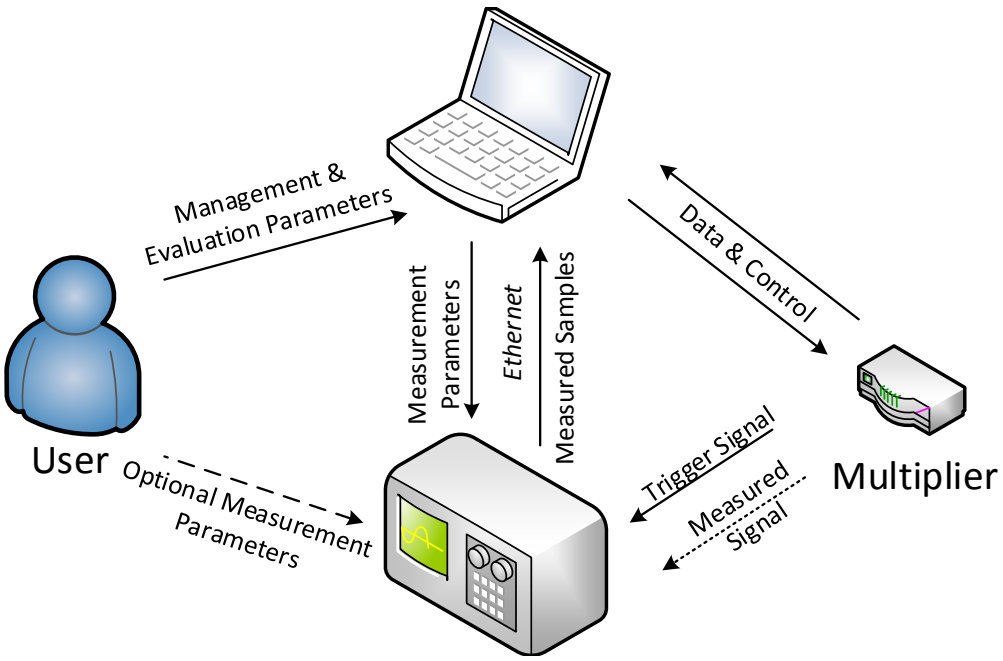

Figure 4.2: Top-Level Measurement & Attack Setup.

4.3 Measurements & Attacks Setup

The measurement setup is depicted in Fig. 4.2. An user uses notebook as an evaluation and management work station which is equipped with applications for measurements and analyses (Sec. 4.4). The workstation manages the whole attack process. It sends data to the cryptographic device (Sec. 4.2); it also sets up the measurement device, in our case one of the oscilloscopes in Sec. 4.5; and finally, it performs the mathematical part of the analysis, the correlation computations, success rate evaluations, and it further traces various indicators in various trends as mentioned in Sec. 4.4 and 4.6. The cryptographic device starts measurements by a trigger signal (Sec. 4.5.3), which is scanned by the measurement device. Thanks to this, power traces can be easily aligned.

4.4 Measurement & Analysis Tools

4.4.1 Application for power consumption measurements & analyses

This C++ application can provide comprehensive data for analysis of CPA. It provides multi-threading features thanks to which conducted analyses are very fast if used on

multi core processors. Thanks to the modularity of the application, the application can be enhanced to future attacks and various leakage models. This application implements also interface for oscilloscope remote controlling, and it is designed to be used in the measurements & attacks setup in Sec. 4.3. The application is configured using initialization files with the .ini suffix. The application traces various indicators in various trends such as listed bellow. The application is designed and developed so it can measure and process millions of power traces independently of operation memory of the computer. For instance, when one power trace has 2000 samples (50ns), the acquisition of 1M of power traces using the oscilloscope LeCroy WavePro 740Zi takes 65 seconds.

4.4.1.1 List of indicators

Correlation coefficient for the correct key hypothesis. Since the CPA is implemented in this application, the maximal correlation coefficient is recorded for the correct key hypothesis separately. This correlation coefficient is then printed into the output file for various trends described bellow.

Difference of correlation coefficients for the correct key hypothesis and incorrect hypothesis. It is the difference of correlation coefficients for the incorrect key hypothesis with the maximal correlation and the maximal one for the correct key hypothesis. This value is than printed into the output file for various trends described below.

Order of the correct key hypothesis. Essentially, the correlation coefficient is used to order the key hypotheses in the way that the key the most correlated hypothesis is the most probable candidate to the correct key, and thus it has the first position. This value is than printed into the output file for various trends described below.

Time moment with max correlation for the correct key hypothesis. The correlation matrix can have more columns than only one. Each column in the correlation matrix is for a time moment (a power trace sample) or time interval (interval of power trace samples that was preprocessed, for instance averaged). Hence, it is an index of the correlation matrix column in which the correct key hypothesis achieved the maximal correlation coefficient. This indicator is traced for various trends.

Success rate of CPA attack. This is relative count of occurrences of the event that the correct key hypothesis was in the first D positions. The D is called threshold and can be configured as a vector traced of thresholds.

Fitness of simulated attack. This application computes all the above trends for measured as well as for simulated power traces. Success rate of CPA attack differs according to the fact whether simulated power traces or measured power traces were used. Regarding the fact the simulated power traces are simulations only, and the measured power traces are the reality, this application record intermediate values to estimate errors of the first and second type, α and β, see Sec. 5.1.4 for definitions and more details. Those errors are traced for all the mentioned trends.

4.4.1.2 List of tracing trends of the indicators

Al the indicators can be recorded with respect to the following variables. These parameters are configured in measurements.ini file.

Part of power trace. This defines the interval of power trace that is considered for analyses. It is used if the power trace is long, and the sensitive information is only in the certain interval in the power trace. If we have no knowledge which part of the power trace it is, we can configure the application for analysis of interval of various length at various positions of the power trace in one turn. This can be used in order to determine the most valuable part of the power trace. Using this feature, we found that for the CPA, it is the most valuable to take the whole cycle when the sensitive intermediate result is registered. This number can be configured as a start, step, and stop value.

Length of preprocessing interval. This value determines number of intervals to which the considered part of the power trace is divided. These intervals are then represented by an average value of each interval. The measurements can be configured to analyze various lengths of preprocessing intervals in one turn. Using this trend we found that, for the CPA, it is the best to averaged the power trace of the whole cycle within which the sensitive intermediate result is registered. This number can be configured as a start, step, and stop value.

Number of power traces. Using this parameter, one can configure how many power traces to record for an attack. This number can be configured as a start, step, and stop value.

Vector of thresh holds. This is the vector of thresholds D_i. The success rate α and β are computed according to these thresholds. It is tested whether the correct key hypothesis index is less than these thresholds. These thresholds than provide view of complexity of the CPA attacks because, essentially, it is a number of ordered key hypotheses that must be take to the account after the CPA. It is because the correct key hypothesis is on the first position very rarely, therefore

we must consider more ordered key hypothesis in order to have the correct key hypothesis between them.

4.4.2 Application for time measurements & analyses

This measurement tool is the side-channel leakage measurement tool Repka [52] employed in the Patterson's algebraic decoding algorithm (Alg. 5.5). As we mentioned above, using the tool, secret error vector, secret permutation, and also the secret goppa polynomial, can be possible to guess. Moreover, power consumption and electromagnetic emanation leakages can be simulated using the measured data provided by this tool.

4.4.2.1 Measurement Type 1

This measurement type records average computation time, standard deviation of the computation time, and if applicable, average values and standard deviations of Hamming Weights and degrees of polynomials processed, and steps performed, during Patterson's algebraic decoding algorithm defined in Alg. 5.5. The purpose is to measure these Indicators (Tab. 4.1) dependency on $\mathrm{HW}(\mathbf{e})$, see Tab. 4.3. Hence, output file of this measurement type is composed as follows. As the first column there is the $\mathrm{HW}(\mathbf{e})$ growth according to that Indicators are recorded. Since there is a possibility to make such record for many random key-pairs, for the next key-pair there is the next such record separated by empty row. As the last data in the measurement file, summarization over all the key-pairs is placed. At the beginning of measurement file the measurement setup, such as stated in Tab. 4.2 is placed.

The average values and the standard deviations are computed from nTests encryptions. The measurement file contains nRandKeyPairs measurement records, each for one random private key. Examples can be found in graphs depicted in Fig. 5.7 and 5.8. Optionally, also the test vectors can be stored in the disk as a text file.

4.4.2.2 Measurement Type 2

This second measurement type is designated to measure Indicators (Tab. 4.1) dependency on secret goppa polynomial (Tab. 4.4). Corresponding to each information about goppa polynomial, Indicators are measured. If moreover dependency on secret permutation is desired to measure, then the flag `is_storeKeys` must be set to 1. Output file of a measurement of this type is composed as follows.

At the beginning of the file, the measurement setup is presented. The measurement setup is arranged in the Tab. 4.2. Regarding the measurement setup, afterwards, information about goppa polynomials (Tab. 4.4), $\mathrm{HW}(\mathbf{e})$ (Tab. 4.3), and the measured Indicators (Tab. 4.1) are stored form left to right respectively. Thus, for each goppa

polynomial there is a row, which displays also step-by-step $\mathrm{HW}(\mathbf{e})$ and indicators for each $i \in [\min(\mathrm{HW}(\mathbf{e})), \max(\mathrm{HW}(\mathbf{e}))]$ according the measurement setup.

Such as in the measurement type 1, the average values and the standard deviations are computed from nTests encryptions. The measurement file contains nRandKeyPairs measurement rows, each for one random private key. Some examples in form of graphs can be seen in Fig. 5.9, ..., 5.14. Optionally, also the test vectors can be stored in the disk as a text file.

4.5 Measurement Devices

Measurement devices and probes used are listed in this section. The measurement devices and probes listed here were used for power consumption and electromagnetic

Table 4.1: Measurement Type 1: Indicators measured to perform side-channel attacks, and determine where the leakages occur. Examples in form of graphs can be found in Fig. 5.7 and 5.8.

Item/Column number	Step of Alg. 5.5	Indicators: avg(.), std(.)	Object
$1, \ldots, 6$	1.	computation time, deg, HW	$S(Z)$
$7, \ldots, 12$	2.	computation time, deg, HW	$T(Z)$
$13, \ldots, 18$	3.	computation time, deg, HW	$\tau(Z)$
$19, 20$	4.	computation time	EEA
$21, \ldots, 24$	4.	deg, HW	$\alpha(Z)$
$25, \ldots, 30$	5.	computation time, deg, HW	$\sigma(Z)$
$31, 32$	6., 7., 8.	computation time	$\mathbf{e}$ construction

Table 4.2: One measurement file header for measurement setup.

Value	Description
m	$\mathbb{F}_{2^m}$
t	deg $g(Z)$
nRandKeyPairs	Number of randomly generated key-pairs
nTests	Number of random messages per key-pair and $\mathrm{HW}(\mathbf{e})$
$\min(\mathrm{HW}(\mathbf{e}))$	Start $\mathrm{HW}(\mathbf{e})$
$\max(\mathrm{HW}(\mathbf{e}))$	End $\mathrm{HW}(\mathbf{e})$

Table 4.3: Information recorded about secret error vector.

Item/Column number	Value
1	$\mathrm{HW}(\mathbf{e})$
$2, \ldots, n+1$	$[\mathbf{e}]_2$

Table 4.4: Measurement Type 2: Information recorded about secret goppa polynomial in order to measure success rate of an attack. Examples in form of graphs can be found in Fig. 5.9, …, 5.14.

Item/Column number	Value
$1, \ldots, (t+1)$	$g_0, \ldots, g_t$
$(t+2), \ldots, (2t+3)$	$\mathrm{HW}(g_0), \ldots, \mathrm{HW}(g_t)$
$(2t+4)$	$\mathrm{HW}(g(Z))$

emanation signal acquisitions.

The time side channel was measured on computer platform by using certain instructions for CPU cycles counting for particular processor. The generic CPU Tick Measurement Library [20] was used.

4.5.1 List of Oscilloscopes

During measurements, the smart, embedded, acquisition memory of the listed oscilloscopes in sequence mode was used. Thanks to this acquisition memory, we were able to achieve very fast signal acquisitions, for instance, when one power trace has 2000 samples (50ns), the acquisition of 100K power traces using the oscilloscope LeCroy WavePro 740Zi takes 6.5 seconds.

LeCroy WavePro 7200A equipped with 8-bit A/D converter, 2GHz bandwidth, 20GS/s sample rate, and 10M points in acquisition memory. This device was used to measure power consumption of FPGA ACTEL FUSION M7AFS600 as voltage drop on resistor placed on power supply of FPGA core. The probe used was active differential voltage probe (Repka, Gaspar, and Fischer [55]). This device was also used to measure power consumption on the ground of μControler PIC18F2520 using passive voltage probe (Repka [50]).

LeCroy WavePro 740Zi equipped with 8-bit A/D converter, 4GHz bandwidth, 40GS/s sample rate, and with extended acquisition memory to 128M points. This device was used to measure power consumption of FPGA ACTEL FUSION M7AFS600 as well.

AGILENT INFINIIUM DSO9404A equipped with 8-bit A/D converter, bandwidth, 20GS/s sample rate, and 10M points acquisition memory. This device was used to measure of power consumption as voltage on the decoupling capacitor (Fig. 4.1.c) placed on power supply of the FPGA ALTERA Cyclon III core. The probe used consisted of coaxial cable as passive voltage probe (Sec. 4.2.1).

4.5.2 List of Probes

Here is the list of probes which were used during measurements performed. We list probes, their connection, and their effectiveness as well.

Passive voltage probe on the ground of μControler PIC18F2520 (Repka [50]). This is the most noised measurement point, since the ground is shared. This measurement technique would not work properly for hardware platform.

Active differential voltage probe connected to the resistor placed in power supply of FPGA core, and other measurement points (Repka, Gaspar, and Fischer [55]). This is very good, but not the best, measurement point. The signal must be amplified by good quality low noise amplifier what makes this measurement the most expensive.

Spot, passive electromagnetic probe which was placed over FPGA core (Repka, Gaspar, and Fischer [55]), essentially over the part of the FPGA where the sensitive operation is performed. Measurements by this electromagnetic probe can achieve significantly better quality of acquisition of signal containing the sensitive information than the active differential voltage probe connected to resistor, but there must be known exact place where the sensitive operation is placed in the FPGA surface to focus the probe exactly to this place. We focused this probe to the embedded RAM memories of the FPGA.

Coaxial cable as a passive voltage probe to measure voltage on the decoupling capacitor (Fig. 4.1.c) placed on power supply of FPGA (Sec. 4.2.1). This is the best measurement point at all. However, there is very important to use certain capacitor. There is possibility use more capacitors of certain characteristics and place them in certain positions to achieve better acquisition of signal containing the sensitive information. This measurement method was used for the 16-bit integer CPA performed in (Repka and Varchola [57]) and improved in (Repka, Varchola, and Drutarovsky [58]).

4.5.3 Triggering the Signal Recording

There are two possibilities where to trigger an oscilloscope to start recording a signal. One possibility is to trigger before the operation is performed and the second one is to trigger afterwards the traced operation is performed. We trigger before the operation is performed.

The reason why we decided to trigger before the operation performance is that if the trigger signal was after the operation performance, there would be needed shifting the timebase to the right. When shifting the timebase to right, the fast acquisition memory of the oscilloscope is used what significantly reduces the memory available for the sequence mode of measurements. The acquisition memory is very expensive, and thus its size is strictly limited. Therefore, in order to have as much acquisition memory available for the measurements as possible, the trigger signal was placed before the traced operation performance. In this case, the trigger signal must be delayed because the trigger signal has non negligible impact to the measured signal that is intended to be used in the attack. Thus, it is important to wait a while to measure as clear signal as the signal is not affected by the trigger signal. If the measured signal was affected by

the trigger signal, the recored signal would have to be averaged as the impact of the the trigger signal would be removed.

4.6 Performed Analyses

4.6.1 CPA of the 16-bit integer multiplier in FPGA

In Sec. 5.1 and 5.2, where CPA attack against DSA & ECDSA is analyzed and improved, we are attacking actually only one integer 16-bit multiplier (Sec. 4.1.1). The integer multiplier is implemented in the DISIPA FPGA board (Sec. 4.2.1). Further in that sections, we approximate the attacks complexities and success rates for N blocks of key, and errors for approximations using simulated attacks. Also the attack is significantly improved. The analyses preformed were conducted using oscilloscope AGILENT INFINIIUM DSO9404A (Sec. 4.5.1) in measurement & analyses setup in Sec. 4.4, by using the C++ application for power consumption and electromagnetic emanation measurements & further analyses described in Sec. 4.4.1.

Let $\mathbf{x}_M$ denotes a vector of M known different second operands. Hence, we have M second operands known and we know the order they were processed. By x_m, we will denote the m-th 16-bit second operand.

By $\mathbf{L}_{M,T}$, we will denote matrix of power traces, where m-th power trace $\mathbf{l}_{m,*}$ consisting of T samples corresponds to processing of the m-th second operand.

While the device is being processing data, the device is emitting some extra (leakage) information through its physical behavior dependent on the data it is processing and operation it is performing. The leakage-information can be for example sound, light (photon emission), computation time, not only the power consumption of the device. Therefore, the multiplication process is not only $y_m = \mathrm{MULT}(x_m, k)$, but rather

$$(y_m, \mathbf{l}_{m,*}) = \mathrm{MULT}(x_m, k), \tag{4.1}$$

where $\mathbf{l}_{m,*}$ is a leakage that is the power consumption in our case.

The first step of our CPA attack is to choose the leakage point of the implementation, to which a hypothetical power consumption will be made. Essentially, we are focused at the power consumption of y_m registration.

The second step of our CPA attack is to collect data important to reveal the key. Hence, we must measure the power consumption of the cryptographic device while it processes different second operands $\mathbf{x}_M$. Since we are using HDPM, it is important to know order in which the second operands was processed. Next, CPA attack needs to have the power traces aligned. We used a trigger signal that starts power traces recording at the start of the multiplication. Power traces are then aligned according to the trigger signal.

The next step of the attack is to calculate hypothetical multiplication results for every possible choice of the key - the constant operand. Therefore, we obtain matrix $\mathbf{H}_{M,K}$, where K is the number of possible values for key. Thus, in the matrix, each row is for each second operand, and each column is for each possible value of the key.

$$h_{m,k'} = \mathrm{MULT}(x_m, k'), \tag{4.2}$$

where k' is a hypothesis to the real k, and $h_{m,k'}$ is thus hypothesis to the real result of the multiplication. The number of possible hypotheses k' is K.

The next step of our CPA attack is to compute hypothetical power consumption $\mathbf{P}_{M-1,K}$ according the hypotheses $\mathbf{H}_{M,K}$, where

$$p_{m,k'} = HD\left(h_{m,k'}, h_{m+1,k'}\right). \tag{4.3}$$

The matrix $\mathbf{P}_{M-1,K}$ contains one row less because of the power model used (Eq. 4.3). There are many possibilities of power models (HWPM, HDPM, bit power model, zero value power model, and at the lower level, one can use differential equations). The lower level of the power model, the precision of the power model is better, but the more information and greater computational power is needed. In our case, we used HDPM. Note that we desire to approximate complexity of this kind of attacks in generic sense. For the approximations, we used only one 16-bit multiplier, and according the results, we approximated complexities and successes of attacks against key consisting of N 16-bit blocks, and relevant errors of thees approximations based on simulated power traces are approximated also.

Now the leakage signal must be compared to the hypothetical power consumption. In our case, we used a compression method to compress the matrix $\mathbf{L}_{\mathbf{M},\mathbf{T}}$ to vector $\mathbf{l}_M$. We simply computed average value for each row $\mathbf{l}_{m,*}$ of the matrix $\mathbf{L}_{\mathbf{M},\mathbf{T}}$. The vector $\mathbf{l}_{m,*}$ contains only samples measured during the multiplication result registration clock. Therefore each row $\mathbf{l}_{m,*}$ is represented by average value $l_m = \mathrm{avg}(l_{m,1}, \ldots, l_{m,T})$. This also improves the success rate of the attack, since neighborhood samples in the power traces are correlated. In order to find which of the key hypotheses is correct, we must compare the hypothetical power consumption to the measured one. As the comparison method, we used the Pearson's Correlation Coefficient. Note that according to the power model used (Eq. 4.3), the matrix $\mathbf{P}_{M-1,K}$ contains one row less, and one m-th row of that matrix corresponds to the $(m+1)$-th element of the vector $\mathbf{l}_M$. Hence, the correlation is computed as:

$$r_{k'} = \rho\left(\mathbf{p}_{*,k'}, (l_2, \ldots, l_M)\right). \tag{4.4}$$

After the comparison, we obtain a correlation vector $\mathbf{r}_K$. According to this correlation coefficients, we will order the key hypotheses in our attacks. Recall that we know that the correlation is negative in our case. Therefore, the key hypothesis with the maximal

Algorithm 4.2 Fault injection & timing analysis of the McEliece PKC key decryption (Alg. 5.4).

