Practical Evaluation of Protected Residue Number System Scalar Multiplication

Louiza Papachristodoulou[1], Apostolos P. Fournaris[2], Kostas Papagiannopoulos[1] and Lejla Batina[1*]

[1] Digital Security Group, Radboud University Nijmegen, The Netherlands,
louiza@cryptologio.org,kostaspap88@gmail.com,lejla@cs.ru.nl
[2] Electrical and Computer Engineering Dpt., University of Patras, Greece,
apofour@ece.upatras.gr

Abstract.
The Residue Number System (RNS) arithmetic is gaining grounds in public key cryptography, because it offers fast, efficient and secure implementations over large prime fields or rings of integers. In this paper, we propose a generic, thorough and analytic evaluation approach for protected scalar multiplication implementations with RNS and traditional Side Channel Attack (SCA) countermeasures in an effort to assess the SCA resistance of RNS. This paper constitutes the first robust evaluation of RNS software for Elliptic Curve Cryptography against electromagnetic (EM) side-channel attacks. Four different countermeasures, namely scalar and point randomization, random base permutations and random moduli operation sequence, are implemented and evaluated using the Test Vector Leakage Assessment (TVLA) and template attacks. More specifically, variations of RNS-based Montgomery Powering Ladder scalar multiplication algorithms are evaluated on an ARM Cortex A8 processor using an EM probe for acquisition of the traces. We show experimentally and theoretically that new bounds should be put forward when TVLA evaluations on public key algorithms are performed. On the security of RNS, our data and location dependent template attacks show that even protected implementations are vulnerable to these attacks. A combination of RNS-based countermeasures is the best way to protect against side-channel leakage.

Keywords: SCA evaluation · TVLA · residue number system · elliptic curve cryptography · scalar multiplication · template attacks

1 Introduction

The security of embedded devices, even devices with dedicated cryptographic processors, is dependent on resilience against side-channel attacks (SCA). Power consumption [KJJ99], electromagnetic emanations [AARR03] and other forms of side channel information leakage can expose the sensitive data of an implementation (usually a cryptographic key), through the use of various types of SCAs, such as Simple Power Analysis (SPA), Differential Power Analysis (DPA) and Template attacks [CRR03, MO09]. There are other powerful attack techniques that require active scenarios, like Fault Attacks (FA), where injecting a fault during the cryptographic operation can be used to extract the key according to the output of the algorithm [BDL97].

*This work was supported in part by the Technology Foundation STW (Project 12624-SIDES and Project 13499 TYPHOON) from the Dutch government and by the European ICT COST Actions TRUDEVICE IC1204 and CRYPTACUS IC1403. A.P. Fournaris' research work was also funded by CIPSEC EU Horizon 2020 project under G.A. No 700378.

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925, Vol. 2019, No. 1, pp. 259–282
DOI:10.13154/tches.v2019.i1.259-282

A broad range of countermeasures to protect cryptographic operations that use secret data have been devised, including randomization of the order of operations or masking sensitive values. In Elliptic Curve Cryptography (ECC) traditional countermeasures are focused on randomizing the scalar, randomizing the input point or manipulating the EC parameters and point representation (e.g. randomized projective coordinates approach) [FV12a]. However, several research groups explore alternative SCA resistance approaches that are focused on non-traditional arithmetic systems like the Residue Number System (RNS).

RNS was originally devised for parallel processing of arithmetic operations, in order to increase computation speed. Since it can effectively represent elements of cyclic groups or finite fields there is merit in adopting it in elliptic curve underlined finite field operations. Efficient RNS hardware implementations of Montgomery multiplication for elliptic curves [BDM06] and RSA [CNPQ04] showed the potentials of using RNS for public key applications. RNS has been recently used for other cryptographic schemes like lattice-based cryptography [BEHZ16] or in the work of Halevi et al. [HPS18], in order to optimize the implementation of the Fan-Vercauteren variant of the scale-invariant homomorphic encryption scheme of Brakerski [FV12b]. Since, this number system can have such a broad applicability to cryptographic operations, using it for SCA resistance seems to be inevitable. However, an extensive practical evaluation of RNS as an SCA countermeasure is still not performed.

Related work: The potentials of RNS as an SCA countermeasure is observed in several research papers, for instance Bajard et al. in [BILT04, BEG13], Guillermin [Gui11], Fournaris et al. [FKSK15, FPBS16], Schinianakis et al. [SS13]. RNS parallel processing of finite field operations apart from speed offers also different representation of the elliptic curve points, which may reduce SCA leakage. Also, taking into account that a single bit fault in an RNS number's moduli can lead to "difficult to trace" changes in an overall finite field element, supports the argument that the RNS can be introduced as SCA and FA combined countermeasure. It has also been observed that the periodic change using base permutation during the modular exponentiation (and consecutively scalar multiplication) computation flow can introduce enough randomness to thwart SCAs. This approach led to the introduction of the leak resistant arithmetic (LRA) technique [BILT04]. LRA has been applied to modular exponentiation designs in two ways, either by choosing a new base permutation once at the beginning of each modular exponentiation or by choosing a permutation once in each modular multiplication during exponentiation [FPBS16].

Theoretical evaluations for secure RNS scalar multiplication are presented in [FPS17, CATB18]. Both papers discuss the resistance of RNS randomization against various SCA attacks. In [FPS17] traces were collected for an initial analysis, but due to the constrains of the target platform, no proper evaluation was made. The SCA analysis of [FPS17] implementation is limited to identifying the rounds of RNS with and without the countermeasures. They imply that an Online Template type of attack (OTA) [BCP+14] should not be possible in their protected implementation, but detailed SCA results are missing. In [CATB18] Monte Carlo simulations are used to verify the resistance of the countermeasure against several attacks, but no practical t-tests are made. To the best of our knowledge, the only practical evaluation of an RNS hardware implementation is presented by Perin et al. [PITM13] but is constrained to an RSA design and is focused on comparative horizontal attacks (doubling/relative doubling attacks) and signal to noise measurement estimations for SCA resistance. The above highlight the need for a practical, broad and generic evaluation approach of scalar multiplication implementations that rely on RNS arithmetic. Such an approach and overall assessment could provide a definite answer on if and how RNS based technique (LRA and/or random moduli operation sequence) can contribute to the design of SCA resistant scalar multiplication. Such an answer should be based not only on theoretical analysis but, most importantly, on actual measurements.

In an effort to formulate a generic and broad methodology for evaluating the security of an RNS scalar multiplier we propose the use of Test Vector Leakage Assessment (TVLA), initially proposed by Goodwill et al. [GJJR11]. This methodology consists of several statistical tests between two trace sets of acquisition and uses Welch's t-test to evaluate if the two sets have significant statistical differences, that would distinguish for example a fixed versus a random input. This test provides results simultaneously for all the intermediate values and indicates potential points of leakage. Since its introduction to public-key algorithms in 2011 [JRW11], a few research groups have used TVLA so far for evaluation of RSA or ECC. Namely, Nascimento et al. [NLD15], Chmielewski et al. [CMV+17] used it for evaluation of Curve25519 on Chipwhisperer and the complete Weierstrass formulas on an FPGA respectively. Tunstall and Goodwill [TG16] give an overview of cases and algorithms that can be applied during public key TVLA evaluations. TVLA requires thousands or even millions of acquired traces to show the existence of leakage. This fact, combined with the low operation speed and the large trace lengths of public key implementations, make TVLA evaluations quite challenging for public-key cryptography.

Contribution: As indicated in the above analysis, the emerging use of RNS systems for cryptographic implementations and the broad belief that RNS offers a potential SCA countermeasure makes practical evaluations of RNS implementations essential. The complexity of RNS together with the fact that software implementations are generally slow, make it challenging to apply common evaluation techniques on such systems. Considering the above, in this paper, we propose a thorough, generic and broadly applicable practical evaluation based on TVLA and template attacks for RNS based scalar multiplication. Our primary goal is to examine and validate the RNS SCA capabilities using practical results. Furthermore, our evaluation is performed on a SCA challenging target implementation, an RNS software implementation using Montgomery Power Ladder (MPL) scalar multiplication that includes both RNS and traditional SCA countermeasures. In this context, the contributions of this paper are the following.

1. We provide a thorough SCA assessment of RNS scalar multiplication that involves a series of statistical tests for a protected RNS implementation with a combination of RNS and traditional countermeasures on a Cortex A8 platform. The RNS random base permutation countermeasure based on the LRA technique is evaluated on its SCA resistance in ECC implementations. This evaluation is extended to include an RNS moduli operation sequence randomization countermeasure for enhancing the SCA resistance. To the best of our knowledge, this work constitutes the *first practical evaluation of an RNS software implementation* for ECC using TVLA in combination with templates attacks, in order to enhance the confidence on the assessment results.

2. In the proposed evaluation approach, we verify and extend previous theoretical results and simulations on the *boundaries of the t-test threshold* for the case of traces with high number of samples that are obtained by a public key cryptographic implementation. A generic methodology to provide those boundaries is proposed.

3. We introduce a complementary template attack methodology, in order to compromise even protected RNS implementations. Our goal is to validate the TVLA results and to expose additional vulnerabilities in the RNS designs. The introduced template attacks are mounted on RNS LRA protected scalar multiplication and, indeed, manage to retrieve the key using data and location dependent leakage. *Location dependent template attacks* are introduced by Heyszl et al. [HMH+12] and used to attack an ECC implementation on an FPGA. We use here location dependent leakage templates for the first time in an RNS-ECC implementation setting.

Organization: The rest of the paper is organized as follows. In Section 2 the basics of RNS arithmetic are highlighted and the RNS implementation that we evaluated is

presented. Section 3 presents the theory of TVLA with focus on challenges for public key implementations. Hereby, we also propose the adaptation of the threshold for the t-test according to the number of samples per trace. The template attacks are presented in Section 4. Section 5 discusses the performance impact of various countermeasures. Finally, Section 6 concludes the paper with a discussion over the potentials of RNS against SCA.

