The Random Fault Model

Siemen Dhooghe and Svetla Nikova

COSIC, KU Leuven, Leuven, Belgium
`firstname.lastname@esat.kuleuven.be`

Abstract. In this work, we introduce the *random fault model* - a more advanced fault model inspired by the random probing model, where the adversary can fault all values in the algorithm but the probability for each fault to occur is limited. The new adversary model is used to evaluate the security of side-channel and fault countermeasures such as Boolean masking, error detection techniques, error correction techniques, multiplicative tags, and shuffling methods. The results of the security analysis reveal new insights both in the novel random fault model as well as in the established random probing model including: shuffling masked implementations does not significantly improve the random probing security over regular masking; error correction providing little security when faults target more bits (versus the significant improvement when using error detection); and the order in which masking and duplication are applied providing a trade-off between random probing and fault security. Moreover, the results also explain the experimental results from CHES 2022 and find weaknesses in the shuffling method from SAMOS 2021.

Keywords: Encoding, Masking, Physical Security, Random Probing, Shuffling

1 Introduction

The field of side-channel analysis, following Differential Power Analysis (DPA) by Kocher *et al.* [18], has made significant progress over the years. Currently, we are capable of practically protecting hardware applications against side-channel attacks using masking. However, such progress did not come without difficulty. Often masking schemes were proposed, broken, and patched. This trial-and-error approach caused the need for theoretical proofs to guarantee the security of new masking methods. The main adversary in the academic literature is considered in the *probing model* proposed by Ishai *et al.* [17]. While this probing model already captures basic attacks and allows for easy proofs, it does not cover more advanced attacks in practice. Instead, the model proposed by Chari *et al.* [7], called the *noisy leakage model*, is much closer to practice but requires a complicated security analysis for countermeasures making the model less used in papers. In 2014, Duc *et al.* [13] made a reduction between the probing and noisy leakage models. This reduction introduced a new intermediate model called the *random probing model* which allows to capture attacks such as horizontal attacks introduced by Clavier *et al.* [9] and which has an easier security analysis compared to the noisy leakage model. A security analysis in the random probing model allows for better insight in the security provided by different countermeasures, such as Boolean masking or shuffling, versus those provided in the standard probing model.

Compared to side-channel analysis, the field of fault attacks, following differential fault analysis by Biham and Shamir [5], is less studied and has not seen the same progress as side-channel analysis. As a result, standard fault security models are not yet accepted. The current most used academic model is an active variant of the probing model where the adversary can inject faults in a circuit up to a threshold number of wires or gates. However, experimental works such as the results by Bartkewitz *et al.* [3] note that a threshold fault model does not properly capture the practice where a single laser fault typically affects multiple values. Other adversary models from this threshold model have not been used to investigate and compare countermeasures. As such, the question remains how effective certain countermeasures are in a more realistic security model. Some popular countermeasures include: Boolean masking introduced by Patarin [15] and Chari *et al.* [7]; error detection methods such as in ParTI [21] and Impeccable Circuits [1]; error correction methods such as in Impeccable Circuits II [22]; multiplicative tags such as in CAPA [20] and M&M [10]; and shuffling such as Rocky [19]. However, none of the above countermeasures have been properly analyzed or compared to one another in a similar adversary model.

Contributions. The work essentially provides contributions in two fields: on random probing security, and on random fault security.

Considering random probing security, we investigate the security of the following countermeasures: Boolean masking, duplication, masked duplication, and shuffling. From the analysis, we made the following interesting observations.

- There is a security difference between masking-then-duplicating a variable versus duplicating-then-masking it.
- Both shuffling and shuffling with masking provides little improvement in random probing security no matter how the shuffling is performed.

Considering fault security, inspired by the random probing model, we propose a new fault adversary, the *random fault model*, which is allowed to fault all values in an algorithm but where each fault has a limited probability to apply. We then use the random fault model to analyze countermeasures in two security models: *correctness*, where the adversary's goal is to have an incorrect output, and *privacy*, where the adversary's goal is to retrieve secret information from the abort state of the algorithm. We analyze Boolean masking, error detection methods, error correction methods, multiplicative tags, and shuffling. From the analysis, we made the following interesting observations.

- There is a security difference between masking-then-duplicating a variable versus duplicating-then-masking it.
- We give a theoretical foundation to the experiments of Bartkewitz *et al.* [3] on faulting encoded variables with a different number of parity bits.

