CONCRETE EVALUATION OF THE RANDOM PROBING SECURITY

A PREPRINT

Vahid Jahandideh
v.jahandideh@gmail.com

June 19, 2021

ABSTRACT

We study the security of Boolean masking countermeasure when an adversary randomly probes internal variables of the masked algorithm, intending to recover non-trivial knowledge about its secrets. We introduce a novel metric called Secret Recovery Probability (SRP) for assessing the informativeness of the probing leakages about the masked secrets. To evaluate SRP, our starting point is to describe the relations of the intermediate variables with a parity equation system where the target secret is an unknown of this system. By a quantitative estimation of the $SRP(n, \epsilon)$, it is possible to measure the security as a function of the masking order n and the leakage rate ϵ for various masked constructions.

In contrast to the previous results in the asymptomatic model, our approach is in a concrete setting. Therefore, it can be used as an analysis tool for practical engineering purposes. Moreover, for the multiplication gadget proposed in *Ches* 2016, with some modifications in the construction, we put forward an upper bound for the $SRP(n, \epsilon)$ regarding the secret operands. The given upper-bound proves the multiplication gadget's security in the random probing for constant leakage rates $\epsilon < .07$. So, we provide the first secure and practical multiplication gadget in the random probing model.

As another contribution, leakage effects of refreshing construction is modeled with an equivalent erasure channel. Appropriate handling of the leakage of refreshing gadgets, instead of neglecting, was a long-standing challenge in the random probing environment. This modeling helps to estimate the $SRP(n, \epsilon)$ for complex structures. In this way, we give an unprecedented masked S-Box implementation with proved security for $\epsilon < 0.03$. We also study $SRP(n, \epsilon)$ of arbitrary order masking of AES, where we derive an innovative security bound that is independent of the size of masked implementation. Furthermore, in this paper, new insights into the connections of the SNI security in the threshold probing model with the security results obtained in the random probing model are developed.

1 Introduction

After more than a couple of decades since the introduction, side-channel attacks are a widely understood concept now. Indeed, multiple times, it is proved that direct implementation of a cryptographic algorithm without any protection might open the door for various adversarial attacks. In these attacks, leakages through physical measurements (also known as side-channel leakages) equip the adversary with extra information. This additional information, combined with publicly known parts of the input/output of the cryptographic algorithm, can even lead to the secret key recovery. Sources of leakages are various; among them are the algorithm execution time, instantaneous power consumption, and electromagnetic emanations of the device [1, 2, 3].

1.1 Modeling side-channel leakages

Expressing the knowledge that an adversary can extract from a side-channel leakage with a proper mathematical model is an essential prerequisite for designing a countermeasure. Various models for bounding the leakage data are given in the literature. However, none of them tightly formulates all kinds of leakage information that an adversary can access, see [4] for a recent review on the leakage models.

For power measurement leakages, the *only computation leaks* (OCL) assumption [5] matches the experiments well. The OCL assumption, at each interval of time, confines the leakage information to be a function of the variables accessed at that interval. Variables of an algorithm include all the operands appearing in the course of its execution.

For a software implementation, the list of variables is dependent on the architecture of the computing processor and the compiler optimization level. Here, we assume that algorithm code is written with the basic operations such as AND, XOR, and the randomness generation gates. So, the compiler does not intervene with processing steps. In our settings, the operands of these operations are variables of the algorithm. Some authors, refer to these operands as wires.

The *independent leakage* (IL) hypothesis simplifies the OCL assumption further. The IL conjecture states that the leakage corresponding to a variable is independent of other variables of the algorithm. In this way, IL implies that one can partition a given power leakage trace into non-overlapping intervals, where each interval provides information about a single variable of the algorithm.

The IL assumption is complying with the single-thread single-task software implementations. However, there are situations (especially at hardware implementations) where the IL assumption does not hold. See [6] and [7] for examples of dependent leakage.

The practically comprehensible *noisy leakage model* is based on the OCL and IL assumptions. According to this model, a noisy version of each variable is leaked to the adversary [8]. Authors of [9] demonstrated that *statistical distance* is an admissible metric for quantifying the *noise* of leakages. For a variable V in the algorithm, the adversary receives a function $\nu(V, R)$ as a final result of processing the leakages corresponding to this variable. Random variable R is the randomness used for modeling the noise. Function $\nu(.)$ is defined δ-noisy if $\mathsf{SD}(V, V|\nu(V, R)) \leq \delta$. Except for being δ-noisy, there is no other restriction over the function $\nu(.)$.

Definition 1. Noisy leakage. For a constant parameter $\delta \in [0, 1]$, the adversary in each execution of the algorithm learns variables of the algorithm with a δ-noisy function.

A somewhat simpler model compared to the noisy leakage model is the *random probing model* (RPM). In this model, the adversary gets (the exact value of) each intermediate variable independently and randomly with constant probability ϵ, and with probability $1 - \epsilon$ learns nothing about this variable. The value of ϵ is constant over the alphabet of the variable, and the adversary has no control over the randomness of the disclosure event. In this way, our definition for RPM is in line with the ϵ-random probing definition given in [9]. Parameter ϵ is also known as *leakage rate*. A somewhat similar definition for random probing is given in [10], which they call ϵ-average probing. In ϵ-average probing, it is assumed that different values of a variable V can experience non-identical values of ϵ. However, we make no such assumption.

Definition 2. Random probing leakage. For a constant parameter $\epsilon \in [0, 1]$, the adversary in each execution of the algorithm learns variables of the algorithm with probability ϵ.

For a practitioner assessing power-traces of cryptographic devices, it seems that the noisy leakage model better fits with the observations. However, the breakthrough work of [9] proved that the (for sufficiently small values of δ) noisy leakage model is reducible to the (noise-free) *random probing model* (RPM). This means that the security results -irrespective of the metric used for measuring the security- obtained in the random probing model can be directly translated to the noisy leakage model.

As pointed in [11], the output of random probing leakage for a variable is distributed identically to its reception via an *erasure channel*. In this paper, as our main technical tool, we will expand this resemblance from just a single variable to more complex structures.

The list of leakage models has another member namely the *threshold probing model* (TPM). In TPM, the adversary learns at most a fixed number of variables of the algorithm [12].

Definition 3. Threshold probing leakage. For a constant parameter $t \in \mathbb{N}$, the adversary in each execution of the algorithm learns the value of at most t variables of the algorithm.

TPM is mathematically tractable and straightforward. Nevertheless, the assumption that leakage information is restricted to a limited number of variables is superficial. Not quite surprisingly, various works have shown that TPM does not conceivably capture real leakage scenarios [13].

For the three discussed leakage models, *masking* is a well studied and practically used countermeasure. A review of Boolean masking with the developed techniques for applying it over block ciphers is presented in part 2.1. At *masking order* n, the masked counterpart of an algorithm $\mathcal{A}$ is denoted by $\mathcal{A}'(n)$ in this paper.

For assessing the effectiveness of a countermeasure, a precise definition of the security is needed.

1.2 Definitions of the security

Security in TPM is stated as a threshold number t, where any set of variables in $\mathcal{A}'(n)$ with at most t members reveal nothing about the variables of $\mathcal{A}$. To be more accurate, not all of the variables of $\mathcal{A}$ are interesting for the adversary. Only the so-called *sensitive* variables that are key-dependent are important.

Definition 4. Security in TPM. Algorithm $\mathcal{A}'$ is secure if any collection of at most t variables of $\mathcal{A}'$ is independent of the variables of $\mathcal{A}$.

t should be less than n; the optimum choice then would be $t = n - 1$. There are firm tools and proof methods for achieving TPM security with $t = n - 1$. The *t-SNI* criteria for building gadgets of $\mathcal{A}'$ defined in [14] guarantee that composition in $\mathcal{A}'$ will be TPM secure. Constructions for AND and refresh gadgets with proved t-SNI security for $t = n - 1$ are available [14].

The unfortunate part of the story is that TPM security cannot assure RPM security. See part 2.4 for their relations. The dominant example is the case of the AND (that is a non-affine field multiplication) algorithm given in [12], for which in [14] t-SNI security for $t = n - 1$ is proved. However, based on the results of [13, 11] it is easy to show that for any fixed (and arbitrarily small) ϵ, there exists an order $\widetilde{n}$, beyond which the random probing leakage of this construction will reveal its internal secrets (i.e., its operands) with increasingly high probability.

For RPM, in [8], the authors used mutual information between leakage data and input plain-text and encryption key as a metric for rating the security of $\mathcal{A}'(n)$. In [9], $\mathcal{A}'(n)$ is defined secure in RPM if receiving leakage of random probing gives no extra benefit to the adversary, as entailed in the definition below.

Definition 5. Security in RPM (informal) [9]. Algorithm $\mathcal{A}'$ is secure if for any adversary $\mathcal{S}_1$ that is observing input/output of $\mathcal{A}'$ and is receiving random probing leakage, there be an adversary $\mathcal{S}_2$ with only access to input/output of $\mathcal{A}'$, such that $\mathsf{SD}(out(\mathcal{S}_1), out(\mathcal{S}_2))$ is sufficiently small. Here, $out(.)$ denotes the output of its argument algorithm. See [9] for the formal definition.

The mutual information metric given in [8] and the indistinguishability definition with the statistical distance measure given in [9] are both of theoretical interest. They just investigate the effectiveness of masking countermeasure when the masking order n is arbitrarily large.

In [15], the authors studied the success probability of the adversary at guessing a target secret of the masked structure. In the direction of their work, this paper gives a new definition for security in the RPM, which we call *RPM security* in the sequel. This definition seems to be more natural and relevant to the practical viewpoint. It is measurable and thus allows us to compare the security of different constructions. We will give multiple linear and non-linear algorithms satisfying RPM security. Further more, we develop techniques for quantitative evaluation of the RPM security of various structures.

Definition 6. RPM security [this paper]. The algorithm $\mathcal{A}'$ is RPM secure if there is a fixed ϵ^o, that for any adversary observing leakage of $\mathcal{A}'$ with $\epsilon \leq \epsilon^o$, the *advantage* of recovering each secret of $\mathcal{A}$ -over trivial guessing- is a monotonically decreasing toward zero by increasing the order n.

A common belief is that in a general non-linear $\mathcal{A}'(n)$, random probing leakages at rates $\epsilon > 1/n$ will reveal secrets of $\mathcal{A}$. Even providing an algorithm for simple multiplication that can withstand constant leakage rates is still an unanswered challenge in the RPM.

In this paper, RPM secure constructions for refresh and multiplication (for constant leakage rates) will be given. Composition rules for assessing the RPM security of complex structures also will be provided. Our contributions are explicitly listed in 1.4.

1.3 Metrics for quantifying the security definitions

Assume variable V is one of the secrets of a masked circuit $\mathcal{A}'(n)$. For simple structures, like refreshing gadgets or even masked S-Box, secret V is unambiguously defined. However, for a complex structure as a masked implementation of the AES, many sensitive variables can be considered as the target secret.

In RPM, the side-channel adversary, receiving leakage $\mathcal{L}$, which is collection of ϵ-random probings of the variables of $\mathcal{A}'(n)$, aims to gain more knowledge about the secret V. Mutual information criteria for security in RPM, estimates $\mathsf{MI}(V; \mathcal{L})$. In this measure, $\mathcal{A}'(n)$ is secure if $\mathsf{MI}(V; \mathcal{L})$ is sufficiently small. Value of $\mathsf{MI}(V; \mathcal{L})$, among the other parameters, is a function of the masking order n and the leakage rate ϵ. For proving the security of a masked implementation of block ciphers in RPM, [8] used this metric, but they proof relied on the existence of *leakage-free* refreshing gadgets, which is often deemed as an impractical assumption.

For evaluation of security based on definition 5, the main technique is to show that the leakage $\mathcal{L}$ and the secret V are independent. For this purpose, one proves that a new $\mathcal{L}^*$ distribution can be created from scratch (without knowing the realized value of V). Such that, $\mathsf{SD}(\mathcal{L}, \mathcal{L}^*) = 0$. In this case, leakage $\mathcal{L}$ is called *simulatable*. Not, all leakages are simulatable. In the sense of definition 5, $\mathcal{A}'(n)$ is secure, if at given (n, ϵ), with sufficiently high probability, a random instance of leakage $\mathcal{L}$ is simulatable. With at least the same probability, $\mathcal{L}$ and the secret V are independent. This approach's main drawback is that it works in asymptotic settings and rarely can be used as piratical security measuring tool.

In our definition of security (definition 6), we also measure the relation of leakage $\mathcal{L}$ and the secret variable V. We estimate how much is probable that an adversary can correctly estimate the k-bit, initially random, secret V with observation of the leakage $\mathcal{L}$. Let the estimation of the adversary for the value of V be $\widetilde{V}$, our measure for security, which we call *Secret Recovery Probability* (SRP) is defined as

$$SRP(n, \epsilon) = |\Pr(\widetilde{V} = V|\mathcal{L}) - \frac{1}{2^k}|. \tag{1}$$

The adversary for which the SRP is the highest is defined as the MAP adversary. Decision rule of the MAP adversary (that is, the optimal rule) is discussed in part 2.3. Afterward, we will only compute and consider SRP for the MAP adversary. When the secret variable is not unique, we specify it with a subscript as SRP_V.

In the sense of our definition, $\mathcal{A}'(n)$ is secure in RPM if $SRP(n, \epsilon)$ for a region of values of ϵ is a decreasing function of n. Compared to definition 5 and its evaluation method, our new definition and SRP metric are measurable and well defined for any masking order and leakage rate.

This paper develops various methods for the evaluation of SRP for both linear and non-linear masked circuits. We will also use this measure for assessing RPM security of a masked implementation of AES.

The main reason for introducing a new security definition and a new measure is that it is practically computable and more sound from the engineering perspective. More on the comparison of definitions 5 and 6 is given in part 2.6.

1.4 Our contributions

In RPM, during each execution of $\mathcal{A}'$, the adversary probes variables of $\mathcal{A}'$ independently with probability ϵ. The primary question is: how much is probable that this leakage information discloses anything about sensitive variables of $\mathcal{A}$? To answer this question, a simple but still novel technique of utilizing the inter-variables parity relations for assessing the RPM security of $\mathcal{A}'$ is developed.

Governing relations describing a masked circuit is reformulated with a linear (or a non-linear) system of equations. Secrets of masking are also unknowns of this system. Then, the derived parity system's informativeness against ϵ random leakage at various orders n is assessed.

Belief propagation is a well-known technique for studying parity systems. As its main limitation, belief propagation requires a sparse representation of equations in the system. Even with sufficiently sparse relations, it can only provide a lower bound on the system's informativeness. However, this paper's approach gives an exact bound for linear systems. It provides upper and lower bounds for non-linear systems to characterize what adversary can achieve with ϵ random probing.

In the following, we briefly give the contributions that are made in this paper

- A systematic approach for evaluating the RPM security of different linear masked implementations is developed. The results are in the concrete (not asymptotic) setting and work for any order n. To our best knowledge, it is the first time that concrete results for arbitrary order of masking are provided.
- For constant leakage rate ϵ, independent of the masking order n, a non-affine multiplication gadget with RPM security is provided. Before the results in this paper, proving RPM security for a non-affine field multiplication circuit was an open issue.
- We rather tightly bound the leakage effects of refreshing gadgets. Handling leakage of refreshing circuits, instead of neglecting them, was a long-standing challenge in the RPM security evaluation frameworks.
- For X^{-1} S-Box used in AES, we give an arbitrary-order masked implementation with proved RPM security for $\epsilon < 0.03$.
- By developing equivalent erasure model for various gadgets and simple combinations, we are able to give a masked AES implementation with proved RPM security. For the first time, our result provides a bound that

is independent of the size of the masked circuit. Previous results were confined to the size of the masked implementation.

We are pretty sure that the tools in this paper will help to explore more masked structures than those considered here.

2 Preliminaries

In this section, introductory concepts in Boolean masking of block ciphers are presented. MAP criterion for making an *optimal* decision based on observations is explored. Moreover, misleading intuitions that are reached by applying t-SNI security results in RPM are also discussed in this section. In the end, state of the art in RPM security proofs is reviewed.

2.1 Boolean Masking

Masking, controlled by a parameter $n \in \mathbb{N}$, is an algebraic tool for re-enforcing the security of a given algorithm $\mathcal{A}$ against an adversary empowered with probing knowledge.

Variable n, also called *masking order*, plays the role of *security parameter*. Increasing n adds to the computational complexity, and one naturally hopes that this will result in a more resistant countermeasure against side-channel leakage.

Boolean masking is the most studied among others. In its simplest form, to mask a stand-alone variable $V \in \mathbb{F}_{q=2^k}$, a vector of n variables as $\overrightarrow{V} = (V_1, V_2, \ldots, V_n) \in (\mathbb{F}_q)^n$ are randomly chosen in a way that $\oplus_{i=1}^{n} V_i = V$.

Boolean masking of a single variable V is best described by two polynomial-time algorithms. One is $\mathsf{Enc}(.)$, a probabilistic sampling function identified by its output vector as $\overrightarrow{V} \leftarrow \mathsf{Enc}(V)$, and the other is a deterministic function given by $\mathsf{Dec}(V) = \oplus_{i=1}^{n} V_i$. Using fresh randomness, the Output of $\mathsf{Enc}(.)$ should uniformly sample $\overrightarrow{V}$ from $(\mathbb{F}_q)^n$ such that, collectively, members of $\overrightarrow{V}$ uniquely determine V, and any subset of $\overrightarrow{V}$ with at most $n-1$ entry is independent of V. Following similarities with secret sharing schemes [16], members of $\overrightarrow{V}$ are sometimes called shares of V, and $\overrightarrow{V}$ itself is called n-sharing of V. In this paper, an over variable right arrow is used to quickly identify the corresponding secret variable of a sharing.

In [17], the soundness of masking is demonstrated in the noisy leakage model with the following experiment; for a binary secret V, an adversary executes $\overrightarrow{V} \leftarrow \mathsf{Enc}(V)$ l times and each time learns a noisy version of resulted shares. The noise being additive with $i.i.d$ zero-mean Gaussian distribution. They showed that, in this case, non-trivially correct guessing of V would require l to be an exponential function of n.

Constructing a $\mathsf{Enc}(.)$ for a single variable V is straightforward; the main challenge of masking (which is the target of numerous research papers) is to develop routines for performing all of the operations respectively on the shares, instead of the initial variables.

2.2 Block ciphers as a common target

Block ciphers are a dominant primitive of cryptography. They usually use the same fixed encryption key during consecutive runs, making them a suitable target for power analysis side-channel attacks. An adversary can combine power traces obtained during different runs to recover some bits or whole of the secret encryption key.

Block ciphers are unvarying function designed with feedforward structure without any feedback loop. Their internal computation is performed in several quite similar rounds. Furthermore, their execution path is branch free and is independent of the values of input and intermediate variables. In this paper, these features will be very beneficial for RPM security analysis of their masked implementations.

2.2.1 Masking a block cipher

Think of $\mathcal{A}$ as a code written to implement a block cipher $C = \mathsf{BC}(P, K)$. Boolean masking provides a systematic way for generating a new code $\mathcal{A}'$ based on $\mathcal{A}$, in a way that for any n, $\mathcal{A}'$ on input $\overrightarrow{P}$ and $\overrightarrow{K}$ (which are n-sharing of the plain-text P and the encryption key K, respectively) produces $\overrightarrow{C}$ (which is an n-sharing of the cipher-text C) as output. $\mathcal{A}'$ is required to be more robust against probing leakages.

Inputs and output of BC are not single elements of $\mathbb{F}_q$. They are usually a block of elements. Generalization of n-sharing for a block of m concatenated $\mathbb{F}_q$ elements as $W = W_1||W_2||...||W_m$ is defined with the following relation.

$$\overrightarrow{W} \leftarrow \mathsf{Enc}(W) = \mathsf{Enc}(W_1)||\mathsf{Enc}(W_2)||...||\mathsf{Enc}(W_m) \tag{2}$$

The pioneering work of [12] founded arbitrary-order Boolean masking. The initial study in [12] was limited to operations in a binary field. The negative impact of this limitation is an undesirably massive overhead in the masking complexity. In [18], an extension to bigger fields took place, which made masking at higher orders computationally affordable.

There are exact **steps to mask** $\mathcal{A}$. Here we informally sketch them.

At **first**, based on the bit-length of variables in $\mathcal{A}$, an appropriate Galois field $\mathbb{F}_q$ is chosen. Then, computation in $\mathcal{A}$ is decomposed into so-called *elementary gates* and S-Box evaluations. Elementary gates include AND operation and all *affine* operations of the field. Permutations over field elements (if any) are left as is. However, permutations over bits (which are not field members) are replaced by their equivalent *lookup-tables*. This means that they are modeled as new S-Boxes. For most block ciphers, this step is obvious and does not require much effort.

The **next step** is to evaluate S-Boxes using elementary gates. $\mathcal{A}$ may incorporate different S-Box structures. For each of them, the common paradigm is to use *Lagrange interpolation* to expand the S-Box output with a polynomial-function of its input, as shown in (3) [19].

$$\mathsf{S\text{-}Box}(X) = \sum_{i=0}^{q-1} \alpha_i X^i \quad \text{with } \alpha_i \in \mathbb{F}_q \tag{3}$$

This polynomial representation can be directly evaluated with elementary gates. But, this trivial approach usually results in high calculation overhead. Many research papers aim to reduce this computational burden by decreasing the total number of non-affine AND invocations [19, 20]. It is proven that for an S-Box representable in a k-bit Galois field, this process will need $\Omega(2^{k/2}/\sqrt{k})$ AND operations [21]. For X^{-1} S-Box in the AES, only 4 AND gates are required.

The **third step** is to replace each elementary gate G with its *shared* or *vectorized* counterpart, denoted by SG and is called *shared gadget*. For a gate G with inputs X and Y (if it has two input operands), and output Z, gadget SG, at any order n, on input $\overrightarrow{X}$ and $\overrightarrow{Y}$ produces $\overrightarrow{Z}$, where the vectors are n-sharing of their respective secrets. Moreover, gadget SG should fulfill some security notions.

$$Z = \mathsf{G}(X, Y) \xrightarrow{\text{replaced by}} \overrightarrow{Z} = \mathsf{SG}(\overrightarrow{X}, \overrightarrow{Y}) \tag{4}$$

In this paper, shared gadget for AND is represented by SAND, and similarly, SXOR stands for the shared implementation of XOR operation. The structure of SG when G is an affine gate is apparent [18]. The main challenge is providing a shared counterpart for AND gate. In [12], SAND-ISW is provided. In [13], a new SAND, which we, with some modifications in this paper, call SAND-Rec, is put forward. Other SAND constructions are given in [22, 23].

The **final step** of masking is to insert a *refreshing* gadget before each reuse of sharing vectors. A refreshing gadget on input $\overrightarrow{X}$ produces a new n-sharing for the same secret X. The principal role of a refreshing gadget is to avoid a fixed n-sharing vector to be the operand of multiple SGs. This step requires many refreshing gadgets. Some works, especially in TPM, argue how to decrease SR invocations [14].

Hereafter, a refreshing gadget is denoted by SR. In the literature, various candidate constructions for SR are given. Some of them are SR-Simple given in [18], SR-SNI [14], and SR-Rot [24]. Other structures for SR are also studied in [9, 25]. To mitigate ambiguities, we have adopted unified names for different SG (including SR) algorithms that we will study in this paper.