2 Residue Number System

2.1 RNS arithmetic for ECC

RNS is an extension of the Chinese Remainder Theorem and is a non-positional arithmetic system where a number is represented by a set of individual n moduli x_i ($x \overset{RNS}{\rightarrow} X : (x_1, x_2, ...x_n)$) of a given RNS basis $B : (m_1, m_2, ...m_n)$ as long as $0 \leq x < M$ where $M = \prod_{i=1}^{n} m_i$ is the RNS dynamic range and all m_i are pair-wise relatively prime. Each x_i can be derived from x by calculating $x_i = \langle x \rangle_{m_i} = x \ mod \ m_i$. In contrast to binary arithmetics, addition, subtraction and multiplication in RNS is performed within each moduli (in n independent channels) thus enabling the parallelism of small bit length operations to come up with an arithmetic result [BDK01] [FKSK15]. Since RNS is a non-positional representation, comparisons, divisions and modular reductions are complex operations, which are performed either by converting the number from RNS to binary representation or by using base extension algorithms.

Binary reconstruction from RNS representation can be done either by using the Chinese Remainder Theorem (CRT) or through a Mixed Radix System (MRS) transformation. While the first approach seem to be more obvious, it requires a computational inefficient final *modulo* M operation that can only be avoided by using and approximating a correction factor (introducing the concept of Cox and Rower [KKSS00]). Using the MRS approach, this correction factor can be avoided but RNS numbers need to be transformed into MRS representation (a weighted moduli RNS variant) [BILT04] and then from this representation to binary numbers. The MRS number $\widetilde{X}$ can be obtained from $X : (x_1, x_2, x_3, ...x_n)$ by executing the Mixed Radix Conversion (MRC) algorithm [FPS17].

For elliptic curves defined over $GF(p)$ (elliptic curves on $GF(2^k)$ are not discussed in this paper), all $GF(p)$ operations (addition, subtraction, multiplication) are modular operations. RNS modular multiplication over $GF(p)$ is the most computationally difficult operation. It is usually realized through the RNS Montgomery multiplication algorithm that avoids modular inversions, but includes base extension operations [BDK97, FPBS16, FPS17].

2.2 RNS Base Extension

Base extension (BE) is used when an RNS number represented in an RNS base $B_n = (m_1, m_2, \ldots, m_n)$ needs to be represented in a different base $\acute{B_n} = (m_{n+1}, m_{n+2}, \ldots, m_{2n})$ ($gcd(m_i, m_j) = 1$ for all $i \in \{1, n\}$ and $j \in \{n+1, 2n\}$). Base extension can be realized in various ways, but essentially, it consists of one step where the RNS number is transformed to binary using base B_n and a second step where the binary number is transformed to RNS using base $\acute{B_n}$. As such, base extension is strongly related to the methods of transforming an RNS number to binary. Two main approaches to base extension are used in practice for RNS arithmetic: the MRS system and the Cox-Rower architecture introduced in [KKSS00].

While Cox-Rower method favors parallelism, the MRS system is often used for RNS Montgomery Multiplication base extension, because it offers benefits in terms of leakage resistance and fault propagation as discussed in [FPS17, BILT04, BEG13]. In MRS base extension, the base B_n RNS number is converted into a base B_n MRS number and then the base B_n MRS number is converted into a base $\acute{B_n}$ RNS number. A similar two step procedure is followed for base extension from $\acute{B_n}$ to B_n respectively. Most studies on

optimal base moduli [ESJ+13, BKP09] agree that moduli of the form $2^k \pm c_i$, $2^k - 2^{t_i} \pm 1$ or 2^k, $2^k - 1$, $2^{k-1} - 1$, $2^{k+1} - 1$ (Mersenne numbers) for various i values provide good performance results as well as n and k numbers that are also optimally determined. The RNS bases B_n and $\acute{B_n}$ dynamic range must be close to p ($4p < M$). Recent results from Bigou and Tisserand in [BT15] show how to perform RNS modular multiplication with a single base bit width instead of a double one, which results in two times faster implementation for the same area, but in this case the Leakage Resistant Arithmetic (LRA) approach followed in this paper cannot be used.

2.3 Using RNS for SCA resistance

In 2004, Bajard et al. in [BILT04] proposed, originally for modular exponentiation, a γ random permutation of the base B_n and $\acute{B_n}$ moduli thus creating $\binom{2n}{n}$ random permutations of B_n and $\acute{B_n}$ (i.e. creating $B_{n,\gamma}$ and $\acute{B_{n,\gamma}}$ for each such permutation). This approach leads to the LRA technique [BILT04] that theoretically can introduce enough randomness to thwart SCAs. Initial attempts to introduce LRA in scalar multiplication have been made in [Gui11] [Gui10], however, they are applicable only to the CRT type BE when pseudo-Mersenne numbers are used for base moduli. As suggested in [FPS17], when a permutation transition is done only once per scalar multiplication the vulnerability to horizontal SCA attacks is not eliminated, while when it is performed in every point operation of every round unacceptable performance overhead is introduced. So the most promising balance between performance and SCA resistance is to employ LRA once per scalar multiplication round [FPS17]. Apart from LRA, when performing modular operations in each individual channel i sequentially (during an RNS calculation), an additional SCA countermeasure can be devised by randomizing the sequence (order) of these modular operations. Considering the high number of RNS operations during a scalar multiplication, this sequence randomization can have a significant impact on information hiding.

2.4 RNS Scalar Multiplication Implementations

Considering that LRA can be a strong randomization tool in an SCA resistant scalar multiplication algorithm, capable of thwarting simple and differential, horizontal and vertical, SCAs, in this paper we adopt a MPL algorithm for scalar multiplication that is proposed and analyzed in [FPBS16, FPS17]. This algorithm realizes LRA as a random RNS Base permutation once per scalar multiplication round and it is combined with more traditional techniques like base point randomization, in order to provide resistance to a broader range of SCA.

In this paper, we extend the approach followed in [FPS17], implement several RNS scalar multiplication algorithms (with various countermeasures) and provide experimental results on the SCA resistance of the LRA technique when used autonomously, with RNS random operation sequence or in combination with traditional SCA techniques. To achieve that, using the MPL algorithm publicly available in [Fou] as a starting point, we implemented four different scalar multiplier variants in C software code for embedded system devices using the GMP 6.1.0 library. The GMP library was chosen for its usability and speed on calculating big-number values during scalar multiplication. Versions of GMP newer than the 6.0.0 release have side-channel resistant functions, such as silent modular operations without branching and constant time. Nevertheless, use of GMP during scalar multiplication is limited to basic RNS $GF(p)$ building blocks (modulo addition, subtraction) that are not directly related to sensitive information. We implemented our own big-number RNS-based Montgomery modular multiplication that is constant time. We also used the random generation function of GMP *mpz_urandomb*(), which is a common

software random generator. Therefore, we do not expect that this choice of library will affect the evaluation of our countermeasures in terms of side-channel leakage.

The first variant implements the original MPL algorithm [JY03] in RNS with no further countermeasures. The second variant implements an MPL optimization with base point randomization [Fou17, FV06], that is resistant against horizontal SCAs. This second variant takes advantage of the intrinsic mathematical coherence between each MPL round's points [Gir06], in order to structure a fault detection mechanism for first and second order fault attacks [FPS17]. Based on this property, We can evaluate if the round number i as well as the scalar e is not modified using Cyclic Redundancy Check (CRC) error detection codes.

To evaluate the LRA information leakage, we infused the MPL algorithm with the LRA technique in order to structure another two scalar multiplier variants. The third implementation consists of the original MPL algorithm with one random base permutation conversion per scalar multiplication round. The fourth scalar multiplication variant offers base point randomization and the LRA technique. We present the algorithms behind the second and fourth scalar multiplier implementations (Alg.1 and Alg.2 respectively). As it can be seen there is a need of transformation to Montgomery format and a Random Base permutation (RBP) conversion for various points in the algorithm. According to [BILT04], RBP can be done by performing two consecutive RNS Montgomery multiplications per point coordinate that use the old and new permutations' bases B_n and $\acute{B}_n$ in reverse order. All four variants of scalar multiplication can use a fixed or a randomized scalar as input and are implemented with and without RNS random moduli operation sequence.