Require: Ciphertext $\mathbf{y}$, number of errors t, length of codeword n, boundary ε.
Ensure: The guess to $\mathbf{e}$.

1: $\texttt{time}_{\texttt{ref}} = \texttt{measure\_time}\left(\texttt{decrypt}(\mathbf{y})\right)$
2: **for** $i = 0; i < n; i++$ **do**
3: $\quad \mathbf{y}_i = \mathbf{y} \oplus \texttt{to\_bin\_vector}\left(2^i\right)$
4: $\quad \texttt{time}_i = \texttt{measure\_time}\left(\texttt{decrypt}(\mathbf{y}_i)\right)$
5: $\quad$ **if** $\left(\texttt{time}_i < \left(\texttt{time}_{\texttt{ref}} - \varepsilon\right)\right)$ or $\left(\texttt{time}_i > \left(\texttt{time}_{\texttt{ref}} + \varepsilon\right)\right)$ **then**
6: $\quad\quad \texttt{time}_i = \infty$
7: $\quad$ **end if**
8: **end for**
9: $\mathbf{ordered\_time\_vec} = \texttt{ascending\_order\_regarding\_time}\left(\left(\texttt{time}_i\right)_{0 \leq i < n}\right)$
10: $\mathbf{e} = \texttt{take\_first\_t\_indexes\_i}(t, \mathbf{ordered\_time\_vec})$
11: **return** $\mathbf{e}$

negative correlation coefficient we assume to be the most probable and it thus has the first position. The key hypothesis with the second maximal negative correlation coefficient has the second position. In our case, we have correlation vector since the compression of the power traces, but in a case a correlation matrix would have more than one column, for each key hypothesis only the maximal one is considered.

4.6.2 Fault injection & timing analysis

This method to attack McEleice PKC, can be used to reveal messages, and some times also to reveal the secret permutation, depending on the certain implementation of the PKC. The notation used here is defined in Sec. 5.4. This attack is based on the idea that decryption of cipher text, which has normally t errors, is faster if it has less than t errors. The algorithm for this attack is listed in Alg. 4.2. This attack is focused on the McEliece PKC key decryption (Alg. 5.4), and it can reveal value of the error vector. The parameter ε is here to reduce false positives. To prefrom this attack the embedded leakage measurement tool can be used. This tool is demonstrated. We analyzed computation time regarding the error vector hamming weight of chosen steps of the decryption process in 5.5. Then we focused on the EEA step for solving the key equation. The computation time analysis of this step of the decryption was improved 5.6.

Chapter 5

Results

5.1 CPA Attack against DSA & ECDSA

5.1.1 Related work & our contribution

In work [32], attack against ECDSA implementation in passive RFID is performed. The ECDSA implementation is based on 163-bit Elliptic Curve, and the sensitive multiplication is performed using a 16-bit integer multiplier. They demonstrate revealing of the first 2 16-bit blocks of the one chosen secret constant operand d (private key) that is denoted as key or k in our work. The attack is aimed against Step 5 in the Algorithm 1 (Signature-generation scheme using ECDSA) listed in their work (it is Step 7 in the ECDSA (Alg. 4.1) in our work. The attack is especially aimed against the integer multiplication dr, where r is known to an adversary (it is public) and d is a private key.

In many works dealing with SCA, often one key is chosen and revealed, many times it is only a part of the key. In this work, we randomly and uniformly generated 665 16-bit keys and tried to reveal them. We used measured as well as simulated power traces using Hamming Distance Power Model. Based on these results, we estimated success rate and complexity of the attack. The complexity is represented by remaining key hypotheses after the CPA attack in both cases (measured and simulated power traces). We also estimated complexity and success of CPA attacks revealing $1 \leq N \leq 21$ 16-bit blocks of the key. We performed these attacks on FPGA, and note that attacking a processor or μcontroller can be less complicated than attacking FPGA or ASIC.

Moreover, we improved this attack using second CPA with different, but still generic, HDPM. While, after the first CPA, it was possible to reveal 21 16-bit blocks (336-bit) of the key with good probability and feasible complexity, afterwards the second CPA, it is possible to reveal 23 16-bit blocks (368-bit) of the key with good probability and feasible complexity. Finally, possible efficient countermeasure is discussed at the end of this work.

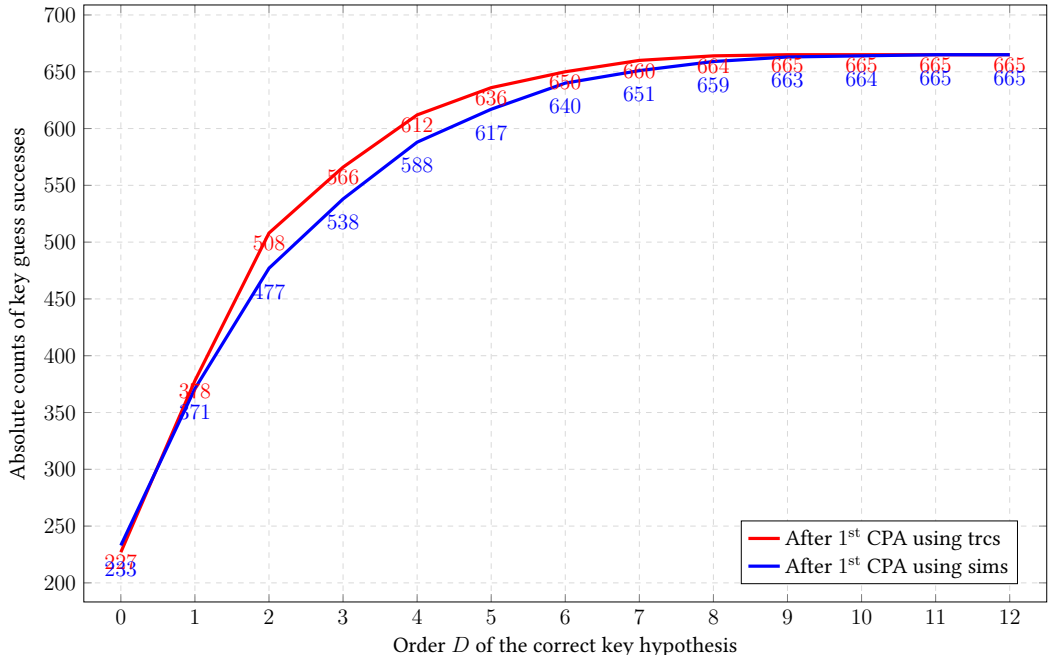

Figure 5.1: Success rates of the 665 CPA attacks using measured power traces (red), and simulated HDPM traces (blue), regarding D - max number of the key hypotheses taken to account after CPA.

5.1.2 Attack complexity & success for one 16-bit block of the key

CPA attacks are aimed against registers for multiplication results (four 8-bit registers). 4096 16-bit LFSR subsequent states was multiplied by the k step by step. We know order of results, thus we can compute HDs of previous and actual results giving us number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions in the result register in time. Hypothetical power consumption for all possible key hypotheses are made by computing HDs of subsequent hypothetical results of multiplications of all the 4096 LFSR states by possible keys. These hypothetical power consumptions are correlated to both measured power-consumption and simulated power traces (just the HDs) afterwards. As an outcome, the correlation vector $\mathbf{l}_M$ is obtained. Then next step is to order the key hypotheses according to the correlation coefficients. We exploited the fact that we have negative correlations.

The question is how many key hypotheses take at least to account to reveal the key after CPA. We will denote this number as D. Recall, we want to show how it can be dangerous to estimate this number by considering simulation only. The correct key hypothesis has thus index $0 \leq i \leq D$ after the CPA attack using HDPM simulated traces. Let j be the index of the correct key hypothesis in reality (using power traces).

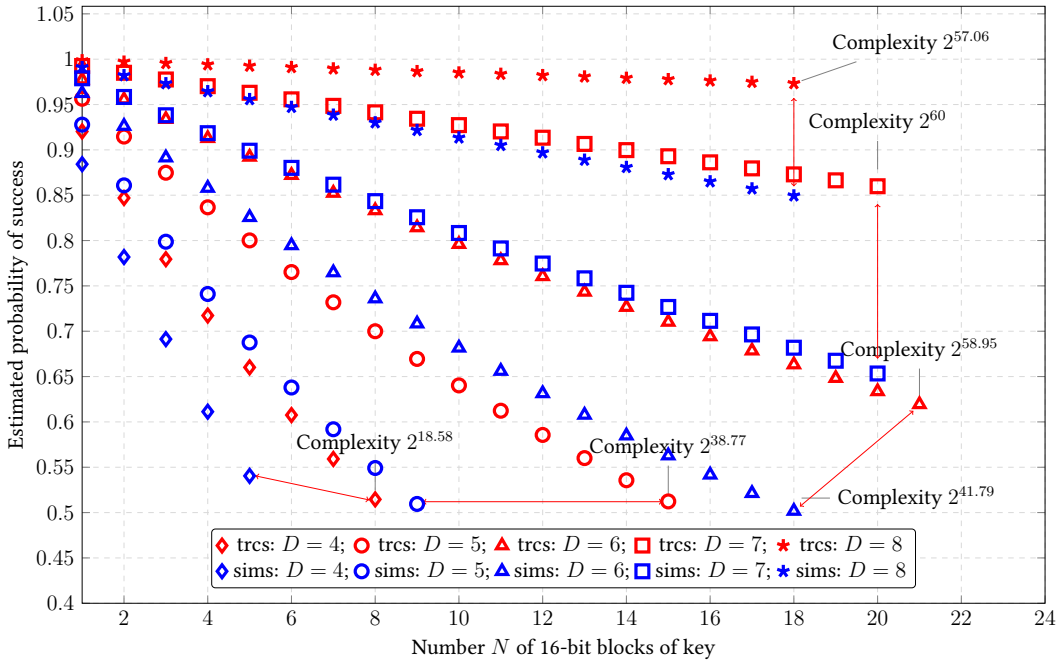

Figure 5.2: Estimation of CPA attacks complexity and success rate against N 16-bit blocks of k using measured (red) and simulated (blue) power traces after 1$^{\text{st}}$ CPA.

From the Fig. 5.1, it can be clearly seen that $j \leq D$ with better probability than $i \leq D$. It means that attack using measured power traces achieves better success rate than attack using simulated power traces. If we took 10 first key hypotheses ($D = 9$) ordered according to correlation coefficients, the real attack would have success in 100%, while when simulated power traces are used, it is 99.7%. If we took 5 first key hypotheses, the attack would succeed in 92.03, and 88.42%, respectively.

5.1.3 Attack complexity & success estimation for N key blocks

Since we can look at revealing of N 16-bit blocks of the key as on independent trials, we can write:

$$P(i_1 > D, \ldots, i_N > D) = P^N(i > D), \tag{5.1}$$

Therefore, also the complexity can be computed as:

$$\text{Complexity} = (D + 1)^N \tag{5.2}$$

According to the results, one can try to estimate power of the attack in terms of complexity (number of the key hypotheses remaining after CPA, the D) and success probability of the attack. We will do this for both simulated power traces and measured

one in order to see the difference between the estimations. Up to now, we attacked one 16-bit block of k. Based on this results, we are going to estimate attack possibilities to reveal N 16-bit blocks of k.

We bound these estimations. The probability of attack success must be greater than 0.5 and the complexity of the attack must be less or equal to 2^{60}.

The estimation is shown in Fig. 5.2. We can see that if the first 5 key hypotheses are taken after CPA against each block of the key ($D = 4$), the estimation using simulated power traces says that 5 blocks of the key can be guessed, while estimation using measured power traces shows that 8 16-bit blocks of the key can be guessed. In case of $D = 5$, 9 and 15 blocks of the key can be revealed for simulated and measured power traces respectively. If $D = 6$ (number of remaining hypotheses after CPA against each block of the key is 7), the estimation based on the simulation indicate that 17 blocks of the key can be revealed, and based on the measured power traces, it is 21. For $D = 7$, the difference between estimation based on simulations and measurements can be observed in the success rate. In the case of simulation, the estimated probability of success is around 0.65, while, in the case of measured power traces, it is approximately 0.86. In both cases 20 16-bit blocks of the key are predicted to be possible to reveal. If 9 first key hypotheses are considered after CPA against each block of the key ($D = 8$), the estimation using simulation indicates estimated probability 0.85. In the case of measured power traces it is close to 1. Here, 18 blocks of the key are indicated in both cases.

This result was published in (Repka and Varchola [57]).

5.1.4 Errors of simulated CPA attack

In order to show relevant error of the approximations of the attacks using HDPM simulated power traces, we performed analysis based on the following formulated hypotheses:

Definition 5.1.1 (Hypothesis H_0:)**.** It is enough to take D first key hypotheses ordered regarding the correlation coefficient. The correct key hypothesis has thus index $0 \leq i \leq D$ after the CPA attack using HDPM simulated traces.

Definition 5.1.2 (Hypothesis H_1:)**.** It is not enough to take the D first key hypotheses, and thus $D < i < K$.

The H_0 is not rejected correctly if $0 \leq i \leq D$ when H_0 is true in reality.

Definition 5.1.3 (Reality:)**.** As the reality, CPA using power traces are considered. Hence, let j be the index of the correct key hypothesis in the reality.

Probability that H_0 is rejected:

$$P(i > D) = P(i > D, j \leq D) + P(i > D, j > D), \tag{5.3}$$

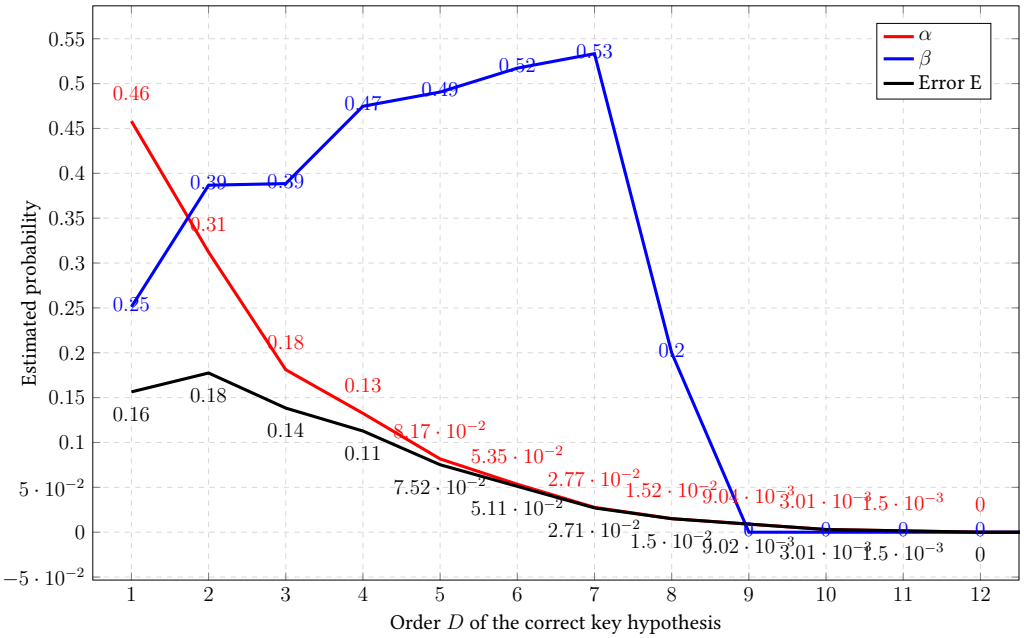

Figure 5.3: Estimation of α, β, and probability of E, for 16-bit key for various D.

$$P(i > D, j \leq D) = P(j \leq D)P(i > D \mid j \leq D), \tag{5.4}$$

$$P(i > D, j > D) = P(j > D)P(i > D \mid j > D). \tag{5.5}$$

Probability that H_0 is not rejected:

$$P(i \leq D) = P(i \leq D, j \leq D) + P(i \leq D, j > D). \tag{5.6}$$

This probability (Eq. 5.6) is depicted in absolute values as *sim* in Fig. 5.1.

Regarding the hypotheses defined, the two types of error can be made. The type one, α, and two, β, errors:

$$\alpha = P(i > D \mid j \leq D), \tag{5.7}$$

$$1 - \alpha = P(i \leq D \mid j \leq D), \tag{5.8}$$

$$\beta = P(i \leq D \mid j > D). \tag{5.9}$$

$$1 - \beta = P(i > D \mid j > D). \tag{5.10}$$

An approximation of both α and β errors is depicted in Fig. 5.3.

Type one error, α, means probability that in each case measurements success, the simulation fails. This error could be dangerous because we could say D is not enough

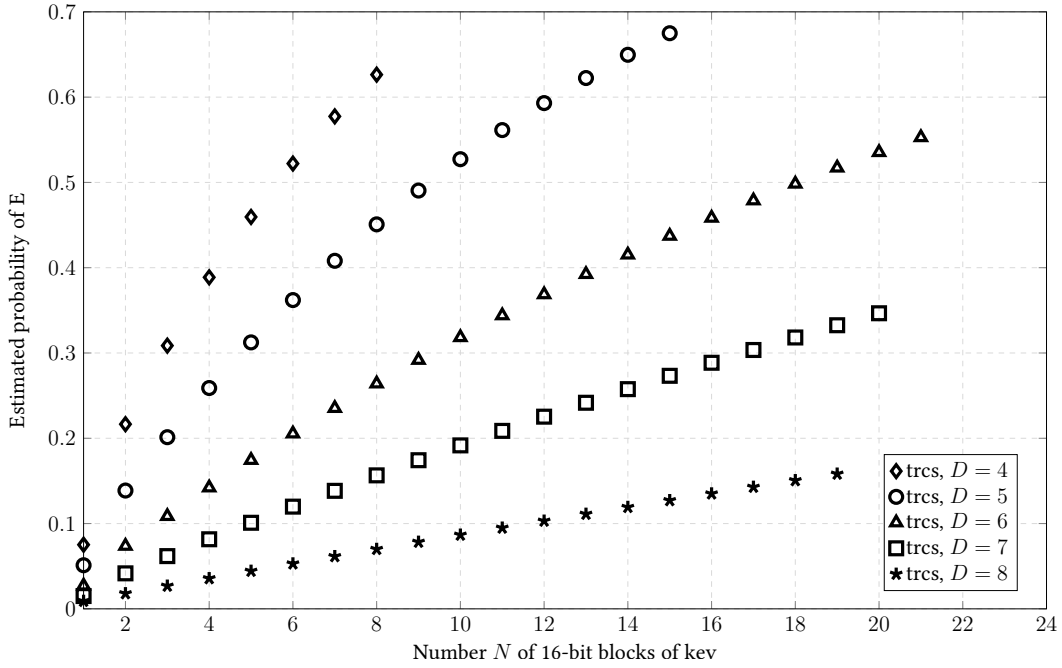

Figure 5.4: Estimation of probability of E for N 16-bit blocks of the key for various D.

yet for an attack, but in reality, when power traces are used, the D would be enough still. This could cause that, according the simulation, the complexity and success rate would looks more complicated as it is in the reality.

Type two error, β, means probability that in each case measurements fail, the simulations successes. Actually, this is not a crucial error for us in this work because we are trying to limit the worst case of the attack. In this situation we do not mind that simulation says that the attack will success for the D still but in reality D is too small because regarding this error we would secure the cryptosystem more than it is important, and this error is not so crucial as the α. For this reason, we stress only α in this paper.

Definition 5.1.4 (Relevant Error E:)**.** As the relevant error, we consider error that is influenced by the error of the first type only.

The probability of occurrence of this error is

$$P(E) = P(i > D) - P(i > D, j > D), \tag{5.11}$$

and its estimation is depicted in Fig. 5.3.