The thus generated circuit above is a masked counterpart of $\mathcal{A}$, and we label it by $\mathcal{A}'$.

An alternative method for computing an S-Box for an n-sharing input is given in [26]. Their approach is based on the *table-randomization* method. In this paper, our analysis best fits with the polynomial based S-Box evaluation. However, it would be an interesting objective to develop similar results for table-randomization based methods.

2.3 MAP adversary

The adversary that makes her decisions based on the *Maximum A Posterior* probability rule is called the MAP adversary. For estimating a secret $V \in \mathcal{V}$, observing leakage $\mathcal{L}$, the MAP adversary outputs the value $\widetilde{V}$ by the following rule as her guess for the value of the V.

$$\begin{aligned}\widetilde{V} = arg\max_{v} \Pr(V = v|\mathcal{L}) &=_{(a)} arg\max_{v} \frac{\Pr(\mathcal{L}|V = v)\Pr(V = v)}{\sum_{\mu\in\mathcal{V}} \Pr(V = \mu)\Pr(\mathcal{L}|V = \mu)} \\ &= arg\max_{v} \Pr(\mathcal{L}|V = v)\Pr(V = v) \\ &=_{(b)} arg\max_{v} \Pr(\mathcal{L}|V = v)\end{aligned} \tag{5}$$

Where (a) is by *Bayes* theorem, and (b) follows from the fact that for the cases we will face in this paper, $\Pr(V)$ is uniform. We define the *Secret Recovery Probability* (SRP) as $SRP = |\Pr(\widetilde{V} = V|\mathcal{L}) - \frac{1}{|\mathcal{V}|}|$ for future references.

The decision of MAP adversary requires knowledge of the joint distribution of $\Pr(\mathcal{L}, V)$. In the side-channel literature, adversary by profiling [27, 28] and some dimensional reduction [29] can estimate this joint probability. For the profiling stage, the adversary should sufficiently run the target device with known and random inputs and record its leakage traces.

If $\Pr(\mathcal{L}, V)$ is not available, the MAP rule is not applicable. Various sub-optimal decision principles such as *correlation attack* and *differential attack* are developed [30].

MAP adversary is computationally unbounded, and for the goal of maximizing $\Pr(\widetilde{V} = V|\mathcal{L})$, this adversary outperforms any other adversary [31].

A closely related alternative is the *Bayesian* adversary given in [15]. In their definition, the adversary considers the first o top values of $\Pr(V|\mathcal{L})$ instead of the single maximum value. Except for adversary definition, our approach for security rating is entirely different from [15].

In the analysis of this paper, we focus on the results for the MAP adversary. Therefore, obtained probability bounds and estimations can be considered as a limit for what all other adversaries might achieve with the same leakage access.

2.4 Implications of t-SNI security in RPM

Security proofs in TPM are *simulation-based*. In this method, at any order n, the distribution of values of any t intermediate variables in $\mathcal{A}'$ is shown to be independent of sensitive variables of $\mathcal{A}$. Usually, there is a group of $t + 1$ variables, knowledge of which will disclose the corresponding secret in the $\mathcal{A}$. To put it short, in the context of threshold probing, for a fixed order n, at any number of probes t, $\mathcal{A}'$ is either completely secure or completely insecure.

In the light of works of [9, 10], we roughly sketch what t-SNI security means in RPM.

Let $\mathcal{A}$ be a multi-round block cipher. Exploiting the method explained in part 2.2.1, at any order n, one can derive algorithm $\mathcal{A}'$, where $\mathcal{A}'$ is only composed of SAND, SXOR, and SR gadgets. The total number of these gadgets is shown by $|\mathcal{A}'|$.

Assume $h(n)$ is an upper bound for the number of variables appearing in any of these gadgets. In this setting, $\mathcal{A}'$ will include at most $h(n)|\mathcal{A}'|$ variables; each of which, in RPM, is independently known to the adversary with the same probability ϵ.

In each execution of $\mathcal{A}'$, let random variable Z be the number of variables leaked to the adversary. Based on the independent nature of leakages, we can bound the expected value of Z as $\mathbb{E}[Z] \leq \epsilon h(n)|\mathcal{A}'|$.

Conditioned on $Z \leq t$, since $\mathcal{A}'$ is assumed to be t-SNI secure for say $t = n-1$, leakages in RPM will be independent of the secrets of $\mathcal{A}$. As a result, $\mathcal{A}'$ is threshold secure with probability at least $p = \Pr(Z \leq t)$.

For practically used shared gadgets, $h(n)$ is bounded as $h(n) \leq an^2$ for some constant a. By application of *Markov* inequality, to ensure that $p \geq 1-1/k$ for an arbitrary $k \in \mathbb{N}$, the expected value of Z should be bounded as $\mathbb{E}[z] \geq t/k$.

By substituting for t and $h(n)$, we can drive an upper bound over ϵ as

$$\epsilon \leq \frac{1}{ak|\mathcal{A}'|n}. \tag{6}$$

As long as ϵ satisfies (6), $\mathcal{A}'$ is guaranteed with probability at least $1 - 1/k$ to be secure. By closer inspection of (6), the following two counterintuitive corollaries are obtained.

1. For constant ϵ, there is an order n, beyond which, t-SNI security of $\mathcal{A}'$ cannot assure concealment of the secrets of $\mathcal{A}$. In other words, increasing the masking order may decrease the security of the masked algorithm.
2. At constant n, as $|\mathcal{A}'|$ grows (for example, by adding to the rounds of the corresponding block cipher), the margin for safe ϵ will shrink. $|\mathcal{A}'|$ is independent of n and is directly proportional to the number of rounds of $\mathcal{A}$. So, one may conclude that if rounds of $\mathcal{A}$ increases, its security will not be assured.

By developing a new measure for assessing RPM security, this paper will give a clear picture to resolve the above strange conclusions.

2.5 State of security proofs in RPM

Thanks to the equivalence proved in [10], one can use the corollaries obtained for either the noisy leakage model and the RPM interchangeably.

In [8], an upper bound for the *mutual information* between noisy leakage (with their definition of the noise) and the secret operands for specific instantiations of SAND and SXOR are given. With these upper bounds in hand, by neglecting possible leakage from SR gadgets, and by summing up the information leakages corresponding to the individual shared gadgets, they gave an information-theoretic bound for the overall leakage of $\mathcal{A}'$.

In [9], state of the art in the RPM analysis was brightly improved. The authors of [9] included leakages of SR gadgets and avoided the arguable random message model adopted in [8]. They also used the more realistic simulation-based method instead of the mutual information metric. Their results inspire the discussion in part 2.4.

In [10], security proof for constant (independent of n) probability ϵ was given for the first time. The results were further improved in [32]. However, their method only works with leak-free SR gadgets. The leakage of SR gadget for a construction similar to SR-Simple [18] is studied in [33]. They could prove the RPM security of their SR gadget.

In a recent work, [34] has considered the security of Boolean masking in the RPM. Our leakage models are identical. However, their security definition is based on the threshold probing criteria, which is stronger and different from our security measure.

In [34], with listing all variables of the target algorithm, subsets of the variables with different sizes are considered. Assume W is the number of variables. For a subset of size $w \leq W$, they have counted the number combinations that result in a simulatable leakage. Let $c_w \leq \binom{W}{w}$ be the number of the subsets that are not simulatable. By doing so, for different subsets of size up to W, they have reached the following measure for the security

$$f(\epsilon) = \sum_{w=t}^{W} c_w \epsilon^w (1-\epsilon)^{W-w}$$

Where t is the TPM security margin. $f(\epsilon)$ is an estimate for the probability that a random leakage may reveal information about the secrets. Because of the exhaustive search approach, their computation load is exponential in the number of variables of the target algorithm. However, for some small gadgets, they have computed the corresponding $f(\epsilon)$. Their main technical tool is an expansion strategy that in several rounds re-mask the masked algorithm of the previous round.

$$\mathcal{A} \to \mathcal{A}' \to (\mathcal{A}')' \to \ldots \to (\ldots(\mathcal{A}')'\ldots)'$$

This expansion drastically increase the size of the final circuit. However, they have proved the security of the final masked result, for particular constructions, at constant leakage rates as $\epsilon = 2^{-8}$, which is a notable contribution. Our security definition and its evaluation techniques are different from theirs. We prove security at substantially higher leakage rates, with ordinary single-round masking.

2.6 More on the new RPM security definition

Our new RPM security given in definition 6, in line with the security notion in [15], considers only a single (not a combination of) variables of $\mathcal{A}(n)$ as the target of the side-channel adversary. This makes definition 6 theoretically weaker than one given by definition 5.

For SR gadgets, only one secret variable is present, so the limitation to a single target variable is not a problem. For assessing the RPM security of SAND gadgets, we will consider the case of identical input operands as a marginal condition. This choice of input operands will limit the secrets of SAND to one. So, again, the limitation of definition 5 will not be an issue. For masked S-Box implementation, again, the secret is uniquely and unambiguously defined.

The pitfall of definition 5 is highlighted when we are considering RPM security of complex structures such as AES masked implementation. For AES, we will prove RPM security (definition 6) by targeting any variable of unmasked AES. Practical side-channel research works try to use all the leakage information to recover or estimate a single variable (usually round-keys of AES). Our RPM security proof for masked AES assures that no adversary can recover any secret of the corresponding (unmasked) AES. However, it does not prove that an adversary cannot obtain an algebraic function of multiple secrets of AES. Nevertheless, there is no evidence that how targeting such algebraic combinations will benefit the adversary. This should not be confused by the concept of the so-called higher-order attacks. In those attacks, the adversary combines multiple leakage information in order to gain knowledge about a single secret variable.

With the aforementioned shortcoming of the new definition, one may wonder what the point of working with a weaker definition is. The answer is that this new definition with the SRP metric is practically measurable, and the limitation of security definition is only of theoretical importance.

As another point, it is needed to note that, definition 6 if fulfilled implies some stronger corollary stated in the following Lemma.

Lemma 1. *Given leakage $\mathcal{L}$, if for variable V derived $SRP_V(n, \epsilon)$ tends to zero by increasing order n, then for any function of V as $f(V)$, $SRP_{f(V)}(n, \epsilon)$ will also limit to zero.*

Proof. In Appendix A □

2.7 Overview of our approach

For each value of order n, we describe $\mathcal{A}'(n)$ with a system of parity equations such as $\mathfrak{P}(n)$. Secret V is also assumed to be an unknown of $\mathfrak{P}(n)$. Then, for random sets of leaked variables, we argue the decision rule of the MAP adversary and also calculate the SRP of the MAP adversary in guessing the secret V at different pairs of (n, ϵ).

For linear algorithms, it is relatively easy to approximate SRP. When $\mathfrak{P}(n)$ is linear, we can use the Gaussian elimination technique to see whether $\mathfrak{P}(n)$ with the leaked variables can determine V or not. However, for non-linear systems, the problem will be more challenging.

In section 3, we study different linear systems and provide tools for approximating SRP. This section is the foundation for the other parts of the paper. For linear $\mathfrak{P}(n)$, we show that an instance of leakage either uniquely determines the secret V or the $\mathfrak{P}(n)$ cannot distinguish between different choices for the secret V with this instance of leakage. This phenomenon will be described by an equivalent erasure channel.

The main examples of linear gadgets are SR gadgets. SR gadgets are mediators between the other gadgets of a complex structure. For SR gadgets, we derive an equivalent model that lets us reduce the RPM security of complex structures to the RPM security of their building gadgets. These results presented in section 4 paves the road for studying non-linear $\mathfrak{P}(n)$ systems.

An SAND gadget is described with a non-linear $\mathfrak{P}(n)$. For a non-linear set of parity equations, our approach is to find a more (or a less) informative but linear counterpart. With this linearization, estimation of SRP will be possible for non-linear SAND gadgets. An SAND gadget with RPM security is presented in section 5.

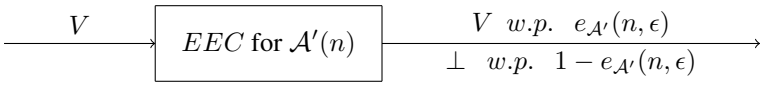
SAND security estimation and SR modeling are enough tools to evaluate S-Box security in section 6. Compared to a single S-Box, a complex structure like AES has many more secret variables, and their parity relations are deeply interwoven. For exploring the RPM security of AES, we combine the techniques developed for linear $\mathfrak{P}(n)$ and the equivalent erasure channel obtained for the S-Box. Results are presented in section 7. Quite surprisingly, we show that RPM security of AES is not much decreasing with increasing its rounds.

3 *EEC* for linear masked circuits

In the RPM, during each execution, the adversary learns variables of $\mathcal{A}'$ independently with probability ϵ. The main question is whether these leaked variables will disclose anything about the secrets of $\mathcal{A}$ or not. In this section, for a restricted class of algorithms, which we call *linear*, a systematic answer for this question is developed.

For linear $\mathcal{A}'$ with a single secret (like SR gadgets), we prove that disclosing a random set of variables either reveals the secret ultimately or expresses nothing about it. This two-level marginal behavior of each leakage instance can be described with an Equivalent Erasure Channel (EEC) with the secret as the input of this channel.

An erasure channel conveys its input V to the output with probability e and puts a special erase symbol $\perp$ at output otherwise. In the latter case, the input is concealed. Parameter e for the EEC of $\mathcal{A}'$ is denoted by $e_{\mathcal{A}'}(n, \epsilon)$, which is the probability that a random instance of leakage variables reveals the secret V.



$\mathcal{A}'$ may contain many variables (especially at higher values of order n), so the direct calculation of $e_{\mathcal{A}'}(n, \epsilon)$ is impractical. Instead, at each pair of (n, ϵ), we give an estimate of $e_{\mathcal{A}'}(n, \epsilon)$ with a *Mont Carlo* based simulation.

As a use case for the developed methods, *EEC* for different SR gadgets is computed and compared at the end of this section. Interestingly, the results obtained put doubts on a widely recognized sense about the security of the SR-Simple gadget and its computationally intensive counterpart SR-SNI.

Recently, [33] also considered SR-Simple in RPM. They demonstrated that SR-Simple, which is deemed insecure (actually, without t-SNI security) in TPM, is still reliable in RPM. Compared to [33], our methods here are more general and, albeit entirely different.

3.1 Preliminary Definitions

Before proceeding further, we pause to give precise definitions of some of the terms already mentioned.

Definition 7. Set of variables of an algorithm. From the perspective of computational complexity theory, $\mathcal{A}'$ is a *non-uniform* family of *circuits*. For each value of the order n, there is a unique circuit $\mathcal{A}'(n)$ belonging to the family. Circuits contain no-loop, and their execution path is branch free.

For each $\mathcal{A}'(n)$, there is a fixed ordered list of variables denoted by $\mathbf{\Sigma}(n)$. The list $\mathbf{\Sigma}(n)$ includes all the operands valued during the computation of $\mathcal{A}'(n)$. Each entry of $\mathbf{\Sigma}(n)$ is valued only once. The terms $\mathcal{A}'$ and $\mathbf{\Sigma}$ (without explicit parameter n) refer to the families, with no interest in a particular value of n.

Definition 8. Erasure channel for a single variable. An erasure channel in a field $\mathbb{F}_q$ is a probabilistic function $\phi : \mathbb{F}_q \to \{\mathbb{F}_q, \perp\}$ defined by the following relation.

$$\phi(x) = \begin{cases} x & \text{with probability } e \\ \perp & \text{otherwise} \end{cases} \tag{7}$$

Where $\perp$ is a special symbol to denote the erasure of the input. In RPM with parameter ϵ, the adversary learns each element of $\mathbf{\Sigma}(n)$ through an erasure channel with $e = \epsilon$.

Definition 9. Linear Algorithm. A linear $\mathcal{A}'$ is only composed of degree-one affine operations. Degree-one affine operations include field XOR and scalar multiplication. Linear $\mathcal{A}'$ will not use higher-degree affine operations such as field squaring. $\mathcal{A}'$ may contain a special Rand gate that, at each invocation, outputs a uniformly random element of $\mathbb{F}_q$.

Definition 10. RPM security (for linear algorithms). $\mathcal{A}'$ is called RPM secure if there is a constant ϵ^o that for any $\epsilon < \epsilon^o$, $e_{\mathcal{A}'}(n, \epsilon)$ monotonically decreases towards zero as the order n increases.

For a fixed security level, a faster decline of $e_{\mathcal{A}'}(n, \epsilon)$ will ease the required order n, and consequently, reduce the computational overhead of masking.

Generally, $e_{\mathcal{A}'}(n, \epsilon)$ decays exponentially with n. However, to remain flexible, we omitted any restriction on the speed of lessening in the definition of RPM security.

3.2 Linear dependencies between variables

Let $\mathbf{\Sigma}(n) = [X_1, X_2, \ldots, X_{T(n)}]$ with $T(n)$ elements all in the same field $\mathbb{F}_q$. Some entries of $\mathbf{\Sigma}(n)$ may be linearly dependent on the others. The secret $V \in \mathbb{F}_q$ is also linearly related to the X_is. The adversary may use these interconnections in her benefit, like parity equations that are used in the coding theory.

Our first target is to extract these dependencies and then assess the power they give to the adversary for estimating the secret V.

A linear relation, also called a parity relation, over $T(n)+1$ elements of the vector $[V, \mathbf{\Sigma}(n)]$ can be expressed by a same-length constant vector of coefficients in $\mathbb{F}_q$ as $U = [u_1, u_2, \ldots, u_{T(n)+1}]$, where in each run of $\mathcal{A}'(n)$ with a random secret V, the following identity always holds.

$$\oplus_{i=1}^{T(n)} u_{i+1} X_i \oplus u_1 V = 0 \tag{8}$$

At each order n, collections of constant vectors like U that are perpendicular to $[V, \mathbf{\Sigma}(n)]$ form a *vector space*. We denote this vector space by $\mathscr{V}_n$.

By assumption, $\mathcal{A}'(n)$ is composed of only affine gates. Therefore, the realizations of $[V, \mathbf{\Sigma}(n)]$ also form a vector space. We denote this later vector space by $\mathscr{S}_n$. By definition, $\mathscr{V}_n$ is the *null space* of $\mathscr{S}_n$. This implies that vectors in $\mathscr{V}_n$ are orthogonal to the vectors of $\mathscr{S}_n$.

Set $\mathbf{\Sigma}(n)$ can include constant values. In this case, $\mathscr{S}_n$ will not be a vector space. However, constant values do not convey any information about the random secret V. So, we can remove them from the definition of $\mathbf{\Sigma}(n)$.

A *basis* $\mathscr{B}$ for a vector space is a finite collection of particular vectors called *component vectors* or *coordinate vectors*. Any vector in a vector space can be uniquely constructed by a (finite) linear combination of these component vectors. The component vectors belonging to a basis $\mathscr{B}$ are linearly independent of each other. In finite fields, there are finitely many basis for a vector space. The number of vectors in each basis is constant and is a feature of the vector space. This number is called the *dimension* of the vector space and is represented by $\mathsf{Dim}(.)$. The *rank-nullity* theorem from linear algebra implies the following identity [35].

$$\mathsf{Dim}(\mathscr{V}_n) + \mathsf{Dim}(\mathscr{S}_n) = T(n) + 1 \tag{9}$$

Assume $\mathscr{B}_n$ is the basis for $\mathscr{V}_n$, and the dimension of $\mathscr{V}_n$ is $D(n)$. We create a new matrix $\mathbf{P}^n_{Lin}$ by the component vectors in $\mathscr{B}_n$. Each row of $\mathbf{P}^n_{Lin}$ is a component vector. In this way, $\mathbf{P}^n_{Lin}$ would have $D(n)$ rows and $T(n)+1$ columns. Since $D(n) \leq T(n)$, the *rank* of $\mathbf{P}^n_{Lin}$ equals $D(n)$.

By the construction, for each realization of $[V, \mathbf{\Sigma}(n)]$, the following identity should hold.

$$\mathbf{P}^n_{Lin} \times [V, \mathbf{\Sigma}(n)]^\top = \mathbf{0} \tag{10}$$

Where $^\top$ denotes matrix transpose.

Interestingly, as far as $\mathcal{A}'(n)$ is linear, $\mathbf{P}^n_{Lin}$ gives a complete and alternative description of $\mathcal{A}'(n)$ in the sense that each $(T(n)+1)$-tuple vector satisfying (10) should be realizable by $\mathcal{A}'(n)$, and each realization of $[V, \mathbf{\Sigma}(n)]$ should satisfy (10).

From the $T(n)$ element of $\mathbf{\Sigma}(n)$, some are input variables of $\mathcal{A}'(n)$, and some are randomness variables (i.e., they are outputs of Rand gadgets). Let $R(n)$ represents the number of non-input and non-randomness variables of $\mathbf{\Sigma}(n)$.

Lemma 2. *If for an algorithm $\mathcal{A}'(n)$ the relation $D(n) = T(n)+1-R(n)$ holds, then there is a one-to-one mapping between $(T(n)+1)$-tuples that satisfy equation (10) and $(T(n)+1)$-tuples that are generated in $\mathcal{A}'(n)$.*

Proof. In Appendix B □

It is easy to see that the condition required in Lemma 2 is always satisfied with linear algorithms. So, for linear $\mathcal{A}'(n)$, one can examine the coefficient matrix $\mathbf{P}^n_{Lin}$, instead of directly working with $\mathcal{A}'(n)$.

In the sequel, our main target will be evaluating the benefits that the equations described by $\mathbf{P}^n_{Lin}$ give to the adversary.

The adversary in the RPM, thorough leakage, learns some unknowns of the system of equations given in (10). After substituting for the learned values, the next step is to process the resulting system to find the most probable estimate for the secret V.

Before proceeding in this direction further, we take a short detour to provide a handy technique for extracting $\mathbf{P}^n_{Lin}$ for a given linear algorithm $\mathcal{A}'(n)$.

3.3 Extracting the linear relations

Although the pseudocode of an algorithm gives a complete description of the relations between the elements of its $\mathbf{\Sigma}$, here we provide an alternative and more straightforward procedure for extracting parity relations in a linear algorithm.

Thanks to the versatile tools of linear algebra, we can derive a complete basis for $\mathscr{V}_n$ in the following steps, without dealing directly with the expressions inside $\mathcal{A}'(n)$.

At **first**, for the desired order n, enumerate variables of $\mathcal{A}'(n)$ and create the list $\mathbf{\Sigma}(n)$. The initial order of variables in $\mathbf{\Sigma}(n)$, once decided, should be kept unchanged; but has no impact on the final results. We assume that all the variables and operations are in an appropriate field $\mathbb{F}_q$.

Then, by $k(n) \gg T(n)$ times executing $\mathcal{A}'(n)$ with fresh randomness and a new random secret V, create a $k(n) \times (T(n)+1)$ matrix $\mathbf{M}$. During each execution of $\mathcal{A}'(n)$, a new row of $\mathbf{M}$ is filled. The secret V is placed in the first column of the current row, and the $T(n)$ values of the list $\mathbf{\Sigma}(n)$ are inserted in the next columns, respectively.

Algorithm 1 Create-M

Input $\mathcal{A}'(n)$, $\mathbf{\Sigma}(n)$
Output $k(n) \times (T(n)+1)$ matrix $\mathbf{M}$

for $i = 1$ **to** $k(n)$ **do**
  $V \leftarrow^{\$} \mathbb{F}_q$
  $\mathbf{M}(i,1) = V$
  Execute $\mathcal{A}'(n)$
  $\mathbf{M}(i, 2 : T(n)+1) = \mathbf{\Sigma}(n)$
return $\mathbf{M}$

The symbol $\leftarrow^{\$}$ is used for the random selection of an element from a set, and the symbol : is used to specify a range of columns or rows of a matrix. If : is used without start and end values, it means all the rows or the columns.