Alg. 1. Blinded SCA-FA MPL [Fou17]

In: V, $R \in E(GF(p))$, $e = (e_{t-1}, e_{t-2}, ...e_0)$
Out: $e \cdot V$ or random value (in case of faults)

1. Choose base B_n and $\acute{B}_n$ (permutation γ_t). Transform V, R to RNS format using perm.γ_t
2. $R_0 = R$, $R_1 = R + V$, $R_2 = -R$ in perm. γ_t
3. *Convert R_0,R_1,R_2 to Montgomery format*
4. **For** $i = t - 1$ **to** 0
   (a) $R_2 = 2R_2$, performed in permutation γ_t
   (b) if $e_i = 1$
   $R_0 = R_0 + R_1$ and $R_1 = 2R_1$ in perm. γ_t
   else
   $R_1 = R_0 + R_1$ and $R_0 = 2R_0$ in perm. γ_t
   end if
6. **If** (i, e not modified (using CRC checks) and $R_0 + V = R_1$)
   then
   (a) return $R_0 + R_2$ in perm. γ_t
   else return (random value)

Alg. 2. LRA SCA-FA Blinded MPL [Fou]

In: V, $R \in E(GF(p))$, $e = (e_{t-1}, e_{t-2}, ...e_0)$
Out: $e \cdot V$ or random value (in case of faults)

1. Choose random initial base permutation γ_t. Transform V, R to RNS format using perm.γ_t
2. $R_0 = R$, $R_1 = R + V$, $R_2 = -R$
3. *Convert R_0,R_1,R_2 to Montgomery format*
4. **For** $i = t - 1$ **to** 0
   (a) $R_2 = 2R_2$, performed in permutation γ_t
   (b) choose a random base permutation γ_i
   (c) RBP from γ_{i+1} to γ_i for R_0 and R_1
   (d) if $e_i = 1$
   $R_0 = R_0 + R_1$ and $R_1 = 2R_1$ in perm. γ_i
   else
   $R_1 = R_0 + R_1$ and $R_0 = 2R_0$ in perm. γ_i
   end if
5. RBP from γ_t to γ_0 for V
6. **If** (i, e not modified (using CRC checks) and $R_0 + V = R_1$)
   then
   (a) RBP from γ_0 to γ_t for R_0
   (b) return $R_0 + R_2$ in perm. γ_t
   else return (random value)

In all four implementations, $GF(p)$ operations are done in RNS. Performance optimizations are out of scope of this work. We aim to evaluate a straight-forward implementation and any sort of optimization might influence our results, which is undesirable at this stage of evaluation. As is typically implemented in literature and in order to avoid excessive operations unrelated to the scalar multiplication, values related only to RNS Bases moduli are precomputed and stored in memory for use in BE and random base permutation algorithms. Following the approach of [ESJ+13] and [BKP09] a 4 moduli RNS bases ($n = 4$) RNS realization was used in all four RNS implementations. For all the above approaches, a $GF(p)$ twisted Edwards EC was adopted with $a = 1$ and $d = 2$ where $p = 2^{192} - 2^{64} - 1$. Twisted Edwards Curves were chosen instead of Weierstrass ones since the first have never been tested under the RNS arithmetic framework. However, the twisted Edwards curve shape was used and not the equivalent Montgomery form that can be combined with MPL, in order to keep the solution generic enough so as to be somewhat usable for other EC types. To retain compatibility with NIST Curves and implementations

of other similar works [ESJ+13], the prime field was left to be $p = 2^{192} - 2^{64} - 1$ (although the implementation can be easily adapted to any Edwards Curve including the popular Curve 25519). The security of the chosen Edwards Curve does not alter the results of our leakage study, since it relies on the employed countermeasures. We performed experiments with the secure Edwards Curve (a=107, d=47, cofactor=4) as proposed in [BBJ+08] and came up with similar results that are presented in subsection 3.6.

For 192-bit length $GF(p)$, we employ RNS bases that have a 200 bit dynamic range consisting of four approximately 50 bit moduli ($k = 50$) for each involved RNS base B_n and $\acute{B_n}$ as suggested in [ESJ+13, FPS17, BKP09]. Since $n = 4$ there exist 70 different base permutations. The security level provided by 70 different bases might not seem enough, however this is a trade-off between memory cost and SCA resistance. Increasing the moduli number would result in a big pre-computation table (currently 70×70 50-bit numbers) used in every RNS base extension operation. Instead of increasing the number of moduli, we propose combining LRA with the low overhead technique of randomizing the RNS moduli operation sequence, offering 24 different combinations (for $n = 4$) for each RNS operation. By randomizing the sequence of the moduli operations, we take advantage of the basic RNS property of parallel computations, which offers a way of shuffling and enhances the side-channel resistance of an RNS implementation as verified by our experiments. This will enable us to create uniquely random computation patterns for each EC point operation and each MPL round.

3 Practical Evaluation of RNS using TVLA

3.1 Theory of TVLA

TVLA is a leakage detection procedure, initially proposed by Cryptography Research (CRI) [GJJR11] and used as a first step towards evaluation of the SCA resistance of device under test. A generic univariate test is used to scan the traces obtained from the device, which is evaluated as non-leaky if the tests at all points of every trace are below a certain threshold. The statistical tests are chosen in a way that is independent of the leakage model. More precisely, we test the case that there is no leakage (null hypothesis) versus the case that there is leakage at a certain intermediate point L_P. Let n_{tr} be the number of traces that the evaluator collects and n_s the number of samples in each trace. Following the notation of [ZDD+17], let $\mathbf{L} = \{L_1, \ldots, L_{n_s}\}$ be the measurement traces representing the realization of our implementation with mean values $\bar{L}_i$. The null hypothesis for our traces means that the expected value from our measurements is the same as the measured value, if there is no leakage, i.e. $\bar{L}_{exp} = \bar{L}_i \quad \forall i \in \{1, \ldots, n_s\}$.

Following the methodology of [GJJR11, TG16, NLD15, CMV+17], there should be two sets of traces, A and B, with $n_{tr}/2$ traces in each set; half of the traces are taken with a fixed input and half with random input. If the null hypothesis holds, there should be no differences in the t-test statistics measured from each trace set. The Welch's t-test is commonly used for TVLA evaluations. The test statistic value is:

$$s_i = \frac{\bar{L}_{i,A} - \bar{L}_{i,B}}{\sqrt{\frac{\sigma^2_{i,A}}{n_A} + \frac{\sigma^2_{i,B}}{n_B}}}. \tag{1}$$

For the Welch's t-test, current TVLA evaluations use the critical value of 4.5 [GJJR11, TG16, NLD15, CMV+17], which corresponds to a statistical significance level of $a < 0.00001$ for the univariate test. In [CMV+17] the authors observe some peaks above the threshold for the protected implementation, which ended up to be ghost peaks. Their rationale is that first they observed few of these peaks in the fixed versus random t-test. Then, they

took two trace sets with random values and applied the same test. The peaks observed before did not appear again at the same spots, meaning that they were not dependent on the input. [BGRV15].

However, in most previous works, the significance level of $a < 0,00001$ does not consider the total number of samples on the trace. As noted in [ZDD+17], the overall significance level increases as the number of leakage points on the trace increases. Therefore, the authors propose to adjust the significance level according to the number of points on a trace. For long traces, meaning for more than 10^5samples per trace, the overall test statistic value will be larger than ± 4.5 and therefore a non-leaky device can not pass the TVLA t-testing with the critical value of ± 4.5. Hence, Balasch et al. [BGG+14] suggested raising the critical value to ± 5 for longer traces based on numerical experiments. For longer traces, as the ones obtained from the implementation of public key algorithms, a non-leaky device can not pass the t-test even with this higher value of ± 5.

3.2 Proposed TVLA threshold for public key algorithms

One of the applications of Welch's t-test is to test the location of one sequence of independent and identically distributed random variables, as the ones obtained from SCA measurements. In such a trace set, we are interested in s different null hypotheses, $H_1, \ldots, H_s$, where s is the number of samples in each trace, and we would like to check if all of them are true. The probability of making one (or more) false discovery when performing multiple hypotheses tests, the so-called family-wise error rate (FWER) or type I errors, is relevant to the number of samples. The more samples we get per trace, the higher the FWER will be. In order to calibrate the significance level, and consequently the threshold of the test, when multiple sequences of variables are tested, the Šidák correction should be applied.

The above mentioned calibration makes more sense in the case where we deal with public key algorithms, where the number of samples per trace are in the order of millions. The Šidák correction as defined in [FHY07] is: $a_{SID} = 1 - (1 - a)^{(1/n_s)}$. This formula appeared recently in the side-channel setting in [ZDD+17], where the authors identified the need to correlate the significance level with the total number of univariate tests. More precisely, they showed that for a trace set of 10^6 samples per trace, the probability that the t-test will fail for the 4.5 threshold is 0.9987, reducing only to the half for a threshold value of 5.

Taking all these into consideration we created a Matlab script which calculates the threshold value based on the number of samples and the variance of each sample in a given trace. We basically calculate the t-value according to the Šidák correction; the larger the number of traces obtained, the closer is the t-test value to the normal distribution's value. By using Alg.3 and $5M$ samples per trace, we obtain a threshold of ± 6, which suggests that the boundaries can be relaxed further. This result is generic and should be applied, in order to define the threshold before any t-test evaluation starts. It makes more sense to use this algorithm when public key cryptographic implementations are used or in any case that the traces contain more than 1M samples, because of the false positives that would appear with the threshold set to ± 4.5

At this point, it is worth noticing that the above observation affects public key cryptographic TVLA evaluations and gives a reasonable explanation for the results of previous papers. For instance, Nascimento et al. [NLD15] with 400k samples and Chmielewski et al. [CMV+17] with 32M samples for their t-tests observed some peaks above ± 4.5, which they discarded as false positives. Using 400k and 32M samples in our Matlab script gives threshold values of ± 6.7 and 7.3 respectively, which would evaluate their implementations as secure with no false positives. Therefore, it is important to adjust the boundaries for every set of traces.

Alg. 3. T-test Threshold

In: number of traces for group A and B nt_A, nt_B, number of samples n_s, sampled standard deviation σ_A, σ_B
Out: threshold value for Welch's t-distribution th_t

1. Choose level of significance a. Here $a = 0.00001$.
2. Family-wise error rate $fwer = 1 - (1 - a)^{n_s}$
3. Šidák correction $sidak_a = 1 - (1 - a)^{(1/n_s)}$
4. $df = \left(\frac{\sigma_A^2}{nt_A} + \frac{\sigma_B^2}{nt_B}\right)^2 / ... \left(\left(\frac{\sigma_A^2}{nt_A}\right)^2 / (nt_A - 1) + \left(\left(\frac{\sigma_B^2}{nt_B}\right)^2 / (nt_B - 1)\right)\right)$
5. Threshold $th_t = tinv(1 - sidak_a/2, df)$

3.3 Experimental setup

For the experiments we chose to load the implementation on a BeagleBone Black, which is a typical processor for portable devices. Apart from being a widely used processor, its high frequency of 1 GHz is necessary for our experiments, since a software implementation of RNS requires a lot of computational power to operate. As an indication, a full scalar multiplication for a 192-bit scalar takes 1,5 seconds when both SCA countermeasures are activated. The experimental setup we used is the following:

- BeagleBone Black with Cortex A8 processor running at 1GHz.
- EMV Langer probe LF B-3, H Field 100kHz up to 50 MHz.
- Lecroy Waverunner 8404M-MS sampling at 2.5GS/sec.