Now we are going to estimate this kind of error for guessing N 16-bit blocks of the key. We denote this error as E_N. It is event that we are wrong in rejecting H_0 at least

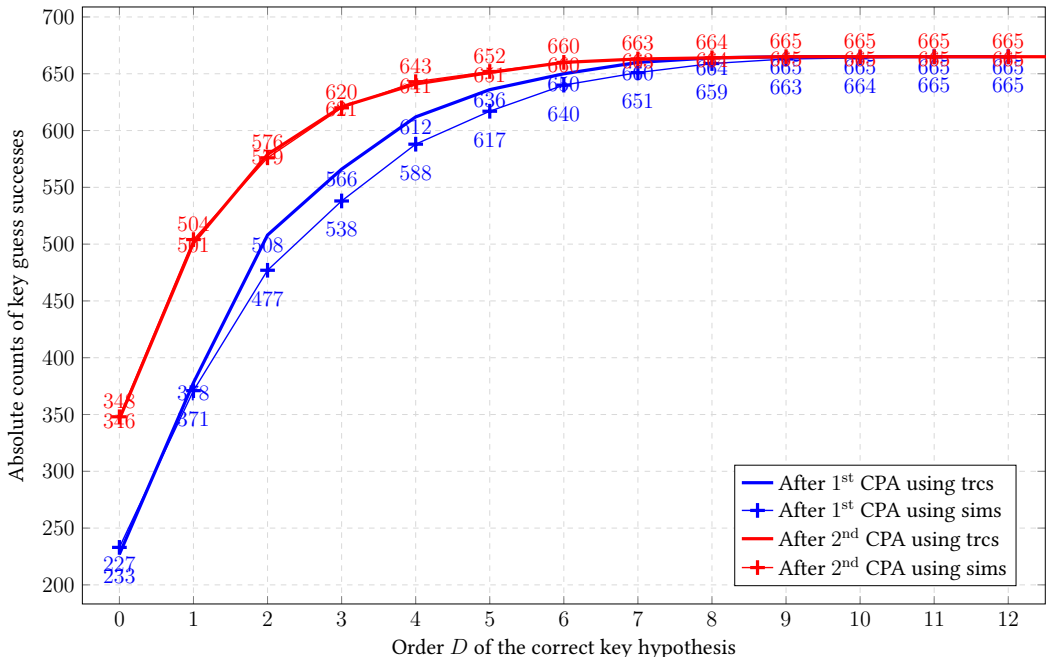

Figure 5.5: Demonstration of the improvement on results of guessing 665 randomly and uniformly generated 16-bit keys.

in one 16-bit block of the key. We can look at this problem as independent trials (Eq. 5.1). Therefore, we can write:

$$P(E_N) = P^N(i > D) - P^N(i > D, j > D). \tag{5.12}$$

The estimation of this probability is depicted in Fig. 5.4. For instance, if we were securing the multiplier regarding the simulated power traces in case of $D = 6$ where 336-bit key can be guessed, we would wrongly reject H_0 in more than 55%. It means that, regarding the simulations, it would seem that the attack is more complex and less success than it is in reality with probability more than 0.55.

5.2 Improving the CPA Attack against DSA & ECDSA

From the results above (Fig. 5.1), we know that if we take 10 first key hypotheses (D = 9), the attack will success in 100% for measured power traces. In order to improve the attack (see Fig. 5.5), we took the first 10 key hypotheses ordered according to the correlation coefficient after the CPA (blue marks) in both cases (measured and simulated power traces), and performed second CPA attack (red marks) in order to reorder the first 10 key hypotheses. In the second CPA attack, we made HDPM only

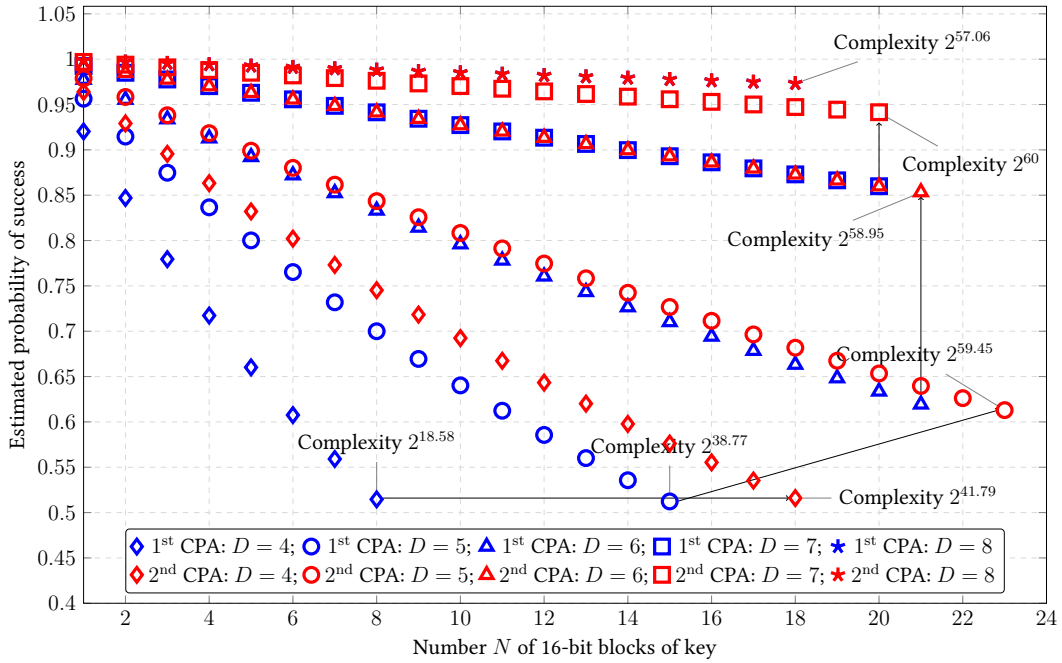

Figure 5.6: Estimations of attack complexity and probability after the second CPA (black markers) in comparison with the estimations after the first CPA (gray markers).

to the vector of the 16 least significant bits of the possible result of multiplication:

$$p_{m,k'} = HD\left(\mathrm{LSB}_{0\ldots15}(h_{m,k'}), \mathrm{LSB}_{0\ldots15}(h_{m+1,k'})\right), \tag{5.13}$$

where the $h_{m,k'}$ and $h_{m+1,k'}$ are hypotheses to the multiplication results regarding the key hypothesis k'. The new order of the 10 first key hypotheses brought significant improvement as can be seen in Figs. 5.5 and 5.6, respectively.

In the Fig. 5.5, we can see that we were successful in more attacks after reordering the 10 first key hypotheses after the second CPA. We can also see that the success for the simulated power traces was improved into the success level of the measured power traces. Actually, now it is slightly better than success rate for the measured power traces. Hence, now there is a negligible deference for D in case of measured and simulated power traces.

When we look at the guessing of N 16-bit blocks after the second CPA attack (Fig. 5.6), we can see the brought improvement since, now, 368-bit ($N = 23$ 16-bit blocks) of the key can be guessed with approximated probability 0.613 and complexity $2^{59.45}$ ($D = 5$), while after the first CPA, only 336-bit ($N = 21$ 16-bit blocks) could be guessed with probability 0.62 and complexity $2^{58.95}$ ($D = 6$). It means that, after the second attack, only the first 6 key hypotheses, instead of 7, for each block of the key can

Table 5.1: Difference between probability of success and complexity of the attack after 1^{st} and 2^{nd} CPA. Data in this table is mentioned only for the most complex attack. Note, the max complexity was bounded by 2^{60}. For more information about the difference, see Fig. 5.6.

Key size	After 1^{st} CPA		After 2^{nd} CPA	
[bits]	Probability	Complexity	Probability	Complexity
368	NA	NA	0.613	$2^{59.45}$
352	NA	NA	0.626	$2^{56.89}$
336	0.619	$2^{58.95}$	0.853	$2^{56.95}$
320	0.856	2^{60}	0.941	2^{60}
304	0.866	2^{57}	0.944	2^{57}
288	0.973	$2^{57.06}$	0.973	$2^{57.06}$

be taken in order to reveal the whole key with higher probability of success and less complexity.

We can see an improvement in success rate or complexity also for other cases. For instance, when $D = 4$, 8 16-bit blocks has been improved to 18 blocks (288-bit) of the key with complexity $2^{41.79}$. Further, when $D = 6$, the probability of success was improved from cca 0.61 to 0.86. For $D = 7$ the success rate was improved from cca 0.86 to 0.94. The success rate for $D = 8$ was not improved significantly.

This result was published in (Repka, Varchola, and Drutarovsky [58]).

5.3 Countermeasure against the CPA Attack against DSA & ECDSA

This CPA needs to have the secret operand constant and to now some second operands of the multiplication. One possible countermeasure to thwart this attack, which does not need special countermeasures, such as hiding (dummy cycles, noise generator, dual-rail-logic [13]) or masking (boolean, multiplicative [2]), is to use the nonce n (the per message randomly and uniformly generated number) to mask the key as follows (Repka, Varchola, and Drutarovsky [58], Varchola et al. [78]):

$$s = n^{-1}\mathrm{Hash}(m) + k\left(n^{-1}r\right) \bmod q. \tag{5.14}$$

Before the private key is multiplied by known r, the r is multiplied by the inversion of an unknown nonce. In order to make these attacks impossible, Step 7 in the ECDSA (Alg. 4.1), must be replaced by Eq. 5.14.

The second option can be to multiply the private key by the nonce, but this would be not effective enough, because the same nonce would be used in both $n^{-1}\mathrm{Hash}(m)$ and $n^{-1}k$, and therefore if some bits of the hash value would have the same value as bits the private key, the power consumption of multiplication those blocks with the equal bits would correlate. If moreover the hash value can be chosen by adversary, the adversary can then reveal the private key, but this time across another leakage.

The cost of this countermeasure is one more multiplication by the inversion of the nonce. This countermeasure is effective since the nonce is random and not public, thus it is not known to the adversary, and this countermeasure is efficient because it costs only one more multiplication and not other special logic, such as in case of an additional masking and hiding.

5.4 Implementation of the Original McEliece PKC

This implementation was published in (Repka [52]). More about code-based cryptography and post-quantum McElice PKC can by found in our work (Repka and Cayrel [54]). For McEliece PKC like cryptosystems, we summarize security in work (Repka and Zajac [59]).

5.4.1 Related work & our contribution

We implemented the most generic original McEliece PKC proposal in order to make the PKC more available. The adjective generic has been achieved using the Number Theory Library (NTL) [69], and the generic CPU Tick Measurement Library [20]. Our implementation is called the McEliece PKC Calculator, since no parameter is fixed in this implementation, and test vectors for all the important intermediate results (for all appropriate m and t in limits of hardware and NTL) can be provided for any: encryption, decryption, or key generation. Thanks to the NTL, the Calculator is easy to understand, use, and modify, since the standard NTL functions, input, and output, are used. Therefore, if a key-pair not generated by the Calculator is desired to be used by Calculator, it is not a problem, it must be just formated accordingly. Moreover, a measurement tool for side-channel analysis has been employed, which test vectors can be also recorded for. Using this tool, timing leakage can be measured, and using the measured data, it is also possible to simulate power-consumption and electromagnetic-emanation leakages. The CPU Tick Library allows to measure CPU Ticks on different families of processors, and operation systems. To our best knowledge, this is the first such an implementation. Although, there exist several implementations, like [75, 17, 72, 12, 29], of the original, and derived, schemes on hardware, embedded, and also on a computer platform. However, parameters are fixed, no test vectors are provided, or no tool for the side channel analysis is employed. The Calculator can be used in the

PKC implementation optimization, and further McEliece/Niederreiter like PKCs properties investigation, as well as in proper key-pairs generation. The Calculator can be also used in proper parameter choice for a hardware implementation, and the leakage-measurement tool can provide information on side-channel vulnerabilities. Description of the implementation details follows.

5.4.2 Binary irreducible Goppa codes for the McEliece PKC

Goppa codes was invited by Goppa [24]. In the original McEliece PKC proposal, random instances of a binary irreducible Goppa code with maximal length are employed. These codes are proposed to be corrected by the Patterson's algebraic decoding algorithm (Alg. 5.5).

Let $\mathbb{F}_{2^m} = \mathbb{F}_2[X]/m(X)$ be the finite field, where $m(X)$ is an irreducible polynomial over $\mathbb{F}_2[X]$, and $\deg m(X) = m$.

Definition 5.4.1 (Binary Irreducible Goppa Polynomial)**.** Binary Irreducible Goppa polynomial is a monic binary irreducible polynomial $g(Z) \in \mathbb{F}_{2^m}[Z]$, where $\deg g(Z) = t$.

Definition 5.4.2 (Code Support)**.** Code support is a vector $\mathbf{\Lambda} \in \mathbb{F}_{2^m}^n$, $\mathbf{\Lambda} = (\lambda_i)_{0 \le i \le n-1}$ consisting of pairwise distinct elements $\lambda_i \in \mathbb{F}_{2^m}$, where $g(\lambda_i) \neq 0$.

Since the Goppa polynomial $g(Z)$ is irreducible, all the field elements are in the code support. Hence, the code length $n = 2^m$.

Definition 5.4.3 (Binary Irreducible Goppa Code)**.** Binary Irreducible Goppa code $\Gamma(\mathbf{\Lambda}, g)$ is a Linear Alternant code defined over $\mathbb{F}_{2^m}$, wherein the g is a binary irreducible Goppa polynomial, and the $\mathbf{\Lambda}$ is a code support. This code has parameters $[n = 2^m, k = n - mt, d = 2t + 1]$, and it is defined as follows:

$$\Gamma(\mathbf{\Lambda}, g) := \{\mathbf{c} \in \mathbb{F}_2^n : S(\mathbf{c}, Z) \equiv 0 \bmod g(Z)\}, \tag{5.15}$$

where

$$S(\mathbf{c}, Z) = \sum_{0 \le i \le n-1} \frac{c_i}{Z - \lambda_i} \tag{5.16}$$

is its syndrome polynomial.

Note, if we have $\mathbf{c} \in \Gamma(\mathbf{\Lambda}, g)$, $\mathbf{y} \in \mathbb{F}_2^n$, and $\mathbf{y} = \mathbf{c} \oplus \mathbf{e}$, where $\mathbf{e}$ is an error vector of $0 \le \mathrm{HW}(\mathbf{e}) \le t$, then $S(\mathbf{e}, Z) \equiv S(\mathbf{c}, Z) \bmod g(Z)$.

Definition 5.4.4 (Error-Locator Polynomial Definition)**.** The error-locator polynomial $\sigma(\mathbf{e}, Z)$ is defined, in binary case, as follows:

$$\sigma(\mathbf{e}, Z) = \Pi_{i=0}^{n-1}(Z - \lambda_i)^{e_i}. \tag{5.17}$$

Algorithm 5.1 McEliece PKC key generation.

Require: $m(X)$, code length n, and $t = \deg g(Z)$.
Ensure: $K_{pub} = G_{pub}$, $K_{priv} = (\Gamma(\mathbf{\Lambda}, g), S, P)$.

1: Generate uniformly a random $g(Z)$. ▷ Determines the secret $\Gamma(\mathbf{\Lambda}, g)$.
2: Find a generator matrix G_{priv} for the random secret $\Gamma(\mathbf{\Lambda}, g)$ code. ▷ Eq. 5.20, 5.21, 5.22
3: Generate uniformly a random $k \times k$ dense invertible binary matrix S.
4: Generate uniformly a random $n \times n$ binary permutation matrix P. ▷ Alg. in Alg. 5.2
5: $G_{pub} = SG_{priv}P$.
6: $K_{pub} = G_{pub}$.
7: $K_{priv} = (\Gamma(\mathbf{\Lambda}, g), S, P)$.
8: **return** K_{pub}, and K_{priv}.

The polynomial is defined over $\mathbb{F}_{2^m}[Z]$, and indexes of its roots in the code support determine error-bit positions in a codeword. In our case, roots are not multiple, and its maximum degree is t.

5.4.3 Key-pairs Generation

Private key K_{priv} consist of a random $\Gamma(\mathbf{\Lambda}, g)$ code, a random permutation matrix P, and a random dense non-singular scramble matrix S.

$$K_{priv} = (\Gamma(\mathbf{\Lambda}, g), S, P)\,. \tag{5.18}$$

The random $\Gamma(\mathbf{\Lambda}, g)$ code means that the g is chosen randomly. The public key K_{pub} is derived from the K_{priv} using P and S. The key generation algorithm is in Alg. 5.1.

$$K_{pub} = G_{pub}. \tag{5.19}$$

As the first step in the key generation phase, the Calculator picks up randomly (or it is chosen by an user) an irreducible polynomial $m(X)$ over $\mathbb{F}_2[X]$, according to that the finite field $\mathbb{F}_{2^m}$ is created. Then a binary irreducible Goppa polynomial $g(Z)$ over $\mathbb{F}_{2^m}[Z]$ is generated randomly. Probability that a random polynomial with degree t is irreducible over the $\mathbb{F}_{2^m}[Z]$ is approximately $1/t$ [42].

Now, the code support is initialized. All the elements of $\mathbb{F}_{2^m}$ are in the support. If the $m(X)$ polynomial is primitive, all elements can be generated using its roots. But it is not the case in general. Therefore, in order to initialize the code support, a generator (field primitive element) should be found. The Calculator searches for a generator using the fact that order of a subgroup divides order of the group. Order of an element that generates the field should be $n - 1$. Let we have all the factors of the

Algorithm 5.2 Random permutation of a sequence of elements.

Require: $\mathbf{p}$, a sequence of elements for permuting.
Ensure: $\mathbf{p}$ with randomly permuted elements.

```
for i = 0; i < p.length; i++ do
    index = rand() mod p.length
    swap(p[index], p[i])
end for
return p
```

integer $n-1$. The generator is found by examination degrees of all the field elements respectively. If an element is found that has the desired degree, the search stops, and the element is used to initialize the code support. We denote this element as λ_1. The first element λ_0 of the initialized code support is always 0.

Generator matrix G_{priv} is found as follows. First, an initial parity check matrix H_{init} is constructed as

$$H_{init}(i,j) = g^{-1}(\lambda_j)\lambda_j^i, \tag{5.20}$$

where $H_{init}(i,j)$ is the i-th row, and the j-th column of the H_{init}. This matrix is then used in its binary form. Therefore, each cell (element of the finite filed $\mathbb{F}_{2^m}$) is represented as the column of sequence m binary digits. Thus, the matrix in the binary form consists of mt rows and n columns. Only the binary form of the parity check matrix is considered hereafter.

The parity check matrix is brought into the reduced row-echelon form using the Gaussian elimination. If the resulted matrix is not in the systematic form, the systematic form is obtained by swapping appropriate columns. Now we have

$$H_{init} = [I|R]. \tag{5.21}$$

The parity coordinates generator matrix R^T has $(n-mt)$ rows and mt columns. Using R^T, the G_{priv} is then defined as

$$G_{priv} := [R^T|I]. \tag{5.22}$$

In order to be able to construct the secret code support $\mathbf{\Lambda}$ for the code generated by the secret G_{priv}, the permutation of elements of the initial code support must be corrected according the swaps performed in order to make the Eq. 5.21 held. For that purpose, only vector of the inverse swaps is important. The vector will be denoted as $\mathbf{b}$ hereafter. Note, the initial parity check matrix H_{init} is not a parity check matrix for the $\Gamma(\mathbf{\Lambda}, g)$ code generated by G_{priv}, since the code support correction.

The dense non-singular matrix S is generated randomly and uniformly by NTL. A random square matrix is invertible with probability approximately $1/3$. One possibility how to determine whether a square matrix is invertible is to examine its determinant

but this would be time consuming. A better approach is to test the actual diagonal element for zero during the elimination when bringing the square matrix into an upper echelon form. If the element is zero the matrix is not invertible. For further optimization, inner instruction parallelism can be used (Zajac and Jokay [81]).

The permutation matrix P is generated randomly and uniformly using the algorithm shown in Alg. 5.2. This algorithm assume a vector $\mathbf{p}$ which is somehow initialized. Hereafter, we consider $\mathbf{p}$ as vector of randomly and without replacement generated integers (a random permutation) that represents the secret permutation matrix P.