Counting the variables depends on the language used for writing the script of the construction and even depends on the target processor's architecture and the used compiler. For the results in this paper, pure C-style coding with no compiler optimization is assumed. Minor modifications in the code scripts may alter the numerical results but seem to be without effect on the main conclusions.

The **next step** is to apply *Gaussian elimination* on the rows of $\mathbf{M}$ to compute the corresponding *row reduced echelon form* of $\mathbf{M}$, which we represent by $\mathbf{E} = \text{Gaussian-Elim}(\mathbf{M})$. The pseudocode for $\text{Gaussian-Elim}(\mathbf{M})$ is given in algorithm 2.

Algorithm 2 Gaussian-Elim

Input $k(n) \times (T(n)+1)$ matrix $\mathbf{M}$
Output Row reduced echelon form of $\mathbf{M}$

$pivot\_row = 1$ ▷ Initialize the pivot row
for $col = 1$ **to** $T(n)+1$ **do**
  for $row = pivot\_row$ **to** $k(n)$ **do**
    if $\mathbf{M}(row, col) \neq 0$ **then**
      $\mathbf{M}(row,:) = \mathbf{M}(row,:)/\mathbf{M}(row,col)$ ▷ Normalize $\mathbf{M}(row,col)$ to 1
      $\mathbf{M}(pivot\_row,:) \leftrightarrows \mathbf{M}(row,:)$ ▷ Swap the rows
      for $i = 1$ **to** $T(n)+1$, $i \neq pivot\_row$ **do**
        $\mathbf{M}(i,:) = \mathbf{M}(i,:) - \frac{\mathbf{M}(i,col)}{\mathbf{M}(pivot\_row,col)}\mathbf{M}(pivot\_row,:)$
      break
  $pivot\_row = pivot\_row + 1$ ▷ Increase the pivot row
$\mathbf{E} = \mathbf{M}$
return $\mathbf{E}$

Finally, the so-called *free* columns, also known as free variables, in the resultant matrix $\mathbf{E}$ are dependent on the other columns. Their corresponding relation also can be directly identified from $\mathbf{E}$. Dependency equations obtained will

constitute the rows of $\mathbf{P}^n_{Lin}$. With algorithm 3, we have $\mathbf{P}^n_{Lin} = \mathsf{Extract\text{-}Linear}(\mathbf{E})$. The number of free variables in $\mathbf{E}$ is equal to the dimension of $\mathscr{V}_n$.

Algorithm 3 Extract-Linear

Input $k(n) \times (T(n)+1)$ matrix $\mathbf{E}$
Output $D(n) \times (T(n)+1)$ matrix $\mathbf{P}^n_{Lin}$

$row_E = 1$
$row_P = 1$
for $col_E = 1$ **to** $T(n)+1$ **do**
  if $\mathbf{E}(row_E, col_E) = 0$ **then** ▷ This column of $\mathbf{E}$ is a free column
    $\mathbf{P}^n_{Lin}(row_P, :) = 0$ ▷ Add a new row to $\mathbf{P}^n_{Lin}$ and initialize it to all-zero
    for $j = 1$ **to** $row_E - 1$ **do**
      $\mathbf{P}^n_{Lin}(row_P, Pivots(j)) = \mathbf{E}(j, col_E)$
    $row_P = row_P + 1$ ▷ Increment current row of $\mathbf{P}^n_{Lin}$
  else ▷ This column of $\mathbf{E}$ is a pivot column
    $Pivots(row_E) = col_E$ ▷ Record pivot column of each row
    $row_E = row_E + 1$; ▷ Increment current row of $\mathbf{E}$
return $\mathbf{E}$

Rows of derived $\mathbf{P}^n_{Lin}$ are component vectors for the vector space $\mathscr{V}_n$. So, they form a basis for $\mathscr{V}_n$. Given one basis for $\mathscr{V}_n$, it is straightforward to construct many other basis for it. Each of these basis -irrespective of their *sparsity*- will work the same for our purpose. *Belief propagation* based methods for solving a system of equation dominantly require sparse representation for the parity equations [36]. Fortunately, our study here has no such prerequisite.

The reliability of the above procedure depends on the value of $k(n)$. An inadequate number of $k(n)$ may introduce erroneous rows in the $\mathbf{P}^n_{Lin}$. As stated in the following lemma, increasing $k(n)$ rejects out the accidental parity relations.

Lemma 3. *With $k(n) = T(n) + m$, by application of the method described above, for $m > 1$, $\mathbf{E}$ will incorporate a complete basis for the vector space $\mathscr{V}_n$, and the probability of introducing a wrong linear relation over $[V, \mathbf{\Sigma}(n)]$ is not more than $(T(n)+1)q^{-(m-1)}$.*

Proof. See Appendix C □

From a practical point of view, Gaussian-Elim and Extract-Linear are agile polynomial-time algorithms; hence, producing $\mathbf{P}^n_{Lin}$ for reasonable values of the order n is relatively fast.

3.4 Driving *EEC* for a linear $\mathcal{A}'$

Before exploring the informativeness of $\mathbf{P}^n_{Lin}$ in RPM, we focus on what a single instance of random leakage can disclose about the secret V.

3.4.1 Single instance of random leakage

Assume some indexes of $\mathbf{\Sigma}(n)$ are declared to the adversary. Let the list of remaining unknown indexes be $\mathbf{\Sigma}^\dagger(n)$. With direct substitution of the known values in (10), the following (possibly) non-homogeneous reduced system of equations will result.

$$\mathbf{P}^{\dagger n}_{Lin} \times [V, \mathbf{\Sigma}^\dagger(n)]^\top = \mathbf{b} \tag{11}$$

The $D(n)$-element column vector $\mathbf{b}$ on the right-hand side of (11) appears after the substitution, and $\mathbf{P}^{\dagger n}_{Lin}$ only contains coefficients corresponding to the remaining unknowns.

Equations in $\mathbf{P}^n_{Lin}$ are linearly independent; however, some rows of $\mathbf{P}^{\dagger n}_{Lin}$ may be linearly dependent (i.e., the rank of $\mathbf{P}^{\dagger n}_{Lin}$ may fall below $D(n)$). The presence of these redundant parities (as will be evident by inspecting the proofs in the rest) will not affect the adversary's success probability.

The adversary knows $\mathbf{P}^{\dagger n}_{Lin}$ and can directly calculate vector $\mathbf{b}$. Then, by obeying the *MAP* decision rule, she tries to find the most probable value for the -initially random- secret V by listing all the possible solutions of (11).

Assume set $\mathcal{S}$ contains non-redundant solutions of (11). Members of $\mathcal{S}$ are $|\mathbf{\Sigma}^{\dagger}(n)|+1$ length vectors $\mathbf{S}_i$ that are satisfying ${\mathbf{P}^{\dagger}}^{n}_{Lin} \times [V, \mathbf{\Sigma}^{\dagger}(n)]^{\top} = \mathbf{b}$. The first entry of each $\mathbf{S}_i$ is the corresponding value for V and is denoted by $\mathbf{S}_{1,i}$. A posterior probability distribution (i.e., the probability after observation) of the secret V, which we denote by $\Pr(\widetilde{V})$ can be directly calculated by partitioning the members of $\mathcal{S}$ based on their first entry. For each value $\alpha \in \mathbb{F}_q$

$$\Pr(\widetilde{V}=\alpha) = \frac{|\mathbf{S}_i \in \mathcal{S},\ \mathbf{S}_{1,i}=\alpha|}{|\mathcal{S}|}. \tag{12}$$

Where $|.|$ represents the cardinality of a set. The MAP adversary would declare the argument of $\max_{\alpha} \Pr(\widetilde{V}=\alpha)$ as her guess for the secret V.

To solve (11), one can apply Gaussian elimination to the rows of ${\mathbf{P}^{\dagger}}^{n}_{Lin}$ and compute the row reduced echelon form of it as $\mathbf{G} = \mathsf{Gaussian\text{-}Elim}({\mathbf{P}^{\dagger}}^{n}_{Lin})$. Coefficients of V reside in the first column of the ${\mathbf{P}^{\dagger}}^{n}_{Lin}$. The secret V is linearly dependent on the variables of $\mathbf{\Sigma}(n)$. Therefore, at least one entry of the first column of ${\mathbf{P}^{\dagger}}^{n}_{Lin}$ is non-zero. This means that V is a *pivot* variable in $\mathbf{G}$, and V will not appear in any other equation of $\mathbf{G}$.

Lemma 4. *Suppose the only equation that contains V (i.e., the first row of $\mathbf{G}$) does not include any other variable. In that case, the adversary can uniquely determine V. Otherwise, any other variable in the first row of $\mathbf{G}$ completely hides the secret V.*

Proof. In Appendix D. □

Based on Lemma 4, for the leakage of any subset of $\mathbf{\Sigma}(n)$, either of the following two cases below will happen.

1. The distribution $\Pr(\widetilde{V})$ is non-zero only on a particular value of the secret V. In this case, the first entry of all the members of $\mathcal{S}$ are equal. This implies that the adversary can uniquely determine the secret V.
2. The distribution $\Pr(\widetilde{V})$ is uniform over $\mathbb{F}_q$. In this case, each candidate value for the secret V occurs with the same abundance in the set $\mathcal{S}$. This uniform distribution is another interpretation for the independence of the secret V from the leaked variables.

For a fixed set $\mathbf{\Sigma}^{\dagger}(n)$, utilizing different randomness in each execution of $\mathcal{A}'(n)$ only alters the corresponding vector $\mathbf{b}$ and does not affect the coefficient matrix ${\mathbf{P}^{\dagger}}^{n}_{Lin}$. A useful observation made in the proof of Lemma 4 is that only ${\mathbf{P}^{\dagger}}^{n}_{Lin}$ determines whether there is a unique answer for V or not, and the right-hand side vector $\mathbf{b}$ has no effect. So the behavior of a fixed $\mathbf{\Sigma}^{\dagger}(n)$ is deterministic. This corollary eases our study of random leakage in the sequel. We need to study any $\mathbf{\Sigma}^{\dagger}(n)$ only once.

3.4.2 Random instance of leakage

In the RPM with parameters ϵ, $\mathbf{\Sigma}^{\dagger}(n)$ is a probabilistic set. $T(n)$ is a polynomial in the security parameter n, and the members of $\mathbf{\Sigma}^{\dagger}(n)$ are independently and identically decided. Therefore, set $\mathbf{\Sigma}^{\dagger}(n)$ is efficiently samplable from the mother set $\mathbf{\Sigma}(n)$. This means that for simulation purposes, we can create indistinguishable (for the MAP adversary) $\mathbf{\Sigma}^{\dagger}(n)$ samples. Each $\mathbf{\Sigma}^{\dagger}(n)$ either discloses the secret V or completely conceals it. One can model this, from the MAP adversary perspective, as an EEC with parameter $e_{\mathcal{A}'}(n,\epsilon)$ and V as input.

Our next goal is to find $e_{\mathcal{A}'}(n,\epsilon)$, which is the percentage of the cases that the sampled $\mathbf{\Sigma}^{\dagger}(n)$ results in complete disclosure of the secret V. For this purpose, instead of cumbersome analytic calculations, we estimate the required probability with a Monte Carlo approach [37]. At each desired pair of (n,ϵ), for a sufficiently large number of times N, different $\mathbf{\Sigma}^{\dagger}(n)$s are sampled from $\mathbf{\Sigma}(n)$. Then, the portion of these samples that reveal the secret V is figured out.

Sampling $\mathbf{\Sigma}^{\dagger}(n)$ and then computing the associated $\mathbf{P}^{\dagger n}_{Lin}$ can be simplified. For each leaked variable, the corresponding column is removed from the $\mathbf{P}^{n}_{Lin}$, and finally, only the columns for the unknown variables remain in $\mathbf{P}^{\dagger n}_{Lin}$. By closer inspection of the Gaussian-Elim algorithm, it is seen that instead of removing columns, we can multiply the leaked columns by zero. With this alternative method, $\mathbf{P}^{\dagger n}_{Lin}$ will have a fixed number of columns, where the columns corresponding to the leaked variables are all-zero. The pseudocode for sampling $\mathbf{P}^{\dagger n}_{Lin}$ from the main system of equations $\mathbf{P}^{n}_{Lin}$ is given in algorithm 4.

Assume $\widetilde{e_N}(n,\epsilon)$ is an estimation of $e_{\mathcal{A}'}(n,\epsilon)$ with the specified number of trials N. According to the *law of big numbers*, the empirical probability of an experiment eventually leads to the expected probability of that event [37]. The

Algorithm 4 Sampl $\mathbf{P}_{Lin}^{\dagger n}$

Input $\mathbf{P}_{Lin}^{n}$ and ϵ
Output Random $\mathbf{P}_{Lin}^{\dagger n}$

$\mathbf{P}_{Lin}^{\dagger n} = \mathbf{P}_{Lin}^{n}$ ▷ $\mathbf{P}_{Lin}^{\dagger n}$ and $\mathbf{P}_{Lin}^{n}$ are of the same size
for $col = 2$ **to** $T(n) + 1$ **do** ▷ Secret is in the first column
  $r \leftarrow^{\$} [0, 1]$
  if $r \leq \epsilon$ **then** ▷ This happens with probability ϵ
    $\mathbf{P}_{Lin}^{\dagger n}(:, col) = \mathbf{0}$ ▷ All the rows of this column are set to zero
return $\mathbf{P}_{Lin}^{\dagger n}$

difference between the empirical measurements and the expected probability lessens as the number of runs increases. Therefore, we can write

$$\lim_{N \to \infty} \widetilde{e_N}(n, \epsilon) = e_{\mathcal{A}'}(n, \epsilon). \tag{13}$$

The necessary condition is that the *variance* of the empirical estimation (i.e., the variance of $\widetilde{e_N}(n, \epsilon)$) decreases by increasing N, at the desired set of (n, ϵ). For the reported results in this paper, sample variance $h(N) = \widetilde{Var}(\widetilde{e_N}(n, \epsilon))$ is evaluated at various values of N. It is observed that $h(N)$ is acceptably decreasing function of N. So, the prerequisite of the limit in (13) is satisfied.

As N increase, the reliability of the estimation increases, but the required time also escalates. So, a middle ground must be chosen. this paper's results, we have verified that the following heuristic thumb rule is passed N.

$$N > \frac{10^4}{\widetilde{e_N}(n, \epsilon)} \tag{14}$$

Pseudocode for approximating $e_{\mathcal{A}'}(n, \epsilon)$ in a range of distinct ϵ and n values is given in algorithm 5, which is essentially a collection of the already described procedures.

Algorithm 5 Approx $e_{\mathcal{A}'}(n, \epsilon)$

Input The family $\mathcal{A}'(n)$
Output EEC parameter $e_{\mathcal{A}'}(n, \epsilon)$

for $n = 2$ **to** n_{max} **do** ▷ Selecting a sufficient n_{max} depends on the application
  $\mathbf{M} =$ Create-M$(\mathcal{A}'(n), \mathbf{\Sigma}(n))$ ▷ This and the following two lines are independent of ϵ
  $\mathbf{E} =$ Gaussian-Elim $(\mathbf{M})$
  $\mathbf{P}_{Lin}^{n} =$ Extract-Linear$(\mathbf{E})$
  for $\epsilon = \epsilon_{min}$ **to** ϵ_{max} **do** ▷ Recommended values are $\epsilon_{min} = 0.01$, step $= 0.01$, and $\epsilon_{max} = 0.3$
    $j = 0$
    for $i = 0$ **to** N **do**
      $\mathbf{P}_{Lin}^{\dagger n} \leftarrow$ Sampl$(\mathbf{P}_{Lin}^{n}, \epsilon)$
      $\mathbf{G} =$ Gaussian-Elim $(\mathbf{P}_{Lin}^{\dagger n})$
      if $\mathbf{G}(1, 2 : T(n)) == 0$ **then** ▷ The secret recovered
        $j = j + 1$
    $\widetilde{e_N}(n, \epsilon) = j/N$ ▷ Work for this (n, ϵ) pair finished
return $e_{\mathcal{A}'}(n, \epsilon)$

In relation to the given producer for approximating e, some points are highlighted here.

- Although the presented algorithms are general and work for any extension field of $\mathbb{F}_2$, for the implementations in this paper, we have used an 8-bit field with the irreducible polynomial given by $X^8 \oplus X^4 \oplus X^3 \oplus X \oplus 1$.
- Run-time for approximating $e_{\mathcal{A}'}(n, \epsilon)$ in a single pair of (n, ϵ) at values of $n < 10$, for the results that are presented throughout this paper, was usually less that one minute.
- Interpolation and extrapolation methods can be used to approximate $e_{\mathcal{A}'}(n, \epsilon)$ beyond the simulation points.

- Based on the physical situation, non-equal values for the leakage rates of different variables can be applied. This may be useful for studying the effects of imperfect randomness.
- Assuring relation (14) may require running the algorithm several times.

3.5 Driving SRP for a linear $\mathcal{A}'$

In part 2.3, we defined SRP as the success probability of the MAP adversary at the correct estimation of the secret V. Based on the EEC result; we can derive an approximate value for the $SRP_{\mathcal{A}'}(n, \epsilon)$.

Note that, when the adversary learns nothing from the leaked set of variables, she still has the chance to find the secret value V by simply guessing it, which will be right with probability $1/q$. So, the SRP corresponding to the linear algorithm $\mathcal{A}$ will be as the following.

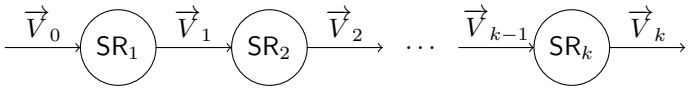
$$SRP_{\mathcal{A}'}(n, \epsilon) = |e_{\mathcal{A}'}(n, \epsilon) + \frac{1}{q}[1 - e_{\mathcal{A}'}(n, \epsilon)] - \frac{1}{q}| = \frac{q-1}{q} e_{\mathcal{A}'}(n, \epsilon) \tag{15}$$

However, in the rest of the paper, especially when handling non-linear systems, we define SRP as the probability of extracting the secret (without the random guessing). In this way, we ignore the multiplicative term $\frac{q-1}{q}$.

3.6 Examining different SR gadgets

In this section, the corresponding e for different linear SR gadgets is evaluated and compared. An SR gadget takes as input an n-sharing vector $\overrightarrow{V}_0$ and produces a new n-sharing $\overrightarrow{V}_1$ for the same secret V. SR gadgets like the other shared gadgets are composed of elementary field gates.

A construction composed of k similar SR gadget in tandem is used for our studies. The output of the ith SR gadget is denoted by $\overrightarrow{V}_i$. Repetitive refreshing of the same secret V, as pointed in [33], has various practical use cases.



The obtained $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon)$ will directly depend on the k. For a valid comparison between different SR candidates, we use a fixed $k = 10$ for deriving the results. The main target of RPM security analysis is to study the dependency of the $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon)$ value to the order n at the desired values of ϵ. the leakage rate is specified from the physical leakage situation. See [38] for estimation of ϵ.

SR-Simple. The first SR candidate is SR-Simple [18], whose pseudocode is given in algorithm 6. This SR implementation once was used in almost any Boolean masking scheme.

Algorithm 6 SR-Simple

Input The n-sharing $\overrightarrow{V}_0 = (V_{1,0}, V_{2,0}, \ldots, V_{n,0})$
Output An n-sharing $\overrightarrow{V}_1 = (V_{1,1}, V_{2,1}, \ldots, V_{n,1})$ for the same secret V

1: $r_n = 0$
2: **for** $i = 1$ **to** $n - 1$ **do**
3: $\quad r_i \leftarrow^{\$} \mathbb{F}_q$
4: $\quad V_{i,1} = V_{i,0} \oplus r_i$
5: $\quad r_n = r_n \oplus r_i$
6: $V_{n,1} = V_{n,0} \oplus r_n$
7: **return** $\overrightarrow{V}_1$

The SR-Simple in each invocation requires $n - 1$ new randomness. Its computational complexity is $O(n)$ and the number of its variables is $|\mathbf{\Sigma}_{\mathsf{SR\text{-}Simple}}(n)| = 4n - 1$. n input elements, r_n before the for loop, $3(n-1)$ variables inside the for loop, and one variable after the for loop. With the method described in part 3.3, $D(n) = 3n$ is obtained.

Since the composition $\mathsf{SR} \to \ldots \to \mathsf{SR}$ is linear, we can derive the family of relations $\mathbf{P}^n_{Lin}$ for it. With the procedure given in algorithm 4, for $k = 10$, at order $n = 5$, with leakage rate $\epsilon = 0.1$, we obtained $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon) \approx 4 \times 10^{-4}$, for SR-Simple. In figure 3, the approximation results for a constant $\epsilon = 0.1$, at different values of order n, are plotted.

SR-SNI. The next linear SR candidate is SR-SNI. Which is proved to be SNI secure in TPM, for $t = n - 1$. Its pseudocode is given in algorithm 7. In each invocation, this SR gadget requires $\frac{n^2-n}{2}$ new randomness and has $O(n^2)$ computational complexity. Common sense is that this extra randomness consumption combined with its higher processing complexity makes the SR-SNI more secure than the SR-Simple. However, quite surprisingly, for $k = 10$ consecutive refreshing structure, our results show that in any point $(n > 2, \epsilon)$, the value of $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon)$ is higher with SR-SNI than with the SR-Simple. For example, with $n = 5$ and $\epsilon = .1$, we obtained $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon) \approx 3\times10^{-3}$, which is nearly ten times more than the corresponding value for SR-Simple.

Algorithm 7 SR-SNI

Input The n-sharing $\overrightarrow{V}_0 = (V_{1,0}, V_{2,0}, \ldots, V_{n,0})$
Output An n-sharing $\overrightarrow{V}_1 = (V_{1,1}, V_{2,1}, \ldots, V_{n,1})$ for the same secret V

for $i = 1$ **to** n **do**
  for $j = i + 1$ **to** n **do**
    $r \leftarrow^{\$} \mathbb{F}_q$
    $V_{i,0} = V_{i,0} \oplus r$
    $V_{i,0} = V_{j,0} \oplus r$
for $i = 1$ **to** n **do**
  $V_{i,1} = V_{i,0}$
return $\overrightarrow{V}_1$

Compared to SR-simple, the defining relations for SR-SNI are *symmetric* over the elements $\overrightarrow{V}_0$ and the elements of $\overrightarrow{V}_1$. In SR-simple, the last share of $\overrightarrow{V}_0$ (i.e., $V_{n,0}$) participates in different equations. However, in SR-SNI, relations are indifferent to members of $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$. In the SR-SNI, symmetry is apparent. This feature will be very beneficial in proving the RPM security of complex structures in the rest of this paper.

SR-Rot. The third SR candidate is SR-Rot. It requires n randomness in each invocation, and its computational complexity is $O(n)$. The pseudocode given in algorithm 8 describes it.

This gadget is introduced in [39], and its SNI security is studied in [40]. The format of SR-Rot that is presented here has not SNI security for $t = n - 1$ at orders $n > 4$. For all values of ϵ, our evaluations show that SR-Rot is RPM secure. The algorithm is symmetric over the input and the output variables. So, it has the above-mentioned symmetry future.

