Low Randomness Masking and Shuffling: An Evaluation Using Mutual Information *

Kostas Papagiannopoulos
kostaspap88@gmail.com, kostaspap88@protonmail.ch

Digital Security Department, Radboud University Nijmegen, The Netherlands

Abstract. Side-channel countermeasure designers often face severe performance overheads when trying to protect a device. Widely applied countermeasures such as masking and shuffling entail generating a large amount of random numbers, which can result in a computational bottleneck. To mitigate the randomness cost, this work evaluates low-randomness versions of both masking and shuffling, namely Recycled Randomness Masking (RRM) and Reduced Randomness Shuffling (RRS). These countermeasures employ memory units to store generated random numbers and reuse them in subsequent computations,making them primarily suitable for implementation on devices with sufficient memory. Both RRM and RRS are evaluated using the MI-based framework in the context of horizontal attacks. The evaluation exhibits the tradeoff between the randomness cost and the noisy leakage security level offered by the countermeasures, enabling the designer to fine-tune a masking or shuffling scheme and maximize the security level achieved for a certain cost.

Keywords: SCA, RNG, masking, shuffling

1 Introduction

The continuously growing Internet of Things (IoT) is rapidly changing modern infrastructure. Several industrial sectors, including construction, IT, agriculture, energy and automotive manufacturing, are already harnessing the transformative impact of IoT on their products. The price drop of IoT devices has enhanced everyday objects with data processing capabilities and network connectivity. Still, the two most important challenges to IoT adoption are the high cost of investment, as well as current concerns about security and privacy [2]. When combined together, these challenges exacerbate the need for devices that offer an adequate level of protection at a reasonable cost, thus motivating the current line of research.

For instance, the option of power/electromagnetic side-channel attacks (SCA) allows adversaries to recover sensitive data, by observing and analyzing the physical characteristics and emanations of a cryptographic implementation [29]. Naturally, these attacks have lead towards countermeasures that perform noise amplification to impede potential adversaries. Two of the most widely deployed countermeasures in cryptographic implementations are masking and shuffling and they are often combined in order to enhance the security level [14, 34, 38]. In order to hinder the attacker, masking applies secret-sharing techniques that randomize intermediate values, while shuffling randomizes the order of the cryptographic blocks and/or the implementation's instructions. As a result, both countermeasures require random numbers to function, making on-chip random number generation (RNG) a useful addition to the device.

*The work described in this paper has been supported by the Netherlands Organization for Scientific Research NWO under project ProFIL (628.001.007).

In order to perform RNG, designers can opt to use a symmetric cipher structure as a pseudo random number generator, relying for instance on either full-round or reduced-round block cipher [23] or on a stream cipher construct. Alternatively, they can opt directly for physical RNG or for structures that use a physical random number generator to seed deterministic random bit generators [5]. When implementing masking or shuffling, the RNG requirement imposes a non-negligible performance overhead that can impact the latency/throughput of the cryptographic implementation. Even if it is possible to perform the RNG procedure during idle phases of the device, the required computations will directly impact the device's energy consumption.

Recently, Faust et al. [20], have introduced the concept of amortizing randomness in a masking scheme, i.e. recycling the available randomness between several gadgets in order to reduce the RNG cost. Their work establishes the notion of security with common randomness (denoted as $t-$SCR) and provides composable ($t-$SNI) gadgets [6] that achieve randomness recycling. However, their analysis relies on simulation-based proofs that do not take into account the effect of recycling on the noise level of the device and on the noise amplification stage of masking.

Our contribution. In this work, we put forward low-randomness versions of standard masking and shuffling countermeasures, which we refer to as Recycled Randomness Masking (RRM) and Reduced Randomness Shuffling (RRS) respectively. RRM and RRS are able to reduce the RNG overhead by employing memory units to store random numbers and reuse them later, e.g. in subsequent executions of the protected cipher. Trading RNG overhead for memory overhead implies that every random number that is reused needs to be stored. As a result, the proposed RRM and RRS schemes are geared towards microcontroller units and possibly high-end FGPAs, since such devices can offer a fairly large amount of memory storage. On the contrary, the more strict area requirements in ASIC devices, encourage recycling on-the-fly, similarly to Faust et al. [20].

Subsequently, we investigate the noisy leakage security of RRM and RRS. We note that the formal approach of Faust et al. [20] has already investigated the $t-$SNI property of certain RRM schemes, introducing $t-$SCR. In order to establish a more holistic notion of security, we complement their approach by performing an analysis of a $t-$NI RRM scheme under the noisy leakage model, using the MI framework for SCA [35]. In particular, we demonstrate how reducing the available randomness for performance/cost reasons interacts with the noise amplification stages of RRM and RRS. Thus, we establish a direct link between the randomness cost and the noisy leakage security level provided by a countermeasure, i.e. *we integrate the noise factor in our analysis*. We conclude that this randomness-security tradeoff constitutes a potent tool in the designer's arsenal that enables us to provide adequate security (in the noisy leakage model), while limiting the computational cost that stems from RNG. In addition to the noise-oriented approach, we provide several efficient $t-$NI custom multiplication gadgets for low-order RRM schemes.

The rest of this paper is organized as follows. In Section 2, we provide background information w.r.t. masking, shuffling as well as the notation used throughout the paper. In Section 3 we provide the improved RRM gadgets and analyze the noise amplification stage of RRM. Similarly, we describe RRS and analyze its noise amplification stage in Section 4. Conclusions and future directions are discussed in Section 5.

2 Background & Related Work

Notation. Random variables are denoted with capital letters. Instances of random variables and constant values are denoted with lowercase letters. Capital bold letters are used for random variable vectors and matrices, unless otherwise specified. Calligraphic font denotes sets and sans serif fonts denote leakage functions. The identity leakage function is

stated as $L_{id}(.)$. Observable leakages of a certain intermediate value V (or its instance v) are denoted using subscript L_V (or L_v respectively). Leakages observed after a specific cipher layer are denoted using superscript $L^{<layer>}$. The average of leakage variables from set $\mathcal{S}$ is denoted as $\overline{L}$, i.e. $\overline{L} = (1/|\mathcal{S}|) * \sum_{v \in \mathcal{S}} L_v$.

Boolean Masking. Chari et al., Goubin et al. and Messerges [14, 21, 30] were the first to suggest randomizing intermediate values with a secret sharing scheme, forcing the adversary to analyze higher-order statistical moments. In detail, a dth-order secure Boolean masking scheme splits a sensitive value x into $d+1$ shares $(x_0, x_1, \ldots, x_d)$, as shown below.

$$x = x_0 \oplus x_1 \oplus \cdots \oplus x_d$$

The shares $(x_0, x_1, \ldots, x_d)$ are also referred to as the $(d+1)$-family of shares corresponding to x [33]. Assuming sufficient noise, it has been shown that the number of traces required for a successful attack grows exponentially w.r.t. the security order d [14, 32], i.e. masking performs *noise amplification*. Several definitions have been used to specify the formal security properties of a masking scheme, and we revisit the most relevant below.

Probing-secure scheme. We refer to a scheme that uses certain families of shares as $t-$probing-secure iff any set of at most t intermediate variables is independent from the sensitive values [27].

Non-interfering scheme. We refer to a scheme as $t-$non-interfering ($t-$NI) iff any set of at most t intermediate variables can be perfectly simulated with at most t shares of each input [6].

Strongly non-interfering scheme. We refer to a scheme as strong non-interfering ($t-$SNI) iff any set of at most t intermediate variables, where t_1 are on the internal variables and t_2 on the output variables, can be perfectly simulated with at most t_1 shares of each input [6].

Multiplying two families of shares under an ISW Boolean masking scheme consists of the computation of all partial products, as well as a compression algorithm that produces the final result, while injecting randomness [8, 27]. Several implementation techniques and evaluation strategies have been suggested in the context of masking. With respect to implementation aspects, the techniques proposed range from lookup-table techniques [15,39] to GF-based circuits [13, 22, 33]. Regarding the evaluation strategies, recent advances by Battistello et al. [7] and Grosso et al. [25], suggest that masked multiplications are prone to horizontal attacks, i.e. attacks that exploit several noisy intermediate values that are computed during the scheme. In this work, we put specific emphasis on the impact of horizontal exploitation to the noisy leakage security level of the scheme.

In the application of Boolean masking schemes, secure multiplications require quadratic data complexity w.r.t. randomness, in order to ensure the refreshing of partial products. Initially, Rivain et al. [33] extended the ISW scheme [27] and put forward a d-private compression algorithm (RP) that can compute dth-order secure multiplications in $GF(2^n)$ using $d(d+1)/2$, n-bit elements. Following, Belaïd, Benhamouda, Passelègue et al. [8] suggested an improved d-private compression (BBP) that performs partial product refreshing using $\lfloor d^2/4 \rfloor + d$ random numbers for security orders $d > 4$. In addition, they derived optimal compression algorithms for security orders $d = 2, 3$ and 4 which have data complexity, respectively $2, 4$ and 5 random elements per multiplication.

Despite recent efforts, it is notable that high-order masking implies a severe RNG overhead. Making for instance a 2nd-order secure AES implementation with optimal compression (2nd-order secure BBP) requires 10240 random bits per block encryption[1]. Generating this amount of random bits with a pseudo AES-based random number generator in ATmega microcontrollers implies an optimistic cost of roughly 20k clock cycles (2-round AES generator) and a pessimistic cost of 100k clock cycles (10-round AES generator) [1, 23].

[1] The cipher runs for 10 rounds consisting of 16 Sboxes, each requiring 32 $GF(2)$ multiplications that need 2 random elements for refreshing the partial products.