5.4.4 Key-pairs Storing

In the Calculator implementation, almost nothing is fixed, even polynomial $m(X)$ is chosen randomly, or can be chosen by user. For the private key K_{priv} reconstruction: the $m(X)$, finite field generator element λ_1, vector of inverse swaps $\mathbf{b}$, permutation vector $\mathbf{p}$, matrix S, and, finally, $g(Z)$, are stored. In case of $m = 11$, $t = 50$, it is $4\ 510\ 452$ bytes.

In order to reconstruct the corresponding public key $K_{pub} = G_{pub}$, only the G_{pub} is stored. For $m = 11$, $t = 50$, it is $6\ 138\ 820$ bytes. We recommend to use any compression method in order to safe the size needed for key-pairs storing. Another possibility is to order permutations in the way that each permutation can be represented by an unique integer [65].

5.4.5 Encryption

The encryption algorithm (Fig. 5.3) is very fast and simple. It can be implemented as several XOR additions in an optimized implementation. In order to generate uniformly a random secret error vector of hamming weight t and length n, the Calculator implementation uses the algorithm listed in Alg. 5.2. From the outcome of the algorithm, only the last t indexes are considered. These indexes determines positions of ones in the error vector.

5.4.6 Decryption

The decryption algorithm is more time consuming than the encryption one. The most time consuming is the Step 2.

The Patterson's algebraic decoding algorithm (Alg. 5.5) is used in order to correct a code word with t errors in the private $\Gamma(\mathbf{\Lambda}, g)$ code. For that purpose, we assume an input binary vector $\mathbf{u} = \mathbf{y}P^{-1}$ that is a codeword in the private code with exactly t errors. Note, t is the maximum number of errors that can be corrected in a $\Gamma(\mathbf{\Lambda}, g)$ code, and also that the algorithm in the Alg. 5.5 is capable to correct.

Algorithm 5.3 McEliece PKC encryption.

Require: $K_{pub} = G_{pub}$, end message $\mathbf{a} \in \mathbb{F}_2^k$, where $k = 2^m - mt$ (the number of rows of G_{pub}).
Ensure: A ciphertext $\mathbf{y} \in \mathbb{F}_2^n$. ▷ $n = 2^m$.

1: Generate uniformly a random binary vector $\mathbf{e_x} \in \mathbb{F}_2^n$ with $\mathrm{HW}(\mathbf{e_x}) = t$. ▷ HW is Hamming weight.
2: $\mathbf{x} = \mathbf{a}G_{pub}$.
3: $\mathbf{y} = \mathbf{x} \oplus \mathbf{e_x}$.
4: **return** y.

Algorithm 5.4 McEliece PKC decryption.

Require: $K_{priv} = (\Gamma(\mathbf{\Lambda}, g), S, P)$, and a ciphertext $\mathbf{y} \in \mathbb{F}_2^n$.
Ensure: Message $\mathbf{a} \in \mathbb{F}_2^k$.

1: $\mathbf{u} = \mathbf{y}P^{-1}$. ▷ $\mathbf{u} = \mathbf{a}SG_{priv} + \mathbf{e_x}P^{-1}$, the vector $\mathbf{p}$ is used instead of P.
2: $\mathbf{e} = \texttt{Patterson}\,(\mathbf{u}, \Gamma(\mathbf{\Lambda}, g))$. ▷ $\mathbf{e} = \mathbf{e_x}P^{-1}$, Alg. 5.5.
3: $\mathbf{v} = \mathbf{u} + \mathbf{e}$. ▷ $\mathbf{v} = \mathbf{a}SG_{priv}$.
4: $\mathbf{w}$ = `GetInformationCoordinates`$(\mathbf{v})$. ▷ $\mathbf{w} = \mathbf{a}S$, the last $n - k$ coordinates of $\mathbf{v}$.
5: $\mathbf{a} = \mathbf{w}S^{-1}$.
6: **return** $\mathbf{a}$.

As the first step of the Alg. 5.5, the syndrome of the error vector is computed. One can compute the syndrome evaluating the syndrome polynomial (Eq. 5.16), but such an evaluation would be the most time consuming step in the decoding algorithm. On the other hand, such an evaluation is very useful on a memory constraint devices. In order to speed up the syndrome evaluation, following look-up table is precomputed $\forall 0 \leq i < n$:

$$\texttt{preSynTab}[i] = (Z - \lambda_i)^{-1} \bmod g(Z). \tag{5.23}$$

Next possibility how to compute the syndrome is to compute the product uH_{priv}^T, wherein the H_{priv} is a parity-check matrix of the secret $\Gamma(\mathbf{\Lambda}, g)$ code, and obtain the syndrome in this way.

The syndrome polynomial $S(\mathbf{e}, Z)$ satisifies

$$S(\mathbf{e}, Z) \equiv \frac{\sigma'(\mathbf{e}, Z)}{\sigma(\mathbf{e}, Z)} \bmod g(Z). \tag{5.24}$$

Since the error of a word is being determined, the first derivative of the error-locator polynomial consists only of all the even terms, i.e. the error-locator polynomial can be split into squares and non-squares:

$$\sigma(\mathbf{e}, Z) = \alpha^2(\mathbf{e}, Z) + Z\beta^2(\mathbf{e}, Z), \tag{5.25}$$

Algorithm 5.5 Patterson's algebraic decoding algorithm.

Require: $\mathbf{u} \in \mathbb{F}_2^n$ (a private code word with t errors), $\Gamma(\mathbf{\Lambda}, g)$.
Ensure: Error vector $\mathbf{e}$ such that $\mathbf{v} = \mathbf{u} + \mathbf{e}$, where $\mathbf{v} \in \Gamma(\mathbf{\Lambda}, g)$ is the code word.

1: $S(\mathbf{e}, Z) \equiv S(\mathbf{u}, Z) \bmod g(Z)$. ▷ Eq. 5.23.
2: $T(\mathbf{e}, Z) \equiv S^{-1}(\mathbf{e}, Z) + Z \bmod g(Z)$. ▷ EEA.
3: $\tau(\mathbf{e}, Z) \equiv \sqrt{T(\mathbf{e}, Z)} \bmod g(Z)$. ▷ Eq. 5.27
4: Find $\alpha(\mathbf{e}, Z)$ and $\beta(\mathbf{e}, Z)$ such that $\beta(\mathbf{e}, Z)\tau(\mathbf{e}, Z) \equiv \alpha(\mathbf{e}, Z) \bmod g(Z)$. ▷ EEA.
5: $\sigma(\mathbf{e}, Z) = \alpha^2(\mathbf{e}, Z) + Z\beta^2(\mathbf{e}, Z)$. ▷ Squaring.
6: Find roots of $\sigma(\mathbf{e}, Z)$. ▷ Evaluation over the $\mathbf{\Lambda}$.
7: Determine indexes of the roots in the support $\mathbf{\Lambda}$.
8: Set 1 in the determined indexes in error vector $\mathbf{e}$.
9: **return** $\mathbf{e}$.

where $\beta^2(\mathbf{e}, Z) = \sigma'(\mathbf{e}, Z)$. After few modifications, the Key Equation can be obtained as:

$$\beta(\mathbf{e}, Z)\sqrt{S^{-1}(\mathbf{e}, Z) + Z} \equiv \alpha(\mathbf{e}, Z) \bmod g(Z). \tag{5.26}$$

Therefore, in the Step 3 of the Alg. 5.5, the square-root modulo $g(Z)$ is computed. Let us denote the term $S^{-1}(\mathbf{e}, Z) + Z$ as $T(\mathbf{e}, Z)$. The Calculator implementation uses the fact of the perfect square, and thus

$$\tau(\mathbf{e}, Z) = \sqrt{T(\mathbf{e}, Z)} \equiv T^{2^{tm-1}}(\mathbf{e}, Z) \bmod g(Z). \tag{5.27}$$

Such an approach can be very useful on memory constraint devices, but on the other hand it is very time consuming operation. Another possibility how to compute that square-root is to use precomputed look-up table, which consist of $\tau_i(Z)$ such that $\tau_i^2(Z) \equiv Z^i \bmod g(Z)$ for $0 \leq i < t$.

Subsequently, the key equation is solved using the Extended Euclidean Algorithm (EEA) that stops when $\deg \alpha_j(Z) \leq \lfloor (t+1)/2 - 1 \rfloor \leq t/2$, where j is the EEA iteration number.

At the time the error-locator polynomial $\sigma(\mathbf{e}, Z)$ is computed, roots of the error-locator polynomial shall be found. The Calculator implementation simply evaluate the $\sigma(\mathbf{e}, Z)$ over the secret code support $\mathbf{\Lambda}$. Therefore, Steps 6, 7, 8 are conducted in one loop. This method is also time consuming, and as an alternative, any other factorization method can be employed. Thus, the decoding algorithm yields the error vector $\mathbf{e}$ in the private code.

When the secret error vector $\mathbf{e}$ is removed (Step 3 in Alg. 5.4), only information coordinates are addressed, and the scrambling matrix is removed. Finally, the decrypted message is obtained.

5.4.7 Basic Use Cases

The Calculator can be used as a basic cryptosystem, for key-pairs generation, encryption, as well as for decryption. It is very important to note that the original McEliece PKC is vulnerable to (adaptive) chosen-ciphertext. Therefore, the Calculator can be used for encryption and decryption only if it is plugged into an IND-CCA2-Secure conversion, like the γ conversion defined in [34].

Essentially, the first main purpose of the Calculator development was the PKC implementation optimization for an FPGA. Test vectors have been used in order to chose particular PKC parameters that we have fixed for the implementation in FPGA. Afterwards, test vectors have been used for the FPGA implementation validation. Further, test vectors can be used in the further McEliece like PKCs properties investigation because all important intermediate results are recorded. The intermediated results can be used in order to verify stated hypotheses, or jut to trace behavior. For the PKC properties investigation, also the information recorded by the side-channel-leakage measurement tool can be used. The test vectors recording can be turned-on appending any command by a file name for the test vectors file. Test vectors are formatted using the standard NTL output.

More details about this McEiece PKC implementation can be found in (Repka [52]).

5.5 Timing Fault Injection Analysis of McEliece PKC Decryption

The side-channel-leakage measurement tool records Indicators (Tab. 4.1) measured in order to preform an attack, and information about secret (Tab. 4.3 and 4.4) used to compute success rate of an attack. Secret error vector, secret permutation, and secret Goppa polynomial, respectively can be guessed using the measured data. Also power-consumption and electromagnetic-emanation leakages can be simulated. Using this tool, particular keys, and proposed countermeasures can be tested. Thanks to the NTL, source codes are easy to read, and it is possible to replace a measured operation for a designer's one. Not only computation time is measured by the tool, also degrees and hamming weights of polynomials processed are recored.

The Step 3 (square root) is the most time consuming step in the Calculator's implementation of the Patterson's decoding algorithm. The Step 3 can be sped up using our proposal to compute p-th roots in finite fields of characteristic $p \leq 2$ in Sec. 5.8. If the syndrome computation was implemented as the polynomial evaluation, that would be the most time consuming step. Steps the computation time depends on $\mathrm{HW}(\mathbf{e})$ can be clearly observed in the Fig. 5.7 and 5.8 respectively. As can be seen from these figures, also HW and $\deg$ of polynomials are recorded.

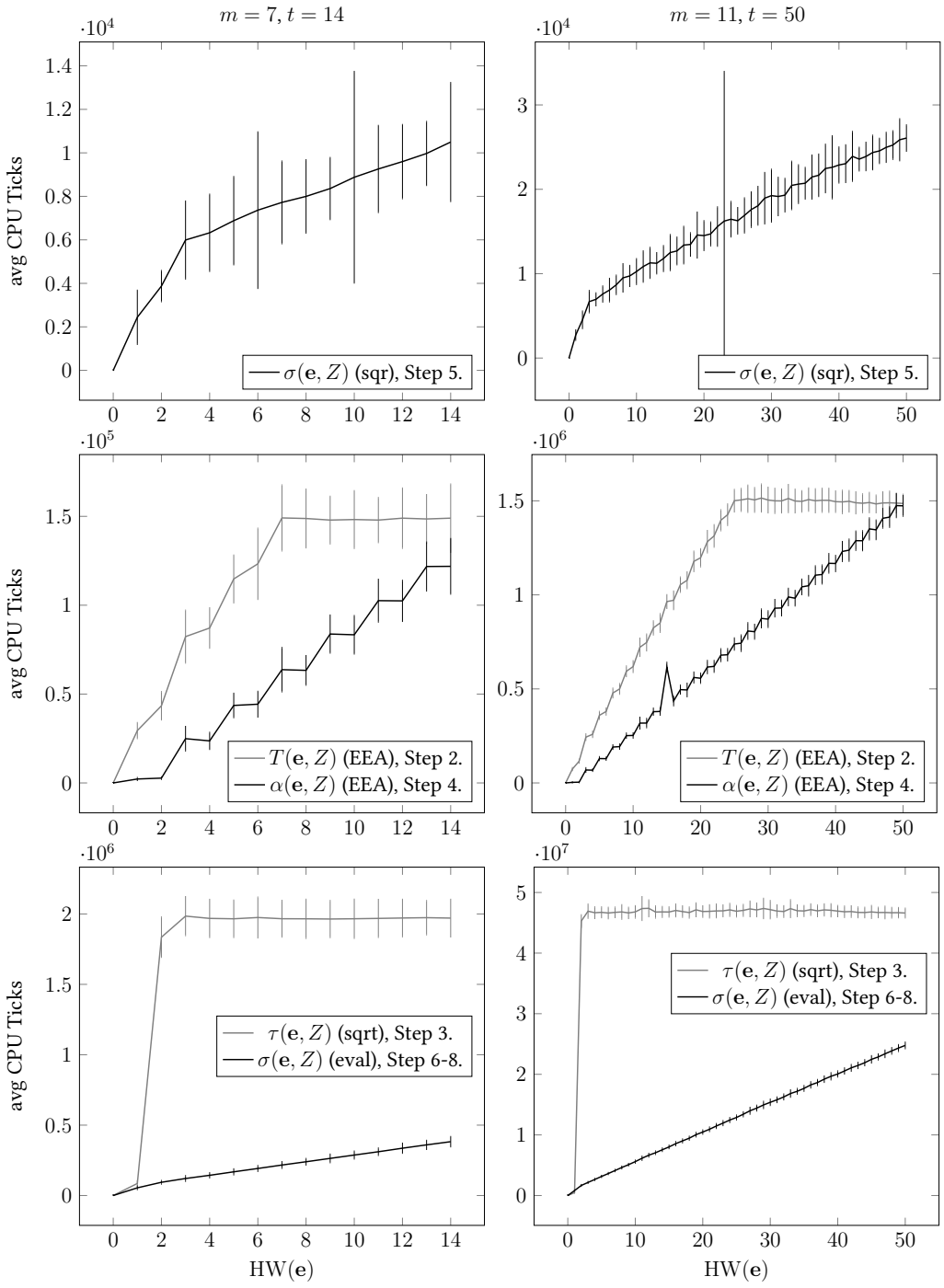

Figure 5.7: Timing analysis of Petterson's decoding algebraic algorithm (Alg. 5.5). One random key pair and 1000 random messages for m, t McEliece PKC.

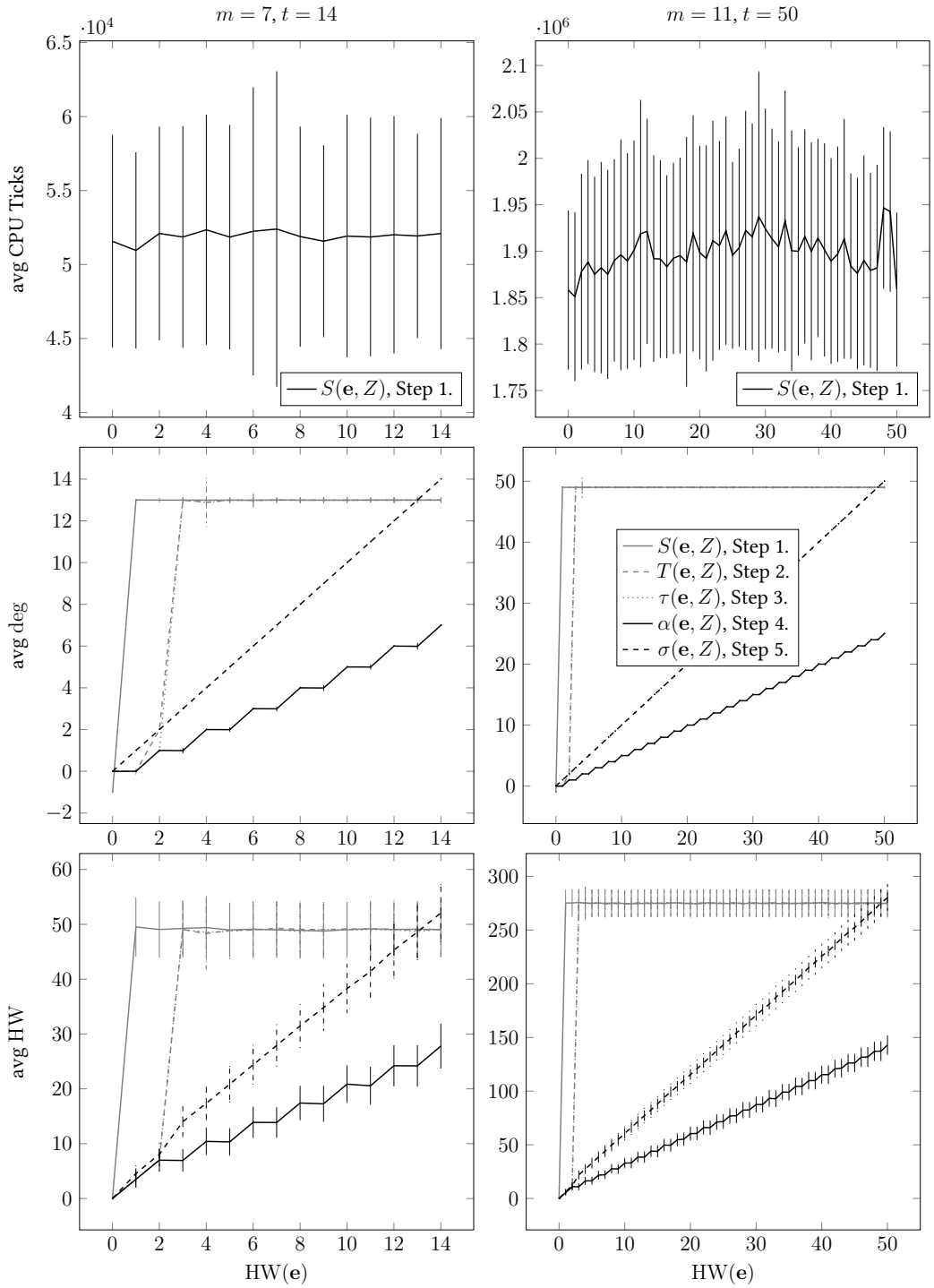

Figure 5.8: Average CPU Ticks, Degrees, and HW of polynomials processed for the same instances of McEliece PKC as in Fig. 5.7.