SR-Rot has comparatively higher leakage rates than SR-Simple. With $k = 10$, $n = 5$, and $\epsilon = 0.1$, we obtained $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon) \approx 6.5 \times 10^{-4}$, which is approximately twice the corresponding value for SR-Simple.

For the three SR gadgets, at $k = 10$, and $\epsilon = 0.1$, results for $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n, \epsilon)$ are plotted in figure 1.

Algorithm 8 SR-Rot

Input The n-sharing $\overrightarrow{V}_0 = (V_{1,0}, V_{2,0}, \ldots, V_{n,0})$
Output An n-sharing $\overrightarrow{V}_1 = (V_{1,1}, V_{2,1}, \ldots, V_{n,1})$ for the same secret V

for $i = 1$ **to** n **do**
  $r_i \leftarrow^{\$} \mathbb{F}_q$
for $i = 1$ **to** 2 **do**
  for $j = 1$ **to** n **do**
    $ind = \mathsf{mod}(i + j - 2, n) + 1$ ▷ To apply a cyclic shift in the used randomness variables
    $V_{i,0} = V_{i,0} \oplus r_{ind}$
for $i = 1$ **to** n **do**
  $V_{i,1} = V_{i,0}$
return $\overrightarrow{V}_1$

3.6.1 Insecurity of SR-SNI at high leakage rates

Before the results in this paper, SR-SNI was considered the most secure linear candidate for refreshing an n-sharing. However, by performing a thorough assessment, we observed that at any order $n > 2$, and any ϵ, $e_{\mathsf{SR\text{-}SNI}}$ has higher

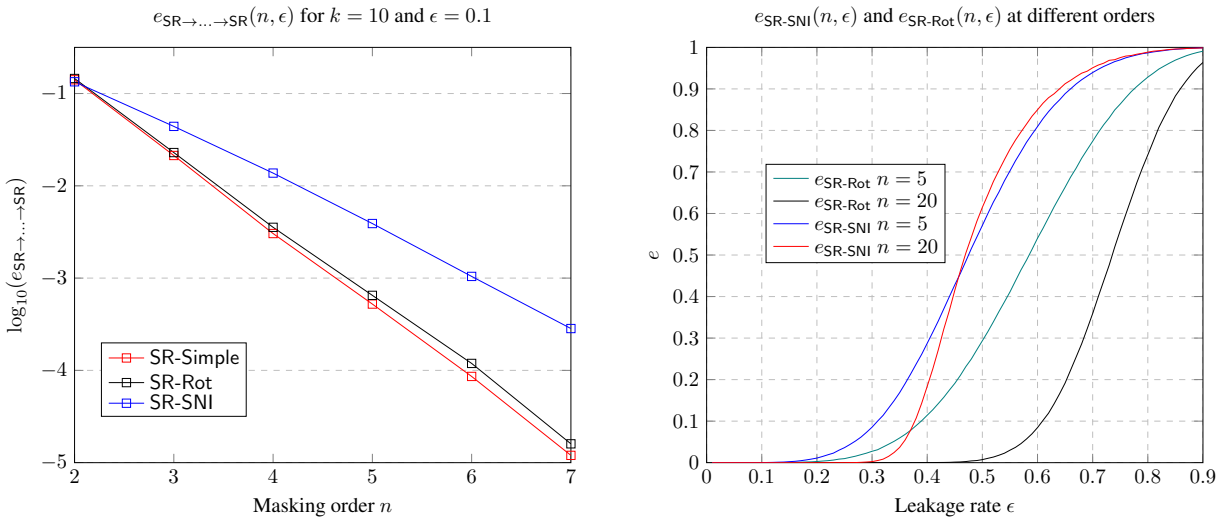


Figure 1: (Left) $e_{\mathsf{SR}\to\ldots\to\mathsf{SR}}(n,\epsilon)$ for different SR gadgets at $k=10$ and leakage rate $\epsilon=0.1$. (Right) $e_{\mathsf{SR\text{-}SNI}}(n,\epsilon)$ and $e_{\mathsf{SR\text{-}Rot}}(n,\epsilon)$ at different orders for $\epsilon<.09$

values compared to the other two SR gadgets. As another negative point for this refreshing, we observed that at higher leakage rates, increasing the masking order also increases the $e_{\mathsf{SR\text{-}SNI}}$. A similar event does not happen for the other two SR gadgets.

In figure 1, $e_{\mathsf{SR\text{-}SNI}}$ and $e_{\mathsf{SR\text{-}Rot}}$ are plotted at different orders n. For $\epsilon>0.45$, the behavior of $e_{\mathsf{SR\text{-}SNI}}$ changes, and it begins to be insecure. In other words, for a single SR-SNI at $\epsilon>0.45$, there is an order $\widetilde{n}$, beyond which the MAP adversary with an increasingly high probability can recover the secret of the input n-sharing. This is what simulation results tell. Nevertheless, it is not clear what is a valid justification for this behavior.

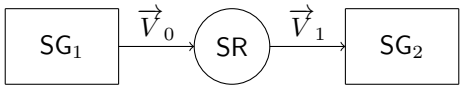
Note that, according to our definition of RPM security, we cannot tell that SR-SNI is insecure. In fact, for at least $\epsilon<0.15$, by exploring and curve-fitting the results of various orders, we can give the following exponentially decaying relation for $e_{\mathsf{SR\text{-}SNI}}$.

$$e_{\mathsf{SR\text{-}SNI}}(n,\epsilon)\leq\epsilon^{0.6n} \tag{16}$$

4 Modeling leakage of SR gadgets

According to part 2.2.1, in the masking procedure, all the computations are rewritten with elementary gates after some preprocessing. Then, these elementary gates are replaced with their shared counterparts, which we call SGs. As the next step, SR gadgets are inserted between any two consecutive SGs that have a common secret.

Assume SG_1 and SG_2 both are computing on the same secret V.The output of SG_1, $\overrightarrow{V}_0$, which is an n-sharing of the secret V, instead of directly passing to SG_2, is first fed into an SR gadget, and then $\overrightarrow{V}_1$, which is a new n-sharing for the secret V, is given to SG_2. Each variable of the whole construction independently may leak to the adversary. The adversary's goal -as before- is to obtain the most probable guess for the secret V.



Intuitive justification of the role of SR is quite tricky. It should be noted that SR makes no change in the results of the computations. However, by introducing fresh randomness, it tries to decrease the usefulness of leaked variables. SR disconnects boundary variables of SG_1 and SG_2. As another role, SR facilitates the *reduction* of the RPM security of the compound construction $\mathsf{SG}_1\to\mathsf{SR}\to\mathsf{SG}_2$ to its building gadgets' RPM security. This reduction is a significant step toward evaluating the security of more sophisticated constructions like a masked implementation of the S-Box of the AES.

4.1 Reduction with leak-free SR gadgets

In [8], by assuming that SR gadgets are leak-free (i.e., an adversary cannot probe their internal variables), the authors could decompose the security of the construction $\mathsf{SG}_1 \rightarrow \mathsf{SR} \rightarrow \mathsf{SG}_2$ -in their metric for the security- to the security of SG_1 and SG_2. A leak-free SR gadget is required for preserving the conditional independence of $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$. This independence is critical for the proofs given in [8].

In the software implementations, the SR gadget (like the other SGs) is a piece of code. The adversary can access its internal variables like any other variable of the entire masked algorithm. Therefore, in software implementations, the requirement of leak-free SR gadgets cannot be fulfilled.

Effectively taking into account the SR gadgets leakages, instead of neglecting, is a long-standing open problem in the RPM. As an answer to this problem, in this section, we provide a quite tight bound on what extra-benefit the adversary can get from the leakage of an SR gadget.

Effects of leakage from SR gadgets. Before diving into the detailed assessments, we briefly describe the consequences of the leakage from SR gadgets.

Leakage from $\mathbf{\Sigma}_{\mathsf{SR}}$ has two effects.

1. Since the secret V depends on the $\mathbf{\Sigma}_{\mathsf{SR}}$, if an adversary learns a good portion of variables in $\mathbf{\Sigma}_{\mathsf{SR}}$, she may uncover the secret V with this knowledge directly.
2. As an indirect effect, by leakage of $\mathbf{\Sigma}_{\mathsf{SR}}$, extra parity relations between $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$ will emerge. These new equations interconnect parity relations of SG_1 and SG_2; this consequently reduces the RPM security of the whole structure.

4.2 A describing system of equations

Up-to here, we were confined to linear algorithms for which a linear system of equations thoroughly describes the relations. Now, in the construction given by $\mathsf{SG}_1(n) \rightarrow \mathsf{SR}(n) \rightarrow \mathsf{SG}_2(n)$, not all the relations are linear. Variables of $\mathbf{\Sigma}_{\mathsf{SG}_1}(n)$ and $\mathbf{\Sigma}_{\mathsf{SG}_2}(n)$ may have non-linear dependencies.

We separate the boundary variables and decompose the sets as $\mathbf{\Sigma}_{\mathsf{SG}_1(n)} = [\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}, \overrightarrow{V}_0]$, $\mathbf{\Sigma}_{\mathsf{SR}(n)} = [\mathbf{\Sigma}^I_{\mathsf{SR}(n)}, \overrightarrow{V}_0, \overrightarrow{V}_1]$, and finally $\mathbf{\Sigma}_{\mathsf{SG}_2(n)} = [\mathbf{\Sigma}^I_{\mathsf{SG}_2(n)}, \overrightarrow{V}_1]$. Based on this new ordering, the system of equations defining SR is denoted as $\mathtt{L}_{\mathsf{SR}(n)}(\mathbf{\Sigma}^I_{\mathsf{SR}(n)}, \overrightarrow{V}_0, \overrightarrow{V}_1, V)$, and the governing system of $\mathsf{SG}_1(n)$ with possibly non-linear relations as $\mathtt{NL}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}, \overrightarrow{V}_0, V)$. Equations set for $\mathsf{SG}_2(n)$ is defined similarly. It is supposed that all the variables and operations are in an appropriate field $\mathbb{F}_q$. The secret V is common to the three sets of equations. Relations in each set are confined only to their argument variables. By merging the three, the system of equations that completely describes $\mathsf{SG}_1(n) \rightarrow \mathsf{SR}(n) \rightarrow \mathsf{SG}_2(n)$ will be as follows.

$$\mathfrak{P}(n) = \begin{cases} \mathtt{NL}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}, \overrightarrow{V}_0, V) & = \mathbf{0} \\ \mathtt{L}_{\mathsf{SR}(n)}(\mathbf{\Sigma}^I_{\mathsf{SR}(n)}, \overrightarrow{V}_0, \overrightarrow{V}_1, V) & = \mathbf{0} \\ \mathtt{NL}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_2(n)}, \overrightarrow{V}_1, V) & = \mathbf{0} \end{cases} \tag{17}$$

To highlight the dependency of the derived relations to the order n, we denote the system (17) with $\mathfrak{P}(n)$.

Intuitively, the combination of the three equations set in $\mathfrak{P}(n)$ will give the adversary more power than a separate assessment of them. As an extreme case, the three combined may uniquely determine the secret V, but V appears to be uniformly random in considering each system alone.

4.3 Strategy of the MAP adversary

The MAP adversary is aware of the construction and the order n. Thus, she can develop a describing system of equations similar to $\mathfrak{P}(n)$.

Before receiving any leakage, the secret V is entirely random. Upon learning some unknowns of the $\mathfrak{P}(n)$ through leakage, she will substitute the disclosed variables in $\mathfrak{P}(n)$. Hence, she will reach to a new (possibly) non-homogeneous system of equations, which we call $\mathfrak{P}^\dagger(n)$. The secret V still is an unknown of this resultant system. Next, she will

list all the possible solutions of $\mathfrak{P}^{\dagger}(n)$. By partitioning the list of answers based on their value for V, she will finally announce the V's most frequent value as her estimate for the secret.

Our main work in this section will be to provide an upper bound on the success probability of the MAP adversary. Note that listing all the solutions of $\mathfrak{P}^{\dagger}(n)$ is impractical for high values of the order n. So we cannot directly evaluate the $SRP_{\mathsf{SG}_1 \to \mathsf{SR} \to \mathsf{SG}_2}(n, \epsilon)$. Instead, we will use tricky ideas to estimate an upper bound on this probability.

Order of variables As for linear gadgets, for non-linear gadgets, any reordering of the variables will change the describing equations. However, the solution set is independent of the representation chosen for the describing equations. As a result, the probability of correct estimation of the secret given any leakage will be invariant to $\mathfrak{P}(n)$ changes.

4.4 Related definitions

Before proceeding further, we pause to give two more definitions.

Definition 11. Primary solution. Among all the solutions of $\mathfrak{P}(n)$, we denote the primary solution by S^*, which is the actual values realized in the current execution of the experiment. In this notation, the realized value of the secret is denoted by V^*.

Definition 12. Helper $\mathcal{H}$. A new entity titled $\mathcal{H}$, who is aware of S^*, and may give some of S^*'s unleaked elements to the adversary. Undoubtedly, the MAP adversary that is receiving help from $\mathcal{H}$ performs better in guessing the secret V.

4.5 Requirements on the gadgets

To reduce the RPM security of the chain $\mathsf{SG}_1 \to \mathsf{SR} \to \mathsf{SG}_2$ to RPM security of its composing gadgets; some limitation on the structure of the participating gadgets is required. It is assumed that the SR gadget is linear. No such restriction on the structure of SG_1 and SG_2 is required. There are two mild prerequisites for the internal relation of variables in $\mathfrak{P}(n)$, which we explain next.

Condition 1. Considering the SR gadget alone that is described by $\mathsf{L}_{\mathsf{SR}(n)}(\mathbf{\Sigma}^I_{SR(n)}, \overrightarrow{V}_0, \overrightarrow{V}_1, V)$, the only relation between the input and the output vectors, $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$, should be their defining equality as $\oplus\overrightarrow{V}_0 = \oplus\overrightarrow{V}_1 = V$. Equivalently, conditioned on the value of V, $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$ should be independent. The SR candidates studied in part 3.6 satisfy this prerequisite.

For a given SR gadget, verification of this condition is straightforward; in section 3.3, it is outlined how to find the linear relations in an ordered set of variables by generating an appropriate matrix $\mathbf{M}$, and computing its row reduced echelon form $\mathbf{E}$. The same procedure can be applied to extract the linear relations of $2n$-tuple $[\overrightarrow{V}_0, \overrightarrow{V}_1]$. Condition 1 requires that there be exactly one free variable in the resultant matrix $\mathbf{E}$.

It is to be noted that the requirement for linear SR gadgets discussed here is not a restriction of our proof methods. We think this condition can be considered as a design role for SR gadgets. In other words, a well-designed SR gadget should satisfy condition 1.

Condition 2. $\mathfrak{P}(n)$ should be *symmetric* in the elements of $\overrightarrow{V}_0$ and also in the elements of $\overrightarrow{V}_1$.

We count on the symmetry of the describing pseudocode of each gadget for justification of this condition. For example, it is clear from the SR-Rot and the SR-SNI pseudocodes that they have at least one system of parity relations, which is symmetric over members of $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$. However, in the SR-Simple, variable $V_{n,0}$ is treated differently. So, $V_{n,0}$ may participate in more parity relations than others. Thus, the SR-Simple does not fulfill the requirement of this condition.

For SG_1 and SG_2 again, the symmetry of the construction, if such symmetry exists, will guarantee that condition 2 holds. We explain more on this when SAND candidates in part 5 are introduced.

4.6 Reduction of the security in RPM

As in the previous section for linear systems, we first focus on a single instance of leakage. Then, we extend the obtained results to the general case of random leakage.

Let the adversary learn some elements of the primary solution S^*. By direct substitution of these values in $\mathfrak{P}(n)$, a new system of equations as (18), denoted by $\mathfrak{P}^\dagger(n)$, will emerge. Description of $\mathfrak{P}^\dagger(n)$ depends on the set of leaked variables and their values.

$$\mathfrak{P}^\dagger(n) = \begin{cases} \mathtt{NL}^\dagger_{\mathsf{SG}_1(n)}(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^\dagger_0, V) & = \mathbf{b}_1 \\ \mathtt{L}^\dagger_{\mathsf{SR}(n)}(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}, \overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) & = \mathbf{b}_2 \\ \mathtt{NL}^\dagger_{\mathsf{SG}_2(n)}(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^\dagger_1, V) & = \mathbf{b}_3 \end{cases} \tag{18}$$

In (18), the upper script † is used to specify the remaining unknown variables and the relations among these variables. The secret V shall be unknown up to this point. $\mathbf{b}_1$ to $\mathbf{b}_3$ are known vectors that are emerged after substitutions.

Some members of $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$ may have leaked to the adversary. The terms $\overrightarrow{V}^\dagger_0$ and $\overrightarrow{V}^\dagger_0$ denote the remaining unknowns of these vectors. Based on the leaked members, we have the following identities.

$$\begin{cases} V & = (\oplus \overrightarrow{V}^\dagger_0) \oplus b_4 \\ V & = (\oplus \overrightarrow{V}^\dagger_1) \oplus b_5 \end{cases} \tag{19}$$

Where $\oplus$ operator before a vector means XOR of all its members, and scalar values b_4 and b_5 depend on the leaked variables.

The equation in the first row of (19) exists in both of $\mathtt{NL}^\dagger_{\mathsf{SG}_1(n)}$ and $\mathtt{L}^\dagger_{\mathsf{SR}(n)}$, and similarly, the second equation exists in both of $\mathtt{NL}^\dagger_{\mathsf{SG}_2(n)}$ and $\mathtt{L}^\dagger_{\mathsf{SR}(n)}$.

We continue by investigating $\mathfrak{P}^\dagger(n)$ for this instance of leakage. The lemma given below, which is proved in Appendix E, can simplify $\mathfrak{P}^\dagger(n)$.

Lemma 5. *By applying elementary row operations, the subsystem which is given by* $\mathtt{L}^\dagger_{\mathsf{SR}(n)}(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}, \overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) = \mathbf{b}_2$ *can be partitioned into two linearly independent sets of equations; without changing the space of solutions of* $\mathfrak{P}^\dagger(n)$*. These two subsystems are as follows.*

$$\begin{cases} \mathtt{L}_1(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}) & = \mathtt{L}_2(\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) \oplus \mathbf{b}^1_2 \\ \mathtt{L}_3(\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) & = \mathbf{b}^2_2 \end{cases} \tag{20}$$

Where $\mathbf{b}^1_2$ *and* $\mathbf{b}^2_2$ *are constant vectors dependent on the value of the leaked variables, and equations specified by* $\mathtt{L}_1$ *are linearly independent of each other.*

For two equation sets as $\mathtt{L}_1$ and $\mathtt{L}_2$ in (20), equivalence means that their respective lines are equal. So, $\mathtt{L}_1$ and $\mathtt{L}_2$ must have the same number of equations.

The description provided in (20) separates relations containing variables of $\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$ from the other equations in the subsystem $\mathtt{L}^\dagger_{\mathsf{SR}(n)}(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}, \overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) = \mathbf{b}_2$.

Since relations in $\mathtt{L}_1$ are linearly independent, any non-trivial linear combination of the equations in $\mathtt{L}_1(\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}) = \mathtt{L}_2(\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) \oplus \mathbf{b}^1_2$ will still have variables from $\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$ with non-zero coefficients.

Next, we show that the subsystem given in the first row of (20) does not change the probability distribution of $\widetilde{V}$.

Lemma 6, which is proved in Appendix F, provides a mathematical condition to rule out the relations containing variables of $\boldsymbol{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$.

Lemma 6. *In a system of equations containing both linear and non-linear relations as*

$$\begin{cases} \mathtt{L}_4(\boldsymbol{\Sigma}_1) & = \mathtt{L}_5(V, \boldsymbol{\Sigma}_2) \\ \mathtt{NL}_1(V, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3) & = \mathbf{0} \end{cases} \tag{21}$$

If variables specified by $\boldsymbol{\Sigma}_1$*,* $\boldsymbol{\Sigma}_2$*, and* $\boldsymbol{\Sigma}_3$ *are segregate, and equations in* $\mathtt{L}_4$ *are linearly independent of each other, then the probability distribution of* $\widetilde{V}$ *will only depend on the system* $\mathtt{NL}_1(V, \boldsymbol{\Sigma}_2, \boldsymbol{\Sigma}_3) = \mathbf{0}$*. So, as much as the estimation of the secret* V *is concerned, the subsystem given by* $\mathtt{L}_4(\boldsymbol{\Sigma}_1) = \mathtt{L}_5(V, \boldsymbol{\Sigma}_2)$ *can be ignored from (21).*

By applying Lemma 5, $\mathfrak{P}^{\dagger}(n)$ given in (18) can be decomposed as follows.

$$\mathfrak{P}^{\dagger}(n) = \begin{cases} \mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^{\dagger}_0, V) & = \mathbf{b}_1 \\ \mathtt{L}_{\text{SR-1}}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}) & = \mathtt{L}_{\text{SR-2}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) \oplus \mathbf{b}^1_2 \\ \mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}^2_2 \\ \mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}_3 \end{cases} \tag{22}$$

Where the linear equation sets denoted by $\mathtt{L}_{\text{SR-1}}(.)$ to $\mathtt{L}_{\text{SR-3}}(.)$ are created from the subsystem defined by $\mathtt{L}^{\dagger}_{\mathsf{SR}(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}, \overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}_2$, and $\mathbf{b}^1_2$ and $\mathbf{b}^2_2$ are constant vectors dependent to $\mathbf{b}_2$.

Note that subsystems represented by $\mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}$ and $\mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}$ are transferred intact from (18) to (22).

By using Lemma 6, we can remove the subsystem in the second row of (22), without affecting the probability distribution of $\widetilde{V}$. The resulted system is shown by $\mathfrak{P}^{\dagger\dagger}(n)$ and is as follows.

$$\mathfrak{P}^{\dagger\dagger}(n) = \begin{cases} \mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^{\dagger}_0, V) & = \mathbf{b}_1 \\ \mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}^2_2 \\ \mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}_3 \end{cases} \tag{23}$$

We can simplify $\mathfrak{P}^{\dagger\dagger}(n)$, especially the subsystem $\mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}^2_2$, further. Let E_{SR} be the event that the subsystem $\mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}^2_2$ alone determines the secret V.

For a given set of equations as $\mathtt{L}_{\text{SR-3}}$ by Lemma 4, it is straightforward to find out whether E_{SR} has occurred or not. If E_{SR} occurs, the job is finished, and the secret is uniquely identified.

Because we have preserved the probability distribution of $\widetilde{V}$, by the connection of the event E_{SR} and the EEC of SR, at each pair of (n, ϵ), we can write

$$\Pr(E_{\mathsf{SR}}) = e_{\mathsf{SR}}(n, \epsilon). \tag{24}$$

Assume E_{SR} has not occurred, we can process $\mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}^2_2$, and remove the secret V from it. As noted in (19), the following equations belong to $\mathfrak{P}^{\dagger\dagger}(n)$ even if we delete $\mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}^2_2$ from $\mathfrak{P}^{\dagger\dagger}(n)$.

$$\begin{cases} V & = (\oplus \overrightarrow{V}^{\dagger}_0) \oplus b_4 \\ V & = (\oplus \overrightarrow{V}^{\dagger}_1) \oplus b_5 \end{cases} \tag{25}$$

For each equation in $\mathtt{L}_{\text{SR-3}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}^2_2$, if the coefficient of V is non-zero, we can add either of the identities in (25) and hence remove V from it. At this point, the trivial relation

$$(\oplus \overrightarrow{V}^{\dagger}_0) \oplus b_4 = (\oplus \overrightarrow{V}^{\dagger}_1) \oplus b_5 \tag{26}$$

will always exist in this system. based on condition 1 given in part 4.5, we know that no other omnipresent equation will exist in this system.