We observe that the pessimistic case is fairly close to the computational cost of the 2nd-order secure AES on AVR devices [3], i.e. it amounts to approximately 38% of the clock cycles. Similarly, a 2nd-order secure PRESENT implementation on an ARM Cortex-M device spends 25% of its execution time for TRNG [18]. The severe overhead of RNG in masking countermeasures can render the implementation cost prohibitive for small embedded devices and has led countermeasure designers towards lightweight alternatives. Low-entropy masking schemes [9,24] reduce the randomness requirements by using masks chosen within a subset of all the possible masks, yet if the leakage function is not linear, they may reduce the security order. Schemes that amortize randomness [20] can achieve similar goals without this shortcoming. In similar lines of work, threshold implementations examined techniques that reduce or even minimize the fresh randomness required to achieve uniformity [10,17]. Still, we stress that several of these schemes need to be evaluated in a fair manner, i.e. by using horizontal leakage exploitation, such as the analysis carried out by Battistello et al. [7] and Grosso et al. [25].

Shuffling. The shuffling countermeasure results in spreading information over n different points in time, according to a random permutation $\mathbf{P}_n$ [34]. The permutation $\mathbf{P}_n$ is defined as a vector $(P_1, \ldots, P_n)$, where P_i represents the new position of element i and thus $\mathbf{P}_n$ is defined over the set of all possible n-dimensional permutations $\mathcal{P}_n$. For instance, assume two independent variables $\mathbf{X} = (X_1, X_2)$ that leak $\mathbf{L} = (L_{X_1}, L_{X_2})$ at different points in time. The shuffling scheme will generate a 2-dimensional permutation $\mathbf{P}_2$ s.t. $L_{X_1} = \mathsf{L}_{id}(X_{P_1}) + noise$ and $L_{X_2} = \mathsf{L}_{id}(X_{P_2}) + noise$. Charvillon et al. [38] have analyzed the security provided by shuffling, in addition to investigating several implementation techniques. Motivated by the increased cost of RNG, Veshchikov et al. [36] investigated cheaper shuffling methods. We will refer to a permutation that shuffles n independent operations of a specific cipher layer as $\mathbf{P}_n^{\{o_1,\ldots,o_n\}}$, where o_i the ith operation in the layer.

Similar to masking, applying the shuffling countermeasure implies a non-negligible randomness cost. Specifically, generating a permutation for shuffling k independent operations of the same type requires $k * \lceil log_2(k) \rceil$ random bits, using a slightly-biased version of the Knuth shuffle algorithm [28,38]. In a practical scenario, shuffling only 16 AES Sboxes requires 640 random bits in total[2]. In order to deal with this RNG overhead, previous work on the shuffling countermeasure opted to reduce the amount of possible permutations (random start index), to shuffle only in selected rounds (partial shuffling) or to use non-homogeneous shuffle patterns, where the amount of possible permutations varied between cipher layers [26,34].

3 Recycled Randomness Masking - RRM

This section puts forward a low-randomness version of the standard Boolean masking schemes, namely it introduces Recycled Randomness Masking (RRM). The novelty of RRM lies in considering two or more masked multiplications simultaneously and sharing randomness between their compression layers. Using this approach, we develop $t-$NI gadgets that reduce the RNG overhead of masked ciphers and enable side-channel protection at a modest budget. We commence with two elementary examples that will be used throughout this section to illustrate the core recycling idea and we introduce additional notation to describe generic RRM schemes (Section 3.1). We continue with section 3.2 which searches for optimized $t-$NI randomness-recycling gadgets using formal verification techniques and applies them to the AES cipher. Finally, Section 3.3 analyzes the noise amplification stage of RRM schemes and demonstrates the impact of recycling in the noisy leakage model.

[2]The cipher runs for 10 rounds, permuting 16 independent operations of the same type (Sbox) per round. Every permutation requires $16 * \lceil log_2(16) \rceil$ random bits.

3.1 Recycling Randomness in Masking

We illustrate the application of RRM using two masked ISW multiplications $z = xy$ and $c = ab$. The multiplications are protected by ISW of security order $d = 1$ (Figure 1) or ISW of security order $d = 2$ (Figure 2). Both examples assume 4 independent families of shares $(x_i)_{0\leq i\leq d}$, $(y_i)_{0\leq i\leq d}$, $(a_i)_{0\leq i\leq d}$ and $(b_i)_{0\leq i\leq d}$ in $GF(2)$. Values $t_0, t_1, t_2, w_0, w_1, w_2$ are random elements in $GF(2)$ that are necessary to maintain probing security[3].

$$
\begin{aligned}
z_0 &= x_0y_0 \oplus \mathbf{w_0} \\
z_1 &= x_1y_1 \oplus (\mathbf{w_0} \oplus x_0y_1) \oplus x_1y_0
\end{aligned}
\qquad\qquad
\begin{aligned}
c_0 &= a_0b_0 \oplus (\mathbf{t_0} \leftarrow \mathbf{w_0}) \\
c_1 &= a_1b_1 \oplus ((\mathbf{t_0} \leftarrow \mathbf{w_0}) \oplus a_0b_1) \oplus a_1b_0
\end{aligned}
$$

(*multiplication 1*) (*multiplication 2*)

Figure 1: RRM scheme applied on two 1st-order secure ISW multiplications, generating random element w_0 in multiplication 1 and recycling it in multiplication 2.

$$
\begin{aligned}
z_0 &= x_0y_0 && \oplus && \mathbf{w_0} && \oplus && \mathbf{w_1} \\
z_1 &= (\mathbf{w_0} \oplus x_0y_1) \oplus x_1y_0 && \oplus && x_1y_1 && \oplus && \mathbf{w_2} \\
z_2 &= (\mathbf{w_1} \oplus x_0y_2) \oplus x_2y_0 && \oplus && (\mathbf{w_2} \oplus x_1y_2) \oplus x_2y_1 && \oplus && x_2y_2
\end{aligned}
$$

(*multiplication 1*)

$$
\begin{aligned}
c_0 &= a_0b_0 && \oplus && (\mathbf{t_0} \leftarrow \mathbf{w_0}) && \oplus && (\mathbf{t_1} \leftarrow \mathbf{w_1}) \\
c_1 &= ((\mathbf{t_0} \leftarrow \mathbf{w_0}) \oplus a_0b_1) \oplus a_1b_0 && \oplus && a_1b_1 && \oplus && \mathbf{t_2} \\
c_2 &= ((\mathbf{t_1} \leftarrow \mathbf{w_1}) \oplus a_0b_2) \oplus a_2b_0 && \oplus && (\mathbf{t_2} \oplus a_1b_2) \oplus a_2b_1 && \oplus && a_2b_2
\end{aligned}
$$

(*multiplication 2*)

Figure 2: RRM scheme applied on two 2nd-order secure ISW multiplications, generating random elements w_0 and w_1 in multiplication 1 and recycling them in multiplication 2.

In Figures 1 and 2, red-annotated variables are fresh random elements and green-annotated variables are recycled random elements. The left-arrow assignment describes the recycling of a random element in a different multiplication. For instance, in the 1st-order secure ISW-based scheme of Figure 1, the element $\mathbf{w_0}$ is generated in multiplication 1 and it is subsequently recycled in multiplication 2 ($\mathbf{t_0} \leftarrow \mathbf{w_0}$). Likewise, the 2nd-order secure example of Figure 2 showcases two ISW multiplications, which originally require 6 random elements: (w_0, w_1, w_2) and (t_0, t_1, t_2). To tackle the RNG overhead, RRM generates 3 fresh random elements (w_0, w_1, w_2) during multiplication 1 and recycles w_0 in t_0 and w_1 in t_1. Thus, the amount of random elements required in multiplication 2 is reduced from three to a single random element (t_2).

The proposed recycling technique can be generalized to more than two multiplications of any order and to describe such generic RRM schemes we introduce the following notation. We assume a gadget consisting of n dth-order secure masked multiplications, where every masked multiplication requires s random elements to maintain probing security, e.g. multiplication i requires random elements $(r_{i,1}, r_{i,2}, \ldots, r_{i,s})$. We assume all the inputs to the masked multiplications to be independent families of shares, which may require fresh randomness in our implementation. In addition, we assume that at least one out of n masked multiplications will generate fresh randomness. Subsequently we define a

[3]Throughout this work we consider elements in $GF(2)$, yet our results and observations remain applicable in larger fields.

recycle set $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$ that consists of sets $\mathcal{R}_i$, $1 \leq i \leq n$. Every set $\mathcal{R}_i$ describes all the the fresh or recycled random elements $r_{i,j}$, $1 \leq i \leq n, 1 \leq j \leq s$ that the multiplication i is using to maintain probing security. Figure 1 for instance has $\mathcal{R} = \{\{r_{1,1}\}, \{r_{2,1}\}\} = \{\{w_0\}, \{w_0\}\}$, since the single random element w_0 is generated and used in multiplication 1 and it is reused (recycled) in multiplication 2. Similarly, in Figure 2, $\mathcal{R} = \{\{r_{1,1}, r_{1,2}, r_{1,3}\}, \{r_{2,1}, r_{2,2}, r_{2,3}\}\} = \{\{w_0, w_1, w_2\}, \{w_0, w_1, t_2\}\}$, since random elements w_0 and w_1 are generated in multiplication 1 and they are recycled in multiplication 2, while element t_2 is generated in multiplication 2. If only a single multiplication generates fresh random elements and all the other multiplications recycle them, then it holds that $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$, where $\mathcal{R}_i = \mathcal{R}_j$ for all $1 \leq i, j \leq n$. Symmetrically, if no randomness gets recycled (a.k.a. standard masking), then it holds that $R_i \cap R_j = \emptyset$ for all $1 \leq i, j \leq n$ and $i \neq j$. To specify the RNG overhead when recycling, we define the *randomness cost* of an RRM gadget with n multiplications as the total amount of fresh random elements generated. E.g. in Figure 1 the randomness cost is 1 and in Figure 2 the cost is 4, while in general the cost of an RRM scheme with recycle set $\mathcal{R}$ is $|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$. The cost of standard masking of n multiplications (without recycling randomness) is equal to $n * s$. In addition, we define the *masking recycle factor* f_{rm} of every random element in the RRM scheme as the number of times it has been used in any multiplication. In the example of Figure 1, $f_{rm}(w_0) = 4$, since it occurs twice in every multiplication. Similarly, in the example of Figure 2, $f_{rm}(w_0) = f_{rm}(w_1) = 4$ and $f_{rm}(w_2) = f_{rm}(t_2) = 2$. It is noteworthy that the recycling of random numbers is similar to the repeated access to shares observed by Battistello et al. [7], where the recycle factor of a share in dth-order secure scheme is shown to be equal to $d + 1$. We will henceforth refer to a dth-order secure masking gadget with n multiplications and recycle set $\mathcal{R}$ as RRM$(d, n, \mathcal{R})$. It is important to stress that RRM necessitates storing and fetching the recycled random elements. Let *gain* $g = n * s - |\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$ be the reduction in randomness cost achieved by an RRM$(d, n, \mathcal{R})$ scheme. RRM requires g less random elements and at most g extra storage units, depending on how many times the elements are recycled. In addition, RRM requires at most g extra store and fetch instructions when recycling. For example, in Figure 2 the gain $g = 2 * 3 - |\{w_0, w_1, w_2\} \cup \{w_0, w_1, t_2\}| = 2$ and it implies 2 extra storage units (w_0 and w_1), 2 extra store instructions and 2 extra fetch instructions.