We used the secure RNS scalar multiplication algorithm from the public repository [Fou] modified with the appropriate functions to perform various types of t-tests and template attacks. For the analysis of our traces we used Matlab R2016b and the Inspector 4.12 software for Side-Channel Attacks provided by Riscure [Ins].

3.4 Processing of traces and alignment technique

Misalignment of traces is a common obstacle in security evaluations. It is often due to the noise of the target device or it is imposed on purpose as "jitter", in order to make the target device more resistant to SCA. Having our implementation on the BeagleBone Black, where other processes of the operating system are running at the same time, it was expected to have some noise from the device. However, this is what makes a target and a leakage assessment more realistic. With some common signal processing techniques, such as the absolute value (ABS) operation, the window resampling and low pass filter, we could get a clear signal and we were able to perform TVLA. The misalignment due to random interrupts is handled by acquiring a big number of traces (in some cases we reached 50k traces), and by throwing out the traces that had interrupts. These were not many, mostly it was about 1/10 of the total number of traces.

The *ABS* operation, as its name suggests, results in traces where the absolute value of each sample is depicted. The Inspector module of ABS computes the average value of each sample and uses it as a reference offset value. The result for each trace s_i in a trace set S is: Abs_offset$(s_i) = |s_i - \text{offset}|$.

Window resampling is a technique, where the samples of the acquired trace are resampled in a window of a desired length, in order to "clear out" some noise from the trace and make it shorter and easier to process. This technique offers a Signal-to-noise-ratio improvement by averaging several samples that carry the same signal. A second advantage is performance improvement. The compression of many samples into one, results in smaller traces which can be processed faster. As with every resampling technique, it results in some loss of information, which could possibly include leakage points and lower the leakage levels. This

loss of information is prevented in our experiments by using a 75% of overlap of samples, at the cost of performance. After experimenting with different window values, 200 samples per window with an overlap of 75% gave good results, with minimal information loss and without affecting the leakage levels.

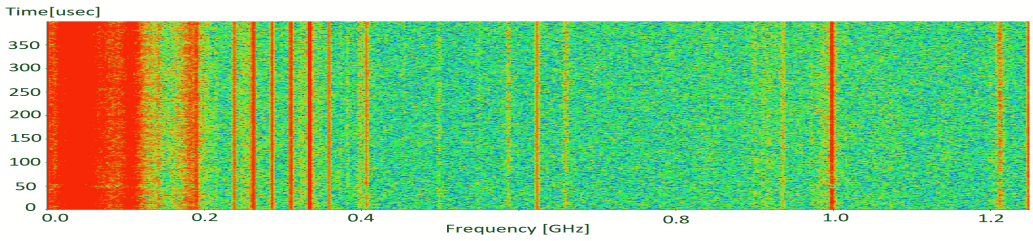

Figure 1: Applying FFT to find the dominant frequencies

A technique that we applied to get clear patterns for alignment is the *Low Pass Filter*, which allows only certain frequencies to pass. From the Fast Fourier Transformation (FFT), we found the dominant frequencies of our device during execution of the cryptographic algorithms. As seen in Figure 1, the maximum energy segments are at $0 - 300$ MHz and a high frequency appears also at 1 GHz, the running frequency of our processor. As seen in Figure 2, applying a low-pass filter in a trace, would make patterns, as the selected one, more clear in all the traces. We chose these repetitive patterns to perform alignment.

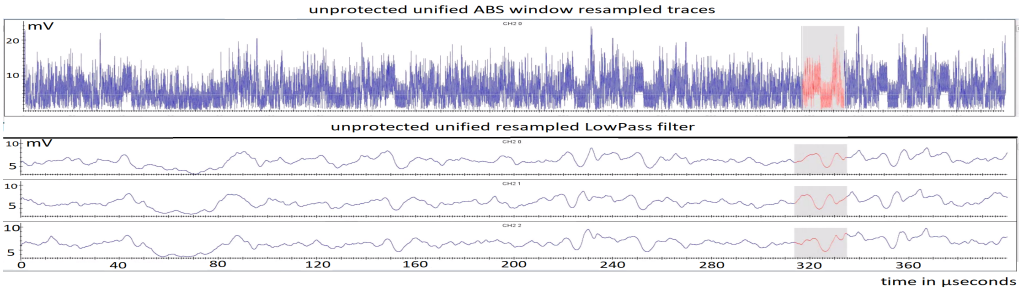

Figure 2: Applying low pass filter to find good alignment points

3.5 TVLA Analysis & Results

The leakage analysis for the RNS implementation was applied for the following four cases:

1. The unprotected version, where no countermeasures are incorporated apart from the fact that a constant time MPL implementation is used (`unprotected()` function).
2. Scalar multiplication with randomized input point (`rdm_point()` function).
3. Scalar multiplication with random base permutations (`LRA()` function).
4. The fully protected version of scalar multiplication, where both point and base permutation randomizations are applied (`LRA_rdm_point()` function).

We performed five sets of experiments. In the first experiment set, we performed t-tests for fixed versus random scalar, when the countermeasure of scalar randomization is not used. The scalar is usually the secret value of the protocols used in public key cryptography; for instance during scalar multiplication or during signing, the aim is to hide the scalar,

i.e. the private key, from an adversary. Therefore, it is interesting to test the correlation of the implementation traces to random versus fixed scalar. In Section 3.5.1, we kept the secret scalar unmasked. In a second set of experiments, we differentiated between random and fixed input point. In Section 3.5.2, we present a series of t-tests for the four variations of the RNS implementation, in order to evaluate the correlation of the implementation leakage to a fixed or a random input point. Then, we implemented and tested scalar randomization. In the experiment set of Section 3.5.3, the t-tests for fixed versus random scalar show the correlation between the randomized scalar and the alternations between random and fixed values of it. Section 3.6 shows the same set of experiments using random versus fixed point and the implementation with unified formulas on the secure twisted Edwards curve. Finally, in Section 3.7 we perform t-tests on the implementations with the LRA technique and the RNS random moduli operation sequence technique.

All sets of experiments that we performed, followed the rationale presented initially in [JRW11]. The acquired traces are compared with a constant scalar−constant input point test vector, in order to reveal any systematic relationship between the power consumption and the secret scalar or the input point respectively. The test fails if there is a single point in time where the t-value for the chosen set exceeds +6 or −6, the new t-test boundaries that we defined in subsection 3.2 due to the large trace length of RNS scalar multiplication algorithm.

3.5.1 T-test random versus fixed scalar

At first we performed the t-test in raw traces, without any pre-processing and alignment. Since we are interested in the leakage of the scalar, we performed the t-tests with fixed versus random scalar[1]. At first, we observed that the levels of noise and misalignment were so high, that even in the unprotected case, the leakage was not significant. This is depicted in Figure 3, where we see that there is leakage in the beginning of the traces, related to the processing of the scalar k. However these peaks disappear in the next rounds due to misalignment. In Figure 4, we notice a peak at the point where we performed alignment. Apart from the fact that window resampling offers a natural way of alignment, we also

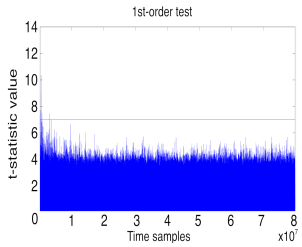

Figure 3: Not aligned traces

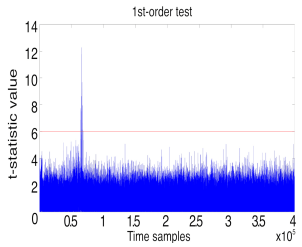

Figure 4: Alignment around 60000 samples

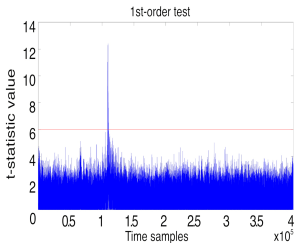

Figure 5: Alignment around 130000 samples

performed static alignment in the resampled traces following the procedure described in 3.4. Aligning at interesting patterns rather than interesting points is useful in our case, since we have a large number of points in all experiments, ranging from 10^5 to $8 \cdot 10^5$.[2] High density of leakage peaks is observed at the places where alignment is applied. It is noticeable that the leakage "moves" on the trace following the alignment points, as shown in Figures 4−5.

[1] At this point, we assume that we are able to control the input of the secret scalar, but in later experiments we have more realistic attack scenarios, where we can manipulate only the input points.

[2] When the number of samples is small, the method to find interesting points of Section 4.2 in [PITM13] can be followed.

It is important to note at this point, that despite having two randomization countermeasures switched on, there is leakage from the unprotected scalar. This fact, indicates the necessity to apply scalar randomization for a secure implementation, even if it might be a costly countermeasure. The following figures show the t-test results for all four cases for aligned traces obtained by interchanging between random and fixed scalars. Figure 6 shows the leakage for the unprotected implementation where both countermeasures are switched off. Figures 7– 8 show the leakage when there is only the randomization of the input point or the random base permutation (LRA) respectively. In Figure 7 we can identify the manipulation of the scalar in every round and the results are in accordance with our expectations, as we have collected about 7 rounds of the implementation for the `rdm_point()` case. An obvious observation is that the introduction of the LRA technique effectively reduces leakage compared to Figure 8 t-test. Thus, it seems that the LRA technique is more effective as a countermeasure when compared to input point randomization. Finally, Figure 9 shows the leakage of the implementation with combined countermeasures. It is important to note here that the leakage of the scalar happens in the beginning of the execution, where most probably the location of the scalar is accessed and the value of the scalar retrieved, in order to get its most significant bits and start the bitwise scalar multiplication. This leakage is expected, since no scalar randomization is implemented in this case. It is also expected to see the leakage disappear after a few rounds, because the other two countermeasures are taking place and they are able to hide the scalar as well. What is not expected is that the combination of randomized base permutation (LRA) with randomized input point increases leakage, especially in the first MPL rounds. This fact might be platform specific and it seems to be possible to launch an attack during the first two MPL rounds of this implementation.