Relation (26) is a dependent equation in $\mathfrak{P}^{\dagger\dagger}(n)$. Because, this identity comes from the XOR of the first line of (25), which is present in the $\mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}$, and the second line of (25), which belongs to the $\mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}$. Therefore, we can remove it from this system. The final system is shown by $\mathtt{L}_{\text{SR-4}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1) = \mathbf{b}^3_2$. By substituting in (23), we get

$$\mathfrak{P}^{\dagger\dagger}(n) = \begin{cases} \mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^{\dagger}_0, V) & = \mathbf{b}_1 \\ \mathtt{L}_{\text{SR-4}}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1) & = \mathbf{b}^3_2 \\ \mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}_3 \end{cases}. \tag{27}$$

Our next goal is to partition $\mathfrak{P}^{\dagger\dagger}(n)$ into two subsystems, one including $\mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}$, and the other including $\mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}$. For this objective, the relations in $\mathtt{L}_{\mathsf{SR}\text{-}4}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1) = \mathbf{b}_2^3$ are the main obstacle.

Most of the time (especially when ϵ is small enough), the system described by $\mathtt{L}_{\mathsf{SR}\text{-}4}$ contains no equation. For the case that $\mathtt{L}_{\mathsf{SR}\text{-}4}$ is not empty, if $\mathcal{H}$ gives the *present unknowns* in $\mathtt{L}_{\mathsf{SR}\text{-}4}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1) = \mathbf{b}_2^3$ to the adversary, then we can do the desired partitioning of the $\mathfrak{P}^{\dagger\dagger}(n)$. This extra leakage will increase the success probability of the MAP adversary.

We show the present unknowns (i.e., unknowns of the system with non-zero coefficients) with $\overrightarrow{V}^{\dagger\dagger}_0$ and $\overrightarrow{V}^{\dagger\dagger}_1$, correspondingly.

Let vectors $\mathbf{b}_6$ and $\mathbf{b}_7$ be the values of $\overrightarrow{V}^{\dagger\dagger}_0$ and $\overrightarrow{V}^{\dagger\dagger}_1$. $\mathcal{H}$ will give the $\mathbf{b}_6$ and $\mathbf{b}_7$ to the adversary. In this way, the new governing system of equation will be

$$\begin{cases} \mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^{\dagger}_0, V) & = \mathbf{b}_1 \\ \overrightarrow{V}^{\dagger\dagger}_0 & = \mathbf{b}_6 \\ \overrightarrow{V}^{\dagger\dagger}_1 & = \mathbf{b}_7 \\ \mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}_3 \end{cases}. \tag{28}$$

Before proceeding in this direction further, we need to point to a practically important issue.

Requesting the minimal help from $\mathcal{H}$. As stated, the disclosed vectors $\mathbf{b}_6$ and $\mathbf{b}_7$ will increase the success probability of the MAP adversary, equations in the system $\mathtt{L}_{\mathsf{SR}\text{-}4}(\overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1) = \mathbf{b}_2^3$ can be linearly processed to make them as spars as possible. A sparse representation will have fewer present variables.

Since for practical evaluations in this paper the size of subsystem $\mathtt{L}_{\mathsf{SR}\text{-}4}$ is not so big, we can find the most sparse representation by exhaustive search. In general, finding the most sparse representation for a given system of equations is a difficult task. Some papers suggest heuristic methods for finding a sparser basis for a given linear system [41].

Back to the study of (28), considering that the secret V is the only common variable of (28), we partition this compound system into the following two subsystems.

$$\mathfrak{P}^{\dagger\dagger}_1(n) = \begin{cases} \mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^{\dagger}_0, V) & = \mathbf{b}_1 \\ \overrightarrow{V}^{\dagger\dagger}_0 & = \mathbf{b}_6 \end{cases} \tag{29}$$

And similarly,

$$\mathfrak{P}^{\dagger\dagger}_2(n) = \begin{cases} \mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^{\dagger}_1, V) & = \mathbf{b}_3 \\ \overrightarrow{V}^{\dagger\dagger}_1 & = \mathbf{b}_7 \end{cases} \tag{30}$$

The probability distribution of $\widetilde{V}$ can be calculated for $\mathfrak{P}^{\dagger\dagger}_1(n)$ and $\mathfrak{P}^{\dagger\dagger}_2(n)$ separately, and then the results can be *properly combined* to get the final approximation of $\Pr(\widetilde{V})$.

Note that we are handling a single instance of random leakage. We showed that by getting help from $\mathcal{H}$, it is possible to bound knowledge of the MAP adversary by two separate systems of equations given in (29) and (30). To simplify these systems further, we need to make use of the probabilistic nature of leakage.

In the random leakage, $\overrightarrow{V}^{\dagger\dagger}_0$ and $\overrightarrow{V}^{\dagger\dagger}_1$ are random sets. These vectors provide extra leakage for the adversary that is studying $\mathtt{NL}^{\dagger}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_1(n)}, \overrightarrow{V}^{\dagger}_0, V) = \mathbf{b}_1$ and $\mathtt{NL}^{\dagger}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SG}_2(n)}, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}_3$.

We aim to utilize the symmetry introduced with condition 2 given in part 4.5, and model the extra random leakage provided via $\overrightarrow{V}^{\dagger\dagger}_0$ and $\overrightarrow{V}^{\dagger\dagger}_1$ as an *equivalent leakage rate* ϵ' on the elements of boundary vectors $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$.

The expected cardinality of $\overrightarrow{V}^{\dagger\dagger}_0$ and $\overrightarrow{V}^{\dagger\dagger}_1$ is a function of SR and is independent of SG_1 and SG_2. We define $f_{\mathsf{SR}}(n, \epsilon)$ for the most sparse representation of $\mathtt{L}_{\mathsf{SR}\text{-}4}$ as

$$f_{\mathsf{SR}}(n, \epsilon) = \frac{1}{2}\mathbb{E}[|\overrightarrow{V}^{\dagger\dagger}_0| + |\overrightarrow{V}^{\dagger\dagger}_1|]. \tag{31}$$

By the symmetry of the studied SR gadgets in this paper, and the identity $|\overrightarrow{V}_0| = |\overrightarrow{V}_1|$, one may expected a simpler definition as $f_{\mathsf{SR}}(n,\epsilon) = \mathbb{E}[|\overrightarrow{V}_0^{\dagger\dagger}|]$. Note that, by appropriately sparsing $\mathtt{L}_{\mathsf{SR}\text{-}4}$, we want to minimize the information that $\mathcal{H}$ should give to the adversary. However, sparsing with the only target of decreasing $|\overrightarrow{V}_0^{\dagger\dagger}|$ may give a non-optimal value for $|\overrightarrow{V}_1^{\dagger\dagger}|$. For establishing an upper bound on the success probability of the MAP adversary, we should ensure that $\mathcal{H}$ gives strictly enough information to the adversary.

Computing $f_{\mathsf{SR}}(n,\epsilon)$, defined in relation (31), for desired pairs of (n,ϵ), can be directly done with a Monte Carlo approach. For a sufficiently big number of times N, in the ith trial, an instance of random leakage is sampled, and the described procedure to find $h_N(i) = \frac{1}{2}(|\overrightarrow{V}_0^{\dagger\dagger}| + |\overrightarrow{V}_1^{\dagger\dagger}|)$ is carried out. The sample average $\frac{1}{N}\sum_{i=1}^{N} h_N(i)$ is reported as $f_{\mathsf{SR}}(n,\epsilon)$.

The following lemma defines the equivalent leakage rate ϵ' as a function of $f_{\mathsf{SR}}(n,\epsilon)$.

Lemma 7. *In RPM with parameter ϵ, the probability distribution of $\widetilde{V}$ conditioned on*

$$\begin{cases} \mathtt{NL}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}, \overrightarrow{V}_0, V) & = \mathbf{0} \\ \overrightarrow{V}_0^{\dagger\dagger} & = \mathbf{b}_6 \end{cases} \tag{32}$$

is statistically equivalent to the probability distribution of $\widetilde{V}$ conditioned on

$$\mathtt{NL}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}, \overrightarrow{V}_0, V) = \mathbf{0}. \tag{33}$$

Where in the latter case, the members of $\overrightarrow{V}_0$, instead of ϵ, leak with the higher rate

$$\epsilon' = \epsilon + \frac{f_{\mathsf{SR}}(n,\epsilon)}{n} - \epsilon\frac{f_{\mathsf{SR}}(n,\epsilon)}{n} \approx \epsilon + \frac{f_{\mathsf{SR}}(n,\epsilon)}{n}. \tag{34}$$

Proof. See Appendix G. □

Combining separate leakages for the secret V. Until here, we have identified three separate sources of information regarding the secret V. First is the occurrence of the event E_{SR}, which reveals the secret. The second is the subsystem defined by $\mathtt{NL}_{\mathsf{SG}_1(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}, \overrightarrow{V}_0, V)$ where the member of $\mathbf{\Sigma}^I_{\mathsf{SG}_1(n)}$ leak with probability ϵ and the members of $\overrightarrow{V}_0$ leak with parameter $\epsilon'(n,\epsilon)$. The third is the subsystem defined by $\mathtt{NL}_{\mathsf{SG}_2(n)}(\mathbf{\Sigma}^I_{\mathsf{SG}_2(n)}, \overrightarrow{V}_1, V)$ with the leak conditions similar to the second case.

Combining leakage information from different sources in general can be quite challenging. Thanks to the derived EECs for the shared gadgets, this combination will be an easy task.

EEC in the next sections is developed for the type of shared non-linear gadgets that we will encounter. These EECs (in contrast to the case of linear gadgets) are valid only for a limited (but still practical) region of ϵ values.

Lemma 8. *If the MAP adversary learns V through a set of independent EECs with parameters $\{e_1, e_2, ..., e_k\}$, then this knowledge is equivalent to a new EEC with parameter $e^* \leq e_1 + e_2 + ... + e_k$.*

Proof. The lemma is obtained by applying the union bound on the probabilities. □

At least for linear gadgets, considering different leakage rates on the elements of its $\mathbf{\Sigma}$ is technically feasible. However, to provide closed-form formulations, we assume that all the members of $\mathbf{\Sigma}$ are leaking to the adversary with the higher probability ϵ'. This assumption, of course, will increase the success probability of the MAP adversary further and hence worsen the obtained bounds. With a simple reduction, it is easy to show that for any gadget SG, at any fixed order n, $e_{\mathsf{SG}}(n,\epsilon)$ is a monotonically increasing function of ϵ.

By directly applying Lemma 8, we can finally give an EEC for the chain $\mathsf{SG}_1 \rightarrow \mathsf{SR} \rightarrow \mathsf{SG}_2$ with the following parameter.

$$\begin{aligned} e_{\mathsf{SG}_1 \rightarrow \mathsf{SR} \rightarrow \mathsf{SG}_2}(n,\epsilon) &= \Pr(\widetilde{V} = V) \\ &= \Pr(\widetilde{V} = V | E_{\mathsf{SR}}) \Pr(E_{\mathsf{SR}}) + \Pr(\widetilde{V} = V | \overline{E_{\mathsf{SR}}}) \Pr(\overline{E_{\mathsf{SR}}}) \\ &\leq \Pr(E_{\mathsf{SR}}) + \Pr(\widetilde{V} = V | \overline{E_{\mathsf{SR}}}) \\ &= e_{\mathsf{SR}}(n,\epsilon) + \Pr(\widetilde{V} = V | \overline{E_{\mathsf{SR}}}) \\ &\leq e_{\mathsf{SR}}(n,\epsilon) + e_{\mathsf{SG}_1}(n,\epsilon') + e_{\mathsf{SG}_2}(n,\epsilon') \end{aligned} \tag{35}$$

Where ϵ' is given in the (34).

Bound given in (35) is derived for those values of ϵ that the corresponding EECs for SG_1 and SG_2 are valid. In the next section, for an SAND gadget, EEC and its validity region will be given.

Condition 1 in part 4.5 for SR-Simple is not satisfied. For SR-SNI by the described procedure, we computed $f_{\mathsf{SR\text{-}SNI}}$ at various pairs of (n,ϵ). By curve-fitting the obtained results, we have found the following estimations.

$$\begin{aligned} &f_{\mathsf{SR\text{-}SNI}}(n,\epsilon) \leq \frac{1}{3} n\epsilon \quad \text{valid for } (n \geq 3, \epsilon \leq 0.1) \\ &f_{\mathsf{SR\text{-}SNI}}(n,\epsilon) \leq 5n\epsilon^2 \quad \text{valid for } (n \geq 4, \epsilon \leq 0.15) \\ &f_{\mathsf{SR\text{-}SNI}}(n,\epsilon) \approx 2n\epsilon^2 \quad \text{valid for } (n \rightarrow \infty, \epsilon \leq 0.15) \end{aligned} \tag{36}$$

Unfortunately, $\frac{1}{n} f_{\mathsf{SR\text{-}Rot}}(n,\epsilon)$ at any value of ϵ grows unboundedly. So, we cannot use SR-Rot for proofing the RPM security of a composition of different gadgets.

It is interesting to note that SR-SNI has the highest leakage rate. Nevertheless, its leakage propagation is much less than SR-Rot.

4.7 Multiple SR gadgets at the output

In the next section, we will encounter structures as in figure 2, where the output of one shared gadget is fed into multiple SR gadgets. In this case, the reduction provided is still possible with some modifications. Parity equations related to each SR gadget is decomposed as in Lemma 5, and the equations containing internal variables are discarded as in Lemma 6. For SR_i, the remaining relations (after sparsing) are of the format $\mathsf{L}_{\mathsf{SR}_i - 4}(\overrightarrow{V}_0^{\dagger}, \overrightarrow{V}_i^{\dagger})$. $\mathcal{H}$ will give the present unknowns to the adversary. The rest of the proof steps are straight-forward, so we state the final result.

For the construction given in figure 2, assuming that the SR gadgets are similar, parameter $e(n,\epsilon)$ is obtained as follows.

$$e(n,\epsilon) \leq k e_{\mathsf{SR}}(n,\epsilon) + e_{\mathsf{SG}}(n, \epsilon + k\frac{f_{\mathsf{SR}}(n,\epsilon)}{n}) \tag{37}$$

4.8 Multiple SR gadgets in the input

In figure 2, if SG_1 was absent, then we face a problem for $k > 1$. In this case, multiple (more than one) SR gadgets are feeding from the input point. This is what happens in the analysis of the S-Box in figure 5.

Taking a similar approach as was done in this section, the following bound for $e(n,\epsilon)$ in this case can be obtained.

$$e(n,\epsilon) \leq k e_{\mathsf{SR}}(n,\epsilon) + [\epsilon + k\frac{f_{\mathsf{SR}}(n,\epsilon)}{n}]^n \tag{38}$$

5 An SAND with RPM security

As an introduction to the rest of this paper, after a brief note on the importance of non-linear algorithms, we point to the primary method for probability analysis over non-linear systems. Then, the crucial role of the SAND algorithm for masking of non-linear functions is stated.

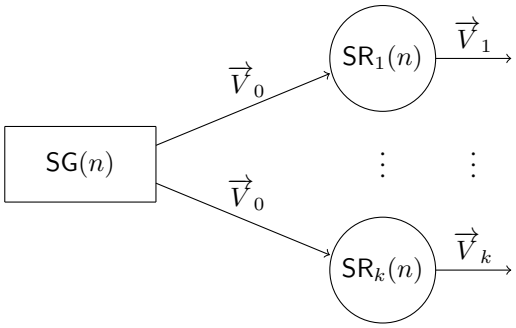


Figure 2: The output of one gadget is feeding multiple SR gadgets.

Role of non-linear algorithms. Linear algorithms cover only a minimal portion of cryptographic functions. In many primitives, including block ciphers, non-linear parity relations are present. Non-linearity is an indispensable ingredient for the security of block ciphers. However, unfortunately, compared to linear systems, for general non-linear systems, Algebra has much fewer tools, so direct approximation of the SRP (as we did for linear systems) is not feasible.

Belief propagation algorithm. For characterizing approximate probability distribution of a random variable $V \in \mathbb{F}_q$ conditioned on (non-)linear equations $\mathfrak{P}(V, .)$, *belief propagation* (BP), also known as *message passing*, is a prevalently used tool.

The BP algorithm's performance dominantly depends on the *sparsity* of underlying equations in $\mathfrak{P}(V, .)$. An equation is labeled as sparse if it is composed of a minimal number of variables. The sparsity of the equations in $\mathfrak{P}(V, .)$ directly affects reliability and the workload of BP.

BP is used in some RPM security evaluations [11, 42]. The inherent suboptimality of this algorithm prohibits it from being a security proof tool [43]. MAP adversary is not computationally bounded and, by definition, outperforms any suboptimal procedure. Therefore, to prove an algorithm's security against a MAP adversary, new technical mechanisms rather than BP are required.

Non-linearity translates to SAND**.** In part 2.2.1, it made clear that through Lagrange interpolation for each S-Box, the SAND gadget handles the non-linearity of the block cipher. Higher degree affine operations such as field squaring are also out of our definitions for linear algorithms. Nevertheless, the main challenge in investigating the after-effect of random leakage will be the intricate SAND gadget.

5.1 Candidates for SAND gadget

The SAND gadget with two n-sharing inputs $\overrightarrow{X}$ and $\overrightarrow{Y}$ outputs an n-sharing vector $\overrightarrow{Z}$, where the relation between participating secrets is $Z = XY$.

The prominent candidate for SAND realization is the SAND-ISW, which was initially proved threshold secure with $t = \lfloor n/2 \rfloor$ [12]. In [18], operations were generalized to extended fields, and security proof modifications were made to declare it secure against $t = n - 1$ probes.

The SAND-ISW construction, with the pseudocode given in algorithm 9, uses n^2 field multiplication and requires $(n^2 - n)/2$ randomness variables. Numerous research works and practical implementations have used this SAND candidate. In TPM, it is proved to be t-SNI secure for $t = n - 1$ [14]. However, in RPM with a method similar to BP, it is shown to be insecure for $\epsilon > 1/n$ [11, 13]. From then, providing a new realization for the SAND with proved RPM security is still an open problem.

In this part, we analyze the SAND candidate introduced in [13], which we call SAND-Rec. With some modifications in the original construction, we prove the RPM security of SAND-Rec for constant values of ϵ. As for the linear algorithms, we give an EEC for SAND-Rec for a region of ϵ values. This EEC will help us to propose an RPM secure S-Box implementation for the AES block cipher in the next section.

To bound the success probability of the MAP adversary, conditioned on a non-linear system of equations governing the SAND-Rec, our primary tool is to introduce a more (and a less) informative but linear conditions to obtain an upper (and a lower) bounds on the success probability of the MAP adversary.

Algorithm 9 SAND-ISW

Input The n-sharings $\overrightarrow{X}$ and $\overrightarrow{Y}$
Output An n-sharing for $\overrightarrow{Z}$ where $Z = XY$

for $i = 1$ **to** n **do**
  for $j = i + 1$ **to** n **do**
    $r_{i,j} \leftarrow^{\$} \mathbb{F}_q$
    $r_{j,i} = (r_{i,j} \oplus X_i Y_j) \oplus X_j Y_i$ ▷ Order of the $\oplus$ operations are important
for $i = 1$ **to** n **do**
  $Z_i = X_i Y_i$
  for $j = 1$ **to** n **,** $j \neq i$ **do**
    $Z_i = Z_i \oplus r_{i,j}$
return $\overrightarrow{Z} = (Z_1, Z_2, \ldots, Z_n)$

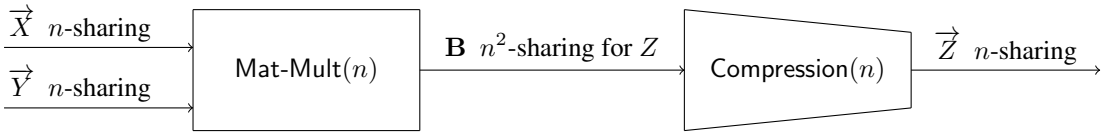


Figure 3: The SAND-Rec composed of the Mat-Mult and the Compression sub-algorithms

5.2 Structure of SAND-Rec

In [13], the authors introduced the idea of *recursive refreshing and multiplication* for implementing an SAND algorithm. They provided a heuristic justification for the RPM security of their algorithm.

Here, we review this construction with some changes. The modifications are done to increase the security in RPM and decrease the algorithm's overall computational load. However, the critical idea of recursive refreshing is still intact. For a description of the original algorithm, refer to [13].

SAND-Rec, as shown in figure 3, is composed of two sub-algorithms working in tandem named Mat-Mult and Compression, respectively. Mat-Mult as input receives $2n$ elements for the two n-sharings of $\overrightarrow{X}$ and $\overrightarrow{Y}$ and produces an n^2-sharing for the secret $Z = XY$. These n^2 values are stored in an $n \times n$ matrix $\mathbf{B}$ for further processing. Next, the Compression algorithm takes $\mathbf{B}$ and uses new fresh randomness to output $\overrightarrow{Z}$, which is an n-sharing for the secret Z.

The pseudocode of Mat-Mult given in [13] is limited to the values of n that are powers of 2. Here we give a description suitable for all values of n.

5.2.1 Mat-Mult sub-algorithm

The pseudocode for Mat-Mult is presented in algorithm 10. We give an explanation of its structure in the following.

Mat-Mult is a recursive algorithm. At the first invocation, it takes n_X-sharing $\overrightarrow{X}$ and n_Y-sharing $\overrightarrow{Y}$. In this call to Mat-Mult, as initial call, n_X and n_Y are equal to the order n.

If n_X and n_Y both be 1, then $\mathbf{B} = X_1 Y_1$ and the function invocation terminates. Otherwise, each of the input vectors $\overrightarrow{X}$ and $\overrightarrow{Y}$ are partitioned into two left and right subvectors. For $\overrightarrow{X}$, the resultant subvectors will be denoted by $\overrightarrow{X_L}$ and $\overrightarrow{X_R}$.

$$\overrightarrow{X_L} = (X_1, X_2, \ldots, X_{\lfloor n/2 \rfloor}) \quad \overrightarrow{X_R} = (X_{\lfloor n/2 \rfloor + 1}, \ldots, X_n) \tag{39}$$

If n_X is even, the length of these left and right vectors will be the same. However, if n_X is odd, then the length of $\overrightarrow{X_L}$ will be one less than the length of $\overrightarrow{X_R}$. $\overrightarrow{Y}$ is partitioned similarly. Following this principle, if $n_X = 1$ and $n_Y > 1$, then $\overrightarrow{X_L}$ will be an empty vector. Empty vectors are represented by the [ ] symbol.