3.2 Efficient RRM Multiplication Gadgets

As demonstrated, the core contribution of RRM is to reduce the randomness cost of n multiplications below the $n * s$ random elements which are required by standard masking. Notably, both ISW and BBP schemes are already reusing random elements during the compression layer of a single multiplication, while maintaining dth-order probing security[4]. Still, excessive recycling between multiplications can lead to RRM gadgets that are no longer probing-secure. For instance, assume the ISW-based RRM$(2, 2, \mathcal{R})$ gadget of Section 3.1, Figure 2 with recycle set $\mathcal{R} = \{\{w_0, w_1, w_2\}, \{w_0, w_1, w_2\}\}$, i.e. 3 fresh elements are generated in multiplication 1 and they are all recycled in multiplication 2. Then, the tuple (z_2, c_2) depends on the sensitive values x, y, a and b simultaneously, because $z_2 \oplus c_2 = x_0y_2 \oplus x_2y_0 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_2 \oplus a_0b_2 \oplus a_2b_0 \oplus a_1b_2 \oplus a_2b_1 \oplus a_2b_2$. Since there exists such a tuple (z_2, c_2), the particular RRM gadget is not 2nd-order probing-secure.

As a result, this section proposes $t-$NI optimized multiplication gadgets that are capable to recycle a large amount of randomness. Analytically, for an RRM$(d, n, \mathcal{R})$ gadget, we search for recycle sets $\mathcal{R}$ that minimize the randomness cost, while the gadget remains $t-$NI. We focus on small orders ($d = 1, 2, 3$) and two multiplications per gadget

[4]ISW uses a symmetric compression structure that reuses every random number once, thus requiring half as many numbers as the naive solution. BBP uses a less regular structure which also reuses every fresh number once.

($n = 2$) due to their practical relevance in implementations. To detect potential security flaws, we use the Lisp-based formal verification tool suggested by Coron [16]. The tool generates all possible tuples of intermediate values (with dimension less or equal to d) that stem from the RRM$(d, n, \mathcal{R})$ gadget and verifies the $t-$NI property using circuit transformations. This process is repeated for all recycle sets $\mathcal{R}$ that ensure the correctness of the scheme, rejecting the insecure choices and identifying the optimized recycle set that minimizes the randomness cost.

The performed brute-force search of Algorithm 3a is carried out for both ISW-based and BBP-based schemes[5] and Figures 3b until 3f demonstrate the optimized $t-$NI gadgets. The randomness and storage requirements of the proposed RRM gadgets are demonstrated in Tables 1 and 2, which confirm that RRM is capable of reducing the randomness cost substantially when compared to standard masking. It remains an open research question to quantify how much the lack of composability (strong non-interference) affects the efficiency, i.e. how many additional refresh layers will be required in the scheme in order to provide a fair comparison with the work of Faust et al. [20].

Table 1: Randomness cost of optimized RRM schemes for $n = 2$ multiplications.

		Scheme compression					
		ISW security order d			**BBP security order d**		
		1	2	3	1	2	3
Recycling	yes	1	4	8	1	2	6
	no	2	6	12	2	4	8

Table 2: Storage cost of optimized RRM schemes for $n = 2$ multiplications, assuming no storage is needed when recycling within a single multiplication (large register file).

		Scheme compression					
		ISW security order d			**BBP security order d**		
		1	2	3	1	2	3
Recycling	yes	1	2	4	1	2	2
	no	0	0	0	0	0	0

On the application of RRM gadgets to the AES Sbox. Applying these novel randomness-recycling gadgets in the AES cipher is extremely straightforward. Assume a 1st-order secure masked AES cipher that uses the Boyar-Peralta decomposition [11] in the Sbox implementation, i.e. the Sbox requires 32 multiplications in $GF(2)$. During the first execution of the AES cipher, we generate all the necessary random elements without any recycling, i.e. for the first full Sbox execution we need $16 * 32 * 1 = 512$ random elements, resulting in $10 * 512 = 5120$ random elements for 10 rounds of Sbox executions. During the second independent execution of the AES cipher every Sbox multiplication can recycle the randomness generated in the respective multiplication of the first execution, since the gadget RRM$(1, 2, \{\{r_1\}, \{r_1\}\})$ is $t-$NI (Figure 3b). Thus, the Sbox-related RNG cost of two AES executions is reduced from 10240 to 5120, i.e. RRM achieves a 50% reduction of the RNG overhead, at the penalty of 5120 element storage, 5120 store and 5120 fetch instructions[6]. In a similar fashion, we can apply the 3rd-order secure BBP-based RRM scheme. During the first AES execution we generate $10 * 16 * 32 * 4 = 20480$ random

[5] Running the tool on an Intel Core i7-4719HQ @ 2.5GHz requires minutes to verify 1st and 2nd-order secure RRM gadgets and can reach approximately 5 hours for the verification of 3rd-order secure RRM gadgets.

[6] The actual number of instructions depends on the architecture.

n: number of multiplications
$\mathcal{R}$: recycle sets
d: security order of multiplications
$\mathcal{T}_d$: d-sized tuples
procedure RecycleSetSearch(n,d)
  for all $\mathcal{R}$ **do**
    Generate $\mathcal{T}_d$ for RRM$(d, n, \mathcal{R})$
    for all tuples $\mathbf{t} \in \mathcal{T}_d$ **do**
      Verify $t-$NI of $\mathbf{t}$
    if secure $\forall \mathbf{t} \in \mathcal{T}_d$ **then**
      Compute $RandomnessCost(\mathcal{R})$

(a) Brute-force search algorithm.

$$z_0 = x_0y_0 \oplus \mathbf{r}_1$$
$$z_1 = x_1y_1 \oplus (\mathbf{r}_1 \oplus x_0y_1) \oplus x_1y_0$$

$$c_0 = a_0b_0 \oplus \mathbf{r}_1$$
$$c_1 = a_1b_1 \oplus (\mathbf{r}_1 \oplus a_0b_1) \oplus a_1b_0$$

(b) ISW RRM$(1, 2, \{\{r_1\}, \{r_1\}\})$ randomness/storage cost = 1

$$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2$$
$$z_1 = x_1y_1 \oplus (\mathbf{r}_1 \oplus x_0y_1) \oplus x_1y_0 \oplus \mathbf{r}_3$$
$$z_2 = x_2y_2 \oplus (\mathbf{r}_2 \oplus x_0y_2) \oplus x_2y_0 \oplus (\mathbf{r}_3 \oplus x_1y_2) \oplus x_2y_1$$

$$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2$$
$$c_1 = a_1b_1 \oplus (\mathbf{r}_1 \oplus a_0b_1) \oplus a_1b_0 \oplus \mathbf{r}_4$$
$$c_2 = a_2b_2 \oplus (\mathbf{r}_2 \oplus a_0b_2) \oplus a_2b_0 \oplus (\mathbf{r}_4 \oplus a_1b_2) \oplus a_2b_1$$

(c) ISW RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$ randomness cost = 4, storage cost = 2

$$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus x_0y_2 \oplus x_2y_0$$
$$z_1 = x_1y_1 \oplus \mathbf{r}_2 \oplus x_0y_1 \oplus x_1y_0$$
$$z_2 = x_2y_2 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus x_1y_2 \oplus x_2y_1$$

$$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus a_0b_2 \oplus a_2b_0$$
$$c_1 = a_1b_1 \oplus \mathbf{r}_2 \oplus a_0b_1 \oplus a_1b_0$$
$$c_2 = a_2b_2 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus a_1b_2 \oplus a_2b_1$$

(d) BBP RRM$(2, 2, \{\{r_1, r_2\}, \{r_1, r_2\}\})$ randomness/storage cost = 2, left-to-right evaluation

$$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$$
$$z_1 = x_1y_1 \oplus (\mathbf{r}_1 \oplus x_0y_1) \oplus x_1y_0 \oplus \mathbf{r}_4 \oplus \mathbf{r}_5$$
$$z_2 = x_2y_2 \oplus (\mathbf{r}_2 \oplus x_0y_2) \oplus x_2y_0 \oplus (\mathbf{r}_4 \oplus x_1y_2) \oplus x_2y_1 \oplus \mathbf{r}_6$$
$$z_3 = x_3y_3 \oplus (x_2y_3 \oplus \mathbf{r}_6) \oplus x_3y_2 \oplus (x_1y_3 \oplus \mathbf{r}_5) \oplus x_3y_1 \oplus (x_0y_3 \oplus \mathbf{r}_3) \oplus x_3y_0$$

$$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus \mathbf{r}_3$$
$$c_1 = a_1b_1 \oplus (\mathbf{r}_1 \oplus a_0b_1) \oplus a_1b_0 \oplus \mathbf{r}_7 \oplus \mathbf{r}_8$$
$$c_2 = a_2b_2 \oplus (\mathbf{r}_2 \oplus a_0b_2) \oplus a_2b_0 \oplus (\mathbf{r}_7 \oplus a_1b_2) \oplus a_2b_1 \oplus \mathbf{r}_6$$
$$c_3 = a_3b_3 \oplus (a_2b_3 \oplus \mathbf{r}_6) \oplus a_3b_2 \oplus (a_1b_3 \oplus \mathbf{r}_8) \oplus a_3b_1 \oplus (a_0b_3 \oplus \mathbf{r}_3) \oplus a_3b_0$$