– Triplication provides significantly less security than duplication methods when the fault targets many bits.
– We observe that the random fault model can explain the success rate of statistical ineffective faults targeting multiple values similar to the random probing model explaining horizontal attacks.
– Multiplicative tags provide an exponential security gain in the field size in the correctness model.
– Shuffling exhibits several weaknesses when used to secure against fault attacks. As a result, we show that the work by Miteloudi *et al.* [19] has vulnerabilities.

2 Background

2.1 Notation

We consider stochastic variables, denoted as capital letters, over a finite field $\mathbb{F}_2$. We denote the probability of a stochastic variable attaining a value x as $\Pr[X = x]$ and the probability of X conditioned on Y as $\Pr[X = x | Y = y]$.

We define random functions as stochastic variables over a set of functions. For example, a function uniform randomly drawn from a set of functions.

2.2 Algorithmic Representation

We represent algorithms as a string of elements or elementary operations over a finite field $\mathbb{F}$. In this work, we consider only the field $\mathbb{F}_2$ for which the elementary operations are the field addition (XOR) and multiplication (AND). We assume that algorithms can sample uniform random field elements. Moreover, an algorithm is also able to abort the computation providing $\perp$ as the output. We give an example of a binary algorithm

$$(x, y, r \leftarrow \$, z \leftarrow x + y, w \leftarrow zy, v \leftarrow y + r)\,.$$

An algorithm can have an *encoding* phase, where its input is, for example, masked or encoded. Oppositely, an algorithm can have a *decoding* phase where, for example, masked variables can be revealed and encoded variables can be checked for errors. For clarity, an error check in a duplication countermeasure is not part of the decoding phase, only the verification of the final output is. These phases are important in the security models of Sect. 2.3 and Sect. 3 since the adversaries cannot target these parts of the algorithm.

2.3 Random Probing Model

In this section, we introduce the *random probing model* as originally introduced by Duc *et al.* [13]. More specifically, its adversary and its security model. Later in Sect. 3, we study the main contribution of the work, namely the random fault model, which can be seen as the fault counterpart of the random probing model.

Random Probing Adversary. Consider the set of two functions $\mathcal{N} = \{f_0, f_1\}$ with $f_0 : \mathbb{F}_2 \to \mathbb{F}_2 \cup \{\perp\} : x \mapsto x$ and $f_1 : \mathbb{F}_2 \to \mathbb{F}_2 \cup \{\perp\} : x \mapsto \perp$. Namely, the function which maps a bit to itself and the function which returns nothing ($\perp$). We consider a Bernoulli distribution over $\mathcal{N}$ with mean $0 \leq \varepsilon \leq 1$. Thus, from the set $\mathcal{N}$ we draw the function f_0 with probability ε and the function f_1 with probability $1 - \varepsilon$. In words, when drawing a random function and evaluating a variable, the adversary has an ε probability to view that variable.

We note for clarity that the adversary can **precisely target** the location of the probes, but the probability for each probe to provide a value is random.

In this paper, we only consider random probes over bits. The model is easily generalized to work over larger fields. However, we note that in practice, the adversary never views the leakage of a large field element but, rather, a function of the bit vector (such as its Hamming weight) which was processed. As a result, the link with practice is weaker when a direct generalization of the random probing model is made.

Security Model. Consider an algorithm and denote the set of all its variables (excluding the encoding and decoding phases) by $\mathcal{V}$, the set of ε-*random probes* on $\mathcal{V}$ is the set $\{F(v)|v \in V, F \overset{Bern(\varepsilon)}{\leftarrow} \mathcal{N}\}$ with V the values the variables $\mathcal{V}$ in the algorithm attained with its input and internal randomness, and where for each value an independent random probe is chosen.