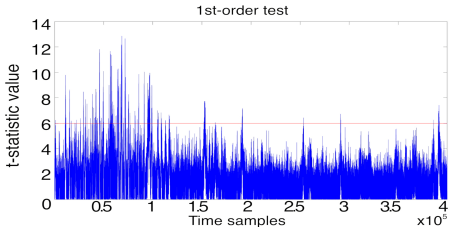

Figure 6: `unprotected()` random vs fixed scalar

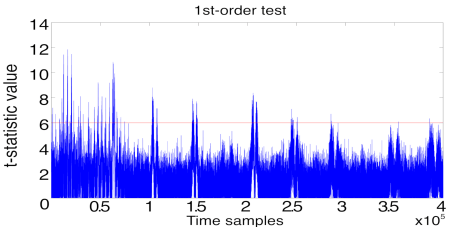

Figure 7: `rdm_point()` random vs fixed scalar

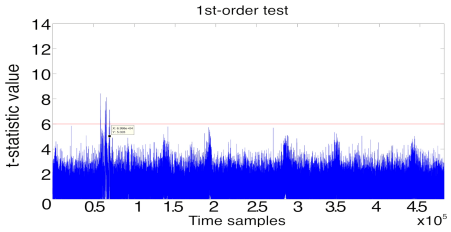

Figure 8: `LRA()` random vs fixed scalar

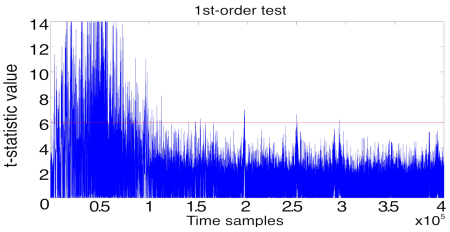

Figure 9: `LRA_rdm_point()` random vs fixed scalar

3.5.2 T-test random versus fixed input point

As a next set of experiments, we take a set of randomly interleaved acquisitions of fixed versus random input point. The scalar is fixed and the countermeasures of randomized

input point and/or random base permutations are applied as before. It is obvious for the t-test illustrations that the leakage in this case is significantly smaller than in Section 3.5.1. In Figure 10 we see that there is significant leakage in the first rounds that reduces later, but this is only due to misalignment caused by the noise. When we set the trigger at a later round, starting for instance at the 10th scalar bit, we got a similar picture, with many leakage peaks in the beginning of the t-test. Figures 11−12 show that applying one of the two proposed countermeasures is enough to reduce the leakage of the implementation only around the aligned area, which might be enough to exploit the implementation, but due to the limited number of leakage points it would be hard to perform a successful attack. Finally, Figure 13 shows that our protected implementation passes the t-test as expected.

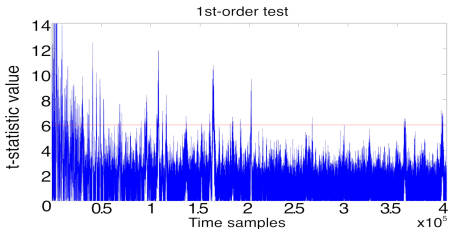

Figure 10: `unprotected`() random vs fixed input point

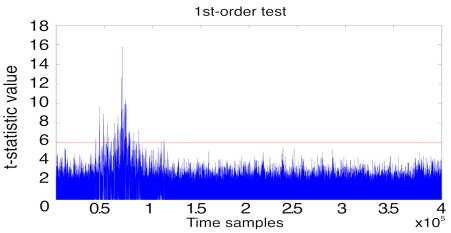

Figure 11: `rdm_point`() random vs fixed input point

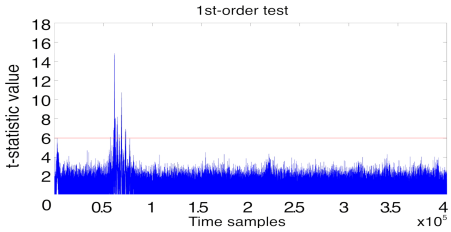

Figure 12: `LRA`() random vs fixed input point

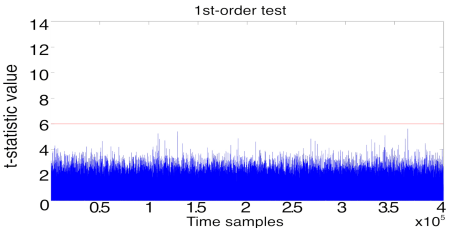

Figure 13: `LRA_rdm_point`() random vs fixed input point

3.5.3 T-test random versus fixed scalar for randomized scalar variation

The evaluation of a countermeasure with TVLA consists of distinguishing between random and fixed values of a certain parameter. If this parameter is randomized and used as countermeasure, then an adversary should not be able to distinguish between a random or a fixed value of it. This is the case for this part of our experiments. T-tests for fixed versus random scalar, when the scalar randomization countermeasure is applied, show that there is no leakage in all four cases as it is expected. The same pre-processing steps of absolute value, window resampling and alignment are applied as before. Figure 14 indicates the t-test results for the scalar multiplication variant of MPL with RNS without input point randomization and LRA, only the scalar randomization is applied. Since we perform 1st order t-tests and we blind the secret scalar, our implementation passes the statistical tests. Depending on the order of the masked value and the order of the t-test, it can happen that higher order t-test will fail. For instance, if additive splitting of the scalar with more than 2 shares is used, then the 1st order t-test should fail. For this work, we performed only 1st order t-tests and we blind the scalar by adding to it a random multiple of the order of the group. As indicated in all figures applying randomized scalar combined with the other RNS countermeasures can give a secure implementation.

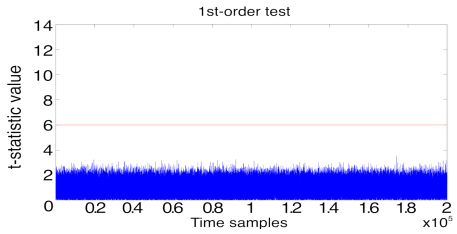

Figure 14: random vs fixed scalar`unprotected_scalar_rdm`()

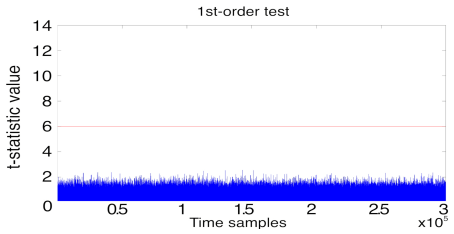

Figure 15: random vs fixed scalar `rdm_point_scalar_rdm`()

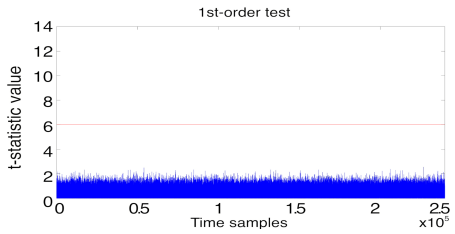

Figure 16: random vs fixed scalar `scalar_rdm_LRA`()

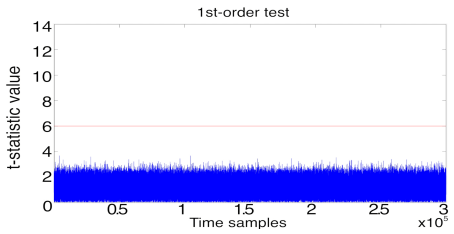

Figure 17: random vs fixed scalar `scalar_rdm_LRA_rdm_point`()

3.6 Secure twisted Edwards curve and unified formulas

The TVLA results presented above are independent of the chosen curve and the applied group law. The results of Sections 3.5.1- 3.5.3 are performed on the Edwards curve $a = 1, d = 2$, which is not included in the list of secure curves according to [BCLN16]. The tests of this section are performed on the secure twisted Edwards Curve $(a = 102, d = 47)$ [BBJ+08] and the results are similar to the previous ones. We also replaced the dedicated group law (which was chosen for efficiency reasons in the first series of experiments), with the unified group law. The figures 18- 21 show the t-tests for the case of random versus fixed input point.

As we see from these figures, the number of samples is of order 10^4, instead of 10^5 as it was in previous experiments. Higher misalignment levels in the new traces compared to previous ones, led us to perform the processing steps with different parameters. Apart from that, the graphs are similar to that of Section 3.5.2, with much leakage for the unprotected version, less leakage when one countermeasure is applied, and no leakage for the fully protected version. In Figure 20, we see that applying only randomization of the point cannot hide the correlation between random and fixed points, and this leakage is spread throughout the trace, not only in the aligned area as before. This might be due to the fact that we achieved better alignment in this set of traces, so the leakage is more obvious.