(e) ISW RRM$(3, 2, \{\{r_1, r_2, r_3, r_4, r_5, r_6\}, \{r_1, r_2, r_3, r_7, r_8, r_6\}\})$, randomness cost = 8, storage cost = 4

$$z_0 = x_0y_0 \oplus \mathbf{r}_1 \oplus x_0y_3 \oplus x_3y_0 \oplus \mathbf{r}_2 \oplus x_0y_2 \oplus x_2y_0$$
$$z_1 = x_1y_1 \oplus \mathbf{r}_3 \oplus x_1y_3 \oplus x_3y_1 \oplus \mathbf{r}_2 \oplus x_1y_2 \oplus x_2y_1$$
$$z_2 = x_2y_2 \oplus \mathbf{r}_4 \oplus x_2y_3 \oplus x_3y_2$$
$$c_3 = a_3b_3 \oplus \mathbf{r}_4 \oplus \mathbf{r}_3 \oplus \mathbf{r}_1 \oplus a_0b_1 \oplus a_1b_0$$

$$c_0 = a_0b_0 \oplus \mathbf{r}_1 \oplus a_0b_3 \oplus a_3b_0 \oplus \mathbf{r}_5 \oplus a_0b_2 \oplus a_2b_0$$
$$c_1 = a_1b_1 \oplus \mathbf{r}_3 \oplus a_1b_3 \oplus a_3b_1 \oplus \mathbf{r}_5 \oplus a_1b_2 \oplus a_2b_1$$
$$c_2 = a_2b_2 \oplus \mathbf{r}_6 \oplus a_2b_3 \oplus a_3b_2$$
$$z_3 = a_3b_3 \oplus \mathbf{r}_6 \oplus \mathbf{r}_3 \oplus \mathbf{r}_1 \oplus a_0b_1 \oplus a_1b_0$$

(f) BBP RRM$(3, 2, \{\{r_1, r_2, r_3, r_4\}, \{r_1, r_5, r_3, r_6\}\})$, randomness cost = 6, storage cost = 2, left-to-right evaluation

Figure 3: Efficient RRM gadgets

elements and in the second AES execution we recycle part of them using the specification of the BBP$(2, 2, \{\{r_1, r_2, r_3, r_4\}, \{r_1, r_5, r_3, r_6\}\})$ gadget of Figure 3f. In this case, RRM achieves a 25% RNG reduction, while storing and fetching 20480 elements. Proving the $t-$NI property for RRM gadgets with more than 2 multiplications ($n > 2$) can enable recycling between more than 2 independent AES executions.

3.3 RRM Noise Amplification

The previous section (Section 3.2) pinpointed the first pitfall of RRM schemes, i.e. how excessive recycling can result in gadgets that are not probing-secure. Having tackled this issue for low-order gadgets with formal methods, we proceed towards the second pitfall of RRM. Namely, excessive recycling is hazardous to the noise amplification stage of masking, even when the gadget is probing-secure. Specifically, this section analyzes the noise amplification stage of several $t-$NI RRM gadgets of Section 3.2, using the mutual information metric suggested by Standaert et al. [35]. In other words, we evaluate the proposed "recycling" countermeasure in the noisy leakage model and compare it to standard masking schemes. The effectiveness of the noise amplification stage of RRM largely depends on the adversary's capability to observe multiple noisy intermediate values during the gadget's execution. We refer to this capability as *horizontal exploitation* and we consider the following cases (C1-C3), in ascending order of adversarial strength:

C1 **Naive-tuple attack.** The adversary exploits a single noisy $(d+1)$-tuple of the RRM gadget and *disregards any repetition* of noisy intermediate values. This scenario is equivalent to an attack against a non-recycling scheme that disregards intra-multiplication repetitions.

C2 **Chosen-tuple attack.** First, the adversary observes the noisy leakage of share repetitions (noted also by Battistello et al. [7]) and the noisy leakage of random element repetitions (noted in this work as "randomness recycling") in the gadget. Second, he averages the observed noisy leakages in order to denoise the side-channel emission. Finally, he exploits a *chosen* leakage $(d+1)$-tuple of the RRM gadget that takes advantage of the denoising.

C3 **Full-state attack.** First, the adversary observes the noisy leakage of share repetitions and random element repetitions in the gadget. Second, he averages the observed leakages in order to denoise the side-channel emission. Finally, he exploits the *full state*, i.e. all leaky intermediate values of the RRM gadget.

For our information-theoretic analysis (cases C1-C3), we introduce the following notation to describe the leaky intermediate values and the noisy leakage of RRM gadgets. In a given $(d+1)$-tuple of intermediate values, let random variable S be the sensitive (key-dependent) intermediate value under attack and let random variables $M_0, \ldots, M_{d-1}$ be the masks used to protect the sensitive value. The leakage of a $(d+1)$-tuple is described using the following random vector: $\mathbf{L} = (L_{S \oplus_{i=0}^{d-1} M_i}, L_{M_0}, \ldots, L_{M_{d-1}}) + \mathbf{N}$, where $L_{S \oplus_{i=0}^{d-1} M_i} = \mathsf{L}_{id}$ $(S \oplus M_0 \oplus \cdots \oplus M_d)$, $L_{M_i} = \mathsf{L}_{id}$ (M_i), $0 \leq i \leq d-1$ and $\mathbf{N}$ is a $(d+1)$-dimensional random vector representing Gaussian noise. We assume independent and equal noise σ^2 in every sample, i.e. diagonal noise covariance matrix and $L_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $0 \leq i \leq d$.

In the naive-tuple case C1, the adversary disregards the multiple accesses to the family shares (due to the structure of the masking scheme) and also disregards the random element repetition (due to recycling), thus he cannot observe any repeated leakages. In other words, the noise amplification stage in the C1 case is equivalent to that of standard Boolean masking. This naive case is only applicable if the evaluator cannot identify and locate the sample positions of the repeated leakages.

Contrary to C1, the chosen-tuple case C2 assumes that the adversary can locate the leakage sample position of repeated shares and recycled random elements, yet he is still limited to exploit a single $(d+1)$-tuple of leaky intermediate values for his attack. The number of repetitions of a specific random element or share v in the RRM gadget is equal to its recycle factor $f_{rm}(v)$. Averaging all available samples that leak value v results in substantial noise reduction, i.e. $\overline{L_v} \sim \mathcal{N}(\mu_v, \sigma^2/f_{rm}(v))$, which the adversary can use in order to diminish the noise amplification effect of masking. Specifically, he can target a carefully chosen $(d+1)$-tuple of leaky intermediate values, whose leakages have been noise-reduced beforehand. For instance, going back to the example of Section 3.1 - Figure 2, a sufficient (yet not efficient) attack tuple for RRM(2,2,$\{\{w_0, w_1, w_2\}, \{w_0, w_1, t_2\}\}$) is (x_0y_0, x_1y_1, x_2y_2). Since all the intermediate values of the tuple appear only once, it holds that $L_{x_iy_i} \sim \mathcal{N}(\mu_{x_iy_i}, \sigma^2)$ for $0 \leq i \leq 2$ and the noise amplification is the same as standard masking. A more efficient choice is tuple (z_2, w_1, w_2), where $L_{z_2} \sim \mathcal{N}(\mu_{z_2}, \sigma^2)$, yet $\overline{L_{w_1}} \sim \mathcal{N}(\mu_{w_1}, \sigma^2/4)$ and $\overline{L_{w_2}} \sim \mathcal{N}(\mu_{w_2}, \sigma^2/2)$, because $f_{rm}(w_1) = 4$ and $f_{rm}(w_2) = 2$.

To highlight the effects of recycling on the noisy leakage model, we performed an MI-based evaluation for 1st and 2nd-order secure ISW RRM gadgets that are proposed in Section 3.2. We make various choices w.r.t. the recycling factor (f_{rm} ranges from 1 to 10) and the strength of horizontal exploitation (we consider both naive-tuple C1 and chosen-tuple C2 adversaries). The experiments are described in Table 3. Naturally, the evaluation depends on the aforementioned parameters, yet we stress that it is adaptable to all RRM choices made by the countermeasure designer. Concretely, computing the MI-metric for a $(d+1)$-tuple requires summing over the randomness vector $\mathbf{M} = (M_0, \ldots, M_{d-1})$ and computing $(d+1)$-dimensional integrations [25]. The resulting MI vs. noise variance plot is visible in Figure 4 (left). In addition to the MI-metric, we use the conjecture of Duc et al. [19], in order to approximate the number of traces required to perform a key recovery in the high-noise regime. Analytically, we use the bound $\#traces \geq \frac{H[S]}{MI(S;L)^{d+1}}$ and the no. of traces vs. noise variance plot is visible in Figure 4 (right).

$$MI(S;\mathbf{L}) = H[S] + \sum_{s\in\mathcal{S}} Pr[s] \cdot \sum_{\mathbf{m}\in\mathcal{M}^d} Pr[\mathbf{m}] \cdot \int_{\mathbf{l}\in\mathcal{L}^{(d+1)}} Pr[\mathbf{l}|s,\mathbf{m}] \cdot log_2 Pr[s|\mathbf{l}] \;\; d\mathbf{l}$$

$$\text{where } Pr[s|\mathbf{l}] = \frac{\sum_{\mathbf{m}^*\in\mathcal{R}} Pr[\mathbf{l}|s,\mathbf{m}^*]}{\sum_{s^*\in\mathcal{S}} \sum_{\mathbf{m}^*\in\mathcal{R}} Pr[\mathbf{l}|s^*,\mathbf{m}^*]}$$

The evaluation results of 1st and 2nd-order secure RRM (cases C1 and C2) are visible in Figure 4 (left), from which we derive three core observations. First, we note that the intermediate values used by the attacker affect directly the RRM evaluation, i.e. the attacker can reduce the security level only by including the average leakage of the repeated random elements or shares in his attack. If the noise-reduced leakages are disregarded (Figure 4 solid red and solid blue lines), then the noise amplification remains intact and equivalent to standard masking of the same order. Second, assuming the right tuple is chosen, we observe that increasing the total recycle factor shifts the MI-curve to the right, i.e. the amount of recycling (modest or excessive) indeed damages the noise amplification stage of the scheme. This shift is visible between the dashed blue line (modest recycling) and the dotted blue line (excessive recycling). Note also that excessive recycling may increase the MI of a 2nd-order secure gadget above the MI of a 1st-order secure scheme. Third, we conclude that the RRM technique is in fact a tradeoff between the mutual information level achieved and the randomness cost required. This fact solidifies it as a lightweight alternative to standard masking that can be used by countermeasure designers when the randomness cost becomes prohibitive in a certain application context. Naturally, the designer needs to always be aware of the device's noise level in order to adapt RRM order and recycle set accordingly.