The *random probing security model* is the bounded query, left-right security game represented in Fig. 1. The game consists of a challenger picking a random bit b, the challenger then creates an oracle $\mathcal{O}^b$ from the algorithm C and provides this to the adversary $\mathcal{A}$. This adversary is computationally unbounded, but it is bounded in the number of queries to the oracle. The adversary provides two secrets k_0, k_1 (for a cipher, a secret is the plaintext and the key) and the set of variables $\mathcal{V}$ which it wants to probe. The oracle then picks the secret k_b, generates its internal randomness, computes the values V on $\mathcal{V}$, and provides the random probing leakage to the adversary. After q queries (for ease, in this work $q = 1$), the adversary guesses the bit b which was chosen by the challenger.

$$\begin{tikzcd}
\begin{array}{c}\mathcal{C}\\ b \leftarrow \$\end{array} \arrow[r, shift left, "\mathcal{O}^b"] & \mathcal{A} \arrow[l, shift left, "b"] \arrow[dr, shift left, "{k_0, k_1, \mathcal{V}}"] & \\
& & \mathcal{O}^b \arrow[ul, shift left, "{\{F(v)|v \in V, F \overset{Bern(\epsilon)}{\leftarrow} \mathcal{N}\}}"]
\end{tikzcd}$$

Fig. 1. The random probing leakage model.

The advantage of $\mathcal{A}$ is defined as

$$\mathrm{Adv}(\mathcal{A}) = |\ \Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1]\ | \,.$$

3 Random Fault Model

We propose a novel adversary model inspired by the previous explained random probing model where the adversary can fault the entire state but there is a limited probability for each fault to occur (following the mechanisms of statistical fault analysis [14]). This random fault security model is novel and is meant to improve over the standard threshold fault model where an adversary can fault a fixed number of values.

We provide a combined adversary model from the random probing and random fault adversary in Appendix A. However, we leave the security evaluation of the combined model of countermeasures for future work.

3.1 Random Fault Adversary

Consider a function $g : \mathbb{F}_2 \to \mathbb{F}_2$ which is the fault the adversary wants to inject. Denote the set of functions $\mathcal{F}_g = \{g, id\}$, with $id : \mathbb{F}_2 \to \mathbb{F}_2 : x \mapsto x$, and consider a Bernoulli distribution $Bern(\kappa)$ on the set to take a random function F. For this random function F, we have that

$$F(x) = \begin{cases} g(x) \text{ with probability } \kappa\,, \\ x \text{ with probability } 1-\kappa\,. \end{cases}$$

Considering the possible fault injections g an adversary can make over bits, there are three possibilities.

- *bitflip*: $\mathbb{F}_2 \to \mathbb{F}_2 : x \mapsto x + 1\,.$
- *set to zero*: $\mathbb{F}_2 \to \mathbb{F}_2 : x \mapsto 0\,.$
- *set to one*: $\mathbb{F}_2 \to \mathbb{F}_2 : x \mapsto 1\,.$

In Sect. 5 and Sect. 6, we use the above function names to indicate which fault is injected. We note for clarity that the adversary can **precisely target** the faults (over bits), but the probability for the fault to occur is random.

We only consider fault over bits and not over larger fields. Since our hardware and software works over bits and not over abstract algebraic structures, faults would hit separate gates or wires causing them to affect bit-by-bit. We leave the generalization of using realistic distributions of fault attacks in the random fault model over words (larger sets of bits) as an open problem.

Similar to the reduction from the random probing adversary to a threshold probing adversary proven by Duc *et al.* [13], the random fault adversary reduces to a threshold fault adversary which we prove in Appendix B.

3.2 Security Model: Correctness

The random fault model has two security models, namely a correctness and a privacy model (after the models by Ishai *et al.* [17]). The goal of the adversary in the correctness game is for the algorithm to output a wrong value (different than the value in case no faults were injected). In case the algorithm detected a fault and aborts, the adversary does not win the game. A fault attack covered by this model is for example differential fault analysis by Biham and Shamir [5]. However, the correctness model is more general as it does not look whether secret information can be gathered from an incorrect output.

Consider an algorithm and denote the set of the variables the adversary targets (excluding the encoding and decoding phases) by $\mathcal{V}$. Denote the set of κ*-random faults* by a set of functions $\mathcal{G} = \{g_i \mid i \in \mathcal{V}\}$.