The rest of the computations are carried on the four above-generated subvectors. The guiding rule for the follow-up is the below simple identity.

$$\begin{aligned} XY &= (X_L \oplus X_R)(Y_L \oplus Y_R) = X_L Y_L \oplus X_L Y_R \oplus X_R Y_L \oplus X_R Y_R \\ &= \oplus \mathsf{Mat\text{-}Mult}(\overrightarrow{X_L}, \overrightarrow{Y_L}) \oplus \mathsf{Mat\text{-}Mult}(\overrightarrow{X_L}, \overrightarrow{Y_R}) \oplus \mathsf{Mat\text{-}Mult}(\overrightarrow{X_R}, \overrightarrow{Y_L}) \oplus \mathsf{Mat\text{-}Mult}(\overrightarrow{X_R}, \overrightarrow{Y_R}) \end{aligned} \tag{40}$$

According to our notations, $X_L = \oplus \overrightarrow{X_L}$, and the other secrets are defined similarly.

The output of Mat-Mult for $\overrightarrow{X_L}$ and $\overrightarrow{Y_L}$ as inputs will be placed in the sub-matrix $\mathbf{B}_{LL}$. The other three Mat-Mult calls will generate $\mathbf{B}_{LR}$, $\mathbf{B}_{RL}$, and $\mathbf{B}_{RR}$, respectively. The final $\mathbf{B}$ matrix will be

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{LL} & \mathbf{B}_{LR} \\ \mathbf{B}_{RL} & \mathbf{B}_{RR} \end{bmatrix}. \tag{41}$$

The key heuristics observation in [13] was that refreshing of input vectors before each call to Mat-Mult -by incorporating fresh extra randomness in the algorithm- may increase the security against random probing leakages.

For length-one inputs, the refreshing gadget will be the identity function. If $\overrightarrow{X_L}$ is empty, then $\mathbf{B}_{LL}$ and $\mathbf{B}_{LR}$ both will be empty. Similarly, if $\overrightarrow{Y_L}$ happens to be empty, then $\mathbf{B}_{LL}$ and $\mathbf{B}_{RL}$ will be empty.

Algorithm 10 Mat-Mult

Input The n-sharings $\overrightarrow{X}$ and $\overrightarrow{Y}$
Output An n^2-sharing for $Z = XY$ saved in the matrix $\mathbf{B}$

if $n_X = 1$ and $n_Y = 1$ **then**
  $\mathbf{B} = X_1 Y_1$ ▷ This is the return point for the recursive calls
else
  $\overrightarrow{X_L} = (X_1, X_2, \ldots, X_{\lfloor n_X/2 \rfloor})$ $\overrightarrow{X_R} = (X_{\lfloor n_X/2 \rfloor + 1}, \ldots, X_{n_X})$
  $\overrightarrow{Y_L} = (Y_1, Y_2, \ldots, Y_{\lfloor n_Y/2 \rfloor})$ $\overrightarrow{Y_R} = (Y_{\lfloor n_Y/2 \rfloor + 1}, \ldots, Y_{n_Y})$
  if $\overrightarrow{X_L} \neq [\,]$ and $\overrightarrow{Y_L} \neq [\,]$ **then**
    $\mathbf{B}_{LL} = \mathsf{Mat\text{-}Mult}(\mathsf{SR}(\overrightarrow{X_L}), \mathsf{SR}(\overrightarrow{Y_L}))$ ▷ Even in this call SR is used
  if $\overrightarrow{X_L} \neq [\,]$ and $\overrightarrow{Y_R} \neq [\,]$ **then**
    $\mathbf{B}_{LR} = \mathsf{Mat\text{-}Mult}(\mathsf{SR}(\overrightarrow{X_L}), \mathsf{SR}(\overrightarrow{Y_R}))$
  if $\overrightarrow{X_R} \neq [\,]$ and $\overrightarrow{Y_L} \neq [\,]$ **then**
    $\mathbf{B}_{RL} = \mathsf{Mat\text{-}Mult}(\mathsf{SR}(\overrightarrow{X_R}), \mathsf{SR}(\overrightarrow{Y_L}))$
  if $\overrightarrow{X_R} \neq [\,]$ and $\overrightarrow{Y_R} \neq [\,]$ **then**
    $\mathbf{B}_{RR} = \mathsf{Mat\text{-}Mult}(\mathsf{SR}(\overrightarrow{X_R}), \mathsf{SR}(\overrightarrow{Y_R}))$
  $\mathbf{B} = \begin{bmatrix} \mathbf{B}_{LL} & \mathbf{B}_{LR} \\ \mathbf{B}_{RL} & \mathbf{B}_{RR} \end{bmatrix}$
return $\mathbf{B}$

In algorithm 10, input vectors, before each new call to Mat-Mult, are refreshed. In [13], the call corresponding to the $\mathbf{B}_{LL}$ is done without refreshing (its input vectors). The tools that we provide in this section can be used to investigate the role of each refreshing in the overall RPM security of the Mat-Mult algorithm.

The SR gadget that is used in [13] is the SR-SNI. In section 3, it was demonstrated that SR-SNI leaks more information than its lightweight counterpart SR-Simple. We use SR-Simple as the SR gadget in Mat-Mult. This choice in comparison with SR-SNI results in much fewer parity relations, which substantially decreases the success probability of the MAP adversary for a given couple of (n, ϵ).

5.2.2 Compression sub-algorithm

The output of Mat-Mult is $\mathbf{B}_{n \times n}$, which is an n^2-sharing for the $Z = XY$. The Compression algorithm squeezes this n^2-sharing into an n-sharing vector for the secret Z.

The pseudocode presented in algorithm 11 is the Compression given in [13]. The security evaluation framework developed in this section can be used to study the effect of different (linear) candidate structures for the Compression gadget.

Algorithm 11 Compression

Input The n^2-sharing for $Z = XY$ saved in the matrix $\mathbf{B}_{n\times n}$
Output An n-sharing $\overrightarrow{Z}$

for $i = 1$ **to** n **do**
  for $j = i + 1$ **to** n **do**
    $r_{i,j} \leftarrow^{\$} \mathbb{F}_q$
    $r_{j,i} = (r_{i,j} \oplus \mathbf{B}_{i,j}) \oplus \mathbf{B}_{j,i}$
for $i = 1$ **to** n **do**
  $Z_i = \mathbf{B}_{i,i}$
  for $j = 1$ **to** n **,** $j \neq i$ **do**
    $Z_i = Z_i \oplus r_{i,j}$
return $\overrightarrow{Z} = (Z_1, Z_2, \ldots, Z_n)$

The n^2-element input and n-element output vectors for the Compression algorithm are linearly independent of each other (see section 4.5). This can be directly verified by the method discussed in part 3.3. Furthermore, both the input vector and the output vector are sharing's for the same secret Z. These similarities with SR gadgets allow us to use results developed in the last section for assessing the RPM security of the combined Mat-Mult $\rightarrow$ Compression construction.

5.3 Bounds for the security of SAND-Rec

We first study the RPM security of the Mat-Mult sub-algorithm alone, neglecting the Compression's presence in its output. For this non-linear construction, an upper and a lower bound for the success probability of the MAP adversary is derived. These bounds are denoted by $SRP^{+}_{\text{Mat-Mult}}(n, \epsilon)$ and $SRP^{-}_{\text{Mat-Mult}}(n, \epsilon)$.

Then, utilizing the similarities of Compression with SR gadgets, we derive an EEC for the combination of Mat-Mult and Compression. However, in contrast to the EEC for linear algorithms, the obtained EEC for SAND-Rec is valid only for a limited portion of ϵ values.

5.3.1 Proof idea

The MAP adversary -observing a set of leaked variables- aims to find the most probable guess for the secrets. The secret can be each of the X, Y, and Z in the SAND gadget.

Our main observation is that the default parity relations describing the Mat-Mult are in a specific format. These parity relations are non-linear; however, we can replace them with linear systems by paying a price. The price is that instead of $SRP_{\text{Mat-Mult}}$, we approximate an upper and a lower value for it.

Technique for deriving $SRP^{+}_{\text{Mat-Mult}}(n, \epsilon)$. Based on the set of variables leaked to the adversary, entity $\mathcal{H}$ (defined in part 4.4) will reveal some extra information to the adversary to omit non-linear terms. With only a linear parity system left, estimating the SRP is straightforward. The result will be an upper bound for the $SRP_{\text{Mat-Mult}}(n, \epsilon)$.

Technique for deriving $SRP^{-}_{\text{Mat-Mult}}(n, \epsilon)$. By assuming that a certain subset of $\mathbf{\Sigma}_{\text{Mat-Mult}}(n)$ will not leak to the adversary, the effective parity relations for describing Mat-Mult will again become a linear system. The result obtained with this linear system will be a lower bound for $SRP_{\text{Mat-Mult}}(n, \epsilon)$.

For $\epsilon < .15$, estimations for $SRP^{+}_{\text{Mat-Mult}}(n, \epsilon)$ and $SRP^{-}_{\text{Mat-Mult}}(n, \epsilon)$ exponentially (in the order n) tend to zero. In these leakage rates, we will approximate $SRP_{\text{Mat-Mult}}(n, \epsilon)$, and based on this approximation, an EEC for the Mat-Mult will be given.

5.3.2 Computing $SRP^{+}_{\text{Mat-Mult}}(n, \epsilon)$

The pseudocode for Mat-Mult (as any other gadget) describes the relations of the variables in $\mathbf{\Sigma}_{\text{Mat-Mult}}(n)$. We call this direct formulation of the parity equations as the *standard-description* and denote it by $\mathfrak{P}_{\mathsf{S}}(n)$.

Variables in the standard-description can be partitioned into two disjoint sets. One set contains only the output variables stored in matrix $\mathbf{B}$, and the other set is composed of all other variables.

For a variable $U \in \mathbf{B}$, there exist a couple of variables in $\mathbf{\Sigma}_{\mathsf{Mat\text{-}Mult}}(n)$ such as X_1 and Y_1, that $U - X_1Y_1 = 0$. In the standard-description, U, except for $U - X_1Y_1 = 0$, does not appear in any other relation. Intuitively, for X_1 and Y_1, the multiplication X_1Y_1 is the last step of computation, and no further processing on the X_1Y_1 is done.

The main observation regarding the standard-description of Mat-Mult is stated in Lemma 9.

Lemma 9. *If $U \in \mathbf{B}$ is not leaked to the adversary, then the relation $U - X_1Y_1 = 0$ can be removed from the standard-description without affecting the success probability of the MAP adversary.*

Proof. See Appendix H. □

However, if U leaks to the adversary, then she learns X_1Y_1. In this case, $\mathcal{H}$ will reveal both of the X_1 and Y_1 to the adversary. This extra information will definitely increase the success probability of the MAP adversary. By substituting X_1 and Y_1 (and U), the relation $U - X_1Y_1 = 0$ will be trivial with no unknowns.

Parity equations like $U - X_1Y_1 = 0$ are the only non-linear relations in $\mathfrak{P}_{\mathsf{S}}(n)$. For a given set of leakage variables, thanks to Lemma 9, and the extra leakage given by $\mathcal{H}$, the non-linear relations either can be removed from $\mathfrak{P}^{\dagger}_{\mathsf{S}}(n)$ or are trivial. Therefore, $\mathfrak{P}^{\dagger}_{\mathsf{S}}(n)$ becomes a linear system for which we can calculate SRP. Here, $\mathfrak{P}^{\dagger}_{\mathsf{S}}(n)$ is a system of equations obtained from $\mathfrak{P}_{\mathsf{S}}(n)$ after substituting for the leaked variables.

Choosing the appropriate secret. Our discussion in section 3 was confined to the case of a single secret. However, the SAND gadget has three secrets with a non-linear dependency among them. Then the question is: SRP should be based on which of these variables? To solve this challenge, again, the structure of standard-description will help.

Computations on X and Y are carried separately until the ending point multiplications. As a consequence, in $\mathfrak{P}^{\dagger}_{\mathsf{S}}(n)$, after removing the non-linear relations by the described procedure, the parity equations for X and Y are separable into two disjoint subsystems. In a way that, each parity relation resides only in one subsystem. In other words, each equation gives information about only X or Y, not both. Parity equations will provide their best benefit to the adversary if they are aligned to the same target. By choosing $X = Y$ as the secret, any parity equation in $\mathfrak{P}^{\dagger}_{\mathsf{S}}(n)$ will be useful.

5.3.3 Computing $SRP^{-}_{\mathsf{Mat\text{-}Mult}}(n, \epsilon)$

If we assume that members of $\mathbf{B}$ will not leak to the adversary, then based on Lemma 9, we know that $\mathfrak{P}^{\dagger}_{\mathsf{S}}(n)$ will be a linear system. For this linear system, SRP can be directly calculated.

Since we have assumed that some variables will not leak, the SRP results will be a lower bound for the success probability of the MAP adversary. For this case, we also assume that $X = Y$ is the secret of the system.

5.3.4 Estimation results for the RPM security of Mat-Mult

For the upper bound, we replace every $U \in \mathbf{B}$ with its corresponding X_1 and Y_1 variables in the matrix $\mathbf{M}$. So, for each U, two new columns will be added to $\mathbf{M}$. These new variables, with probability ϵ, both will be known to the adversary. Deriving $\mathbf{P}^{n}_{Lin}$ is done with the Extract-Linear algorithm. However, $\mathsf{Sampl}(\mathbf{P}^{n}_{Lin}, \epsilon)$ function should be slightly modified. One random variable $r \xleftarrow{\$} [0, 1]$ should control the leakage event of each pair of the above mentioned X_1 and Y_1 auxiliary variables. The rest of the work is done according to algorithm 5. For the lower bound, variables $U \in \mathbf{B}$ are crossed out from $\mathbf{M}$, and the next steps are done based on algorithm 5. Note that non-linear parity relations are ignored (with no modification in the procedures).

In figure 4, the estimation results for $SRP^{+}_{\mathsf{Mat\text{-}Mult}}(n, \epsilon)$ and $SRP^{-}_{\mathsf{Mat\text{-}Mult}}(n, \epsilon)$ are plotted. For obtaining these results, SR-Simple is used for the realization of the SR functions.

By curve-fitting the results for $n < 30$, the following two approximations, for region $\epsilon < 0.15$, are derived.

$$\begin{aligned} SRP^{+}_{\mathsf{Mat\text{-}Mult}}(n, \epsilon) &\leq \epsilon^{0.3n} \\ SRP^{-}_{\mathsf{Mat\text{-}Mult}}(n, \epsilon) &\approx \epsilon^{0.8n} \end{aligned} \tag{42}$$

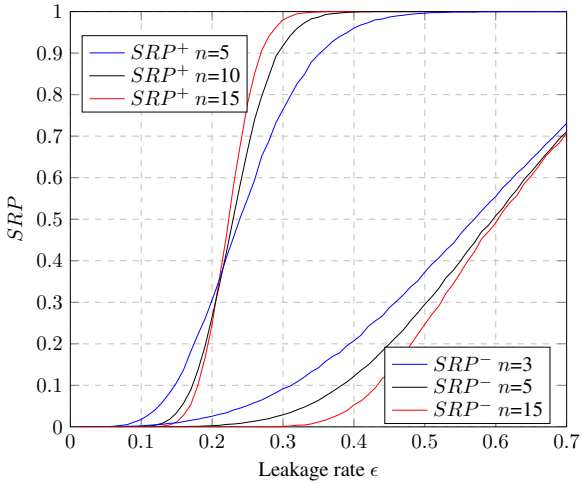


Figure 4: SRP^+ and SRP^- for Mat-Mult at different orders

By the linearity of the final systems of parity equations, we can write

$$SRP^-_{\text{Mat-Mult}}(n,\epsilon) \leq e_{\text{Mat-Mult}}(n,\epsilon) \leq SRP^+_{\text{Mat-Mult}}(n,\epsilon). \tag{43}$$

Considering the plotted upper bound in figure 4, it is clearly seen that for $\epsilon > 0.22$, as the masking order increases, the upper bound for RPM security of the structure also increases. Hence, Mat-Mult may be insecure for $\epsilon > 0.22$. Interestingly, for Mat-Mult with SR-SNI, SRP^- starts being insecure for $\epsilon > 0.3$. So, Mat-Mult with SR-SNI is definitely insecure for $\epsilon > 0.3$.

5.3.5 Adding the leakage of Compression

From algorithm 10 and 11, it is clear that Mat-Mult and Compression are symmetric. So, condition 2 of section 4.5 is satisfied. For Compression, we have verified condition 1, and it is also satisfied.

With almost the same procedure used in section 4, we can find an upper bound on $e_{\text{SAND-Rec}}(n,\epsilon)$. The only difference is that the Compression input vector has n^2 variables, so the computation of ϵ' should be modified.

For Compression, the definition of $f_{\text{Compression}}(n,\epsilon)$ should be modified. In this linear refreshing algorithm, the input is an n^2-sharing, and the output is an n-sharing. After applying Lemmas 5 and 6, a linear system as $\text{L}_{\text{Compression-4}}(\mathbf{B}^\dagger, \overrightarrow{Z}^\dagger)$ will appear. In this system, $\mathbf{B}^\dagger$ are the remaining unknowns of the input $\mathbf{B}$, and $\overrightarrow{Z}^\dagger$ are the remaining unknowns of the $\overrightarrow{Z}$.

Here, we only need to define ϵ' for the connection between Mat-Mult and Compression. For this purpose, based on the present unknowns of $\text{L}_{\text{Compression-4}}$, after sparsing, we define $f_{\text{Compression}}$ as follows.

$$f_{\text{Compression}}(n,\epsilon) = \mathbb{E}[|\mathbf{B}^{\dagger\dagger}|] \tag{44}$$

With this definition, with a procedure similar to Lemma 7, ϵ' for output port of Mat-Mult is obtained as

$$\epsilon' \approx \epsilon + \frac{f_{\text{Compression}}(n,\epsilon)}{n^2}. \tag{45}$$

The final bound on $e_{\text{SAND-Rec}}(n,\epsilon)$ will be

$$e_{\text{SAND-Rec}}(n,\epsilon) \leq e_{\text{Mat-Mult}}(n,\epsilon') + e_{\text{Compression}}(n,\epsilon). \tag{46}$$

In a region of ϵ values, $f_{\mathsf{Compression}}(n,\epsilon)$ and $e_{\mathsf{Compression}}(n,\epsilon)$, for $n < 30$, are approximated by a sufficiently large number of trails. The results are as follows.

$$\begin{aligned} f_{\mathsf{Compression}}(n,\epsilon) &\leq \epsilon n^2 \quad \text{valid for } (n \geq 4, \epsilon \leq .07) \\ e_{\mathsf{Compression}}(n,\epsilon) &\leq \epsilon^{0.6n} \quad \text{valid for } (n \geq 2, \epsilon \leq 0.15) \end{aligned} \tag{47}$$

5.4 EEC for the SAND-Rec

By substituting estimations in (47) and (42), in the $e_{\mathsf{SAND\text{-}Rec}}(n,\epsilon)$ relation given in the (46), for $(n \geq 4, \epsilon \leq .07)$, we will have

$$e_{\mathsf{SAND\text{-}Rec}}(n,\epsilon) \leq (2\epsilon)^{0.3n} + (\epsilon)^{0.6n} \approx (2\epsilon)^{0.3n}. \tag{48}$$

This proves the RPM security of the proposed SAND-Rec structure.

The implementation provided in this paper for Mat-Mult exhibits a much lower arithmetic workload compared to its counterpart given in [13]. This reduction in the complexity gets practically more critical for higher values of n.

The computational complexity of the SAND given in [13] is $O(n^2 \log(n))$, and the complexity of SAND-Rec in this paper is $O(n^2)$, which is the same as SAND-ISW [12].

This section's results put an end to the first counterintuitive corollary made in part 2.4 by giving the first practically-used (non-affine) gadget with proved RPM security at constant (independent of n) ϵ values. The second corollary will also be proved to be incorrect in the rest of this paper.

6 RPM secure masking for X^{-1} S-Box

For X^{-1} S-Box used in the AES, RPM security for a new arbitrary-order masking construction is proved in this section. Based on our notations, the masked counterpart of S-Box is denoted by S-Box$'$. For the proposed S-Box$'$, SAND-Rec is used to realize the SAND gadgets, and SR-SNI is used for the SR gadgets.

In a limited region of ϵ values, an EEC for the new masked construction is derived. EEC models the random leakage consequences as a black-box that either reveals X with probability $e_{\mathsf{S\text{-}Box}'}(n,\epsilon)$ or conveys nothing about X. The results will help characterize the RPM security of a masked AES at arbitrary order n in the next section.

For an $X \in \mathbb{F}_q$, S-Box$'$ accepts an n-sharing $\overrightarrow{X}$ as input and outputs an n-sharing $\overrightarrow{Y}$, where $Y = \mathsf{S\text{-}Box}(X)$. In [18], the authors used the following identity to implement their S-Box$'$.

$$X^{-1} = X^{254} = [(X^2X)(X^2X)^4]^{16}(X^2X)^4X^2 \tag{49}$$

The multiplication X^2X is performed just once. Computing X^{2^i} is an affine operation in $\mathbb{F}_{q=2^8}$. The given identity has only four non-affine multiplications. These multiplications are done with SAND gadgets. S-Box in the AES also has a constant XOR operation in its output, which we have ignored here for simplicity.

Our implementation, given in figure 5, for the shared computation of X^{-1} is different from the one given in [18]. We have used more refreshing gadgets, one between any consecutive SGs with a common secret. The extra SR gadgets enable usage of the results that we have developed so far.

By directly applying the upper bound given in section 4, we can reduce the RPM security of the composition in figure 5 to the RPM security of its building gadgets. Furthermore, since we have already developed EEC for the individual gadgets, by Lemma 8, we can derive an EEC for the entire S-Box$'$.

Choosing the appropriate secret. Secrets of the SGs in figure 5 are not the same; they are different powers of the input X. To provide an upper bound, we assume that by the knowledge of X^j, the MAP adversary will determine X uniquely. Mathematically, this is true only if $\gcd(j, 255) = 1$. Some power of X appearing in the construction fail to satisfy this condition. To solve this issue, we can assume that $\mathcal{H}$ (defined in part 4.4) helps the adversary and, by receiving (j, X^j), gives X to the adversary. Obtaining help from $\mathcal{H}$ will increase the success probability of the MAP adversary.