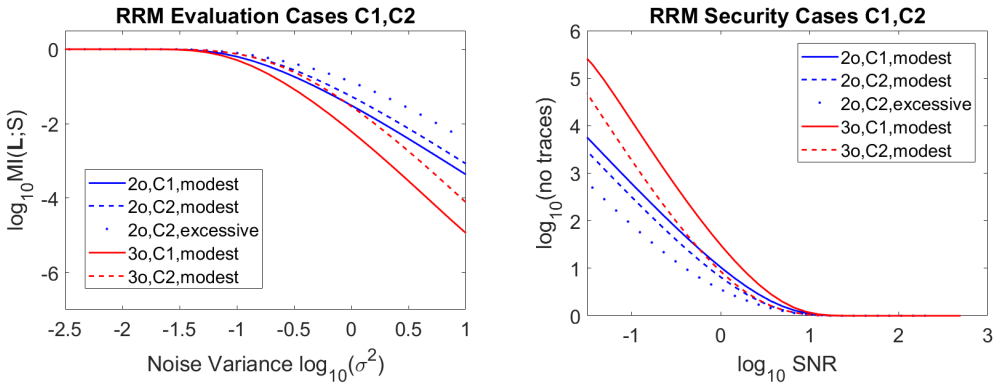


Figure 4: MI evaluation and no. traces bound for 1st and 2nd-order secure RRM schemes with 2 multiplications, assuming naive-tuple (C1 - equivalent to std. masking) and chosen-tuple (C2) adversaries. The evaluation considers gadgets with modest and excessive recycling. Blue lines denote 2nd-order attack (vs. 1st-order RRM) and red lines denote 3rd-order attack (vs. 2nd-order RRM).

Table 3: $t-$NI RRM gadgets analyzed in the noisy leakage model assuming naive-tuple (C1) and chosen-tuple (C2) adversaries. The attacks exploit a large amount of the available recycling (case C2, excessive recycling) or a small amount of recycling (case C2, modest recycling) or they disregard recycling (case C1).

Attack description	**RRM$(d, n, \mathcal{R})$ gadget**	**Attack tuple**	**Recycle factor $\mathbf{f_{rm}}$**
Naive-tuple 2nd-order attack modest recycling	1st-order secure RRM$(1, 2, \{\{r_1\}, \{r_1\}\})$	(x_0y_0, x_1y_1)	$f_{rm}(x_0y_0) = 1$ $f_{rm}(x_1y_1) = 1$
Chosen-tuple 2nd-order attack modest recycling	1st-order secure RRM$(1, 2, \{\{r_1\}, \{r1\}\})$	(z_1, r_1)	$f_{rm}(z_1) = 1$ $f_{rm}(r_1) = 2$
Chosen-tuple 2nd-order attack excessive recycling	1st-order secure RRM$(1, 10, \{\{r_1\}, \ldots, \{r_1\}\})$	(z_1, r_1)	$f_{rm}(z_1) = 1$ $f_{rm}(r_1) = 10$
Naive-tuple 3rd-order attack modest recycling	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	(z_0, z_1, z_2)	$f_{rm}(z_0) = 1$ $f_{rm}(z_1) = 1$ $f_{rm}(z_2) = 2$
Chosen-tuple 3rd-order attack modest recycling	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	(z_2, r_2, r_3)	$f_{rm}(z_2) = 1$ $f_{rm}(r_2) = 4$ $f_{rm}(r_3) = 2$

In case C3, the adversary is capable of exploiting the full state of a multiplication, which implies a computational overhead in the MI-formula due to the increase in the integral dimension and the scheme order. In order to bypass this limitation, we simplify the computation of MI, using the approach established by Grosso et al. [25]. Analytically, in order to include the $(d+1)^2$ partial products in our evaluation, we use the information bound established by Prouff et al. [32], stating that the multiplication's leakage is roughly $1.72(d+1) + 2.72$ times the leakage of a $(d+1)$-tuple. Computing the bound reduces the evaluation of an RRM multiplication to the evaluation of a single $(d+1)$-tuple. We also employ the independent shares' leakage assumption to reduce the leakage vector from the information of a $(d+1)$-tuple $\mathbf{X}$ to the information of a single share X_i, i.e. $\mathbf{L_X} = L_{X_i}$ [19]. However, simplifying to a single-share evaluation does not directly capture the noise reduction issue of RRM, caused by random element and/or share repetitions.

To incorporate the noise reduction in our evaluation, we consider the worst-case scenario were the adversary is able to reduce the noise of *all intermediate values* by a recycle factor f_{rm}^{max}. The factor f_{rm}^{max} is the maximum recycle factor observed in any random number or share, e.g. in the gadget RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$, the maximum recycling is observed on random numbers r_1 and r_2, thus $f_{rm}^{max} = 4$. Note that f_{rm}^{max} may stem from either repetitions of random numbers or repetitions of shares. The bound constructed is conservative, since we assume an adversary that can average *every noisy intermediate value* of the encoding by the maximum recycle factor, i.e. $L_{X_i} \sim \mathcal{N}(\mu_{X_i}, \sigma^2/f_{rm}^{max})$, $0 \leq i \leq d$. Still, it provides an efficient alternative to direct computation of the MI formula and demonstrates the evaluation trend for RRM schemes in the high-noise regime. It remains open whether closer bounds can be derived for such scenarios. In Figure 5 (left) we demonstrate the MI evaluation of 2nd and 3rd-order secure RRM schemes, with known recycle factor f_{rm}^{max} shown in Table 2, using the conservative bound which raises $MI(X_i; L_{X_i})$ to the security order. In Figure 5 (right) we demonstrate the no. of traces bound.

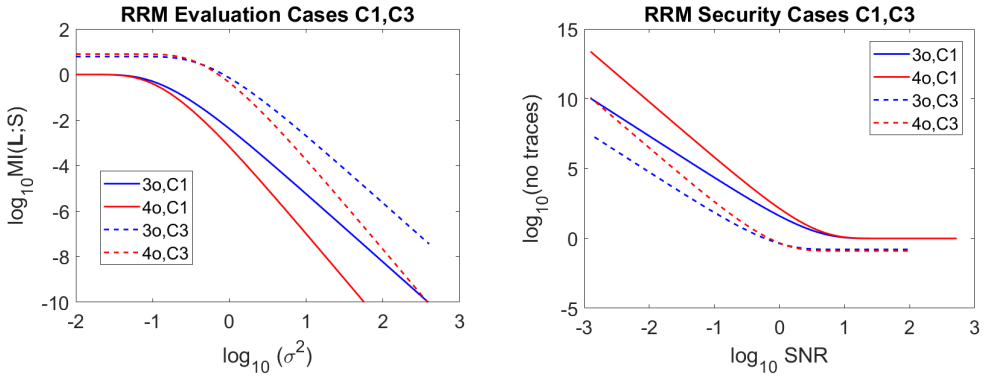


Figure 5: MI evaluation for 2nd and 3rd-order secure RRM schemes comparing naive-tuple (C1) with full-state attack (C3). Blue lines denote 3rd-order attack (vs. 2nd-order RRM) and red lines denote 4th-order attack (vs. 3rd-order RMM).

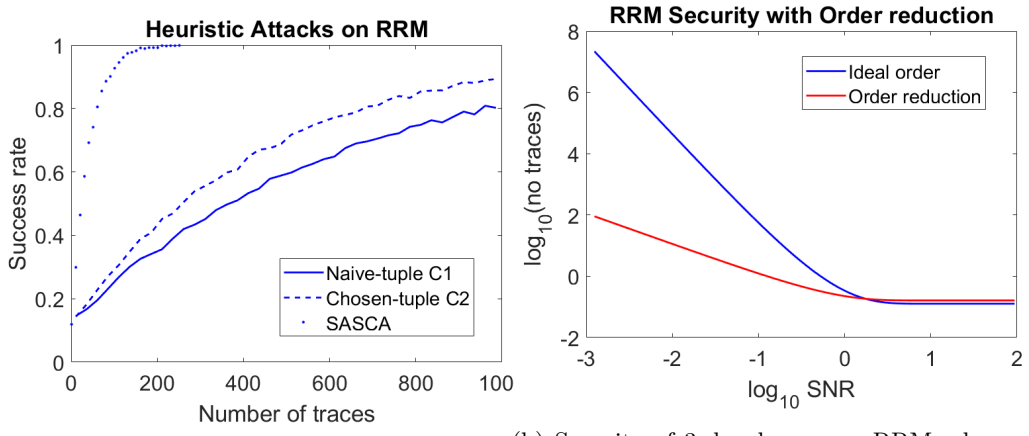
Table 4: $t-$NI RRM gadgets analyzed using a conservative bound assuming naive-tuple (C1) and full-state (C3) adversaries.