The correctness game of the random fault model consists of an adversary querying the oracle implementing the algorithm providing it with the input secret k and the set of faults $\mathcal{G}$. The oracle then implements the algorithm with the secret k faulting its values with random functions $\mathcal{F}_\mathcal{G}$ following a $Bern(\kappa)$ distribution. The oracle outputs 1 if the output was correct or abort $\perp$, and 0 if the output was incorrect. This is depicted in Fig. 2. We require, for the algorithm to be useful, that it outputs a correct result in case no faults were present.

$$\begin{tikzcd}\mathcal{A} \arrow[r, shift left, "{k, \mathcal{F}_\mathcal{G}}"] & \mathcal{O} \arrow[l, shift left, "b"] \arrow[loop right] \end{tikzcd}$$

$$\text{Apply } \{G_i \overset{Bern(\kappa)}{\longleftarrow} \mathcal{F}_{g_i} | g_i \in \mathcal{G}\}$$

Fig. 2. The correctness game of the random fault model.

The advantage is the probability the oracle outputs zero after a single query.

$$\text{Adv}(\mathcal{A}) = \Pr[\mathcal{O}(k, \mathcal{F}_\mathcal{G}) = 0]$$

3.3 Security Model: Privacy

The goal of the adversary in the privacy game is to uncover internal information (the input) of the algorithm from the algorithm's abort state after faulting it. In case the algorithm does not have an abort signal (such as with masking or error correction), the countermeasure is automatically secure in the privacy model and its protection is solely determined by the correctness model. Examples of fault attacks in the privacy model include Clavier's ineffective faults [8] and statistical ineffective faults by Dobraunig *et al.* [12].

The privacy game of the *random fault model* is the bounded query, left-right security game represented in Fig. 3. The game consists of a challenger picking a random bit b, the challenger then creates an oracle $\mathcal{O}^b$ from the algorithm and provides this to the adversary $\mathcal{A}$. This adversary is computationally unbounded, but it is bounded in the number of queries to the oracle. The adversary provides two secrets k_0, k_1 together with a set of functions on the values (excluding encoding and decoding phases) $\mathcal{G} = \{g_i \mid i \in \mathcal{V}\}$. The oracle then picks the secret

k_b, generates its internal randomness, computes the algorithm, and applies the random fault functions $\mathcal{F}_\mathcal{G}$ on the targeted variables $\mathcal{V}$ (following a $Bern(\kappa)$ distribution). The oracle returns the state of the abort signal of the algorithm. After q queries (for ease, in this work $q = 1$), the adversary returns the bit b which was chosen by the challenger.

$$\begin{tikzcd}
\begin{array}{c}\mathcal{C}\\ b \leftarrow \$\end{array} \arrow[rr, shift left, "\mathcal{O}^b"] & & \mathcal{A} \arrow[ll, shift left, "b"] \arrow[dr, shift left, "{k_0, k_1, \mathcal{F}_\mathcal{G}}"] & \\
& & & \mathcal{O}^b \arrow[ul, shift left, "\perp"] \arrow[loop right] \\
& & & \text{Apply } \{G_i \overset{Bern(\kappa)}{\leftarrow} \mathcal{F}_{g_i} | g_i \in \mathcal{G}\}
\end{tikzcd}$$

Fig. 3. The privacy game of the random fault model.

The advantage of $\mathcal{A}$ is defined as

$$\mathrm{Adv}(\mathcal{A}) = |\ \Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1]\ |\ .$$

4 Case Studies: Random Probing Model

In order to showcase the random fault model, we apply it to several popular countermeasures with the goal to find general bounds over the parameters of the countermeasure. However, in order to properly provide the security of each countermeasure, we also evaluate them over the established random probing model (as introduced in Sect. 2.3). For example, to show that masking might not improve the bound over the random fault model, but that it does increase the security in the random probing model.

Although the random probing model has already been established for some time, some of the results in this section are novel. For example, as far as we are aware, no concrete random probing bounds have been given for duplicate-and-mask countermeasures. In addition, we show that shuffling methods do not significantly improve the security over the random probing model.

4.1 Influence of Duplication

We investigate what happens if you view the same variable m times (for example if the value is duplicated to defend against fault attacks) or when the variable is an element of $\mathbb{F}_{2^m}$.

Consider a uniform random variable $X \in \mathbb{F}_2$ ($\Pr[X = x] = 1/2$) and m independent random probes $F_0, ..., F_{m-1}$ taken from the set $\mathcal{N}$ following a $Bern(\varepsilon)$

distribution. Then, the probability that at least one probe views a value is $1-(1-\varepsilon)^m$. Thus, the advantage of a random probing adversary is

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= |\ \Pr[\mathcal{A}^{\mathcal{O}^0}=1]-\Pr[\mathcal{A}^{\mathcal{O}^1}=1]\ | \\ &= 1-(1-\varepsilon)^m \leq m\varepsilon\,,\end{aligned}$$

where the last inequality is Bernoulli's inequality.