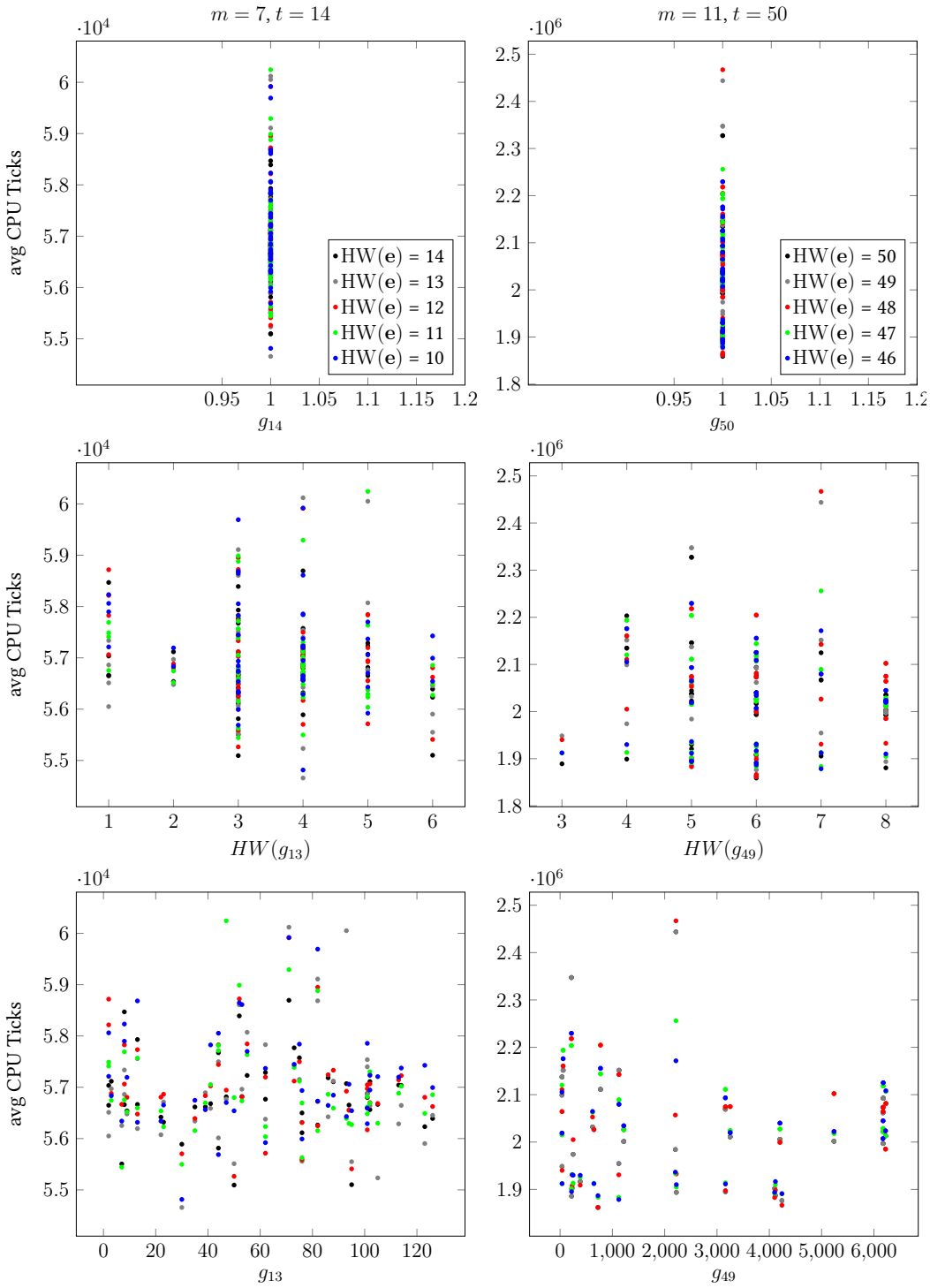

Figure 5.9: Plot for $S(\mathbf{e}, Z)$ (inv), Step 1 (Alg. 5.5), measurements of type II (Sec. 4.4.2). Data are plotted only for the last and the penultimate coefficients of 46 goppa polynomials (random key pairs) each used with 1000 random messages.

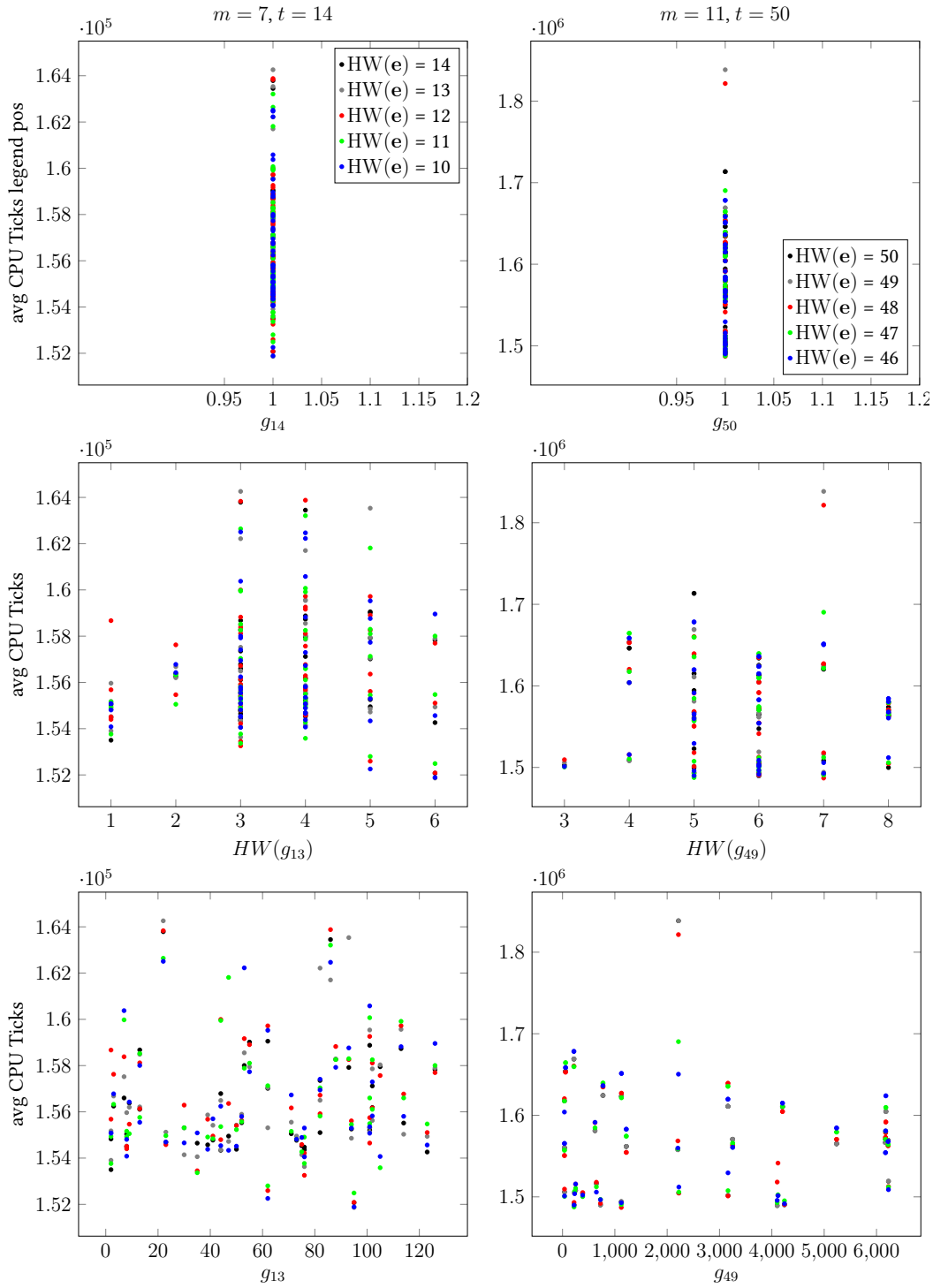

Figure 5.10: Plot for $T(\mathbf{e}, Z)$ (EEA), Step 2 (Alg. 5.5), measurements of type II (Sec. 4.4.2). Data are plotted only for the last and the penultimate coefficients of 46 goppa polynomials (random key pairs) each used with 1000 random messages.

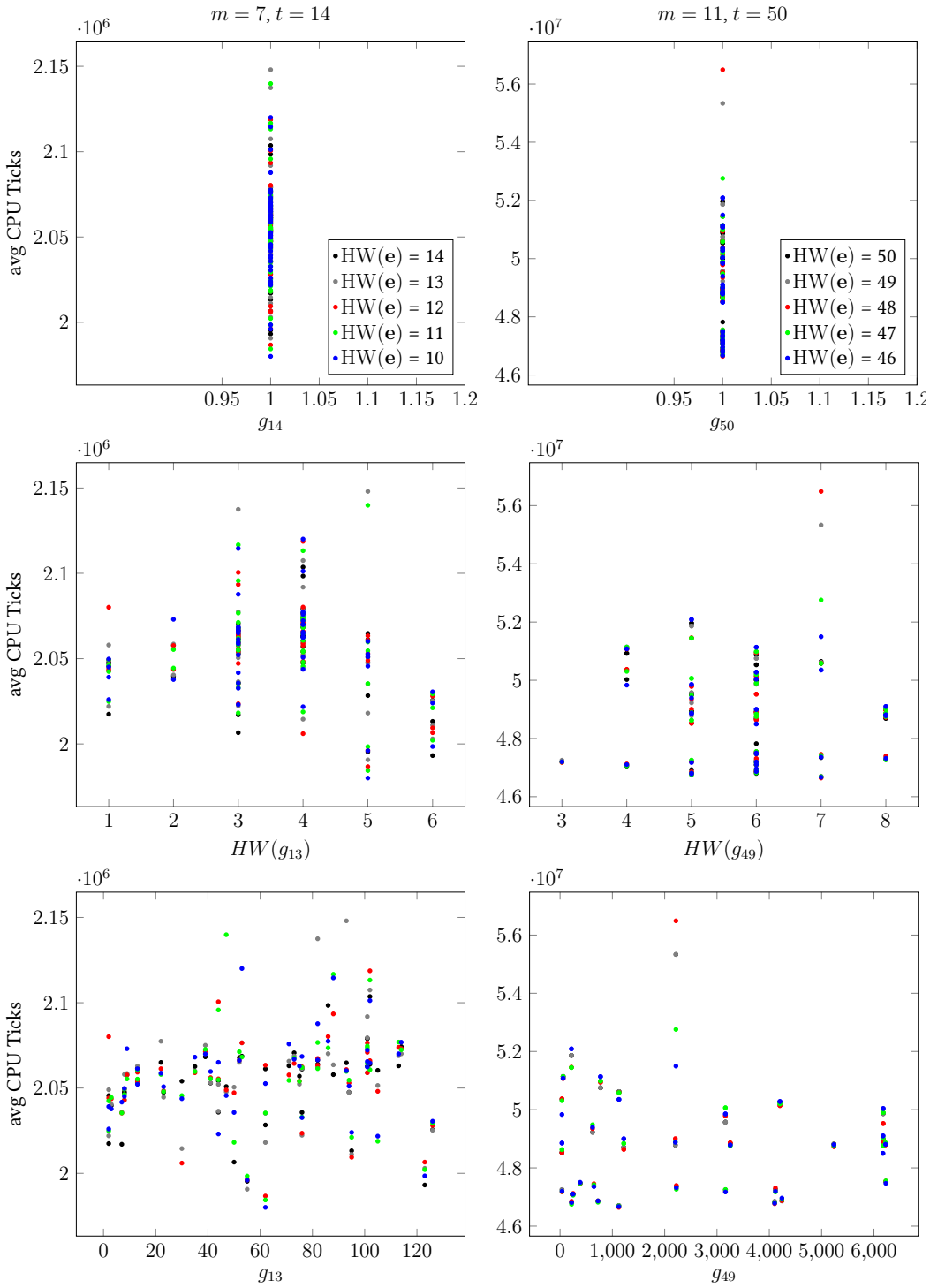

Figure 5.11: Plot for $\tau(\mathbf{e}, Z)$ (sqrt), Step 3 (Alg. 5.5), measurements of type II (Sec. 4.4.2). Data are plotted only for the last and the penultimate coefficients of 46 goppa polynomials (random key pairs) each used with 1000 random messages.

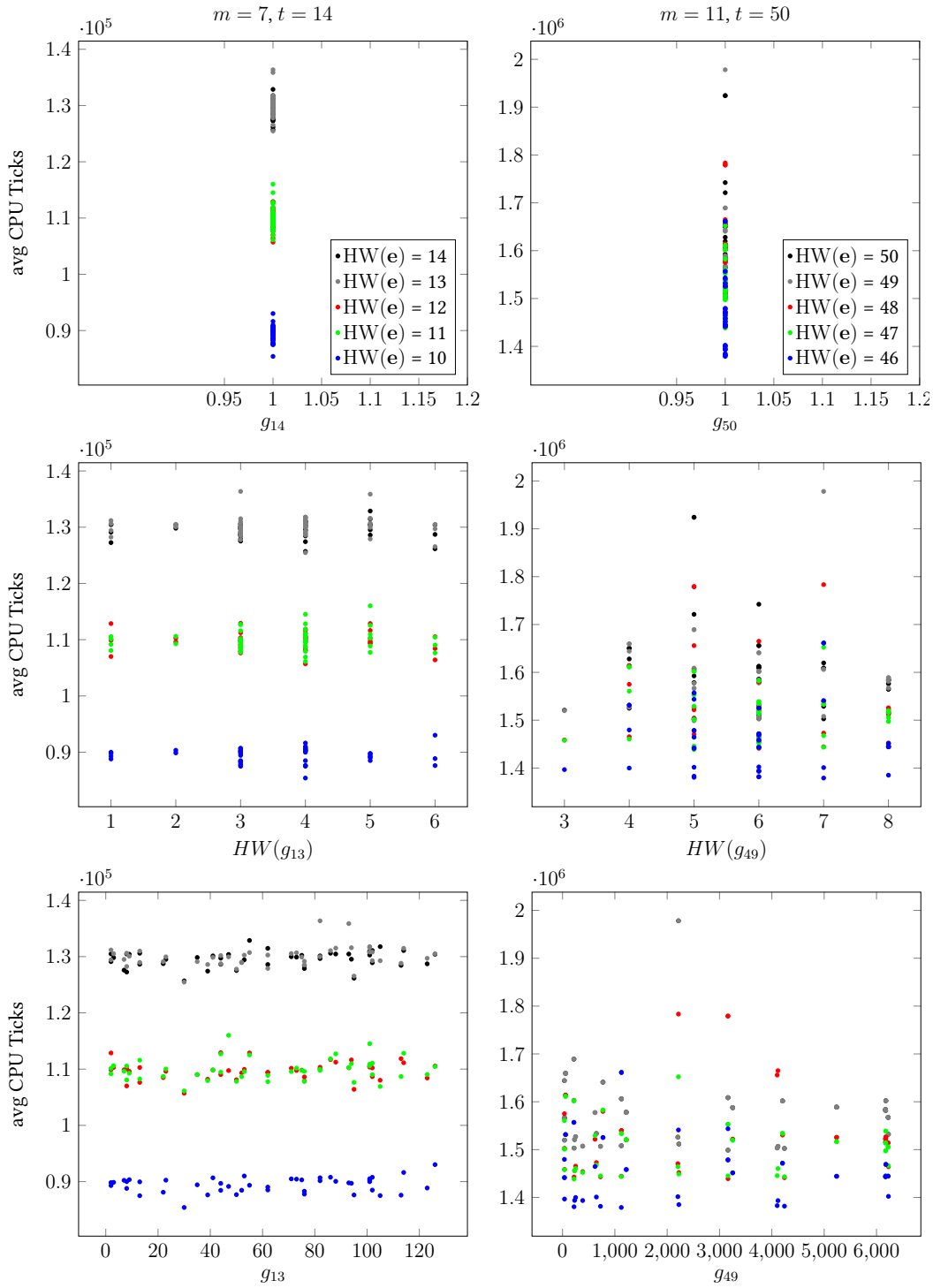

Figure 5.12: Plot for $\alpha(\mathrm{e}, Z)$ (EEA), Step 4 (Alg. 5.5), measurements of type II (Sec. 4.4.2). Data are plotted only for the last and the penultimate coefficients of 46 goppa polynomials (random key pairs) each used with 1000 random messages.

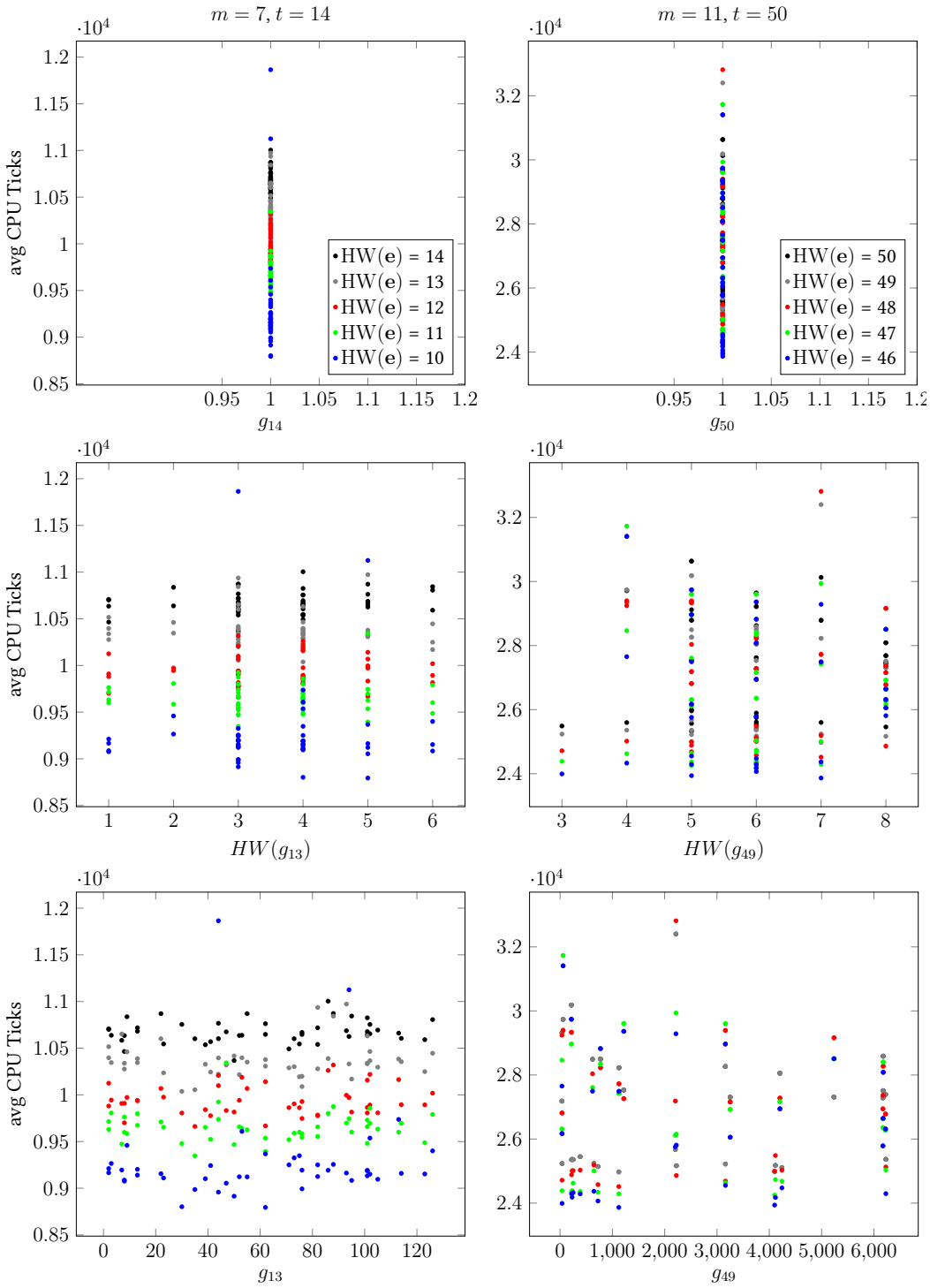

Figure 5.13: Plot for $\sigma(\mathbf{e}, Z)$ (sqr), Step 5 (Alg. 5.5), measurements of type II (Sec. 4.4.2). Data are plotted only for the last and the penultimate coefficients of 46 goppa polynomials (random key pairs) each used with 1000 random messages.