Attack description	**RRM$(d, n, \mathcal{R})$ gadget**	**Attack tuple**	**Recycle factor f_{rm}**
Naive-tuple 3rd-order attack	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	full state	$f_{rm}^{max} = 4$
Chosen-tuple 3rd-order attack	2nd-order secure RRM$(2, 2, \{\{r_1, r_2, r_3\}, \{r_1, r_2, r_4\}\})$	full state	$f_{rm}^{max} = 4$
Naive-tuple 4th-order attack	3rd-order secure RRM$(3, 2, \{\{r_1, r_2, r_3, r_4, r_5, r_6\}, \{r_1, r_2, r_3, r_7, r_8, r_6\}\})$	full state	$f_{rm}^{max} = 4$
Naive-tuple 4th-order attack	3rd-order secure RRM$(3, 2, \{\{r_1, r_2, r_3, r_4, r_5, r_6\}, \{r_1, r_2, r_3, r_7, r_8, r_6\}\})$	full state	$f_{rm}^{max} = 4$

On practical attacks and realistic leakage models. The non-trivial data complexity of the optimal attacks for large order d and large amount of integral dimensions, has led to the development of heuristic attacks that combine the horizontal information of several leaking instructions in a sub-optimal, yet efficient manner. To demonstrate this, we provide two types of *heuristic horizontal attacks* on simulated leakages of a 1st-order secure gadget with 16 multiplications, namely RRM$(1, 16, \{\{r_1\}, \ldots, \{r_1\}\})$. First, we employ the chosen-tuple attack (C2), where the attacker chooses a $(d+1)$-tuple of leakages

whose values have been sufficiently denoised by averaging the respective repetitions. The horizontal exploitation of this heuristic attack implies a small overhead for the adversary, namely he needs to perform an averaging pre-processing step. Consecutively, the adversary will employ the noise-reduced tuple in order to attack using Correlation Power Analysis (CPA) [12]. The second heuristic attack that we use in order to exploit horizontally the simulated traceset is a Soft Analytical Side-Channel Attack (SASCA) [37], applied in the context of masking [25]. The SASCA performs the same averaging during the preprocessing step of the first heuristic attack. Continuing, it exploits the full state of a multiplication by constructing a factor graph and using a belief propagation algorithm. The horizontal exploitation of SASCA implies an overhead depending on the factor graph. The results of the two heuristic attacks and the results of the naive-tuple CPA attack without noise averaging (C1) are visible in Figure 6a. As expected, the additional effort w.r.t. horizontal exploitation of the SASCA attack improves the success rate compared to C1 and C2.

Moreover, throughout this work we assumed an idealized leakage noisy leakage model, namely independent shares that leak according to the identity function. In practice, several devices showcase order reduction due to various device effects such as glitches, distance-based leakages and coupling [31]. We demonstrate this effect on Figure 6b, using a 3rd-order secure RRM scheme and the order-reduction theorem of Balasch et al. [4], which states that distance-based leakages can reduce the security order from d to $\lfloor \frac{d-1}{2} \rfloor$. The red and blue lines of Figure 6b give the lower and upper security bounds caused by a large class of real-world leakage flaws.



(a) Success rate of naive, chosen-tuple and SASCA attacks on simulated traces of 1st-order secure RRM with $f_{rm} = 16$.

(b) Security of 3rd-order secure RRM scheme under ideal (blue line) and order-reduced leakage (red line). The order-reduced line is equivalent to 1st-order secure RRM.

Figure 6: Practical attacks and realistic leakage models

On the necessity of a noise-based analysis. We conclude this section by showcasing the importance of a noise-oriented analysis of RRM using the following custom scenario. Assume an RRM gadget that recycles a single random number between n 1st-order secure ISW multiplications with mutually independent inputs, where n is large. Trivially, an ISW-based proof shows such a case to be secure, because the adversary can probe only a single intermediate value and thus cannot view multiple recyclings. Thus, we can recycle a random number infinitely while the scheme remains probing-secure. In practice however, the noise level of the leaking random number can be eliminated by averaging the recyclings. A noise-based analysis such as the MI metric or the SASCA can exploit recycling horizontally and is essential to quantify the security damage. If instead the attack remains naive, it may lure the evaluator into a false sense of security.

4 Reduced Randomness Shuffling - RRS

Motivated by the recycling ideas of Section 3, we use a similar approach on the popular shuffling countermeasure against side-channel analysis. Analytically, we put forward the Reduced Randomness Shuffling (RRS) countermeasure which consists of three shuffling variants that can alleviate the randomness cost involved. In Section 4.1 we analyze how RRS reduces the randomness cost compared to standard shuffling. Section 4.2 analyzes the susceptibility of RRS to horizontal/multivariate attacks in the noisy leakage model.

4.1 Reducing Randomness in Shuffling

To achieve the goal of RNG reduction, we explore the following three variants: *partitioned, merged* and *recycled* shuffling. We demonstrate these three variants using a generic structure of *layers* and *independent operations*. In particular, we assume that the cipher we want to shuffle can be described by the *layer set* $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ that consists of sets $\mathcal{L}_i$, $1 \leq i \leq n$. Every set $\mathcal{L}_i$ describes s independent operations that constitute this layer, e.g. $\mathcal{L}_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,s}\}$. The partitioning of a cipher into layers and of layers into independent operations rests upon the countermeasure designer and it is closely related to the cipher implementation. For instance, the independent operations may range from whole cipher parts (e.g. shuffling Sboxes) to individual assembly operations (e.g. shuffling key-dependent instructions). We will refer to a Reduced Randomness Shuffling scheme that shuffles independent operations according to the layer set $\mathcal{L}$ as RRS($\mathcal{L}$). In addition we specify the *randomness cost* of the RRS countermeasure as the total RNG overhead required to shuffle the cipher according to layer set $\mathcal{L}$. Figures 7a-7d illustrate the application of RRS on a layered structure.

The example of Figure 7a commences with an RRS scheme that shuffles a cipher structure, using $n = 2$ layers and $s = 4$ independent operations per layer, i.e. layer set $\mathcal{L} = \{\{o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}\}, \{o_{2,1}, o_{2,2}, o_{2,3}, o_{2,4}\}\}$. Both layers are shuffled using two different permutations on 4 operations, namely permutations $\mathbf{P}_4^{\mathcal{L}_1}$ and $\mathbf{P}_4^{\mathcal{L}_2}$. Thus, the randomness cost for a single execution of the 2-layer structure is $|\mathcal{R}| * |\mathcal{R}_i| * \lceil log_2(|\mathcal{R}_i|) \rceil = 2 * 4 * log_2(4) = 16$ bits.

To scale down the randomness cost, *partitioned shuffling* splits *vertically* a set of independent operations $\mathcal{R}_i$ into two or more smaller subsets that are cheaper to shuffle. For instance, in Figure 7b, instead of shuffling a single set of 4 independent operations $\mathcal{L}_1 = \{o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}\}$, we opt to partition $\mathcal{L}_1$ in two subsets of 2 independent operations each. Thus, we will partition $\mathcal{L}_1$ to $\mathcal{L}'_1 = \{o_{1,1}, o_{1,2}\}$ and $\mathcal{L}''_1 = \{o_{1,3}, o_{1,4}\}$. An analogous partitioning is done in $\mathcal{L}_2$, resulting in $\mathcal{L}'_2 = \{o_{2,1}, o_{2,2}\}$ and $\mathcal{L}''_2 = \{o_{2,3}, o_{2,4}\}$. We define the granularity of this vertical partitioning as the *partition factor* f_p, where $f_p = 1$ implies no partitioning. Performing partitioned shuffling with factor f_p on $|\mathcal{R}_i|$ independent operations reduces the randomness cost of layer i to $|\mathcal{R}_i| * \lceil log_2(|\mathcal{R}_i|/f_p) \rceil$. In the example of Figure 7b, we use $f_p = 2$ on both cipher layers and we replace $\mathbf{P}_4^{\mathcal{L}_1}$ and $\mathbf{P}_4^{\mathcal{L}_2}$ with $\mathbf{P}_2^{\mathcal{L}'_1}$ and $\mathbf{P}_2^{\mathcal{L}'_2}$ respectively, reducing the cost of a single execution from 16 to 8 bits.

To similar ends, the *merged shuffling* variant combines several cipher layers *horizontally* in order to permute them together. The example of Figure 7c views $\mathcal{L}_1$ and $\mathcal{L}_2$ as a single layer and shuffles them using the same permutation. That is, we merge $\mathbf{P}_4^{\mathcal{L}_1}$ and $\mathbf{P}_4^{\mathcal{L}_2}$ into permutation $\mathbf{P}_4^{\mathcal{L}''}$, s.t. $\mathcal{L}'' = \{\{o_{1,1}, o_{2,1}\}, \{o_{1,2}, o_{2,2}\}, \{o_{1,3}, o_{2,3}\}, \{o_{1,4}, o_{2,4}\}\}$. We define the granularity of this horizontal combination as the *merge factor* f_m, where $f_m = 1$ implies no merging and observe that merged shuffling can reduce the randomness cost of a single iteration to $(|\mathcal{R}|/f_m) * k * \lceil log_2(|\mathcal{R}_i|) \rceil$. Naturally, merging and partitioning can be combined, resulting in randomness cost of $(|\mathcal{R}|/f_m) * |\mathcal{R}_i| * \lceil log_2(|\mathcal{R}_i|/f_p) \rceil$ bits per iteration. Still, different cipher layers can present a different number of independent operations for partitioned/merged shuffling and thus may need to be homogenized by

shuffling additional dummy operations.

Last, *recycled shuffling* opts for the "external" recycling of the generated permutation, i.e. we reuse a permutation between different executions or rounds of the cipher structure. In Figure 7d, the layer $\mathcal{L}_1$ of cipher execution no. 1 and the layer $\mathcal{L}_1$ of cipher no. 2 are independent, yet they are shuffled with the same permutation $\mathbf{P}_4^{\mathcal{L}_1}$. We define the *recycle factor of shuffling* f_{rs} as the number of repetitions of a permutation in different cipher iterations, i.e. $f_{rs} = 1$ implies no recycling. Recycled shuffling can reduce the randomness cost of a certain layer i from $(\#executions) * |\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|) \rceil$ to $(\#executions/f_{rs}) * |\mathcal{L}_i| * \lceil log_2(|\mathcal{L}_i|) \rceil$.