3.7 Secure twisted Edwards curve with randomized RNS operations

In this section, we examine the TVLA results of two RNS-specific countermeasures, namely the LRA technique and the RNS random moduli operation sequence technique. We notice that the leakage is significantly less compared to the results of the previous sections for the case of LRA. This is due to the fact that extra randomization is added in the implementation. Figure 22 shows the results for random versus fixed scalar, when the scalar is unblinded. Leakage is observed as expected, but it is much less compared to figure8 from previous results. Figure 23 shows the results for random versus fixed scalar, when the

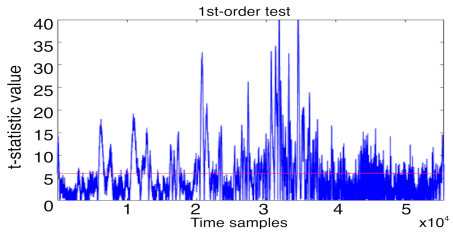

Figure 18: `unprotected_new_curve`() rdm vs fixed point

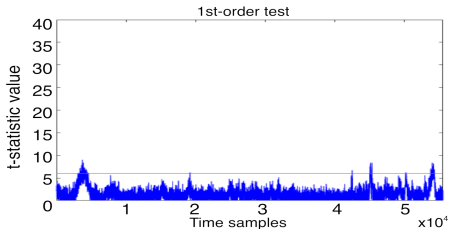

Figure 19: `LRA_new_curve`() rdm vs fixed point

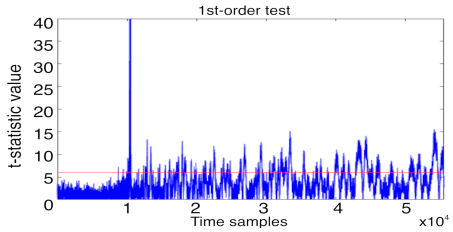

Figure 20: `rdm_point_new_curve`()rdm vs fixed point

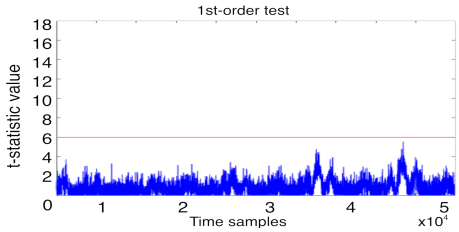

Figure 21: `LRA_rdm_point_new_curve`() rdm vs fixed point

scalar is randomized. We notice only one peak around the 38000 sample, which is above the threshold and can be considered a ghost peak, since it disappears when we perform random vs random TVLA. Finally, Figure 24 shows the results for random versus fixed point, when the point is random and shows clearly no leakage. This is an improvement over previous results for LRA in Section 3.5.2. From the above results can be concluded that the combination of the two RNS countermeasures lead to reduced leakage compared to implementations with single countermeasures or LRA with base point randomization.

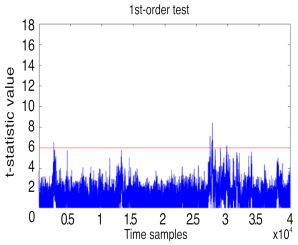

Figure 22: rdm vs fixed scalar

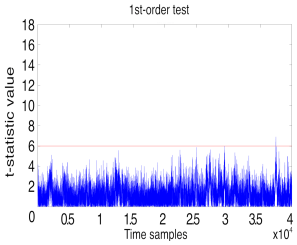

Figure 23: rdm vs fixed scalar

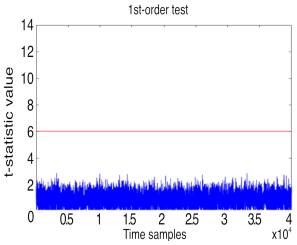

Figure 24: rdm vs fixed point

4 Template Attacks on RNS scalar multiplication

TVLA offers a generic framework for evaluating an implementation and the t-tests presented in the previous section show the expected leakage of RNS with the combination of LRA and traditional countermeasures. In this section, we perform template attacks in the different variations of the implementation, in order to further examine the behavior of the RNS-LRA and randomization or RNS operations techniques and to validate the fact that certain

countermeasures are necessary to prevent leakage of the implementation. The attacks explore data dependent and location dependent leakage and give similar results for all cases. When the randomized scalar or operations countermeasures are activated the results are different as expected. As an evaluation metric for the leakage of the RNS algorithm we used *Perceived Information (PI)* introduced to SCA by Renauld et al. [RSVC+11]. Compared to the mutual information metric, which assumes a hypothetical adversary who can perfectly profile the leakage, PI uses actual estimation procedures and practical traces to profile the Probability Density Function (PDF) of the implementation. More precisely, following the steps to estimate the PI of the implementation of RNS on the BeagleBone, as defined by Durvaux et al. in [DSVC14] we first collected profiling traces to estimate the leakage model. Then, we collected a set of test traces to estimate PI corresponding to the actual leakage of the chip. Finally, we evaluated the estimation errors and the assumption errors from which the misclassification percentage can be calculated.

Estimation errors occur when the number of collected traces is too low to estimate the model properly. In our approach, we avoid estimated errors by collecting 50k traces. After alignment about 20k traces are left, for which the template classification results were not so high. For 80% alignment threshold and the resulted 20k aligned traces, we get success rates between $65-70\%$. These success rates are rather low, if we want to avoid assumption errors. Assumption errors can occur when the template model may not be able to correctly predict the distribution of samples, even after intensive profiling. When the alignment is stricter and by using a group of 10 traces for detection of the correct group instead of a single trace each time, then we can reach success rate of 99% as described later.

4.1 Data dependent leakage

Data dependent leakage is observed when the value of a secret variable can be monitored by an adversary. This happens when the variable is unprotected. Leakage can also be observed when that specific variable is protected, but at the observation time the variable is sent or retrieved from a memory location in clear view. In our case, the unprotected scalar is used during scalar multiplication in the variation of the unprotected RNS implementation and the one with LRA countermeasure activated. The only case that behaves differently in data dependent leakage is the variation that uses scalar randomization or LRA randomized operations.

The key dependent assignment, the `if`-statement in Algorithm 1-step 4b (or Algorithm 2-step 4d), is the one that could be observed for this experiment. We created templates for scalars `s1 = 0x000001000···000` and `s2 = 0xFFF0···00F`. Since we wanted to observe only one instruction, we collected 50k traces of 700 samples each. After alignment, we kept around 3k-7k traces (more precisely 3385 traces for the unprotected case, 3132 traces for the protected one, 7622 traces for LRA-rdm_operations), from which we used half to create templates and the other half to test if the classification in the correct template group is successful. Figure 25 shows the acquired traces from the protected (top) and unprotected (bottom) implementations. The selected part from each trace is used for alignment for both variations of the algorithm for 700 samples. For the training of the templates and the classification we used only the aligned part for each group of traces, which is about 350 samples long as shown in Figure 26. The success rate of the unprotected algorithm is 91,73% for group 1 corresponding to the current bit e_i being 0, and 90,54% correctly classified traces to group 2, corresponding to the current bit being 1. The corresponding results for the LRA_rdm_point protected version is $97,47\%$ for $e_i = 0$ and $82,12\%$ for $e_i = 1$. The results when one countermeasure is activated give similar percentages of classification, above 90% of success rate. We did not perform the actual attack to recover each key bit, we just show here with high classification results, that with template attacks it is possible to distinguish bit 0 from bit 1.

Scalar randomization and randomized RNS operations seem to be an efficient counter-

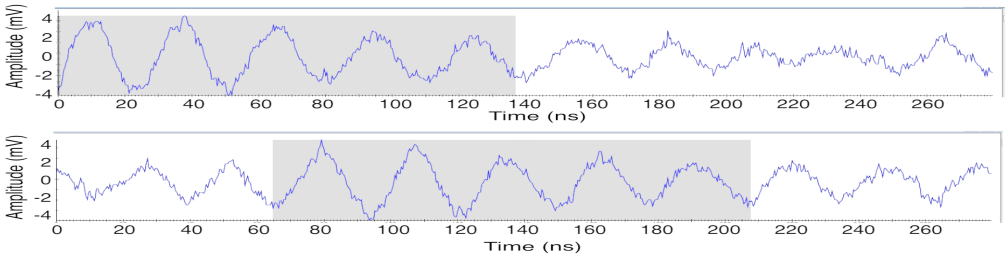

Figure 25: The selected part for alignment the protected and unprotected implementation

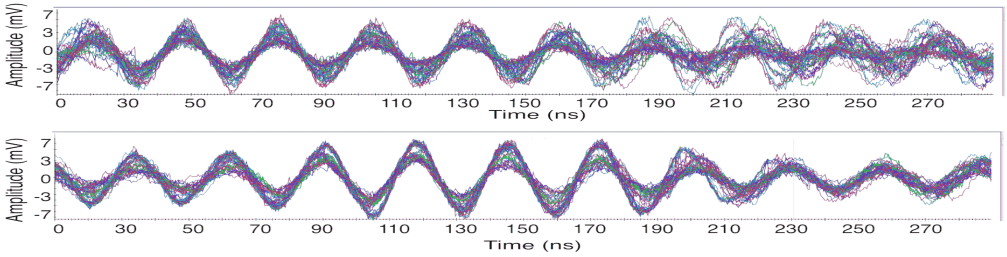

Figure 26: Aligned template traces for scalar assignment protected (top) and unprotected (bottom) algorithms

measures against data dependent template attacks. We performed the same procedure as with the previous sets of traces. More specifically, we collected 50k traces for all variations of the implementation. Then we performed alignment, and trained templates from the aligned part of the traces. We note here that we use only the selected part to train and classify templates.

The initial results give a success rate of 65-72% for all four variations of scalar randomization, which are quite low to give us confidence for the classification results. This might be a result of the bias of the GMP randomization function. We could not achieve higher success rate for this amount of traces and it is possible that more traces would eliminate the bias and give better results. However, since we perform the attack on the blinded scalar, it is expected that the data dependent leakage is very low in this case and therefore, simple classification techniques cannot separate the two sets as before. Clustering algorithms from machine learning, as applied in [OPB16, PITM14] might give better results for the scalar randomization version. Finally, we tried to classify the template traces for the LRA_rdm_operations() function and we got $55 - 58\%$ of correct classification, which shows that by using this combination of RNS countermeasures randomizes the data enough to make template attacks hard to succeed.