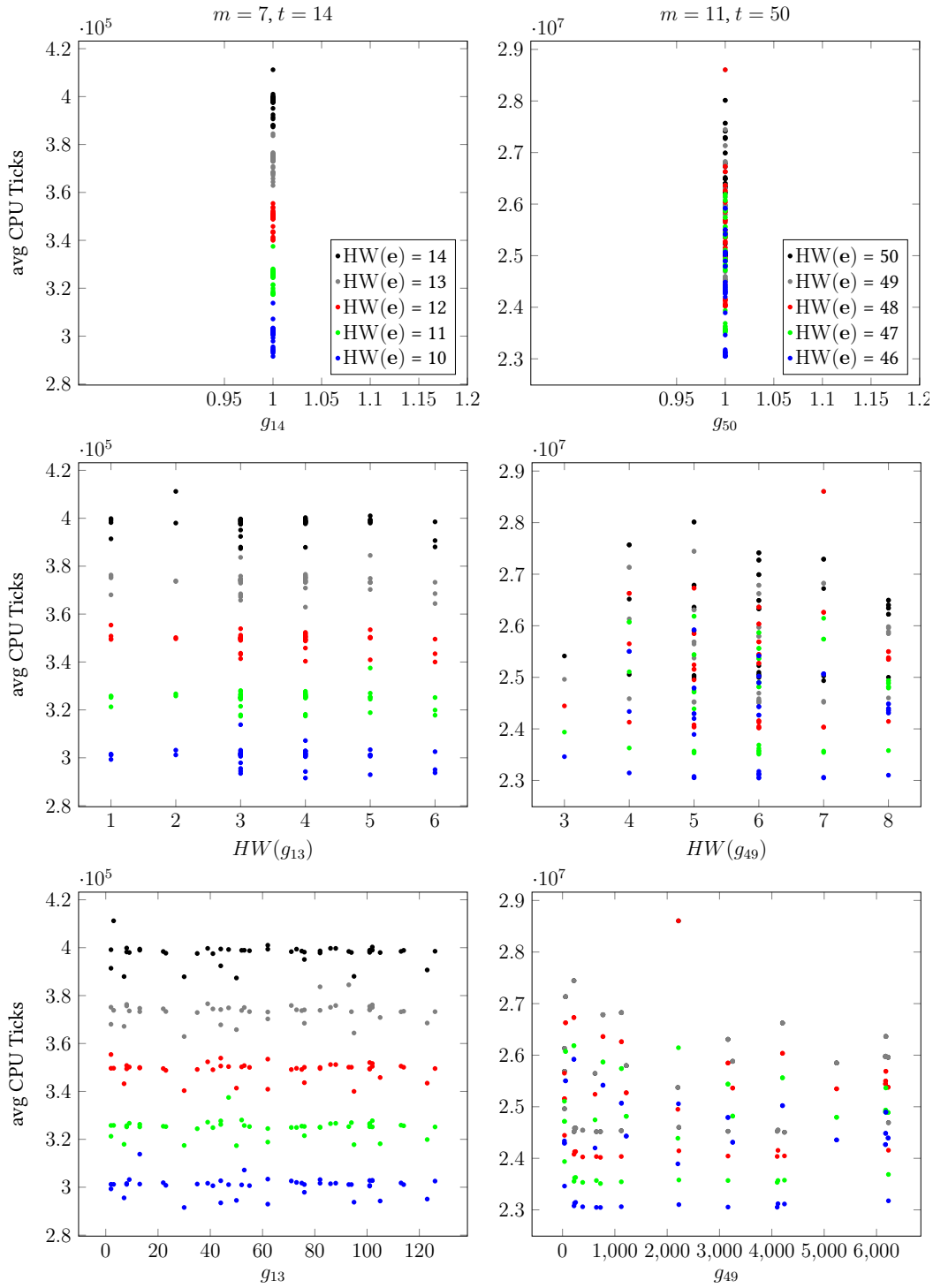

Figure 5.14: Plot for $\sigma(\mathbf{e}, Z)$ (eval), Step 6-8 (Alg. 5.5), measurements of type II (Sec. 4.4.2). Data are plotted only for the last and the penultimate coefficients of 46 goppa polynomials (random key pairs) each used with 1000 random messages.

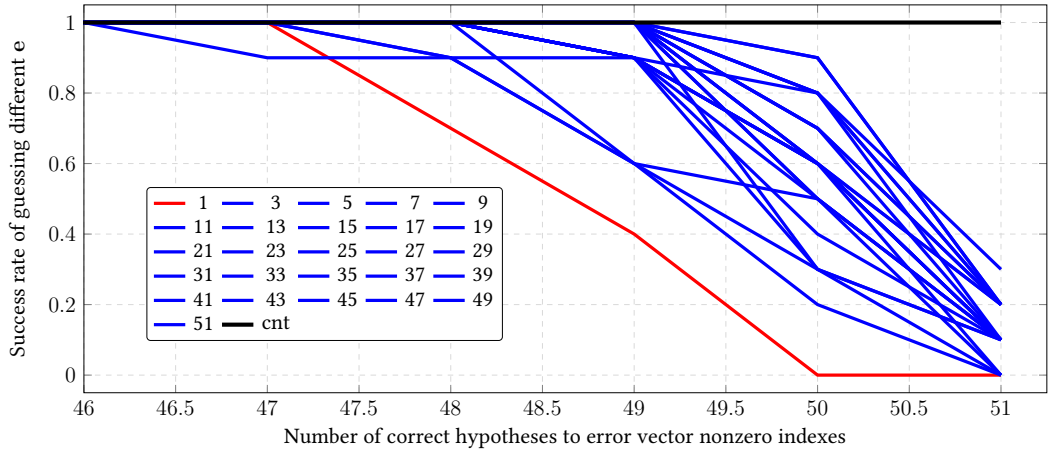

Figure 5.15: Plot for success rate of error vector guessing for McEliece PKC with $m = 11$, $t = 51$. The red line (1) is for guessing error vectors using Alg. 4.2. The next blue lines (3-51) are for guessing as well but taking error guesses of previous attacks (Tab. 5.2). The black line (cnt) is for guessing regarding number of occurrences at index in the first t position from all the performed attacks.

5.6 Improving the Timing Fault Injection Analysis

This improvement is demonstrated on the guessing the error vector using Alg. 4.2, focused on the EEA solving the key equation (Step 4 in Alg. 5.5). Plots for computation time regarding the hamming weight of error vector can be found in Fig. 5.7. That plots show which steps of the decoding algorithm have got linearly dependent computation time on the hamming weight of the error vector. Regarding the timing information hypotheses to the error vector can be constructed. The measured probability a hypothesis is correct for the case $m = 11$, $t = 51$ shows the red line in Fig. 5.15. The improvement is to use counts of hypotheses occurrences at lower positions than $t + 1$, instead of to directly take timing information to order the hypotheses. The counts are counted from the whole attack vector (Tab. 5.2).

5.7 Countermeasure against the Timing Fault Injection Analysis

The vulnerability of the Patterson's algebraic decoding algorithm (Alg. 5.5) in this issue is that computation time of the Steps 4, 5, and (6, 7, 8) of the Alg. 5.5 are linearly dependent enough to guess the HW of the error vector. Therefor, there is a possibility to guess number of iterations of these steps. This leakage can be exploited (Alg. 4.2) to guess value of error vectors in order to reveal messages and sometimes also the secret

# of e bit positions	1	3	5	7	9	11	13	15	17	19	21	23	25
# of repetitions	1	10	10	10	10	10	5	5	5	5	5	1	3
# of e bit positions	27	29	31	33	35	37	39	41	43	45	47	49	51
# of repetitions	3	3	3	3	3	3	3	3	3	3	3	3	2

Table 5.2: Attack vector for plot in Fig. 5.15. For instance, 1:1 means guessing error bit position 1 time using Alg. 4.2; 3:10 means that guessing of error bit position was performed using 2 guesses with the lowest time from the previous attack, repeated 10 times; 5:10 means that guessing of error bit position was performed using 4 guesses with the lowest time from the previous attack, repeated 10 times.

permutation regarding the McEliece PKC implementation. The countermeasure that would be effective in this case is to make the computation of these steps of constant time. For this purpose, one can use dummy cycles processing random data. Further, the number of iterations of the EEA and also the degrees of polynomials, intermediate results, such as syndrome, $T(\mathbf{e}, Z)$, and error-locator polynomial, should be checked in order to detect this kind of attacks.

5.8 Computing p^{th} roots in extended finite fields of characteristic $p \geq 2$

This part shows how to directly compute p^{th} roots efficiently and scalable in extended finite fields of characteristic $p \geq 2$, wherein the reduction polynomial can be even random without constrains. This method was published in (Repka [49]).

Let $\mathbb{GF}(p^m) := \mathbb{GF}(p)[X]/m(X)$ be the finite subfield, where $m(X)$ is an irreducible polynomial of $\deg m(X) = m > 0$, and p is a prime ≥ 2.

Let $g(Z)$, $\deg g(Z) = t > 0$, be an irreducible polynomial in $\mathbb{GF}(p^m)[Z]$, then the extended finite field is defined as $\mathbb{GF}(p^{mt}) := \mathbb{GF}(p^m)[Z]/g(Z)$.

We deal with computation of p^{th} root $r(Z)$ of a polynomial $n(Z) \in \mathbb{GF}(p^m)[Z]$, $0 \leq \deg n(Z) \leq t-1$, modulo the $g(Z)$:

$$r(Z) \equiv \sqrt[p]{n(Z)} \bmod g(Z). \tag{5.28}$$

5.8.1 Related work & our contribution

The p^{th} root computation is important in many applications, like in coding theory [46, 37] and cryptography [29, 68, 7]. Thus, there exist alternatives how to compute the p^{th} root, however particularly for $p = 2$, like to exploit perfect square, or to utilize inversion of a squaring matrix, as well as to exploit discrete logarithm of a primitive element [52, 16, 67, 28, 33, 15, 73]). In some cases [45] can be used.

5.8.2 The computation of p^{th} root

The p^{th} root (5.39) is split to the Left and Right parts, vectors, in the way the Left part contains terms which do not have to be reduced, and the Right part contains terms that must be reduced, as follows:

$$\mathbf{L} := (L_j)_{0 \le j < t_1}, \tag{5.29}$$

$$t_1 := \left\lceil \frac{t}{p} \right\rceil, \tag{5.30}$$

$$L_j := jp, \tag{5.31}$$

and

$$\mathbf{R} := (R_k)_{0 \le k < t_2}, \tag{5.32}$$

$$t_2 := t - t_1. \tag{5.33}$$

For $k : 0 \le k \le p-2, k < t_2$:

$$R_k := (k+1)\pi, \tag{5.34}$$

where

$$\pi := p^{mt-1} \equiv p^{-1} \pmod{p^{mt}-1}, \tag{5.35}$$

$$c_{R_k}(Z) \equiv Z^{R_k} \bmod g(Z), \tag{5.36}$$

and for $k : (p-1) \le k < t_2$:

$$R_k := R_{k-(p-1)} + 1. \tag{5.37}$$

$$c_{R_k}(Z) \equiv c_{R_{k-(p-1)}}(Z) Z \bmod g(Z). \tag{5.38}$$

The p^{th} root can then be written as:

$$\sqrt[p]{n(Z)} = \bigoplus_{j=0}^{t_1-1} Z^j \sqrt[p]{n_{L_j}} \oplus \bigoplus_{k=0}^{t_2-1} Z^{R_k} \sqrt[p]{n_{\lambda(k)}}, \tag{5.39}$$

where

$$\lambda(k) := k + 1 + \left\lfloor \frac{k}{p-1} \right\rfloor. \tag{5.40}$$

The carry terms which must be reduced, the terms in the Right part of the p^{th} root in (5.39), are substituted to the corresponding carry polynomials in (5.36) and (5.38) respectively.
Therefore, the p^{th} root modulo $g(Z)$ in (5.28) can be written as:

$$r(Z) = \bigoplus_{j=0}^{t_1-1} Z^j n_{L_j}^{p^{-1}} \oplus \bigoplus_{k=0}^{t_2-1} c_{R_k}(Z) n_{\lambda(k)}^{p^{-1}}. \tag{5.41}$$

The products are then added together, such as in (5.41) and (5.42). Multiplications, additions, and p^{th} roots are operations in the subfield.

$$\begin{array}{c} \\ Z^0 \\ \vdots \\ \\ Z^j \\ \vdots \\ Z^{t_1-1} \\ \vdots \\ Z^{t-1} \end{array} \begin{array}{c} \begin{array}{cccc} Z^{R_0} & Z^{R_1} & \cdots & Z^{R_{t_2-1}} \end{array} \\ \begin{pmatrix} & & & \\ & & & \\ & & & \\ \mathbf{c_{R_0}} & \mathbf{c_{R_1}} & \cdots & \mathbf{c_{R_{t_2-1}}} \\ & & & \\ & & & \\ & & & \end{pmatrix} \end{array} \begin{pmatrix} n_{\lambda(0)}^{p^{-1}} \\ n_{\lambda(1)}^{p^{-1}} \\ \vdots \\ n_{\lambda(t_2-1)}^{p^{-1}} \end{pmatrix} \oplus \begin{pmatrix} n_{L_0}^{p^{-1}} \\ \vdots \\ n_{L_j}^{p^{-1}} \\ \vdots \\ n_{L_{t_1-1}}^{p^{-1}} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{r} \tag{5.42}$$

where $\mathbf{c_{R_k}}$ are column vectors of coefficients of the corresponding carry polynomials $c_{R_k}(Z)$.

5.8.3 Summary

Direct p^{th} root computation in extended finite fields of a prime characteristic $p \geq 2$ with reduction polynomial without constrains was shown. The matrix in (5.42) has t rows and t_2 columns, see (5.30) and (5.33). Carry polynomials (5.38) for (5.37) are computed very quickly based on (5.36). This method is very efficient, scalable and can be parallelized well, and it was published in (Repka [49]). Some notes on modular reduction in extended finite fields can be found in the work (Repka [53]).

Chapter 6

List of Publications & Contributions

In this chapter, chosen contributions of the author are listed. The categorization of the contributions is made regarding the criteria to evaluate level of research, development, art, and other creative activities in the complex accreditation of the high school.

6.1 Zoznam príspevkov kategórie A

6.1.1 Current Contents

[49] M. Repka. "Computing pth roots in extended finite fields of prime characteristic p >= 2". In: *Electronics Letters* 52.9 (Apr. 2016), pp. 718–719. ISSN: 0013-5194. DOI: 10.1049/el.2015.4141.

6.1.2 Science Citation Index Expanded

[52] Marek Repka. "McEliece PKC Calculator". In: *Journal of Electrical Engineering* 65.6 (2014), pp. 333–341.

[53] Marek Repka. "Note on modular reduction in extended finite fields and polynomial rings for simple hardware". In: *Journal of Electrical Engineering* 67.1 (2016), pp. 56–60.

[58] Marek Repka, Michal Varchola, and Miloš Drutarovsky. "Improving CPA against DSA and ECDSA". In: *Journal of Electrical Engineering* 66.3 (2015), pp. 159–163.

6.1.3 Pozvané články

[79] Michal Varchola, Miloš Drutarovský, and Marek Repka. "Robust FPGA based True Random Number Generator utilizing Oscillatory Metastability in Transition Effect Ring Oscillators - Invited Paper". In: *Proceedings of the 9th International Conference on Circuits, Systems, Signal and Telecommunications (CSST'15), Dubai, United Arab Emirates, February 22-24, 2015*. Ed. by Nikos E. Mastorakis and Zoran Bojkovic. WSEAS press, 2015, pp. 90–98.

6.2 List of Contributions of Category B

[21] Lubos Gaspar et al. "Cryptoprocessor with Native Resistance against Side Channel and Fault Injection Attacks". In: *Proceedings of the 13th International Conference on Telecommunications and Informatics (TELE-INFO'14), Istanbul, Turkey, December 15-17, 2014*. Ed. by Nikos E. Mastorakis et al. WSEAS press, 2014, pp. 88–97.

[54] Marek Repka and Pierre-Louis Cayrel. In: *Multidisciplinary Perspectives in Cryptology and Information Security*. Ed. by Sattar B. Sadkhan Al Maliky and Nidaa A. Abaas. IGI Global, 2014. Chap. Cryptography Based on Error Correcting Codes: A Survey, pp. 133–156.

[57] Marek Repka and Michal Varchola. "Correlation Power Analysis using Measured and Simulated Power Traces based on Hamming Distance Power Model – Attacking 16-bit Integer Multiplier in FPGA". In: *International Journal of Computer Network and Information Security(IJCNIS)* 7.6 (2014), pp. 10–16.

[59] Marek Repka and Pavol Zajac. "Overview of the McEliece Cyptosystem and its Security". In: *Tatra Mt. Math. Publ* 60.3 (2014), pp. 57–83.

[78] M. Varchola et al. "Side Channel Attack on Multiprecision Multiplier Used in Protected ECDSA Implementation". In: *2015 International Conference on ReConFigurable Computing and FPGAs, ReConFig15, Cancun, Mexico, December 7-9, 2015*. IEEE.

6.3 List of Talks at Conferences

[55] Marek Repka, Lubos Gaspar, and Viktor Fischer. "Correlation Power Analysis of an AES Implementation in FPGA". In: *TatraCrypt 2012: Book of abstracts*. 2012.

[60] Marek Repka et al. "Hamming Distance Power Model Analysis of AES at Architecture Level". In: *COSADE 2012: 3rd International Workshop on Constructive Side-Channel Analysis and Secure Design. Darmstadt, Germany, May 3-4, 2012. Darmstadt, Technische Universität Darmstadt.*

Chapter 7

Conclusion

Hardware is more challenging to attack using CPA and protect against the CPA than software. We can say that ASIC is more challenging to attack and protect than FGPA and processors respectively. Attacks like DPA [35], CPA [9], DEMA [47] or CEMA [14] are the most common side channel attacks. They require an appropriate description of the data-dependent power consumption or electromagnetic emanation. The most common one is HDPM, and, secondly, it is HWPM. HDPM is more complex –but also more fitting– than HWPM. In work (Repka et al. [60]) it is shown that by contemplating a cryptographic algorithm RTL regarding SCAs, some leakages can be suppressed or even eliminated. This zero-cost countermeasure is presented on various AES-128 RTL architectures with 128-bit data path and, thus, 16 S-boxes conducted parallel. CPA of the most vulnerable AES-128 RTL architecture was presented in (Repka, Gaspar, and Fischer [55]). Countermeasures against this kind of attacks are investigated in our work (Gaspar et al. [21]).

The Goal 2, which is defined in Sec. 3.2, was met by implementing the original McEliece PKC (Sec. 5.4). This implementation was published in article (Repka [52]). It can provide test vectors for all important intermediate results. The side-channel leakage measurement tool (Sec. 4.4) is embedded in the implementation. The tool was demonstrated in (Repka [51]), and in Sec. 5.5 and 5.6 where timing fault injection analysis is performed and finally improved. Countermeasure was discussed in Sec. 5.7. More about code-based cryptography and post-quantum McElice PKC can by found in (Repka and Cayrel [54]). Summarization of McEliece like PKCs security can be found in (Repka and Zajac [59]).

In order to meet the Goal 1 defined in Sec. 3.1, an application for power consumption measurements and analyses was made. This application is able to manage the whole attack analysis process (Sec. 4.3). It remotely configure oscilloscopes (Sec. 4.5), manages the cryptographic devices (Sec. 4.2), and provides data for further analyses. Detailed description of the application is in Sec. 4.4. Afterwards, the analysis of

the CPA attack against (EC)DSA was performed (Repka, Tomeček, and Varchola [56]; Sec. 5.1). Possible errors of success rate and complexity approximations according to simulated CPA was investigated in (Repka and Varchola [57]; Sec. 5.1.4). Finally, the Goal 1 was met by improving the CPA attack against (EC)DSA (Repka, Varchola, and Drutarovsky [58]; Sec. 5.2; Fig. 5.6; Tab. 5.1). The improvement was demonstrated on 16-bit integer multiplier with one constant secret operand, implemented in FPGA. First CPA is performed to order key hypotheses from the most fitting to the worst fitting the reality. Afterwards, second CPA is used to reorder the first 10 hypotheses which came from the first CPA. Both CPAs use HDPM, however while, in the first CPA, HDPM of the whole multiplication result is considered to order all the possible key hypotheses, for the second CPA, HDPM of the first half least significant bits is made to reorder the first 10 possible key hypothesis ordered according to the first CPA. Thanks to the improvement, the CPA is more successful, more blocks of key can be guessed, and errors of simulated CPA are eliminated. Hence, the CPA attack can be simulated in order to approximate its success rate and complexity after the second CPA. This improvement is valuable to use especially when ECDSA is implemented on 32-bit or wider platforms. Such an attack can be thanks to the improvement simulated with negligible errors. An effective and efficient countermeasure is discussed in Sec. 5.3. Demonstration of this improvement, such as discussion of the countermeasure, were published in the work (Repka, Varchola, and Drutarovsky [58]). The countermeasure is discussed more deeply in (Varchola et al. [78]).