We note that only recycled shuffling implies an overhead due to storage units and store/fetch instructions, while partitioned and merged shuffling simply use less randomness. The overhead relates to the recycle factor of shuffling, i.e. reusing the same permutation results in f_{rs} extra store/fetch instructions and a memory unit to store the random number.

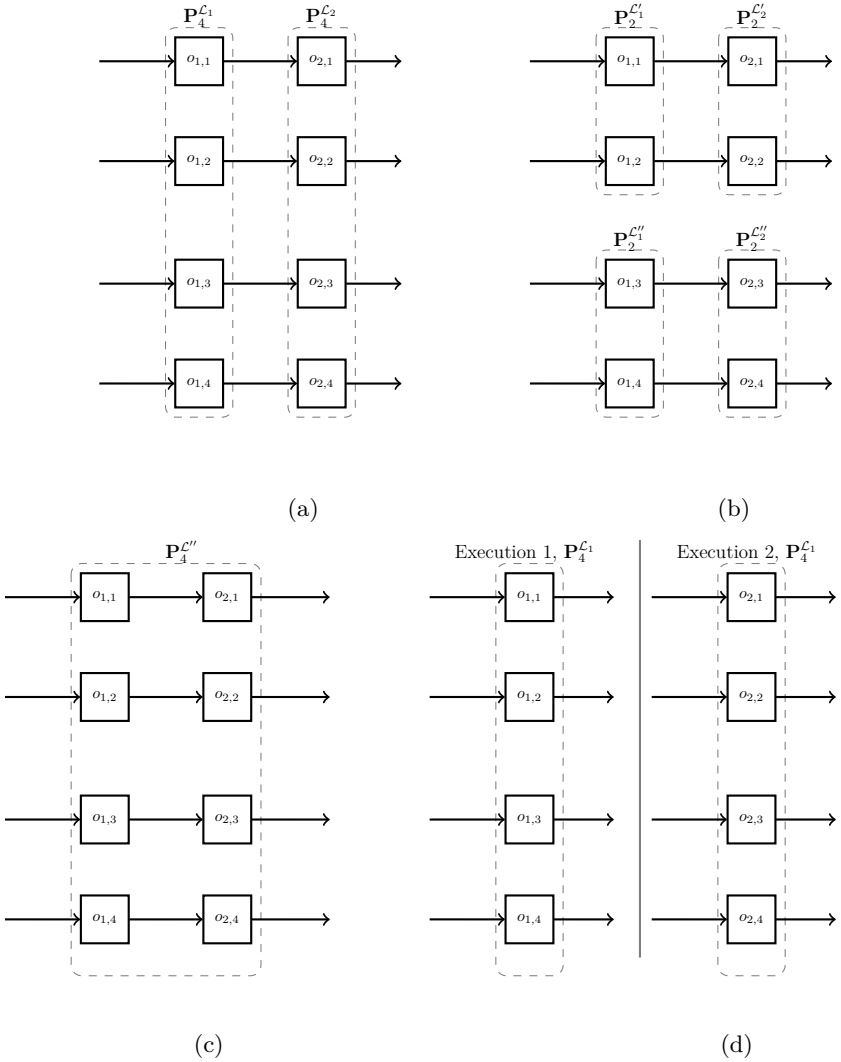


Figure 7: Initial, partitioned, merged and recycled shuffle is applied to the layered cipher structure in Figures (a) - (d). Dashed-line boxes indicate the operations and layers that are shuffled with the same permutation. The arrows indicate the information flow between layers.

On the application of RRS to the AES cipher. Assume that the countermeasure designer focuses on the first two layers of the AES cipher, namely KeyAddition (ka) and Sbox (sb). The standard way to shuffle them would require two permutations on 16 independent operations, i.e. $\mathbf{P}_{16}^{KA}$ and $\mathbf{P}_{16}^{S}$, costing 128 random bits per round, resulting in 1280 bits for 10 rounds of AES. Alternatively, the designer can opt to partition both layers with partition factor $f_p = 4$, i.e. split $\{ka_1, \ldots, ka_{16}\}$ and $\{sb_1, \ldots, sb_{16}\}$ into $\{ka_1, \ldots, ka_4\}, \{ka_5, \ldots, ka_8\}, \{ka_9, \ldots, ka_{12}\}, \{ka_{13}, \ldots, ka_{16}\}$ and $\{sb_1, \ldots, sb_4\}, \{sb_5, \ldots, sb_8\}, \{sb_9, \ldots, sb_{12}\}, \{sb_{13}, \ldots, sb_{16}\}$ respectively. Thus, the cost is reduced to 640 bits ($\mathbf{P}_4^{KA}$ and $\mathbf{P}_4^{S}$ for 50% RNG reduction). In a similar fashion, the designer can merge the KeyAddition and Sbox layers into a single layer, i.e. $\mathcal{L} = \{kasb_1, \ldots, kasb_{16}\}$, reducing again the cost to 640 bits ($\mathbf{P}_{16}^{KA,S}$ for 50% RNG reduction). Finally, any generated permutations on KeyAddition, Sbox can be recycled in subsequent AES executions, reducing RNG even further, at the penalty of extra storage.

4.2 RRS Noise Amplification

As expected, reducing the randomness cost of shuffling has a direct impact on the noise amplification effect of the countermeasure, offering an interesting randomness-security tradeoff for the designer. Similarly to Section 3.3, we evaluate the variants of RRS via the mutual information framework and consider an adversary that can exploit horizontally more than a single cipher layer. We perform our evaluation on the layered cipher structure used previously, where the adversary attempts to recover any part of the key $\mathbf{k} = (k_0, k_1, k_2, k_3)$ that is related to the 4 independent operations of $\mathcal{L}_1$ and $\mathcal{L}_2$. To that end, he may exploit the leakage from both layers as well as direct leakage from the permutations used to shuffle these layers. Below, we introduce the random variable notation that describes shuffling in the noisy leakage model.

- The adversary can observe the leakage vector after every cipher layer, namely $\mathbf{L}^{\mathcal{L}_1}$ and $\mathbf{L}^{\mathcal{L}_2}$. The leakage variables L_i depend on the layer permutations $\mathbf{P}_n^{\mathcal{L}_1}$ and $\mathbf{P}_n^{\mathcal{L}_2}$, thus it holds that $L_i^{\mathcal{L}_1} = L_{id}\ (X_{P_i^{\mathcal{L}_1}}) + noise$ and $L_i^{\mathcal{L}_2} = L_{id}\ (Y_{P_i^{\mathcal{L}_2}}) + noise$, where $noise$ represents additive Gaussian noise $\mathcal{N}(0, \sigma^2)$.
- The adversary can observe the direct permutation leakage of every shuffled layer, namely $\mathbf{L'}^{\mathcal{L}_1}$ and $\mathbf{L'}^{\mathcal{L}_2}$. For layer permutations $\mathbf{P}_n^{\mathcal{L}_1}$ and $\mathbf{P}_n^{\mathcal{L}_2}$, it holds that $L_i'^{\mathcal{L}_1} = L_{id}\ (P_i^{\mathcal{L}_1}) + noise$ and $L_i'^{\mathcal{L}_2} = L_{id}\ (P_i^{\mathcal{L}_2}) + noise$, where $noise$ represents additive Gaussian noise $\mathcal{N}(0, \sigma^2)$.

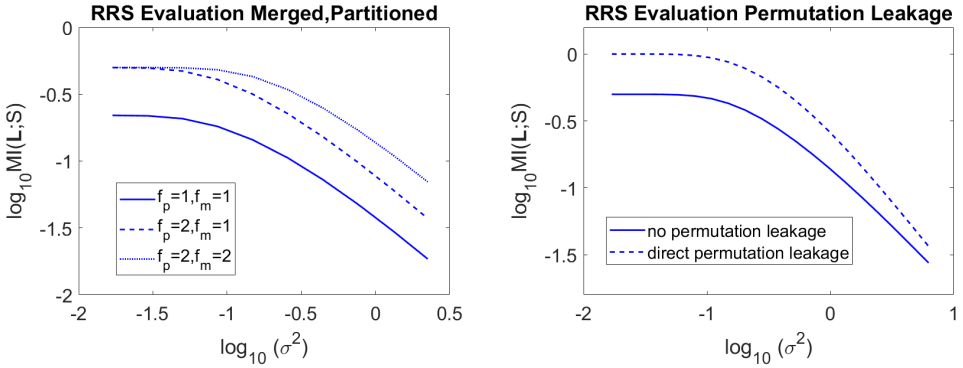
To analyze the tradeoff between the MI level and the randomness cost, we perform the MI-based evaluation for several versions of the RRS and attack options. The cases are demonstrated in Table 5. The evaluation uses the formula by Charvillon et al. [38], which we update in order to account for the partitioned, merged and recycled shuffling with factors f_p, f_m and f_{rs} respectively.

$$MI(K_t; \mathbf{L}) = H[K_t] + \sum_{k_t \in \mathcal{K}_t} Pr[k_t] \cdot \int_{\mathbf{l} \in \mathcal{L}^\eta} Pr[\mathbf{l}|k_t] \cdot log_2 Pr[k_t|\mathbf{l}]\ d\mathbf{l} \quad \text{where}$$

$$Pr[k_t|\mathbf{l}] = \frac{Pr[\mathbf{l}|k_t]}{\sum_{k_t^* \in \mathcal{K}_t} Pr[\mathbf{l}|k_t^*]} \ \text{ and } \ Pr[\mathbf{l}|k_t] = \sum_{\mathbf{p} \in \mathcal{P}_\theta} \frac{Pr[\mathbf{l'}|\mathbf{p}]}{\sum_{\mathbf{p}^* \in \mathcal{P}_\theta} Pr[\mathbf{l'}|\mathbf{p}^*]} \cdot Pr[\mathbf{l}|k_t, \mathbf{p}]$$

In the formula above, we assume that the adversary attacks a certain key part K_t, where $t \in \{0, 1, 2, 3\}$. We also note that the adversary in general exploits η-dimensional leakage vectors $\mathbf{L}, \mathbf{L'}$ and performs summations over the set of θ-dimensional permutations. In the following analysis we show how parameters η and θ relate to the particular RRS variant

used as well as the adversary's horizontal capabilities (i.e. the number of layers attacked). The results of the MI-based evaluation are visible in Figures 8a,8b.