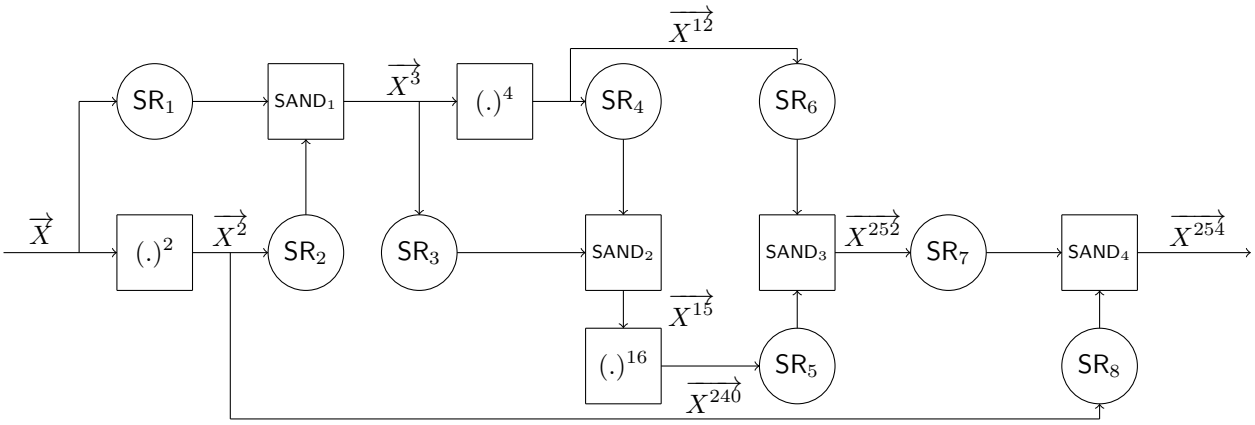


Figure 5: Proposed masked implementation for the X^{-1} S-Box

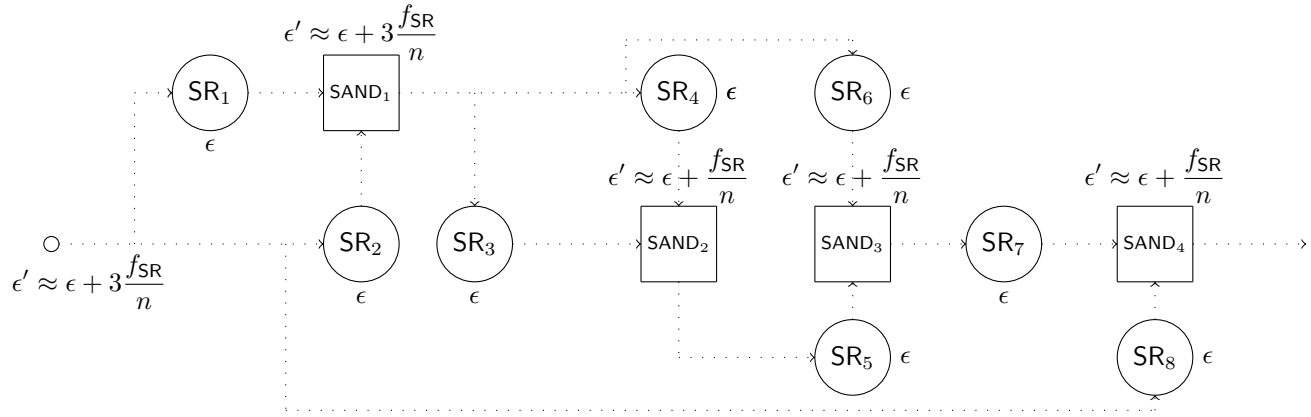


Figure 6: Equivalent leakage rate ϵ' for each gadget in the proposed construction

Including leakage of affine gadgets. Affine operations, such as $(.)^{2^i}$, introduce no extra intermediate variable and are operated on each share separately. Therefore, taking their leakage effect into account is relatively easy. Consider a simple composition as $(.)^{2^i} \rightarrow \mathsf{SR}$ with $\overrightarrow{X^l}$ as input. From Lemma 7, we know that the SR gadget increases ϵ value on the output of $(.)^{2^i}$ to $\epsilon' \approx \epsilon + f_{\mathsf{SR}}(n,\epsilon)/n$. Random leakage on the shares of X^{l2^i} is equivalent to random leakage on the input X^l shares, because we have assumed that adversary with knowledge of (j, X^j) can recover X. So, we can ignore the presence of $(.)^{2^i}$ gadget, and conclude that shares of input X^l are leaking randomly with ϵ'.

In figure 5, input vector X^l is feeding three SR gadgets, so ϵ' at input point will be $\epsilon' \approx \epsilon + 3f_{\mathsf{SR}}(n,\epsilon)/n$. Also, SAND_1 is connected to three SR gadgets in its output. The detail parameters for ϵ' corresponding to each SG is written in its vicinity in figure 6.

To use the previous EEC results, we have confined the leakage rate to $\epsilon < 0.03$ and the masking order to $n \geq 4$.

Finally, by Lemma 8, the EEC for the construction given for X^{-1} evaluation in figure 6 will be as follows.

$$
\begin{aligned}
e_{\mathsf{S\text{-}Box}'}(n,\epsilon) &\leq 8e_{\mathsf{SR\text{-}SNI}}(n,\epsilon) \\
&\quad + 3e_{\mathsf{SAND\text{-}Rec}}(n,\epsilon + \frac{f_{\mathsf{SR\text{-}SNI}}(n,\epsilon)}{n}) + e_{\mathsf{SAND\text{-}Rec}}(n,\epsilon + 3\frac{f_{\mathsf{SR\text{-}SNI}}(n,\epsilon)}{n}) \\
&\quad +_{(a)} [\epsilon + 3\frac{f_{\mathsf{SR\text{-}SNI}}(n,\epsilon)}{n}]^n \\
&\leq 8\epsilon^{0.6n} + 3(\frac{8}{3}\epsilon)^{0.3n} + (4\epsilon)^{0.3n} + (2\epsilon)^n \approx 4(4\epsilon)^{0.3n}
\end{aligned}
\tag{50}
$$

Where we used the approximation $f_{\mathsf{SR\text{-}SNI}}(n,\epsilon) \leq \frac{1}{3}n\epsilon$ that was given in (36). The reason for the additive term specified by (a) is explained in part 4.7. From (50), by increasing the order n, $e_{\mathsf{S\text{-}Box}'}$ will decrease, which proves the given structure's RPM security.

6.1 Relation of the RPM security with the number of used gadgets

From relation (50), it seems that the RPM security of a construction is inversely proportional to its complexity. i.e., as the number of utilized SGs in a construction increases, its overall RPM security decreases. This is rather easy to justify in the case of our proposed masked S-Box; in this structure, there are a lot of leaking gadgets, and all these leakages are about the same secret.

One may deduce that, in general, as the number of SGs in a masked implementation grows, its RPM security declines (see part 2.4 and relation (6) with its accompanying explanation). However, in the next section, we will show that this conclusion is not correct for the AES, which is much more complicated than a single S-Box.

As another point, it worth noting that connecting the output of one SAND gadget to multiple SR gadgets substantially decreases the RPM security of the entire structure. In the proposed S-Box′, the output of SAND_1 is effectively connected to three SR gadgets. Consequently, admissible ϵ values are limited to a narrower region compared to a stand-alone SAND-Rec gadget. It would be an interesting topic to find a remedy for this situation. Maybe designing a new refresh gadget that can provide multiple output vectors alleviates this issue.

7 RPM secure masking of AES

As our last contribution, we study the RPM security of a masked implementation for the AES at arbitrary order n. In the AES, S-Box is the only non-linear function, and all the other operations are affine in $\mathbb{F}_{q=2^8}$. Details of Boolean masking of AES can be found in [18]. Here, we analyze the RPM security of AES′, using the S-Box′ implementation given in the last section.

Effect of complexity on the RPM security. As was explained in part 2.4, previous bounds suggest that RPM security is inversely proportional to the number of participating SGs. In a complex structure like AES′, there are many gadgets. In RPM, each gadget leaks information about its secrets. The proposed S-Box′ is made out of only 4 SAND gadgets. However, AES′ is roughly composed of 600 SAND gadgets, which shows an about 150 fold increase in the used SANDs. SR gadgets are also much more compared to a single S-Box′. In this regard, the main question is: to what extent are these leaking gadgets reinforcing each other. To be more specific, is it possible that they sum up and reveal the secrets of AES, even at moderately small ϵ values?

In [11, 42], the authors have used the BP algorithm to combine the leakages throughout the entire masked implementation. In contrast to expectations, their results demonstrate no sign of massive advantage over targeting a single gadget. Since the BP algorithm is suboptimal, its failure at combining leakage of different gadgets does not mean that such a combination cannot occur. Our results in this section will give a clear explanation for this contradiction.

In the remainder of this section, we first briefly review the structure of AES′ with a focus on the initial rounds. Then, explain our method for approximating the RPM security of the presented AES′. In the end, the success probability of the MAP adversary targeting the first sub-key of the first round is approximated.

7.1 Structure of the AES′

AES is a symmetric-key block cipher that operates on 128-bit blocks of plain-text inputs and has several variants for the input key-length. The one with a 128-bit key is the most used option. In this choice for the key-length, computation of the 128-bit cipher-text is carried in 10 rounds.

Operations are in an 8-bit field. So, the input plain-text is partitioned into 16 concatenated bytes. We use boldface letters with a vector over them, such as $\overrightarrow{\mathbf{P}}$, for denoting the associated 16 concatenated n-sharing vectors.

$$\overrightarrow{\mathbf{P}} = \overrightarrow{P_1}||\overrightarrow{P_2}||\dots||\overrightarrow{P_{16}} \tag{51}$$

Based on the 16-byte input key, different keys are produced for each round of AES. Keys for each round are 16-byte and are called *round-keys*.

In the first round, the plain-text is bit-wise XORed with its round-key. The operations in the 2nd to the 9th rounds are similar. Three functions, S-Box, Shift-Rows, and Mix-Columns, are applied on the 16-byte input of each round. At the

end of each round, the 16-byte round-keys are XORed with the results. The last round is quite different. In that round, the Mix-Columns is absent.

AES is composed of non-linear S-Box calculation and affine Shift-Rows and Mix-Columns functions. For the masked implementation, input to both of the Shift-Rows′ and Mix-Columns′ are 16 n-sharings. Their outputs are also 16 n-sharings. Detail description of these two algorithms can be found in [44].

In algorithm AES′-Rounds$(1:2)$, which shows the procedures in the first two rounds of AES′, the masked functions Shift-Rows′ and Mix-Columns′ are used. For our study here, since these two algorithms are affine, their internal structure makes no difference. So, we do not point to their interior structure, even though their effects are precisely taken into account in the RPM security results.

Algorithm 12 AES′-Rounds$(1:2)$

Input 16 n-sharings of input plain-text, represented by $\overrightarrow{\mathbf{P}} = \overrightarrow{P_1}||\overrightarrow{P_2}||\dots||\overrightarrow{P_{16}}$
Output 16 n-sharings as input to third round, represented by $\overrightarrow{\mathbf{V}} = \overrightarrow{V_1}||\overrightarrow{V_2}||\dots||\overrightarrow{V_{16}}$

1: **for** $i = 1$ **to** 16 **do**
2: $\quad\overrightarrow{K_i^1} \leftarrow \mathsf{SR}(\overrightarrow{K_i^1})$ ▷ Round-keys are refreshed before each new run
3: $\quad\overrightarrow{T_i} = \overrightarrow{P_i} \oplus \overrightarrow{K_i^1}$
4: $\quad\overrightarrow{R_i} \leftarrow \mathsf{SR}(\overrightarrow{T_i})$ ▷ End of round 1
5: $\quad\overrightarrow{S_i} = \mathsf{S\text{-}Box}'(\overrightarrow{R_i})$
6: $\quad\overrightarrow{S_i} \leftarrow \mathsf{SR}(\overrightarrow{S_i})$
7: $\overrightarrow{\mathbf{U}} = \mathsf{Shift\text{-}Rows}'(\overrightarrow{\mathbf{S}})$
8: $\overrightarrow{\mathbf{U}} = \mathsf{Mix\text{-}Columns}'(\overrightarrow{\mathbf{U}})$
9: **for** $i = 1$ **to** 16 **do**
10: $\quad\overrightarrow{K_i^2} \leftarrow \mathsf{SR}(\overrightarrow{K_i^2})$
11: $\quad\overrightarrow{V_i} = \overrightarrow{U_i} \oplus \overrightarrow{K_i^2}$ ▷ End of round 2
12: **return** $\overrightarrow{\mathbf{V}} = \overrightarrow{V_1}||\overrightarrow{V_2}||\dots||\overrightarrow{V_{16}}$

In algorithm AES′-Rounds$(1:2)$, round-keys for different rounds are specified by an upper script. Since the round-keys are fixed, they can be computed once and saved in a memory for future uses, instead of recalculating them in each encryption.

The $\oplus$ operation over two same-length vectors is defined as XOR of their respective elements. SR gadget in algorithm AES′-Rounds$(1:2)$ is SR-SNI. For better readability, some variables in the algorithm are valued multiple times. This makes no issue in RPM security evaluation, in matrix $\mathbf{M}$, we will give separate names (and so, separate columns) for each usage of a variable. Furthermore, We assume that the MAP adversary is aware of the used plain-texts; this means that she knows the 16 secrets of the $\overrightarrow{\mathbf{P}}$.

7.2 Approximating the RPM security of the AES′

Each variable of AES can be considered as a target for the MAP adversary. However, since the input plain-text is assumed known, the sub-keys of the first rounds, i.e., K_i^1s, for $i = 1$ to 16, are mostly targeted. Here we study RPM security with K_1^1 as the secret.

P_1 (the secret of the n-sharing $\overrightarrow{P_1}$) is known to the adversary. So, if the adversary learns K_1^1, she learns S_1 (the secret of the n-sharing $\overrightarrow{S_1}$). Also, by using the relation $\mathsf{S\text{-}Box}^{-1}(.)$, knowledge of S_1 leads to K_1^1. Therefore, we can write

$$SRP_{K_1^1}(n,\epsilon) = SRP_{S_1}(n,\epsilon). \tag{52}$$

The only non-linear operation in the first two rounds of the AES′ is the 16 S-Box′ calculations of the second round. In the last section, we derived and EEC for the S-Box′. This EEC can help us eliminate the non-linear parities, and consequently, calculate SRP_{S_1} directly with the methods presented for the linear algorithms.

Lemma 10. *In the RPM, for algorithm* $\mathsf{AES'\text{-}Rounds}(1:2)$*, in* $i = 1$ *to* 16*, at region* $(n \geq 4, \epsilon < 0.02)$*, leakage from the lines*

$$\begin{aligned} \overrightarrow{K_i^1} &\leftarrow \mathsf{SR}(\overrightarrow{K_i^1}) \\ \overrightarrow{T_i} &= \overrightarrow{P_i} \oplus \overrightarrow{K_i^1} \\ \overrightarrow{R_i} &\leftarrow \mathsf{SR}(\overrightarrow{T_i}) \\ \overrightarrow{S_i} &= \mathsf{S\text{-}Box'}(\overrightarrow{R_i}) \end{aligned} \tag{53}$$

can be replaced with $\Phi(S_i)$*, where the function* $\Phi(.)$ *is an erasure channel with parameter* $\phi(n,\epsilon) = e_{\mathsf{S\text{-}Box'}}(n,\epsilon') + 2e_{\mathsf{SR}}(n,\epsilon) + 2(\epsilon')^n$*, where* $\epsilon' = \epsilon + f_{\mathsf{SR}}(n,\epsilon)/n$*.*

Proof. See Appendix I. □

With the direct application of Lemma 10, we can approximate $SRP_{s_1}(n,\epsilon)$ using algorithm Approx $e_{\mathcal{A}'}(n,\epsilon)$ with some considerations.

The function Create-M should be modified to include equivalent leakages from the secret S_is. For $i = 1$ to 16, S_is are placed in the first 16 columns of $\mathbf{M}$. Rest of the variables in algorithm $\mathsf{AES'\text{-}Rounds}(1:2)$, corresponding to the lines 6 to the end, will be the next columns of $\mathbf{M}$.

Members of $\overrightarrow{V_i}$ for $i = 1$ to 16 are also in $\mathbf{M}$. However, we cannot assume that their shares are just leaking with probability ϵ. $\overrightarrow{V_i}$s are the only variables that bridge the second round with the next rounds. So, they should, in some way, reflect the leakage consequences of the next rounds. To provide an answer, we consider two marginal cases.

- As one extreme, we consider that leakages in rounds 3 to end are very high, such that shares of $\overrightarrow{V_i}$ are completely extractable for the adversary. In this case, we show the success probability of the MAP adversary by $SRP^{+}_{S_1}(n,\epsilon)$.
- As the other extreme, we assume that leakages in rounds 3 to the end have a minimal effect on the recovery of the shares of $\overrightarrow{V_i}$s, and they are only leaking with probability ϵ. In this case, the success probability of the MAP adversary is denoted by $SRP^{-}_{S_1}(n,\epsilon)$.

Deriving the corresponding $\mathbf{P}^n_{Lin}$ is done with Extract-Linear algorithm. However, $\mathbf{P}^{\dagger n}_{Lin} = \mathsf{Sampl}(\mathbf{P}^n_{Lin},\epsilon)$ function should apply different probabilities on its columns. The columns 1 to 16 corresponding to S_1 to S_{16} will leak with probability $e_{\mathsf{S\text{-}Box'}}(n,\epsilon')$. Note that S_1 is our target, that we also assume leaks with probability $e_{\mathsf{S\text{-}Box'}}(n,\epsilon')$. Share of $\overrightarrow{V_i}$s, for computing $SRP^{+}_{S_1}(n,\epsilon)$, should leak with probability one. The remaining columns will leak independently with probability ϵ.

If S_1 is leaked to the adversary, then there is no need to solve $\mathbf{P}^{\dagger n}_{Lin}$. However, if S_1 still to be unknown, $\mathbf{G} = \mathsf{Gaussian\text{-}Elim}\ (\mathbf{P}^{\dagger n}_{Lin})$ should be invoked. The rest of Approx $e_{\mathcal{A}'}(n,\epsilon)$ (algorithm 5) is routinely computed.

7.3 Approximation results

Secret S_1 is known directly with probability $\phi(n,\epsilon)$ (given in Lemma 10), and also may be recovered with the system of equations described by $\mathbf{G}$. Simulation results demonstrate that the probability $\phi(n,\epsilon)$ is much higher than the probability of recovering secret S_1 from $\mathbf{G}$. For this reason, we estimate the latter probability separately, and at the end, use union bound to combine the two probabilities.

For $\epsilon < 0.02$, the difference between $SRP^{-}_{S_1}(n,\epsilon)$ and $SRP^{+}_{S_1}(n,\epsilon)$ is very faint and practically negligible. Therefore, we assume that $SRP^{-}_{S_1}(n,\epsilon) \approx SRP^{+}_{S_1}(n,\epsilon) \approx SRP_{S_1}(n,\epsilon)$. With curve-fitting the obtained results, we put forward the following estimations.

$$SRP_{S_1}(n,\epsilon) \leq \phi(n,\epsilon) + 3(\epsilon)^{n-1} \approx e_{\mathsf{S\text{-}Box'}}(n,\frac{4}{3}\epsilon) \approx 4(5.3\epsilon)^{0.3n} \tag{54}$$

The derived bound for $SRP_{S_1}(n,\epsilon)$ is exponentially decaying, and this proves the RPM security of the proposed masking for the AES.

Practical equivalence of $SRP^{+}_{S_1}(n,\epsilon)$ and $SRP^{-}_{S_1}(n,\epsilon)$ means that increasing the rounds does not appreciably affect the RPM security. By inspecting the relations in $\mathbf{P}^{n}_{Lin}$, it becomes clear that the main reason for why leakages in different rounds are not amplifying each other hides in the confusion that round-keys are introducing. They act as a barrier and avoid the spread of leakage results between the rounds.

8 Conclusion and future works

Studying the RPM security of masked implementations by utilizing their defining parity equations is a new idea founded in this paper. With this approach, we could provide RPM secure gadgets. Moreover, with appropriately modeling leakage of SR gadgets, we could prove the RPM security of a combination of gadgets as in a masked S-Box and a masked AES.

The techniques developed in this paper can be used to evaluate the RPM security of more structures than those worked on here. Furthermore, our results demonstrated that most of the constructions are insecure for high values of leakage rates such as $\epsilon > 0.2$. Designing alternative gadgets (or proof techniques) that are able to withstand more leakages is an exciting challenge for future works.

References

[1] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[2] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[3] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[4] Yael Tauman Kalai and Leonid Reyzin. *A Survey of Leakage-Resilient Cryptography*, pages 727–794. Association for Computing Machinery, New York, NY, USA, 2019.

[5] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[6] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked aes hardware implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 157–171, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[7] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[8] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[9] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 423–440, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[10] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. Noisy leakage revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 159–188, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[11] Qian Guo, Vincent Grosso, and François-Xavier Standaert. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Cryptol. ePrint Arch.*, 2018:498, 2018.

[12] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 463–481, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[13] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the isw masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 23–39, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[14] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 116–129, 2016.

[15] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[16] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612613, November 1979.

[17] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 398–412, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[18] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 413–427, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[19] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michael Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In Anne Canteaut, editor, *Fast Software Encryption*, pages 366–384, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] Arnab Roy and Srinivas Vivek. Analysis and improvement of the generic higher-order masking scheme of fse 2012. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, pages 417–434, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[21] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 170–187, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[22] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 135–156, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[23] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Private multiplication over finite fields. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 397–426, Cham, 2017. Springer International Publishing.

[24] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 535–566, Cham, 2017. Springer International Publishing.

[25] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log n)$ leakage rate. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*, page 586615, Berlin, Heidelberg, 2016. Springer-Verlag.

[26] Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 441–458, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[27] François-Xavier Standaert, François Koeune, and Werner Schindler. How to compare profiled side-channel attacks? In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security*, pages 485–498, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[28] 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.

[29] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is more. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 22–41, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[30] Carolyn Whitnall and Elisabeth Oswald. A fair evaluation framework for comparing side-channel distinguishers. *Journal of Cryptographic Engineering*, 1(2):145, Aug 2011.

[31] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 55–74, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[32] Stefan Dziembowski, Sebastian Faust, and Maciej Skórski. Optimal amplification of noisy leakages. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography*, pages 291–318, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[33] Stefan Dziembowski, Sebastian Faust, and Karol Żebrowski. Simple refreshing in the noisy leakage model. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 315–344, Cham, 2019. Springer International Publishing.

[34] Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Abdul Rahman Taleb. Random Probing Security: Verification, Composition, Expansion and New Constructions. In *CRYPTO 2020 - 40th Annual International Cryptology Conference*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368, Santa Barbara, CA / Virtual, United States, August 2020. Springer.

[35] S. Banerjee and A. Roy. *Linear Algebra and Matrix Analysis for Statistics*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2014.

[36] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, page 133136. AAAI Press, 1982.

[37] M. Mazhdrakov, D. Benov, and N. Valkanov. *The Monte Carlo Method: Engineering Applications*. Acmo Academic Press, 2018.

[38] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 401–429, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[39] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 535–566, Cham, 2017. Springer International Publishing.

[40] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Improved parallel mask refreshing algorithms: generic solutions with parametrized non-interference and automated optimizations. *Journal of Cryptographic Engineering*, 10(1):17–26, Apr 2020.

[41] Lee-Ad Gottlieb and Tyler Neylon. Matrix sparsification and the sparse null space problem. *Algorithmica*, 76(2):426–444, Oct 2016.

[42] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 282–296, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[43] J. M. Mooij and H. J. Kappen. Sufficient conditions for convergence of the sumproduct algorithm. *IEEE Transactions on Information Theory*, 53(12):4422–4437, 2007.

[44] Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, Lawrence James Foti, E. Bassham, E. Roback, and James F. Dray Jr. *Advanced Encryption Standard*. (NIST FIPS) - 197. Federal Inf. Process. Stds., 2001.