4.2 Location dependent leakage

In this subsection, we present location dependent template attacks. The templates are created based on the storage structure that handles a key-dependent instruction, in our case the doubling during scalar multiplication. As it is indicated in Algorithm 2 step 4(d), during an MPL round an addition and a doubling of points on the curve happen every time in the same order. The only thing that differs according to the current scalar bit is the manipulated register. More specifically, when the current scalar bit is 1, then the content

of storage variable R_1 is doubled, otherwise R_0 is doubled. We exploit this vulnerability and the fact that the implementation has no memory address randomization. By capturing the doubling operation, we can successfully create and classify templates for R_0 and R_1. In this way the scalar could be recovered bit-by-bit. This sort of memory access leakage is exploited in [PITM14] for the case of RSA using RNS, where the authors used unsupervised learning and clustering algorithms to classify their traces. We show here how to obtain high classification results using template attacks for the case of ECC with RNS on our software implementation.The template classification results for all four variations of scalar multiplication vary between 87-99% and thus give a very high probability of a successfully launched template attack. We calculated the two normal distributions for $e_i = 0$ and $e_i = 1$ for every variation of the implementation and they are indeed very different ($N(-24.3, 9, 7)$ and $N(19.6, 6.1)$). This is why the misclassification percentage is very low.

For the doubling operation, we collected 50k traces 3000 samples long. After alignment, we used the remaining 14k traces for the location dependent attack. We use half of the traces, i.e. 7k traces of 3000 samples, for training the templates, and the remaining 7k traces for classification[3]. Figure 27 shows the raw traces and selected part for alignment and Figure 28 shows the aligned traces for doubling in the case of the randomized scalar implementation. As previously, we use the 451-samples-long area to train and classify traces. The result of classification is correct with $99,44\%$ for group 1 and $99,97\%$ for group 2. For the case where randomization of the scalar is activated together with LRA, the situation is similar and the success rate of correct classification of templates reaches 95%. Only the case of LRA_rdm_operations shows lower classification levels of $70 - 83\%$ which indicates that combining two RNS-specific countermeasures makes template attacks harder to perform. Acquiring more traces, might give better results, but this might discourage the attacker, since already for 50k traces we need 20 hours of acquisition time.

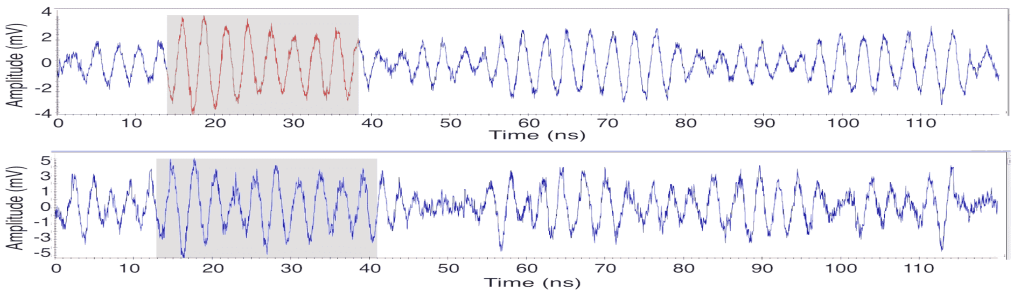

Figure 27: Selected area for alignment for protected and unprotected doubling

The registers mentioned in the algorithms are not really single registers that hold the result of doubling. In RNS all values are stored in 50-bit chunks (in our case), so the result of doubling is stored in 4 different memory locations. Therefore, the location dependent leakage is not an expected result for an RNS implementation and especially with the LRA countermeasure activated. We attribute the high success rate of this attack to the leakage of the platform and the fact that those chunks of values are probably stored in capacitors next to each other.

The fact that scalar randomization does not seem to be an efficient countermeasure against localized templates is also worth extra justification. We believe that the bits of the randomized scalar is what is correctly classified and can be recovered, not the initial,

[3]This number of traces may seem too low for an evaluation of a symmetric key implementation. For instance, Unterstein et al. in [UHSS17] performed EM location dependent templates for the case of AES S-Box and they used 1M traces for profiling. Acquiring 1M is unrealistic for public key evaluations, because it is not possible for the analysis tools to process this amount of information. Therefore, we have to draw our conclusions with the amount of traces that we can acquire.

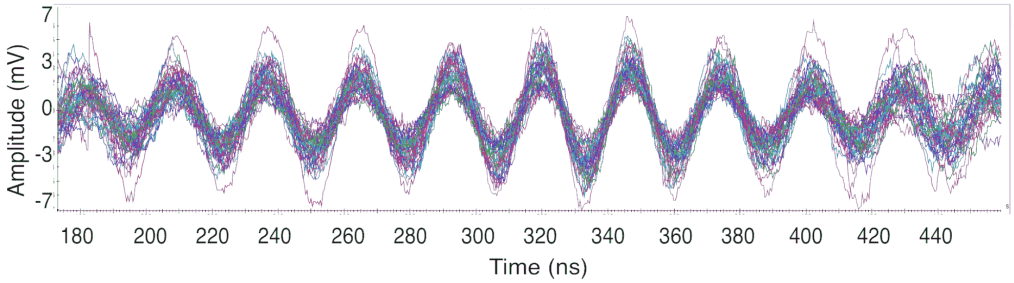

Figure 28: Zoom-in aligned traces for doubling operation with randomized scalar

unblinded scalar. Therefore, we can consider the location dependent template attack unsuccessful in this case.

5 Performance impact of countermeasures

In this section, we discuss the performance impact of the various countermeasures we implemented. At first, we note that performance optimizations were out of scope of our work. Therefore, we did not apply SIMD instructions or parallel computations of RNS operations. An unprotected implementation of RNS takes 0,726 sec. to perform a 192-bit scalar multiplication.

As a reference point we take the unprotected version of RNS and we measure the extra time needed as we add countermeasures. Applying point randomization costs more than performing random base permutations, because of one extra point doubling that is necessary. So we have 1,101 sec. for point randomization instead of 1,090 for the LRA algorithm. When both countermeasures are applied, the scalar multiplication takes 1,525 sec., which is more than double compared to the unprotected version. Adding scalar randomization to all the algorithms would add a $1-8\%$ performance overhead (PO). Adding unified formulas makes our algorithms $16-30\%$ slower. Using the LRA technique with random RNS operations takes 1,278 sec. and adds 76% overhead, which is much less than the combination of LRA and randomization of the point. At the same time, this combination leaks less and make data dependent template attacks not possible. It is therefore proposed as a better equivalent to tranditional SCA countermeasures.

Table 1 shows the performance evaluation and the attack possibilities for every algorithm in terms of this work. The ✗ sign shows that the corresponding algorithm does not pass the t-test or it is not protected against template attacks, while the ✓ sign shows a secure algorithm according to TVLA or against templates. The $-$ sign shows that we did not perform this attack, because we can expect very similar results. The N/A indicates that this type of attack is not applicable to the algorithm. The percentage numbers show the PO compared to the unprotected RNS implementation.

The TVLA experiments for RNS-MPL shows that the proposed RNS-LRA algorithm with combined countermeasures can indeed provide high levels of security compared to the simple RNS implementation or an implementation with a single countermeasure. More precisely, the systematic correlation between the power consumption and the input point can be prevented by using the classical countermeasure of point randomization and by randomizing the base point representation that is used in every round. A very interesting observation is that by using RNS LRA representation and randomizing the input point, but not the scalar, is leaking secret information. Another important remark is that the LRA countermeasure is the best trade-off between security and performance compared to traditional point blinding, which is usually $2-20\%$ more expensive to implement in RNS.

Table 1: Collective Evaluation Table

Algorithm	Welch t-test r-vs-f scalar	Welch t-test r-vs-f point	TA Data	TA Location	PO
`unprotected()`	✗	✗	✗	✗	0%
`rdm_point()`	✗	✗	✗	✗	52%
`LRA()`	✗	✗	✗	✗	50%
`protected_LRA()`	✗	✓	✗	✗	110%
`unprotected_rdm_scalar()`	✓	N/A	✓	✗	19%
`rdm_point_rdm_scalar()`	✓	N/A	✓	✗	54%
`LRA_rdm_scalar()`	✓	N/A	✓	✗	51%
`protected_rdm_scalar()`	✓	N/A	✓	✓	110%
`unprotected_unified()`	✗	✗	✗	✗	19%
`rdm_point_unified()`	✗	✗	✗	✗	99%
`LRA_unified()`	✗	✗	✗	✗	72%
`protected_unified()`	✓	✓	✗	✗	144%
`LRA_nc_rdm_operat()`	✗	✓	✓	✓	76%
`LRA_nc_rdm_operat_rdm_scal()`	✓	N/A	✓	✓	76%

6 Conclusions

In this section we summarize the conclusions of our practical evaluation of RNS implementations with various countermeasures. Different RNS representations of elliptic curve points, randomization of the input point, scalar randomization and the regularity of MPL during scalar multiplication are good countermeasures to protect against horizontal type of attacks and SPA. However, even in the presence of these countermeasures the TVLA results show that leakage is still present most of the times. When two countermeasures are combined, but the secret scalar is not randomized, there seems to be still exploitable leakage. The TVLA results are verified by high classification levels for two types of template attacks in the presence of countermeasures. When we randomize secret data, it is shown that data dependent template attacks are not successful, but the manipulated registers can still give very high classification results for the correct scalar guess.

Another important contribution of this work is for evaluators; they should not take the TVLA bounds as rigid, since our evaluation shows that for public key cryptography the typical threshold of ± 4.5 is not correct. TVLA threshold should be established every time according to the distribution of the traces and the number of samples.