(a) MI evaluation for partitioned and merged RRS schemes on 2 layers, without direct permutation leakage (cases D1, D2, D3).

(b) MI evaluation with and without direct permutation leakage, for parameters $f_p = 2, f_m = 2, frs = 2$ (cases D4, D5).

Figure 8: RRS MI evaluation

Table 5: Reduced Randomness shuffling schemes analyzed in the context of single-layer and two-layer horizontal attacks, with/without direct permutation leakage.

Case	No. of layers attacked	Direct permutation leakage exploited	Partition factor f_p	Merge factor f_m	Recycle factor f_{rs}
No RRM (D1)	1	no	1	1	1
Partitioned (D2)	1	no	2	1	1
Partitioned, merged (D3)	2	no	2	2	1
Partitioned, merged, recycled (D4)	2	no	2	2	2
Partitioned, merged, recycled, direct perm. leakage (D5)	2	yes	2	2	2

In Table 5, case D1 , the adversary attacks a standard shuffling scheme where no randomness reduction is performed. We assume again the reduced block cipher structure with two layers, $k = 4$ independent operations per layer and a two 4-dimensional permutations $\mathbf{P}_4^{\mathcal{L}_1}$, $\mathbf{P}_4^{\mathcal{L}_2}$ for shuffling them. The adversary attacks a single layer, i.e. he focuses solely on the 4-dimensional leakages $\mathbf{L}^{\mathcal{L}_1}$ observed in layer $\mathcal{L}_1$. Disregarding layer $\mathcal{L}_2$ implies integration over 4-dimensional leakages, i.e. parameter $\eta = k = 4$. In addition, the permutation that the adversary needs to consider in the attack is 4-dimensional, resulting in parameter $\theta = k = 4$. Continuing, case D2 evaluates a single-layer attack on an RRS scheme where both $\mathcal{L}_1$ and $\mathcal{L}_2$ layers use partitioned shuffling with factor $f_p = 2$. In this case, the adversary evaluates by focusing on the 2-dimensional leakages of layer $\mathcal{L}_1$, which are shuffled by a 2-dimensional permutation $\mathbf{P}_2^{\mathcal{L}_1}$. In other words, it holds that $\eta = \theta = k/f_p = 2$. The MI curve of D2 is shifted to the right of curve D1, so we show that reducing the number of available permutations via partitioned shuffling is detrimental to the noise amplification stage.

Case D3 evaluates a two-layer attack on an RRS scheme that combines partitioned and merged shuffling with factors $f_p = 2$ and $f_m = 2$. Specifically, layers $\mathcal{L}_1$ and $\mathcal{L}_2$ are merged and shuffled together with permutation $\mathbf{P}_2^{(\mathcal{L}_1,\mathcal{L}_2)}$. The adversary takes advantage of this fact and targets both layers in order to extract more information horizontally. The attack uses leakage vectors $(\mathbf{L}^{\mathcal{L}_1}, \mathbf{L}^{\mathcal{L}_2})$, so $\eta = (\#no\_attacked\_layers) * (k/f_p) = 4$ and the only permutation in place is $\mathbf{P}_2^{(\mathcal{L}_1,\mathcal{L}_2)}$, thus $\theta = k/f_p = 2$. The MI curve of D3 is shifted

to the right of curve D2, showing that merged shuffling can improve the effectiveness of multi-layer horizontal attacks and it is detrimental to the MI level.

Last, we compare partitioned, merged and recycled shuffling (case D4) with equivalent shuffling that can observe the repeated direct permutation leakage. Specifically, in both cases, the adversary exploits horizontally two partitioned layers that use the same permutation, i.e. $\eta = 4$ and $\theta = 2$. Note however, that in case D5 the adversary can also observe the repeated direct permutation leakage, i.e. he has access to ${\mathbf{L}'}_j^{\mathcal{L}_1}$ for all executions $j = 1, \ldots, f_{rs}$, while D4 assumed equiprobable permutations. As a result, in case D5, the adversary can reduce the noise level of the direct permutation leakage by computing $\overline{\mathbf{L}'^{\mathcal{L}_1}} \sim \mathcal{N}(\boldsymbol{\mu}^{\mathcal{L}_1}, (1/f_{rs}) * \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ a diagonal covariance matrix. Figure 8b shows how exploiting the direct permutation leakage enhances the attack.

5 Conclusions & Future Directions

In this work, we have performed an in-depth investigation of low-randomness alternatives to standard masking and shuffling, namely RRM and RRS. The first core outcome is that RRM and RRS can offer effective tradeoffs between randomness cost and security. A designer of side-channel countermeasures can now rely on the MI-based evaluation and provide optimized and flexible protection that reduces the randomness cost.

The second core outcome of this work is demonstrating the importance of horizontal exploitation in masking and shuffling. We have shown that univariate (or partially horizontal) evaluations provide us with only a part of the whole picture and may lure the evaluator into a false sense of security. By examining the multivariate adversarial model, we exploit a larger quantity of the available leakage and provide a more complete security evaluation.

Last, this work has demonstrated the necessity of noise-based analysis as a complement to formal methods. We maintain that a sound evaluation approach is to start from a provably secure scheme and enhance it with a noise-based analysis in order to provide a more holistic view.

With regards to future work, we note that multivariate evaluation techniques are still at a nascent stage when it comes to real-world devices. In fact, research efforts concentrate on a fairly high abstraction layer, i.e. they only consider leaky cipher operations, disregarding many peculiarities of the hardware and physical layers. Future research needs to strive towards closing the gap between theoretical and applied evaluations and improve the attacks that are able to exploit horizontally RRM and RRS.

Moreover, a long-term vision is research towards unifying several side-channel and fault injection countermeasures under the MI framework. Based on this unification, the countermeasure designer will possess a plethora countermeasure options at his disposal, which he can combine and tweak in order to maximize side-channel and fault injection security w.r.t. a given budget in clock cycles, silicon area or power/energy consumption.

6 Acknowledgments

I would like to thank Lejla Batina, Joan Daemen and Vincent Grosso for their feedback comments and the discussions towards SASCA.

References

[1] Rijndaelfast implementation on atmel avr. http://point-at-infinity.org/avraes/.

[2] The internet of things business index 2017, The Economist Intelligence Unit. https://www.eiuperspectives.economist.com/sites/default/files/EIU-ARM-IBM%20IoT%20Business%20Index%202017%20copy.pdf.

[3] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 724–754, Cham, 2017. Springer International Publishing.

[4] 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.

[5] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf.

[6] Barthe, Gilles and Belaïd, Sonia and Dupressoir, François and Fouque, Pierre-Alain and Grégoire, Benjamin and Strub, Pierre-Yves and Zucchini, Rébecca Strong non-interference and type-directed higher-order masking. In *ACM Conference on Computer and Communications Security, CCS '16, 2016, Vienna, Austria*, pages 116–129.

[7] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 23–39, 2016.

[8] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 616–648, 2016.

[9] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. A low-entropy first-degree secure provable masking scheme for resource-constrained devices. In *Proceedings of the Workshop on Embedded Systems Security*, WESS '13, pages 7:1–7:10, New York, NY, USA, 2013. ACM.

[10] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order DPA resistant implementations of Keccak. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, pages 187–199, 2013.

[11] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In Paola Festa, editor, *Experimental Algorithms*, pages 178–189, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[12] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[13] D. Canright and Lejla Batina. A very compact "perfectly masked" s-box for AES. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, pages 446–459, 2008.

[14] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 398–412, 1999.

[15] Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.

[16] Jean-Sebastien Coron. Formal verification of side-channel countermeasures via elementary circuit transformations. Cryptology ePrint Archive, Report 2017/879, 2017. https://eprint.iacr.org/2017/879.

[17] Joan Daemen. *Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharing*, pages 137–153. Springer International Publishing, Cham, 2017.

[18] Wouter de Groot, Kostas Papagiannopoulos, Antonio de La Piedra, Erik Schneider, and Lejla Batina. Bitsliced masking and arm: Friends or foes? In Andrey Bogdanov, editor, *Lightweight Cryptography for Security and Privacy*, pages 91–109, Cham, 2017. Springer International Publishing.

[19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 401–429, 2015.

[20] Sebastian Faust, Clara Paglialonga, and Tobias Schneider. Amortizing randomness complexity in private circuits. Cryptology ePrint Archive, Report 2017/869, 2017. http://eprint.iacr.org/2017/869.

[21] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[22] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? Cryptology ePrint Archive, Report 2016/264, 2016. http://eprint.iacr.org/2016/264.

[23] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. multiparty computation: how large is the gap for aes? *J. Cryptographic Engineering*, 4(1):47–57, 2014.

[24] Vincent Grosso, François-Xavier Standaert, and Emmanuel Prouff. Low entropy masking schemes, revisited. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, pages 33–43, 2013.

[25] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight (and how to exploit it in security evaluations). Cryptology ePrint Archive, Report 2017/116, 2017. http://eprint.iacr.org/2017/116.

[26] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, pages 239–252, 2006.

[27] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[28] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[29] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[30] Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.

[31] Kostas Papagiannopoulos and Nikita Veshchikov. Mind the gap: Towards secure 1st-order masking in software. In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 282–297, 2017.

[32] 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.

[33] 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, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[34] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 171–188, 2009.

[35] François-Xavier Standaert, Tal 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, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.

[36] Nikita Veshchikov, Stephane Fernandes Medeiros, and Liran Lerman. Variety of scalable shuffling countermeasures against side channel attacks. In *Journal of Cyber Security and Mobility, 2016*, Volume 5, Issue 3, pages 195–232.

[37] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 282–296, 2014.

[38] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 740–757, 2012.

[39] Junwei Wang, Praveen Kumar Vadnala, Johann Großschädl, and Qiuliang Xu. Higher-order masking in practice: A vector implementation of masked AES for ARM NEON. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 181–198. Springer, 2015.