Appendices

A Proof of Lemma 1

Assume V is a k-bit uniformly distributed random variable, and $\mathcal{L}$ is a random instance of leakage. For simplicity of notations, we define a new random variable V' as $V' = (V|\mathcal{L})$. Note that, here $\mathcal{L}$ is a realization of leakage. So the defining conditional probability of V' is unambiguous. For $\mathsf{SD}(V;V')$ we can write

$$\begin{aligned}\mathsf{SD}(V;V') &= \frac{1}{2}\sum_{v\in\mathcal{V}} |\Pr(V=v) - \Pr(V'=v)| = \frac{1}{2}\sum_{v\in\mathcal{V}} |\frac{1}{2^k} - \Pr(V'=v)| \\ &=_{(a)} \sum_{v\in\mathcal{V},\Pr(V'=v)>\frac{1}{2^k}} [\Pr(V'=v) - \frac{1}{2^k}] \leq 2^k \max[\Pr(V'=v) - \frac{1}{2^k}] = 2^k SRP_V(n,\epsilon),\end{aligned} \tag{55}$$

where (a) can be derived by a simple partitioning of the given summation. By the above relation and noting that

$$\max[\Pr(V'=v) - \frac{1}{2^k}] \leq \sum_{v\in\mathcal{V},\Pr(V'=v)>\frac{1}{2^k}} [\Pr(V'=v) - \frac{1}{2^k}], \tag{56}$$

we conclude that

$$SRP_V(n,\epsilon) \leq \mathsf{SD}(V;V') \leq 2^k SRP_V(n,\epsilon). \tag{57}$$

For function $f : \mathcal{V} \to \mathcal{Y}$, we show that $\mathsf{SD}(f(V);f(V')) \leq \mathsf{SD}(V;V')$.

$$\begin{aligned}\mathsf{SD}(f(V);f(V')) &= \frac{1}{2}\sum_{y\in\mathcal{Y}} |\Pr(f(V)=y) - \Pr(f(V')=y)| \\ &= \frac{1}{2}\sum_{y\in\mathcal{Y}} |\Pr(V\in f^{-1}(y)) - \Pr(V'\in f^{-1}(y))| \\ &= \frac{1}{2}\sum_{y\in\mathcal{Y}} |\sum_{t\in f^{-1}(y)} [\Pr(V=t) - \Pr(V'=t)]| \\ &\leq \frac{1}{2}\sum_{y\in\mathcal{Y}} [\sum_{t\in f^{-1}(y)} |\Pr(V=t) - \Pr(V'=t)|] \\ &= \mathsf{SD}(V;V').\end{aligned} \tag{58}$$

Now we have the following chain of inequalities

$$SRP_{f(V)}(n,\epsilon) \leq \mathsf{SD}(f(V);f(V')) \leq \mathsf{SD}(V;V') \leq 2^k SRP_V(n,\epsilon). \tag{59}$$

Since k is fixed, if $SRP_V(n,\epsilon)$ moves toward zero by increasing n, so will do $SRP_{f(V)}(n,\epsilon)$.

B Proof of Lemma 2

In $\mathbb{F}_q$ with a $D(n) \times (T(n)+1)$ matrix $\mathbf{P}^n_{Lin}$ of $\mathsf{rank}(\mathbf{P}^n_{Lin}) = D(n)$, the equation system defined by

$$\mathbf{P}^n_{Lin} \times [V, \mathbf{\Sigma}(n)]^\top = \mathbf{0}_{D(n)\times 1} \tag{60}$$

has $T(n)+1-D(n)$ free, and $D(n)$ pivot variables. Therefore, the number of different solutions for (60) in $(\mathbb{F}_q)^{T(n)+1}$ are $q^{T(n)+1-D(n)}$. We call the collection of these solutions set $\mathcal{S}_1$.

On the other hand, in $\mathcal{A}'(n)$, The $R(n)$ randomness and input variables are free to take any value in $\mathbb{F}_q$. So, the vector $[V, \mathbf{\Sigma}(n)]$ should have at least $q^{R(n)}$ realizations. Let the collection of these realizations be set $\mathcal{S}_2$.

By the assumption in the lemma, we have $D(n) = T(n) + 1 - R(n)$. Therefore, for the cardinality of the two sets, we can write

$$|\mathcal{S}_2| \leq |\mathcal{S}_2|. \tag{61}$$

By definition of $\mathbf{P}^n_{Lin}$, any $T(n)+1$ element vector $[V, \mathbf{\Sigma}(n)]$ in $\mathcal{S}_2$ is in $\mathcal{S}_1$. Thus, we conclude that

$$\mathcal{S}_1 = \mathcal{S}_2. \tag{62}$$

C Proof of Lemma 3

Matrix $\mathbf{M}$ is formed by $k(n)$ times executing $\mathcal{A}'(n)$ and putting the realized values of the vector $[V, \mathbf{\Sigma}(n)]$, in a fixed order at the consecutive rows of $\mathbf{M}$.

We show that if in $[V, \mathbf{\Sigma}(n)]$, the jth element is dependent on the previous $j-1$ entries, then column j of $\mathbf{E}$ will be a free variable. For instance, there will be a linear combination of initial $j-1$ columns of $\mathbf{M}$ that produces the jth column. To prove this, we do the following reasoning.

First, note that in $\mathbf{E} = \mathsf{Gaussian\text{-}Elim}(\mathbf{M})$, only linear operations are done over the rows of $\mathbf{M}$. So, any linear relation on its columns should be preserved and reflected equivalently in the columns of $\mathbf{E}$. For contradiction, assume that j is not a free variable in $\mathbf{E}$. This means that $\mathbf{E}(I, j) = 1$ for some I, and $\mathbf{E}(i, j) = 0$ for any $i < I$. In this case, it is easy to see that no combination of columns 1 to $I-1$ of $\mathbf{E}$ (or of $\mathbf{M}$) can create the jth column of $\mathbf{E}$ (or $\mathbf{M}$), and this inconsistency completes our proof.

We proved that a dependent column in $\mathbf{M}$ always will be a free variable in $\mathbf{E}$. In other words, dependent relations will be correctly identified by the given algorithms.

On the other hand, if the jth column of $\mathbf{M}$ is independent, it is possible to be erroneously declared as a free variable in $\mathbf{E}$. Let the $(T(n)+1)$-length coefficients vector U describes the dependency relation obtained for this falsely-identified free variable. It is required that U be perpendicular to all of the rows of $\mathbf{M}$. Since a free variable in column j means that this column is dependent only on the columns 1 to $j-1$, $U(j)$ should be 1 and $U(j : T(n)+1)$ should be 0.

Mathematically, the event of incorrectly declaring column j as a free variable is the same as that a fixed set of coefficients in $U(1 : j-1)$ be orthogonal to each row of sub-matrix $\mathbf{M}(1 : j, pivot\_row : k(n))$. It can be shown by the independence of the rows, and the uniformity of the randomness variables, that the probability of this event is $q^{k(n)-pivot\_row}$. This probability decrease as the variable $pivot\_row$ increases. The maximum possible value for the $pivot\_row$ is $D(n)$. So, the probability of introducing a wrong relationship in a column is at most $q^{k(n)-D(n)}$.

For matrix $\mathbf{M}$, initially, we are not aware of $D(n)$. However, from algebra, we know that $D(n) \leq T(n)+1$. So, in the absence of extra knowledge, we conclude that a false relation is possible only with probability $q^{k(n)-T(n)+1}$. By substituting $k(n) = T(n) + m$, the final probability will be q^{m-1}.

It is recommended for a safe margin to choose $k(n) = 2T(n)$. In this way, we are pretty sure that no column will give an invalid relation.

D Proof of Lemma 4

From linear algebra, we know when a non-homogeneous linear system given below has a solution and what is the general structure of its solutions.

$$\mathbf{P}^{\dagger n}_{Lin} \times [V, \mathbf{\Sigma}^{\dagger}(n)]^{\top} = \mathbf{b} \tag{63}$$

Since the adversary's probing is noiseless, this system has at least one solution. Because it is based on a physical experiment, unknown variables of (63) were valued in that experiment, and these values satisfy (63).

The complete collection of unique vectors $[V, \mathbf{\Sigma}^{\dagger}(n)]$ that satisfies (63) are representable as follows.

$$[V, \mathbf{\Sigma}^{\dagger}(n)] = [V, \mathbf{\Sigma}^{\dagger}(n)]_p \oplus [V, \mathbf{\Sigma}^{\dagger}(n)]_f \tag{64}$$

Where $[V, \mathbf{\Sigma}^{\dagger}(n)]_p$ is called the *particular solution* of (63). A particular solution is any solution of (63), and $[V, \mathbf{\Sigma}^{\dagger}(n)]_f$ can be any vector in the vector space defined by

$$\mathbf{P}^{\dagger n}_{Lin} \times [V, \mathbf{\Sigma}^{\dagger}(n)]^{\top} = \mathbf{0}. \tag{65}$$

The number of unique solutions of (65) directly depends on the number of free variables of $\mathbf{P}^{\dagger n}_{Lin}$. Assume $\mathbf{P}^{\dagger n}_{Lin}$ has l free variables, and label them as $Z_1, Z_2, ..., Z_l$.

As stated in the text, the coefficients of V are in the first column of $\mathbf{P}^{\dagger n}_{Lin}$. The secret V is linearly dependent on the variables of $\mathbf{\Sigma}(n)$. Therefore, at least one entry of the first column of $\mathbf{P}^{\dagger n}_{Lin}$ is non-zero. Consequently, V is not a free variable.

The final relation for the solution of V (based on (64)) will be

$$V = v_p \oplus Z_1 v_1 \oplus Z_2 v_2 ... \oplus Z_l v_l. \tag{66}$$

Where v_p is the value of V in $[V, \mathbf{\Sigma}^{\dagger}(n)]_p$, and v_i is the value of V in (65) when all the free variables are set to zero, except Z_i, which is set to 1.

In relation (66), Z_1 to Z_l are free to take any value in $\mathbb{F}_q$. So, any value for V (irrespective of v_p) will have the same abundance.

To find a unique answer for V, it is necessary that all the $v_i = 0$. By algebra, we know that in the row reduced echelon form of $\mathbf{P}^{\dagger n}_{Lin}$ as $\mathbf{G} = \mathsf{Gaussian\text{-}Elim}(\mathbf{P}^{\dagger n}_{Lin})$, values of v_is are $v_i = -\mathbf{G}(1, i)$. Therefore, if the condition $\mathbf{G}(1, 2 : T(n) + 1) = 0$ is satisfied , then V is uniquely determined. Any non-zero value in $\mathbf{G}(1, 2 : T(n) + 1)$ with its accompanying free variable Z_i, will hide the particular solution v_p.

Note that, behaviors described above do not depend on the right-hand side constant vector $\mathbf{b}$. In other words, as long as, $\mathbf{b}$ is based on a real experiment, the number of solutions for V is independent of the realized values in $\mathbf{b}$.

E Proof of Lemma 5

Let $\mathbf{P}^n$ be the coefficient matrix for the system of equations defined by $\mathtt{L}^{\dagger}_{\mathsf{SR}(n)}(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}, \overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V) = \mathbf{b}_2$. In this way, we can write

$$\mathbf{P}^n \times [\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}, \overrightarrow{V}^{\dagger}_0, \overrightarrow{V}^{\dagger}_1, V]^{\top} = \mathbf{b}_2. \tag{67}$$

Define *augmented matrix* $\mathbf{P'}^n$ by appending the column vector $\mathbf{b}_2$ to the right of $\mathbf{P}^n$ as $\mathbf{P'}^n = [\mathbf{P}^n | \mathbf{b}_2]$. Next, obtain the row reduced echelon form of $\mathbf{P'}^n$ as $\mathbf{G} = \mathsf{Gaussian\text{-}Elim}(\mathbf{P'}^n)$. From $\mathbf{G}$, we can derive the $\mathtt{L}_1$, $\mathtt{L}_2$, and $\mathtt{L}_3$ as defined in the lemma.

The system defined in (67) has at least one solution. Because it is based on a physical experiment and the noise-less information (ϵ random probing leakage) that the adversary has received.

By the structure of $\mathbf{G}$, we know that if row i of $\mathbf{G}$ is all-zero, then all the rows beneath the ith row should be all-zero. Assume I is the first row of $\mathbf{G}$ that is all-zero. In any row $i < I$ of $\mathbf{G}$, there exist a column J such that $\mathbf{G}(i, J) = 1$, and for every $j < J$, $\mathbf{G}(i, j) = 0$. Column J is called a *pivot* variable. Let function $\mathsf{A}(.)$, for each row $i < I$, give its corresponding column J. It can be seen that $\mathsf{A}(.)$ is a monotonically increasing function.

In $\mathbf{P}^n$, define the last column containing elements of $\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$ with $k = |\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}|$.

We can partition $\mathbf{G}$'s columns into three parts as $\mathbf{G} = [\mathbf{G}_1 | \mathbf{G}_2 | \mathbf{G}_3]$. Where, $\mathbf{G}_1 = \mathbf{G}(:, 1 : k)$, $\mathbf{G}_2 = \mathbf{G}(:, k + 1 : end - 1)$, and $\mathbf{G}_3 = \mathbf{G}(:, end)$. $\mathbf{G}_1$ is over $\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$, and $\mathbf{G}_3$ is the last column which was initially filled by $\mathbf{b}_2$.

Parity equations are rows of $\mathbf{G}$. Next, We partition rows of $\mathbf{G}$ into the following disjoint sets.

- Rows $i = 1$ to $R = \min[I-1, \mathsf{A}^{-1}(k)]$. In these rows, the corresponding parity equations will have a pivot variable from set $\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$. Because, at least we have $\mathbf{G}(i, \mathsf{A}(i)) = 1$. Since $\mathsf{A}(.)$ is monotonically increasing, $\mathsf{A}(i) \leq \mathsf{A}(R) = \min[\mathsf{A}(I-1), k] \leq k$. For these parity equations we can write

$$\mathbf{G}_1(i,:) \times [\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}]^\top \oplus \mathbf{G}_2(i,:) \times [\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V]^\top = \mathbf{G}_3(i). \tag{68}$$

  Based on (68), we label the equations defined by $\mathbf{G}_1(1:R,:)$ as $\mathtt{L}_1$, the equations defined by $\mathbf{G}_2(1:R,:)$ as $\mathtt{L}_2$, and the constants at the column vector $\mathbf{G}_3(1:R)$ as $\mathbf{b}^1_2$.
  Each row of $\mathbf{G}_1(1:R,:)$ has a pivot variables. So, these rows (that are also shown by $\mathtt{L}_1$) are linearly independent.
- Rows $i = R+1$ to $I-1$. These rows exist only if $\mathsf{A}^{-1}(k) < I-1$. For these rows, we have $\mathsf{A}(i) > k$. Consequently, $\mathbf{G}_1(i,:) = 0$. So, no variable from $\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}$ will exist in the equations defined by these rows. For these equations that are corresponding to the rows $R+1 : I-1$, we can write

$$\mathbf{G}_2(i,:) \times [\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V]^\top = \mathbf{G}_3(i). \tag{69}$$

  Based on (69), we label the equations defined by $\mathbf{G}_2(R+1:I-1,:)$ as $\mathtt{L}_3$, and the constants at the column vector $\mathbf{G}_3(R+1:I-1)$ as $\mathbf{b}^2_2$.

Finally, collection of the parity equations in $\mathbf{G}$ will be of the following structure.

$$\begin{cases} \mathtt{L}_1(\mathbf{\Sigma}^{\dagger I}_{\mathsf{SR}(n)}) & = \mathtt{L}_2(\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) \oplus \mathbf{b}^1_2 \\ \mathtt{L}_3(\overrightarrow{V}^\dagger_0, \overrightarrow{V}^\dagger_1, V) & = \mathbf{b}^2_2 \end{cases} \tag{70}$$

F Proof of Lemma 6

We know that a system of equations in a finite field $\mathbb{F}_q$ has a limited number of unique solutions. Assume, $\mathbf{S}^1$ to $\mathbf{S}^N$ are all the solutions of the system defined by $\mathtt{NL}_1(V, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3) = \mathbf{0}$, where $\mathbf{S}^i$ is a vector of values as $\mathbf{S}^i = (V^i, \mathbf{\Sigma}^i_2, \mathbf{\Sigma}^i_3)$.

If we show that corresponding to each $\mathbf{S}^i$, there a fixed number of unique answers for $\mathbf{\Sigma}_1$ that the system $\mathtt{L}_4(\mathbf{\Sigma}_1) = \mathtt{L}_5(V, \mathbf{\Sigma}_2)$ is satisfied, then the lemma is proved (see relation (12)).

For each $\mathbf{S}^i$, $\mathtt{L}_5(V, \mathbf{\Sigma}_2)$ is a known and fixed vector, and for simplicity, we denote it with $\mathbf{d}^i$. We are seeking the number of solutions of $\mathtt{L}_4(\mathbf{\Sigma}_1) = \mathbf{d}^i$. Because it is assumed that the equations in $\mathtt{L}_4$ are independent of each other, from linear algebra, we know that this system has at least one solution. This solution is labeled $[\mathbf{\Sigma}_1]_p$ and is called particular solution. Structure of answers for $\mathtt{L}_4(\mathbf{\Sigma}_1) = \mathbf{d}^i$ is given in the following.

$$\mathbf{\Sigma}_1 = [\mathbf{\Sigma}_1]_p \oplus [\mathbf{\Sigma}_1]_f \tag{71}$$

Where $[\Sigma_1]_f$ is any vector satisfying the homogeneous system defined by

$$\mathtt{L}_4(\mathbf{\Sigma}_1) = \mathbf{0}. \tag{72}$$

The cardinality of answers in (71) is independent of the value of $\mathbf{d}^i$. So, we conclude that the frequency of answers for V is solely controlled by sub-system defined by $\mathtt{NL}_1(V, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3) = \mathbf{0}$.

G Proof of Lemma 7

The set of equations defined by $\mathtt{NL}_{\mathsf{SG}_1}$ is deterministic, and only their set of known variables are probabilistic. Moreover, the set of known variables in the two systems are distributed identical, except (possibly) for the known elements of $\overrightarrow{V}_0$. Therefore, we just need to show that the distribution of the known elements of input $\overrightarrow{V}_0$ are also identical in the two systems.

In the first system, members of $\overrightarrow{V}_0$ are known to the adversary from two separate sources. One is the usual ϵ probing on the SG_1 side, and the other is $\overrightarrow{V}^{\dagger\dagger}_0$, resulted from the SR side. Since the parities corresponding to the SR gadget

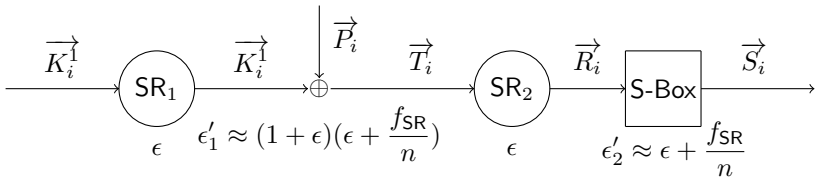


Figure 7: Operations before S-Box evaluations in the first round of AES, with equivalent leakage rate ϵ' for each gadget in the construction

are symmetric on the members of $\overrightarrow{V}_0$ and $\overrightarrow{V}_1$, we are sure that each member of $\overrightarrow{V}_0$ will be in the set $\overrightarrow{V}_0^{\dagger\dagger}$ independently with probability $\mathbb{E}(|\overrightarrow{V}_0^{\dagger\dagger}|)/n = f_{\mathsf{SR}}(n,\epsilon)/n$. Finally, because the two leakage sources are independent, their combination will give the following probability for leakage of each member of $\overrightarrow{V}_0$.

$$\epsilon' = \epsilon + \frac{f_{\mathsf{SR}}(n,\epsilon)}{n} - \epsilon\frac{f_{\mathsf{SR}}(n,\epsilon)}{n} \tag{73}$$

In the second system, members of $\overrightarrow{V}_0$ are directly leaking with probability ϵ'. So, the inputs to the two systems are statistically equivalent. Consequently, they should produce statistically equivalent results.

H Proof of Lemma 9

Consider the following system of equations.

$$\begin{cases} U - X_1Y_1 & = 0 \\ \mathtt{NL}(V, \mathbf{\Sigma}, X_1, Y_1) & = \mathbf{0} \end{cases} \tag{74}$$

Where V is the target secret and $U \notin \mathbf{\Sigma}$. We want to show that the relation $U - X_1Y_1 = 0$, does not affect the distribution of $\widetilde{V}$.

Our reasoning in the proof of this lemma is similar to those in Lemma 6.

A system of equations in a finite field $\mathbb{F}_q$ has a limited number of unique solutions. Let, $\mathbf{S}^1$ to $\mathbf{S}^N$ be all the solutions of $\mathtt{NL}(V, \mathbf{\Sigma}, X_1, Y_1) = 0$. where $\mathbf{S}^i$ is a vector of values as $\mathbf{S}^i = (V^i, \mathbf{\Sigma}^i, X_1^i, Y_1^i)$.

For each $\mathbf{S}^i$, the relation $U - X_1Y_1 = 0$, after substitution, will be $U = X_1^iY_1^i$. So, associated with each solution for $\mathtt{NL}(V, \mathbf{\Sigma}, X_1, Y_1) = 0$, there is a unique solution for $U - X_1Y_1 = 0$. Therefore, the frequency of values of V in the solutions of (74) is solely determined by $\mathtt{NL}(V, \mathbf{\Sigma}, X_1, Y_1) = 0$.

I Proof of Lemma 10

The block diagram of the operations for a single value of i is depicted in figure 7.

From all the variables in the lemma, only the members of $\overrightarrow{S_i}$ and its secret, S_i, are present in the rest of the computations. So, variables such as shares of $\overrightarrow{K_i^1}$, $\overrightarrow{T_i}$, and $\overrightarrow{R_i}$, are important as much as they give information about the members of $\overrightarrow{S_i}$, and the S_i itself.

Since P_i is assumed known, recovery of K_i^1 is sufficient to derive T_i, R_i, and S_i. In this way, we can assume that the only secret of the construction in figure 7 is S_i.

Except for the SXOR (the shared XOR), computing the secret recovery probability is directly possible with Lemma 7 and Lemma 8.

The probability of recovering the secret at the SXOR gadget is estimated in the following. Considering SXOR gadget, the secret is recovered only if

1. Each share of T_i is known, or the corresponding shares of P_i and K_i^1 are both known.

2. Each share of K_i^1 is known, or the corresponding shares of P_i and T_i are both known.

Shares of T_i and K_i^1 (without using the algebraic relation $\overrightarrow{T_i} = \overrightarrow{K_i^1} \oplus \overrightarrow{P_i}$) are known independently with the probability $\epsilon + f_{\mathsf{SR}}(n, \epsilon)/n$, and each share of P_i is independently known with probability ϵ. So, each share of T_i (by using the algebraic relation), utilizing the union bound, will be known to the adversary with the probability

$$\epsilon_1' \leq \epsilon(\epsilon + \frac{f_{\mathsf{SR}}(n, \epsilon)}{n}) + (\epsilon + \frac{f_{\mathsf{SR}}(n, \epsilon)}{n}) \approx \epsilon + \frac{f_{\mathsf{SR}}(n, \epsilon)}{n}. \tag{75}$$

The interesting regions for ϵ and $f_{\mathsf{SR}}(n, \epsilon)/n$ are usually in $(0$ to $.1)$. The probability of the cases 1 and 2, above, are equal and are $(\epsilon_1')^n$.

Not that the output of S-Box is directly connected to an SR gadget (not shown in figure 7). From the perspective of the output (and the input), we derive the ϵ_2' by Lemma 7 as $\epsilon_2' \approx \epsilon + f_{\mathsf{SR}}(n, \epsilon)/n$.

Finally, the overall probability for the leakage of S_i by Lemma 8 will be

$$2e_{\mathsf{SR}}(n, \epsilon) + e_{\mathsf{S\text{-}Box}'}(n, \epsilon_2') + 2(\epsilon_1')^n. \tag{76}$$

This completes the proof.