Finally, an efficient method for p-th roots computations in extended finite fields of characteristic p was invented (Sec. 5.8).

Chapter 8

Resumé

Kapitola 1 uvádza motiváciu práce a dôvody potreby skúmania útokov postrannými kanálmi a protiopatrení proti útokom postrannými kanálmi. Súčasný stav poznania útokov postrannými kanálmi a protiopatrení voči postranným kanálom je zmapovaný v Kapitole 2. Nachádza sa tu zoznam postranných kanálov, delenie postranných kanálov, popis krokov útoku postrannými kanálmi, niekoľko príkladov útokov postrannými kanálmi, ako aj delenie opatrení proti útokom postrannými kanálmi. Ciele dizertačnej práce je možné nájsť v Kapitole 3. Počas realizácie popisovaných experimentov a dosahovaní požadovaných výsledkov boli analyzované symetrické ako aj asymetrické kryptosystémy, a to AES, ECDSA a McEliece PKC. Metodológia je podrobne popísaná v Kapitole 4.

AES bol implementovaný v dvoch analyzovaných laboratórnych kryptografických zariadeniach. V μkontrolery PIC18F2520 a v FPGA ACTEL FUSION M7AFS600, ktoré sa líšili technológiou ako aj prípadmi použitia. V μkontroler PIC18F2520 bol implementovaný AES-128 so sekvenčným spracovávaním stavu AES po 8 bitoch, pretože je to 8-bitový μkontroler. Útok na tento μkontroler s AES-128 bol prezentovaný v (Repka [50]). V FPGA ACTEL FUSION M7AFS600 bol implementovaný AES-128 so 128-bitovou dátovou cestou. Celý stav AES-128 bol spracovávaný paralelne. ECDSA bol analyzovaný v FPGA ALTERA Cyclone III, presnejšie bola implementovaná len časť ECDSA, ktorá bola podrobená útokom. Bola to 16-bitová celočíselná násobička. McEliece PKC s nástrojom na meranie rôznych únikov postrannej informácie bol implementovaný softvérovo na operačnom systéme Linux a Windows v jazyku C++ za použitia knižnice NTL, pričom experimenty boli realizované na 64-bitovej platforme.

Uvedené kryptografické zariadenia implementácie AES a ECDSA boli podrobené skúmaniu útokov korelačnou analýzou spotreby a McEliece PKC bol podrobený skúmaniu útokov analýzou doby výpočtu v spojení s vnášaním chýb do výpočtu. Na vykonanie analýz boli vyvinuté aplikácie v programovacom jazyku C++ na meranie elektrickej spotreby, elektromagnetického vyžarovania a meranie doby výpočtu (Časť 4.4).

Taktiež boli vyvinuté aplikácie na vykonávania samotných útokov v C++ a na vykonanie analýz boli vyvinuté aplikácie v C++, MATLAB a R. Aplikácie boli vyvinuté s podporou viac jadrových procesorov, čo zrýchlilo niekoľko násobne realizované výpočty. V Časti 4.3 Kapitoly 4 je uvedená topológia útokov a analýz.

Pri zrealizovaných analýzach a útokoch boli použité tri rôzne meracie zariadenia pre rôzne meracie body (Časť 4.5). Bol použitý osciloskop LeCroy WavePro 7200A s 8-bitobým A/D prevodníkom s 2GHz šírkou pásma a 20GS/s vzorkovaciu frekvenciou, ktorý bol použitý na meranie spotreby na zemi μkontroler PIC18F2520 pomocou pasívnej napäťovej sondy. Druhým meracím zariadeným bol osciloskop LeCroy WavePro 740Zi s 8-bitovým A/D prevodníkom, 4GHz šírkou pásma a 40GS/s vzorkovaciu frekvenciou, ktorý bol použitý na meranie spotreby FPGA ACTEL FUSION M7AFS600 na rezistore umiestnenom na napájacej vetve jadra FPGA pomocou aktívnej diferenciálnej napäťovej sondy, a tiež bodovou pasívnou elektromagnetickou sondou. Tretie meracie zariadenie, osciloskop AGILENT INFINIIUM DSO9404A s 8-bit A/D prevodníkom, 4GHz šírkou pásma a 20GS/s vzorkovacou frekvenciou, bolo použité na meranie spotreby na kondenzátore na napájacej vetve jadra FPGA ALTERA Cyclone III pomocou koaxiálneho kábla. Pri meraniach bola použitá sekvencia, vďaka čomu boli merania veľmi rýchle. Napríklad meranie 1M priebehov spotreby trvá 65 s, pričom jeden priebeh spotreby obsahuje 2000 vzoriek (50 ns).

Útočiť na registre v programovateľnej logike FPGA (ďalej len „registre“) je jednoduchšie (hlavne čo sa týka ACTEL), ako útok na vstavané pamäte, ktoré tvoria súčasť RAM FPGA, tzv. embedded memory (ďalej len „RAM“). Táto skutočnosť bola prezentovaná na FPGA rodiny FUSION M7AFS600 od firmy ACTEL, kde bol implementovaný AES S-box. V tomto FPGA sú pamäte RAM smart pamäťami implementovanými pri jadre FPGA. Tieto pamäte sú napevno dané, ich logika sa neprogramuje, je možné len do nich zapisovať, alebo z nich čítať. V prípade registra tomu tak nie je. Registre sú v prípade ACTEL FUSION M7AFS600 tvorené pomocou tzv. „VersaTile“ buniek, prostredníctvom ktorých je možné vytvoriť štandardné logické elementy, ako sú napríklad OR, NAND, XOR, multiplexery, registre a latche. Princíp programovania je založený na prepínačoch, ktoré sú realizované flash pamäťovými bunkami. Prostredníctvom týchto prepínačov je možné zvoliť aký element má daný VersaTile implementovať. Technológia s prepínačmi tvorenými flash pamäťovými bunkami sa nazýva nevolatilna, pretože po odpojení napájania sa nestratí konfigurácia FPGA. Zneužiteľná spotreba teda závisí od zložitosti hardvéru, resp. dĺžky spojov a elementov umiestnených na týchto spojoch, cez ktoré sa musí vstupný signál pretransformovať, aby sa stal signálom výstupným. Zabudovanými RAM pamäť je menšia, a teda na realizáciu svojej funkcie spotrebuje menej elektrickej energie ako register v logike, ktorý je väčší, súčasťou programovateľnými VersaTiles. Práve preto je vo všeobecnosti ťažšie zaútočiť na ASIC, než na FPGA, pretože ASIC je jednoúčelový a optimalizovaný na jeho jednoúčelovú funkciu.

Útok na μkontroler alebo procesor je zväčša jednoduchší, než na FPGA. Dôvodom

je, že procesory majú rovnomernú zbernicu, po ktorej dáta prúdia sem a tam. Táto zbernica je symetrická a je konštante široká. Napríklad μkontroler PIC18F2520, ktorý bol podrobený útokom, má 8-bitov širokú zbernicu, a teda vie spracovávať 8-bitové slová v jednom takte. Ďalej môže byť šírka zbernice napríklad 16, 32, alebo 64 bitov. Pričom platí, že čím širšia je zbernica, tým je útok zložitejší, pretože, buď jE SNR pre útok menej priaznivé, alebo je počet hypotéz po zrealizovanom útoku vyšší. Keďže sú zbernice symetrické, je možné spotrebu a vyžarovanie modelovať jednoduchšie, napr. HW modelom spotreby. Naopak pri FPGA sú zbernice nesymetrické, nie sú rovnako dlhé, a nie je na každom vodiči rovnaký počet logických elementov. Dokonca pri opätovnej konfigurácii FPGA rovnakou funkciou, je funkcia inak implementovaná, keďže syntéza a spôsob smerovania (rooting) nie sú deterministické. Toto spôsobuje, že HW model spotreby je v prípade FPGA omnoho menej efektívny než HD model spotreby. Tento fakt bol potvrdení pri porovnaní útoku na AES-128 v 8-bitovom μkontroléri v (Repka [50]), kde boli spracovávaný stav jedného kola sekvenčne S-box po S-boxe, narozdiel od FPGA v (Repka, Gaspar, and Fischer [55]), kde bolo implementované spracovávanie celého stavu paralelne a teda všetkých 16 S-boxov súčasne. V práci k dizertačnej skúške ďalej bolo ukázané, že počet hypotéz, závisí od architektúry kryptografického algoritmu. Kryptografický algoritmus môže byť implementovaný rôznym usporiadaním funkčných blokov a registrov. Práve preto je možné učiniť útok postranným kanálom zložitejším a niekdy aj nemožným, ak sa tieto funkčné bloky a registre usporiadajú s ohľadom na možnosti zostrojovania hypotéz. Takéto protiopatrenie zväčša nevyžaduje žiadne ďalšie zdroje a tak môže byť takéto protiopatrenie klasifikované ako „cost-efficient“, dokonca niekedy aj „zero-cost“ protiopatrenie. Takéto protiopatrenie bolo prezentované na rôznych možnostiach RTL architektúry šifry AES-128 v (Repka et al. [60]) a v (Repka [48]). Na úrovni architektúry bol tento typ protiopatrení využitý aj pri návrhu kryptoprocesora v (Gaspar et al. [21]), ktorý vďaka tomuto prístupu získal prirodzenú odolnosť voči niektorým typom postranných útkov, ako FIA, CPA, CEMA, na určitej úrovni. Využilo sa usporiadanie funkčných blokov a návrh multiplexovania tak, aby niektoré útoky neboli vôbec možné, a iné zasa viac zložité. K týmto protiopatreniam môžeme zaradiť aj návrh hierarchie manažmentu šifrovacích kľúčov a nastavenie počtu použití kľúčov na rôznych úrovniach, pretože univariačné a multivariačné útoky z rodiny DPA a CPA potrebujú, aby sa rovnaký kľuč použil niekoľko krát. Chvíľka pozornosti bola venovaná aj boolovskému maskovaniu S-boxu, ktorého aplikovanie musí byť tiež zvážené na úrovni algoritmu, architektúry a implementácie, s ohľadom na možnosti zostrojovania hypotéz, pretože pri nesprávnom alebo nedostatočnom maskovaní nie je citlivý medzivýsledok zamaskovaný dostatočne, alebo je zamaskovaný na jednom mieste, avšak na iných miestach zamaskovaný nemusí byť vôbec. Pri správnom prístupe maskovania má byť maskovanie aplikované hneď na vstupe ešte pred prvou registráciou v obvode a nová nezávislá maska má byť aplikovaná pred každým ďalším spracovávaním alebo registrovaním. Výsledok má byť demaskovaný až pri výstupe z

obvodu po čítaní výsledku z registra. Maska má byť nezávislá nie len naprieč časovej osi tak, ako sa dáta spracúvajú, ale aj naprieč šírky dát, napríklad ak sa dáta delia na menšie bloky, ktoré sú spracovávané samostatne inými alebo rovnakými funkciami.

Kapitola 5 popisuje ďalšie dosiahnuté výsledky. Okrem útokov na symetrické šifry boli skúmané aj útoky na asymetrickú kryptografiu. Skúmanie bolo upriamené na elektronický podpis ECDSA, ktorý je dnes veľmi využívaný (Časť 5.1.2). Doposiaľ bolo publikovaných veľa útokov postrannými kanálmi na eliptické krivky a tiež adekvátne protiopatrenia na eliminovanie publikovaných útokov. Pri útoku na ECDSA bolo ukázané, že nie len samotné operácie na eliptických krivkách môžu byť zraniteľné, a teda, že nestačí aplikovať protiopatrenia len na ne, ale že aj operácie ako celočíselné násobenie môžu poskytnúť únik o privátnom kľúči. Toto platí hlavne pri podpisoch z rodiny DSA, pretože architektúra algoritmov týchto podpisov používa privátny kľúč až pri konci algoritmu ako jeden z operandov celočíselného násobenia, kde druhý operand tohto násobenia je známy, je časťou podpisu. Vďaka tomu sú útoky typu DPA, CPA, a CEMA účinnými. Keďže tieto podpisy sú dnes zväčša implementované na smart kartách, alebo iných zariadeniach využívajúce procesory, ktoré obsahujú 8, 16, alebo 32 bitové celočíselné násobičky, tieto útoky radikálne zužujú množinu možných privátnych kľúčov. V prácach (Repka, Tomeček, and Varchola [56]), (Repka and Varchola [57]), (Repka, Varchola, and Drutarovsky [58]), bol vyšetrovaný tento aspekt. Úspešnosť útoku s reálne nameranými dátami dosahuje vyšších hodnôt, ako úspešnosť útoku so simulovanými vzorkami.Preto malo prínos uvažovať chybu odhadu zložitosti útoku (reprezentovanú počtom zostávajúcich hypotéz po útoku) a úspešnosti útoku (pravdepodobnosti, že medzi tými zostávajúcimi hypotézami je tá správna). V rámci tejto úvahy boli vysvetlené chyby prvého a druhého druhu, ako aj bola zadefinovaná relevantná chyba z pohľadu analýzy úrovne zraniteľnosti kryptosystému. Významným prínosom bolo zlepšenie úspešnosti v oboch prípadoch (Časť 5.2), v prípade nameraných vzoriek, ako aj v prípade simulovaných vzoriek. Boli aplikované v sérii dva CPA útoky s HD modelom spotreby. Prvý CPA útok zoradil všetky možné hypotézy o bloku kľúča. V danom prípade bola použitá 16-bitová násobička v ALTERA FPGA. Za účelom dosiahnutia čo najpriaznivejšieho SNR pre útok, bol použitý HD model spotreby popisujúci celú 32-bitovú šírku výsledku násobenia. Z predošlých experimentov bolo zistené, že ak sa zoberie prvých 10 takto zoradených hypotéz po takto zrealizovanom CPA, tak je medzi nimi správna hypotéza o privátnom kľúči s pravdepodobnosťou veľmi blízkou 1. V sérii druhý útok bol aplikovaný na týchto prvých 10 hypotéz za účelom ich prezoradenia. V druhom útoku bol použitý HD model spotreby popisujúci len spodných 16 bitov výsledku násobenia. Tento postup priniesol značné zlepšenie pri útoku s použitím nameraných vzoriek a ešte väčšie zlepšenie úspešnosti v prípade použitia simulovaných vzoriek. Prínos sa prejavil aj v znížení zložitosti a zvýšení úspešnosti útoku. Po zrealizovaní tohto druhého útoku sa chyba simulovaného útoku stáva zanedbateľnou, čo ma veľký význam pri odhadoch zložitosti a úspešnosti útokov na základe simulácií.

Porovnanie úspešnosti a zložitosti útoku po prvom a druhom CPA je možné nájsť v (Repka, Varchola, and Drutarovsky [58]; Sec. 5.2; Obr. 5.6; Tab. 5.1).

Ako protiopatrenie voči tomuto útoku bolo navrhnuté efektívne protiopatrenie nevyžadujúce žiadne ďalšie zdroje, až na jedno násobenie navyše (Repka, Varchola, and Drutarovsky [58]; Varchola et al. [78]; Časť 5.3). Na maskovanie bolo použité náhodné číslo, ktoré sa používa v rámci (EC)DSA na konci výpočtov. Toto náhodné číslo bolo použité na zamaskovanie citlivého násobenia známeho výstupného operandu s privátnym (16-bit blok privátneho kľúča).

V práci (Varchola, Drutarovský, and Repka [79]) sa tiež zaoberáme skutočne náhodnými generátormi.

Doposiaľ boli skúmané útoky využívajúce únik citlivej informácie v spotrebe elektrickej energie alebo v elektromagnetickom vyžarovaní. Avšak v práci boli skúmané aj útoky využívajúce postranný kanál doby výpočtu algoritmu alebo jeho častí. Pri skúmaní tohto typu postranného útoku bol analýze doby výpočtu podrobený Pattersonov algebraický dekódovací algoritmus, ktorý je súčasťou dešifrovacieho procesu McEliece kryptosystému s verejným kľúčom. Ešte pred samotným podrobením tohto kryptosystému analýze doby výpočtu bolo nutné tento algoritmus implementovať. Bola vyvinutá implementácia originálneho McEliece PKC (Časť 5.4), presne podľa popisu samého McEliece (1978). Táto implementácia, ak je správne použitá (CCA2 bezpečne), je považovaná za post-kvantový kryptosystém s verejným kľúčom, teda kryptosystém odolný voči kvantovej kryptoanalýze, kryptoanalýze využívajúcej kvantový počítač. Táto implementácia bola nazvaná McEliece PKC Calculator (Repka [52]), práve preto, že poskytuje testovacie vektory pre každý relevantný medzivýsledok, a navyše implementuje aj nástroj na meranie doby výpočtu jednotlivých krokov Pattersonovho algebraického dekódovacieho algoritmu. Tento nástroj na meranie doby výpočtu je prezentovaný v (Repka [51]). Prehľad rôznych variant tohto kryptosystému a jeho bezpečnosti je možné nájsť v (Repka and Zajac [59]), ako aj v (Repka and Cayrel [54]), kde je poskytnutý celkový pohľad na kryptografiu založenú na lineárnych samoopravných kódoch. Analýza doby výpočtu rôznych krokov dekódovacieho algoritmu je uvedená v Častiach 5.5 a 5.6 kde je aj táto analýza vylepšená. Protiopatrenie je diskutované v Časti 5.7. Niekoľko poznámok nad modularnou redukciou v rozšírených konečných poliach je možné nájsť v práci (Repka [53]). V poradí posledným zaujímavým výsledkom je efektívna metóda na výpočet p-tej odmocniny v rozšírených konečných poliach charakteristiky $p \geq 2$ (Sec. 5.8). Táto metóda bola publikovaná v Repka [49]. Kapitoly 6 a 7 uvádzajú zoznam príspevkov autora a záver.

References

[1] SPS Project Number: 984520. *Secure implementation of post-quantum cryptography*. 2013. URL: http://re-search.info/node/27.

[2] Mehdi-Laurent Akkar and Christophe Giraud. "An Implementation of DES and AES, Secure against Some Attacks". English. In: *Cryptographic Hardware and Embedded Systems — CHES 2001*. Ed. by ÇetinK. Koç, David Naccache, and Christof Paar. Vol. 2162. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 309–318. ISBN: 978-3-540-42521-2. DOI: 10.1007/3-540-44709-1_26. URL: http://dx.doi.org/10.1007/3-540-44709-1_26.

[3] Mehdi-Laurentm Akkar and Christophe Giruad. "An Implementation of DES and AES, Secure against Some Attacks". In: *Cryptographic Hardware and Embedded Systems - CHESS 2001, Third Interational Workshop, Paris, France, May 14-16, 2001, Proceedings*. Springer, 2001, pp. 309–318.

[4] Alessandro Barenghi et al. "Exploring the Feasibility of Low Cost Fault Injection Attacks on Sub-Threshold Devices through an example of a 65nm AES implementation". In: *Workshop on RFID Security – RFIDSec'11*. Amherst, Massachusetts, USA, 2011.

[5] Lyonel Barthe, Pascal Benoit, and Lionel Torres. "Investigation of a Masking Countermeasure against Side-Channel Attacks for RISC-based Processor Architectures". English. In: *FPL'10: Field Programmable Logic and Applications*. Milan, Italy, 2010, pp. 139 –144. URL: http://hal-lirmm.ccsd.cnrs.fr/lirmm-00548802/en/.

[6] Lejla Batina et al. "Mutual Information Analysis: a Comprehensive Study". English. In: *Journal of Cryptology* 24.2 (2011), pp. 269–291. ISSN: 0933-2790. DOI: 10.1007/s00145-010-9084-8. URL: http://dx.doi.org/10.1007/s00145-010-9084-8.

[7] DanielJ. Bernstein, Tanja Lange, and Christiane Peters. "Wild McEliece Incognito". English. In: *Post-Quantum Cryptography*. Ed. by Bo-Yin Yang. Vol. 7071. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 244–254. ISBN: 978-3-642-25404-8. DOI: 10.1007/978-3-642-25405-5_16. URL: http://dx.doi.org/10.1007/978-3-642-25405-5_16.