As future work, it would be interesting to investigate further the possibilities of location dependent template attacks in the presence of countermeasures. The classification rates for templates of the algorithms that use randomization of the RNS operations is also an interesting follow-up work that could investigate clustering and other algorithms from machine learning. Furthermore, evaluating the combination of these countermeasures in a FPGA implementation of RNS with parallel execution of the RNS operations for various moduli sizes would give further insights in the security of RNS.

Acknowledgments

We would like to thank N. Sklavos for encouraging initial findings of this research and I. Kizhvatov for technical support at the lab of Radboud. We appreciate the recommendations and useful comments of the anonymous reviewers that helped us to improve this work.

References

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side channel(s). In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 29–45, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[BBJ+08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards Curves. In Serge Vaudenay, editor, *Progress in Cryptology – AFRICACRYPT 2008*, volume 5023 of *LNCS*, pages 389–405. SV, 2008. http://cr.yp.to/papers.html#twisted.

[BCLN16] Joppe W. Bos, Craig Costello, Patrick Longa, and Michael Naehrig. Selecting elliptic curves for cryptography: an efficiency and security analysis. *J. Cryptographic Engineering*, 6(4):259–286, 2016.

[BCP+14] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In *proc. of 15th International Conference on Cryptology in India INDOCRYPT 2014, New Delhi, India, Dec. 14-17, 2014*, pages 21–36, 2014.

[BDK97] J.-C. Bajard, L.-S. Didier, and P. Kornerup. An RNS Montgomery modular multiplication algorithm. In *Proc.13th IEEE Symp. on Comp. Arithmetic*, pages 234–239. IEEE Comput. Soc, 1997.

[BDK01] J.-C. Bajard, L.-S. Didier, and P. Kornerup. Modular multiplication and base extensions in residue number systems. In *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001*, pages 59–65. IEEE Comput. Soc, 2001.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT'97*, pages 37–51, 1997.

[BDM06] Jean-Claude Bajard, Sylvain Duquesne, and Nicolas Meloni. Combining Montgomery Ladder for Elliptic curves defined over Fp and RNS Representation. 01 2006.

[BEG13] Jean-Claude Bajard, Julien Eynard, and Filippo Gandino. Fault Detection in RNS Montgomery Modular Multiplication. In *IEEE 21st Symp. on Comp. Arithmetic*, pages 119–126. IEEE, April 2013.

[BEHZ16] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, pages 423–442, 2016.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, pages 64–81, 2014.

[BGRV15] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, Bitslicing and Masking at 1 GHz. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 599–619, 2015.

[BILT04] Jean-Claude Bajard, Laurent Imbert, Pierre-Yvan Liardet, and Yannick Teglia. Leak resistant arithmetic. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 62–75, 2004.

[BKP09] Jean-Claude Bajard, Marcello Kaihara, and Thomas Plantard. Selected RNS Bases for Modular Multiplication. In *2009 19th IEEE Symp. on Comp. Arithmetic*, pages 25–32. IEEE, June 2009.

[BT15] Karim Bigou and Arnaud Tisserand. Single base modular multiplication for efficient hardware RNS implementations of ECC. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 123–140, 2015.

[CATB18] Jerome Courtois, Lokman Abbas-Turki, and Jean-Claude Bajard. Evaluation of Resilience of randomized RNS implementation. Cryptology ePrint Archive, Report 2018/009, 2018. https://eprint.iacr.org/2018/009.

[CMV+17] Lukasz Chmielewski, Pedro Maat Costa Massolino, Jo Vliegen, Lejla Batina, and Nele Mentens. Completing the complete ECC formulae with countermeasures. *Journal of Low Power Electronics and Applications*, 7(1), 2017.

[CNPQ04] Mathieu Ciet, Michael Neve, Eric Peeters, and Jean-Jacques Quisquater. Parallel FPGA implementation of RSA with Residue Number Systems - Can side-channel threats be avoided? - extended version. Cryptology ePrint Archive, Report 2004/187, 2004.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[DSVC14] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. How to certify the leakage of a chip? In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 459–476, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[ESJ+13] Mohammad Esmaeildoust, Dimitrios Schinianakis, Hamid Javashi, Thanos Stouraitis, and Keivan Navi. Efficient RNS implementation of elliptic curve point multiplication over gf(p). *IEEE Trans. VLSI Syst.*, 21(8):1545–1549, 2013.

[FHY07] Jianqing Fan, Peter Hall, and Qiwei Yao. To how many simultaneous hypothesis tests can normal, student's t or bootstrap calibration be applied? *Journal of the American Statistical Association*, 102:1282–1288, 2007.

[FKSK15] Apostolos P. Fournaris, Nicolaos Klaoudatos, Nicolas Sklavos, and Christos Koulamas. Fault and power analysis attack resistant RNS based Edwards curve point multiplication. In *Proceedings of the 2nd Workshop on Cryptography and Security in Computing Systems, CS2 at HiPEAC 2015, Amsterdam, Netherlands, January 19-21, 2015*, pages 43–46, 2015.

[Fou] Apostolos P. Fournaris. RNS_LRA_EC_scalar Multiplier. https://github.com/afournaris/RNS_LRA_EC_Scalar_Multiplier.

[Fou17] Apostolos P Fournaris. *Fault and Power Analysis Attack Protection Techniques for Standardized Public Key Cryptosystems*, pages 93–105. Springer International Publishing, Cham, 2017.

[FPBS16] A. P. Fournaris, L. Papachristodoulou, L. Batina, and N. Sklavos. Residue number system as a side channel and fault injection attack countermeasure in elliptic curve cryptography. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–4, April 2016.

[FPS17] Apostolos P. Fournaris, Louiza Papachristodoulou, and Nicolas Sklavos. Secure and Efficient RNS Software Implementation for Elliptic Curve Cryptography. In *2017 IEEE Eur. Symp. Secur. Priv. Work.*, pages 86–93. IEEE, apr 2017.

[FV06] Guillaume Fumaroli and David Vigilant. Blinded fault resistant exponentiation. In *FDTC*, pages 62–70, 2006.

[FV12a] Junfeng Fan and Ingrid Verbauwhede. An updated survey on secure ECC implementations: Attacks, countermeasures and cost. *Cryptography and Security: From Theory to Applications*, 6805:265–282, January 2012.

[FV12b] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[Gir06] Christophe Giraud. An RSA implementation resistant to Fault Attacks and to Simple Power Analysis. *IEEE Trans. on Computers*, 55(9):1116–1120, 2006.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. P.: A testing methodology for side channel resistance validation, 2011. NIST non-invasive attack testing workshop.

[Gui10] Nicolas Guillermin. A high speed coprocessor for elliptic curve scalar multiplications over Fp. *Lecture Notes in Computer Science Advances in Cryptology Cryptographic Hardware and Embedded Systems CHES 2010*, pages 48–64, 2010.

[Gui11] Nicolas Guillermin. A coprocessor for secure and high speed modular arithmetic. *IACR Cryptology ePrint Archive*, 2011.

[HMH+12] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 231–244, 2012.

[HPS18] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. Cryptology ePrint Archive, Report 2018/117, 2018. https://eprint.iacr.org/2018/117.

[Ins] Inspector SCA Tool. https://www.riscure.com/security-tools/inspector-sca/. Accessed: 2017-12-14.

[JRW11] Josh Jaffe, Pankaj Rohatgi, and Marc Witteman. Efficient side channel testing for public key algorithms: RSA case study, 2011.

[JY03] Marc Joye and Sung-Ming Yen. The Montgomery Powering Ladder. In *4th CHES 2003*, pages 291–302, UK, 2003. Springer-Verlag.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *in Proc. of CRYPTO '99*, pages 388–397. Springer-Verlag, 1999.

[KKSS00] Shinichi Kawamura, Masanobu Koike, Fumihiko Sano, and Atsushi Shimbo. Cox-rower architecture for fast parallel montgomery multiplication. In *Advances in Cryptology EUROCRYPT*, 2000.

[MO09] Marcel Medwed and Elisabeth Oswald. Template attacks on ECDSA. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *Information Security Applications*, volume 5379, pages 14–27, 2009.

[NLD15] Erick Nascimento, Julio López, and Ricardo Dahab. *Efficient and Secure Elliptic Curve Cryptography for 8-bit AVR Microcontrollers*, pages 289–309. Springer International Publishing, Cham, 2015.

[OPB16] Elif Özgen, Louiza Papachristodoulou, and Lejla Batina. Template attacks using classification algorithms. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2016, McLean, VA, USA, May 3-5, 2016*, pages 242–247, 2016.

[PITM13] Guilherme Perin, Laurent Imbert, Lionel Torres, and Phillipe Maurine. Electromagnetic analysis on RSA algorithm based on RNS. In *2013 Euromicro Conference on Digital System Design (DSD)*, number 1, pages 345–352, 2013.

[PITM14] Guilherme Perin, Laurent Imbert, Lionel Torres, and Philippe Maurine. Attacking randomized exponentiations using unsupervised learning. In *COSADE*, volume 8622 of *Lecture Notes in Computer Science*, pages 144–160. Springer, 2014.

[RSVC+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 109–128, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[SS13] Dimitrios Schinianakis and Thanos Stouraitis. Hardware-fault attack handling in RNS-based Montgomery multipliers. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 3042–3045. IEEE, May 2013.

[TG16] Michael Tunstall and Gilbert Goodwill. Applying TVLA to Public Key Cryptographic Algorithms. *IACR Cryptology ePrint Archive*, 2016:513, 2016.

[UHSS17] Florian Unterstein, Johann Heyszl, Fabrizio De Santis, and Robert Specht. Dissecting leakage resilient prfs with multivariate localized EM attacks - A practical security evaluation on FPGA. In *COSADE*, volume 10348 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2017.

[ZDD+17] Liwei Zhang, A. Adam Ding, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. *IACR Cryptology ePrint Archive*, 2017:287, 2017.