[8] Andrey Bogdanov et al. "Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs". In: *Topics in Cryptology - CT-RSA 2010*. Ed. by Josef Pieprzyk. Vol. 5985. Lecture Notes in Computer Science. 10.1007/978-3-642-11925-5_17. Springer Berlin / Heidelberg, 2010, pp. 235–251. ISBN: 978-3-642-11924-8. URL: http://dx.doi.org/10.1007/978-3-642-11925-5_17.

[9] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation Power Analysis with a Leakage Model". In: *CHES*. 2004, pp. 16–29.

[10] Florent Bruguier et al. "A New Process Characterization Method for FPGAs Based on Electromagnetic Analysis". English. In: *FPL'11: 21st International Conference on Field Programmable Logic and Applications*. Greece, Sept. 2011, N/A. URL: http://hal-lirmm.ccsd.cnrs.fr/lirmm-00616954/en/.

[11] David Brumley and Dan Boneh. "Remote timing attacks are practical". In: *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*. Washington, DC: USENIX Association, 2003, p. 1. URL: http://portal.acm.org/citation.cfm?id=1251354.

[12] Pierre-Loius Cayerl. *Code based cryptography*. 2012. URL: http://cayrel.net/research/code-based-cryptography/code-based-cryptosystems/.

[13] J.-L. Danger et al. "Overview of Dual rail with Precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors". In: *Signals, Circuits and Systems (SCS), 2009 3rd International Conference on*. 2009, pp. 1–8. DOI: 10.1109/ICSCS.2009.5412599.

[14] Guo Liang Ding et al. "Correlation Electromagnetic Analysis for Cryptographic Device". In: *Proceedings of the 2009 Pacific-Asia Conference on Circuits, Communications and Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 388–391. ISBN: 978-0-7695-3614-9. DOI: 10.1109/PACCS.2009.144. URL: http://dl.acm.org/citation.cfm?id=1636711.1637668.

[15] Javad Doliskani and Éric Schost. "Computing in Degree 2^K 2K-extensions of Finite Fields of Odd Characteristic". In: *Des. Codes Cryptography* 74.3 (Mar. 2015), pp. 559–569. ISSN: 0925-1022. DOI: 10.1007/s10623-013-9875-7. URL: http://dx.doi.org/10.1007/s10623-013-9875-7.

[16] Martin Döring. "On the Theory and Practice of Quantum-Immune Cryptography". PhD thesis. Darmstadt: Technische Universität, 2008. URL: http://tuprints.ulb.tu-darmstadt.de/1072/.

[17] Thomas Eisenbarth et al. "MicroEliece: McEliece for Embedded Devices". In: *CHES*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. LNCS. Springer, 2009, pp. 49–64. ISBN: 978-3-642-04137-2.

[18] Daniela Engelbert, Raphael Overbeck, and Arthur Schmidt. "A Summary of McEliece-Type Cryptosystems and their Security". In: *J. Mathematical Cryptology* 1.2 (2007), pp. 151–199.

[19] Junfeng Fan et al. "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures". In: *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*. 2010, pp. 76–87. DOI: 10.1109/HST.2010.5513110.

[20] Matteo Frigo and Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on Program Generation, Optimization, and Platform Adaptation, pp. 216–231. URL: http://www.fftw.org/.

[21] Lubos Gaspar et al. "Cryptoprocessor with Native Resistance against Side Channel and Fault Injection Attacks". In: *Proceedings of the 13th International Conference on Telecommunications and Informatics (TELE-INFO'14), Istanbul, Turkey, December 15-17, 2014*. Ed. by Nikos E. Mastorakis et al. WSEAS press, 2014, pp. 88–97.

[22] Daniel Genkin, Adi Shamir, and Eran Tromer. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis". In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Springer, 2014, pp. 444–461. ISBN: 978-3-662-44370-5. DOI: 10.1007/978-3-662-44371-2_25. URL: http://dx.doi.org/10.1007/978-3-662-44371-2_25.

[23] Daniel Genkin et al. "Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation". In: *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. 2015, pp. 207–228. DOI: 10.1007/978-

3-662-48324-4_11. URL: http://dx.doi.org/10.1007/978-3-662-48324-4_11.

[24] V. D. Goppa. "A New Class of Linear Error Correcting Codes". In: *Probl. Pered. Inform.* 6 (Sept. 1970), pp. 24–30.

[25] Christopher Hadnagy. *Social engineering : the art of human hacking*. Hoboken, N.J. Wiley Chichester: John Wiley, 2011. ISBN: 978-0-470-63953-5. URL: http://opac.inria.fr/record=b1133491.

[26] J. Alex Halderman et al. "Lest We Remember: Cold-boot Attacks on Encryption Keys". In: *Commun. ACM* 52.5 (May 2009), pp. 91–98. ISSN: 0001-0782. DOI: 10.1145/1506409.1506429. URL: http://doi.acm.org/10.1145/1506409.1506429.

[27] Annelie Heuser and Michael Zohner. "Intelligent Machine Homicide". English. In: *Constructive Side-Channel Analysis and Secure Design*. Ed. by Werner Schindler and SorinA. Huss. Vol. 7275. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 249–264. ISBN: 978-3-642-29911-7. DOI: 10.1007/978-3-642-29912-4_18. URL: http://dx.doi.org/10.1007/978-3-642-29912-4_18.

[28] Stefan Heyse. "Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers". In: *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings*. Ed. by Nicolas Sendrier. Vol. 6061. Lecture Notes in Computer Science. Springer, 2010, pp. 165–181. ISBN: 978-3-642-12928-5. DOI: 10.1007/978-3-642-12929-2_13. URL: http://dx.doi.org/10.1007/978-3-642-12929-2_13.

[29] Stefan Heyse and Tim Güneysu. "Towards One Cycle per Bit Asymmetric Encryption: Code-Based Cryptography on Reconfigurable Hardware". In: *CHES*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. LNCS. Springer, 2012, pp. 340–355. ISBN: 978-3-642-33026-1.

[30] Martin Hlaváč. "Využití postranných kanálů v symetrické a asymetrické kryptoanalýze". In: *Disertačná práce*. Universita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra algebry, 2010.

[31] Naofumi Homma et al. "High-Resolution Side-Channel Attack Using Phase-Based Waveform Matching". In: *Cryptographic Hardware and Embedded Systems - CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. Lecture Notes in Computer Science. 10.1007/11894063_15. Springer Berlin / Heidelberg, 2006, pp. 187–200. ISBN: 978-3-540-46559-1. URL: http://dx.doi.org/10.1007/11894063_15.

[32] Michael Hutter et al. “Attacking ECDSA-Enabled RFID Devices”. English. In: *Applied Cryptography and Network Security*. Ed. by Michel Abdalla et al. Vol. 5536. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 519–534. ISBN: 978-3-642-01956-2. DOI: 10.1007/978-3-642-01957-9_32. URL: http://dx.doi.org/10.1007/978-3-642-01957-9_32.

[33] “IEEE Standard Specifications for Public-Key Cryptography”. In: *IEEE Std 1363-2000* (2000), pp. 1–228. DOI: 10.1109/IEEESTD.2000.92292.

[34] Kazukuni Kobara and Hideki Imai. “Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC”. In: *Public Key Cryptography*. Ed. by Kwangjo Kim. Vol. 1992. LNCS. Springer, 2001, pp. 19–35. ISBN: 3-540-41658-7.

[35] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. “Differential Power Analysis”. In: *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO ’99. London, UK: Springer-Verlag, 1999, pp. 388–397. ISBN: 3-540-66347-9. URL: http://dl.acm.org/citation.cfm?id=646764.703989.

[36] PaulC. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. English. In: *Advances in Cryptology — CRYPTO ’96*. Ed. by Neal Koblitz. Vol. 1109. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1996, pp. 104–113. ISBN: 978-3-540-61512-5. DOI: 10.1007/3-540-68697-5_9. URL: http://dx.doi.org/10.1007/3-540-68697-5_9.

[37] J.S. Lemos-Neto and V.C. da Rocha. “Cyclically permutable codes specified by roots of generator polynomial”. In: *Electronics Letters* 50.17 (2014), pp. 1202–1204. ISSN: 0013-5194. DOI: 10.1049/el.2014.0296.

[38] Huiyun Li, Keke Wu, and Fengqi Yu. “Enhanced correlation power analysis attack against trusted systems”. In: *Security and Communication Networks* 4.1 (2011), pp. 3–10.

[39] H. Maghrebi et al. “Evaluation of countermeasure implementations based on Boolean masking to thwart side-channel attacks”. In: *Proc. 3rd Int Signals, Circuits and Systems (SCS) Conf.* 2009, pp. 1–6. DOI: 10.1109/ICSCS.2009.5412597. URL: http://dx.doi.org/10.1109/ICSCS.2009.5412597.

[40] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007. ISBN: 978-0-387-30857-9.

[41] Philippe Maurine et al. “Local and Direct Power Injection on CMOS Integrated Circuits”. English. In: *The 8th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC 2011)*. Nara, Japan, Sept. 2011, pp. 001 –010. URL: http://hal-lirmm.ccsd.cnrs.fr/lirmm-00607868/en/.

[42] R. J. McEliece. “A public-key cryptosystem based on algebraic coding theory”. In: *DSN progress report* 42.44 (1978), pp. 114–116. URL: http://www.cs.colorado.edu/~jrblack/class/csci7000/f03/papers/mceliece.pdf.

[43] Marcel Medwed and Elisabeth Oswald. “Template Attacks on ECDSA”. English. In: *Information Security Applications*. Ed. by Kyo-Il Chung, Kiwook Sohn, and Moti Yung. Vol. 5379. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 14–27. ISBN: 978-3-642-00305-9. DOI: 10.1007/978-3-642-00306-6_2. URL: http://dx.doi.org/10.1007/978-3-642-00306-6_2.

[44] Elisabeth Oswald et al. “Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers”. English. In: *Topics in Cryptology – CT-RSA 2006*. Ed. by David Pointcheval. Vol. 3860. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 192–207. ISBN: 978-3-540-31033-4. DOI: 10.1007/11605805_13. URL: http://dx.doi.org/10.1007/11605805_13.

[45] D. Panario and D. Thomson. “Efficient pth root computations in finite fields of characteristic p”. English. In: *Designs, Codes and Cryptography* 50.3 (2009), pp. 351–358. ISSN: 0925-1022. DOI: 10.1007/s10623-008-9236-0. URL: http://dx.doi.org/10.1007/s10623-008-9236-0.

[46] N. J. Patterson. “The algebraic decoding of Goppa codes”. In: *IEEE Transactions on Information Theory* 21 (2 1975), pp. 203–207. DOI: 10.1109/TIT.1975.1055350.

[47] Jean-Jacques Quisquater and David Samyde. “ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards”. In: *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*. E-SMART ’01. London, UK, UK: Springer-Verlag, 2001, pp. 200–210. ISBN: 3-540-42610-8. URL: http://dl.acm.org/citation.cfm?id=646803.705980.

[48] Gaspar Ľuboš Repka Marek. “Hamming Distance Power Model is Architecture Dependent”. In: *ELOSYS. Elektrotechnika, informatika a telekomunikácie 2012 Trenčín, 9.-12.10.2012*. Nakladateľstvo STU, 2012.

[49] M. Repka. “Computing pth roots in extended finite fields of prime characteristic p >= 2”. In: *Electronics Letters* 52.9 (Apr. 2016), pp. 718–719. ISSN: 0013-5194. DOI: 10.1049/el.2015.4141.

[50] Marek Repka. “DPA of an AES Implementation”. In: *KOZÁKOVÁ, A. ELITECH 2011: 13th Conference of Doctoral Students Faculty of Electrical Engineering and Information Technology. Bratislava, Slovak Republic, 17 May, 2011.* Nakladateľstvo STU, 2011, pp. 1–6. ISBN: 978-80-227-3500-1.

[51] Marek Repka. “Leakage Measurement Tool of McEliece PKC Calculator”. In: *Proceedings of the 13th International Conference on Telecommunications and Informatics (TELE-INFO’14), Istanbul, Turkey, December 15-17, 2014*. Ed. by Nikos E. Mastorakis et al. WSEAS press, 2014, pp. 124–132.

[52] Marek Repka. “McEliece PKC Calculator”. In: *Journal of Electrical Engineering* 65.6 (2014), pp. 333–341.

[53] Marek Repka. “Note on modular reduction in extended finite fields and polynomial rings for simple hardware”. In: *Journal of Electrical Engineering* 67.1 (2016), pp. 56–60.

[54] Marek Repka and Pierre-Louis Cayrel. In: *Multidisciplinary Perspectives in Cryptology and Information Security*. Ed. by Sattar B. Sadkhan Al Maliky and Nidaa A. Abaas. IGI Global, 2014. Chap. Cryptography Based on Error Correcting Codes: A Survey, pp. 133–156.

[55] Marek Repka, Lubos Gaspar, and Viktor Fischer. “Correlation Power Analysis of an AES Implementation in FPGA”. In: *TatraCrypt 2012: Book of abstracts*. 2012.

[56] Marek Repka, Jozef Tomeček, and Miachal Varchola. “Correlation Hamming Distance Power Analysis of 16-bit Integer Multiplier in FPGA”. In: *Proceedings of the 13th International Conference on Telecommunications and Informatics (TELE-INFO’14), Istanbul, Turkey, December 15-17, 2014*. Ed. by Nikos E. Mastorakis et al. WSEAS press, 2014, pp. 49–53.

[57] Marek Repka and Michal Varchola. “Correlation Power Analysis using Measured and Simulated Power Traces based on Hamming Distance Power Model – Attacking 16-bit Integer Multiplier in FPGA”. In: *International Journal of Computer Network and Information Security(IJCNIS)* 7.6 (2014), pp. 10–16.

[58] Marek Repka, Michal Varchola, and Miloš Drutarovsky. “Improving CPA against DSA and ECDSA”. In: *Journal of Electrical Engineering* 66.3 (2015), pp. 159–163.

[59] Marek Repka and Pavol Zajac. “Overview of the McEliece Cyptosystem and its Security”. In: *Tatra Mt. Math. Publ* 60.3 (2014), pp. 57–83.

[60] Marek Repka et al. “Hamming Distance Power Model Analysis of AES at Architecture Level”. In: *COSADE 2012: 3rd International Workshop on Constructive Side-Channel Analysis and Secure Design. Darmstadt, Germany, May 3-4, 2012. Darmstadt, Technische Universität Darmstadt.*

[61] Matthieu Rivain. “Differential Fault Analysis on DES Middle Rounds”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 457 –469.

[62] Matthieu Rivain. “On the Exact Success Rate of Side Channel Analysis in the Gaussian Model”. English. In: *Selected Areas in Cryptography*. Ed. by RobertoMaria Avanzi, Liam Keliher, and Francesco Sica. Vol. 5381. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 165–183. ISBN: 978-3-642-04158-7. DOI: 10.1007/978-3-642-04159-4_11. URL: http://dx.doi.org/10.1007/978-3-642-04159-4_11.

[63] Matthieu Rivain and Emmanuel Prouff. “Provably Secure Higher-Order Masking of AES”. In: *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. Lecture Notes in Computer Science. Springer, 2010, pp. 413 –427.

[64] Werner Schindler, Kerstin Lemke, and Christof Paar. “A Stochastic Model for Differential Side Channel Cryptanalysis”. English. In: *Cryptographic Hardware and Embedded Systems – CHES 2005*. Ed. by JosyulaR. Rao and Berk Sunar. Vol. 3659. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 30–46. ISBN: 978-3-540-28474-1. DOI: 10.1007/11545262_3. URL: http://dx.doi.org/10.1007/11545262_3.

[65] Robert Sedgewick. “Permutation Generation Methods”. In: *ACM Comput. Surv.* 9.2 (June 1977), pp. 137–164. ISSN: 0360-0300. DOI: 10.1145/356689.356692. URL: http://doi.acm.org/10.1145/356689.356692.

[66] Nicolas Sendrier, ed. *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings*. Vol. 6061. Lecture Notes in Computer Science. Springer, 2010. ISBN: 978-3-642-12928-5. DOI: 10.1007/978-3-642-12929-2. URL: http://dx.doi.org/10.1007/978-3-642-12929-2.

[67] Daniel Shanks. “Five number-theoretic algorithms”. In: *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972)*. Winnipeg, Man.: Utilitas Math., 1973.

[68] Abdulhadi Shoufan et al. “A Novel Cryptoprocessor Architecture for the McEliece Public-Key Cryptosystem”. In: *IEEE Trans. Computers* 59.11 (2010), pp. 1533–1546.

[69] V. Shoup. *NTL: A Library for doing Number Theory (version 6.0.0)*. 2013. URL: http://www.shoup.net/ntl/.

[70] Sergei P. Skorobogatov. "Semi-invasive attacks – A new approach to hardware security analysis". In: UCAM-CL-TR-630. 2005. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf.

[71] François-Xavier Standaert, TalG. Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks". English. In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 443–461. ISBN: 978-3-642-01000-2. DOI: 10.1007/978-3-642-01001-9_26. URL: http://dx.doi.org/10.1007/978-3-642-01001-9_26.

[72] Falko Strenzke. "A smart card implementation of the mceliece PKC". In: *The 4th IFIP WG 11.2, Proceedings*. WISTP'10. Passau, Germany: Springer-Verlag, 2010, pp. 47–59. ISBN: 3-642-12367-8, 978-3-642-12367-2. DOI: 10.1007/978-3-642-12368-9_4.

[73] Falko Strenzke. "A Smart Card Implementation of the McEliece PKC". In: *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, 4th IFIP WG 11.2 International Workshop, WISTP 2010, Passau, Germany, April 12-14, 2010. Proceedings*. 2010, pp. 47–59. DOI: 10.1007/978-3-642-12368-9_4. URL: http://dx.doi.org/10.1007/978-3-642-12368-9_4.

[74] Falko Strenzke. "A Timing Attack against the Secret Permutation in the McEliece PKC". In: *PQCrypto*. Ed. by Nicolas Sendrier. Vol. 6061. Lecture Notes in Computer Science. Springer, 2010, pp. 95–107. ISBN: 978-3-642-12928-5. DOI: 10.1007/978-3-642-12929-2. URL: http://dx.doi.org/10.1007/978-3-642-12929-2.

[75] Falko Strenzke. "Efficiency and Implementation Security of Code-based Cryptosystems". PhD thesis. Germany: Universitat Darmstadt, 2013.

[76] Falko Strenzke. "Timing Attacks against the Syndrome Inversion in Code-Based Cryptosystems". In: *PQCrypto*. Ed. by Philippe Gaborit. Vol. 7932. LNCS. Springer, 2013, pp. 217–230. ISBN: 978-3-642-38615-2.

[77] Falko Strenzke et al. "Side Channels in the McEliece PKC". In: *PQCrypto*. Ed. by Johannes Buchmann and Jintai Ding. Vol. 5299. LNCS. Springer, 2008, pp. 216–229. ISBN: 978-3-540-88402-6.

[78] M. Varchola et al. "Side Channel Attack on Multiprecision Multiplier Used in Protected ECDSA Implementation". In: *2015 International Conference on ReConFigurable Computing and FPGAs, ReConFig15, Cancun, Mexico, December 7-9, 2015*. IEEE.

[79] Michal Varchola, Miloš Drutarovský, and Marek Repka. “Robust FPGA based True Random Number Generator utilizing Oscillatory Metastability in Transition Effect Ring Oscillators - Invited Paper”. In: *Proceedings of the 9th International Conference on Circuits, Systems, Signal and Telecommunications (CSST'15), Dubai, United Arab Emirates, February 22-24, 2015*. Ed. by Nikos E. Mastorakis and Zoran Bojkovic. WSEAS press, 2015, pp. 90–98.

[80] Serge Vaudenay. “Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ...” In: *EUROCRYPT*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 534–546. ISBN: 3-540-43553-0.

[81] Pavol Zajac and Matus Jokay. “Computing indexes and periods of all Boolean matrices up to dimension n= 8”. In: *Computing and Informatics* 31.6 (2013), pp. 1329–1344.

[82] Xin jie Zhao et al. “MDASCA: An Enhanced Algebraic Side-Channel Attack for Error Tolerance and New Leakage Model Exploitation”. In: *COSADE*. Ed. by Werner Schindler and Sorin A. Huss. Vol. 7275. Lecture Notes in Computer Science. Springer, 2012, pp. 231–248. ISBN: 978-3-642-29911-7.