It is the above observed gain in advantage when viewing the same variable multiple times which causes horizontal attacks [9] to be effective.

4.2 Influence of Masking

Masking was introduced by Goubin and Patarin [15] and Chari *et al.* [7] in 1999. The definition for Boolean masking is given as follows.

Definition 1 (Boolean masking). *The n-shared Boolean masking of a variable $x \in \mathbb{F}_2$ consists of a vector $(x^0,\dots,x^{n-1}) \in (\mathbb{F}_2)^n$ such that $x=\sum_{i=0}^{n-1} x^i$.*

In a countermeasure, a share vector is made using random bits. For example, to mask a secret x in two shares one can use a random bit r and create the vector $(x+r,r)=(x^0,x^1)$. That way, each share x^0 or x^1 is uniform random.

We start by showing that masking indeed improves the protection against a random probing adversary. Given a uniform value X^0 and X^1 such that $X^0+X^1=0$ and two independent random probing functions F_0 and F_1 with probability ε to observe the value, then

$$\begin{aligned}\Pr[F_0=f_{id},F_1=f_{id}]\Pr[X^0\oplus X^1=0|F_0(X^0),F_1(X^1)] &= \varepsilon^2\\ \Pr[F_0=f_{\perp},F_1=f_{id}]\Pr[X^0\oplus X^1=0|F_0(X^0),F_1(X^1)] &= \varepsilon(1-\varepsilon)/2\\ \Pr[F_0=f_{\perp},F_1=f_{\perp}]\Pr[X^0\oplus X^1=0|F_0(X^0),F_1(X^1)] &= (1-\varepsilon)^2/2\,,\end{aligned}$$

with $f_{id}: x\mapsto x$ and $f_\perp: x\mapsto\perp$. Similarly,

$$\begin{aligned}\Pr[F_0=f_{id},F_1=f_{id}]\Pr[X^0\oplus X^1=1|F_0(X^0),F_1(X^1)] &= 0\\ \Pr[F_0=f_{\perp},F_1=f_{id}]\Pr[X^0\oplus X^1=1|F_0(X^0),F_1(X^1)] &= \varepsilon(1-\varepsilon)/2\\ \Pr[F_0=f_{\perp},F_1=f_{\perp}]\Pr[X^0\oplus X^1=1|F_0(X^0),F_1(X^1)] &= (1-\varepsilon)^2/2\,,\end{aligned}$$

As a result, the advantage of the adversary is ε^2 (the absolute subtraction of the three corresponding equations). Similarly, for n shares, the adversary only guesses correctly when all random probes return a value which happens with probability ε^n (which is then the advantage).

4.3 Influence of Masked Duplication

Consider the case where a variable is both masked and duplicated. In practice, this happens when we require both fault protection and side-channel security. We distinguish two cases.

- Mask-then-duplicate: A variable is first masked and then duplicated. For two shares and two duplicates, this means a bit $x \in \mathbb{F}_2$ is encoded to $(x_0^0, x_0^1), (x_1^0, x_1^1)$ where $x_0^0 + x_0^1 = x$ and $x_0^0 = x_1^0$, $x_0^1 = x_1^1$. Examples of countermeasures which use this technique include [11,16].
- Duplicate-then-mask: A variable is first duplicated and then masked. For two shares and two duplicates, this means a bit $x \in \mathbb{F}_2$ is encoded to $(x_0^0, x_1^1, x_2^0, x_3^1)$ where $x_0^0 + x_1^1 = x_2^0 + x_3^1 = x$. Examples of countermeasures which use this technique include [10,20,21].

Mask-then-duplicate. For the first case with two shares ($n = 2$) and two duplicates ($k = 2$), we have that

$$\mathrm{Adv}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1]| = (1 - (1 - \varepsilon)^2)^2 .$$

Namely, the adversary only gets an advantage if it observes both shares, but since each share is duplicated, observing a single share happens with probability $1 - (1 - \varepsilon)^2$. By combining the advantages for duplication and for masking, for n shares and k duplicates, we have an advantage of

$$\mathrm{Adv}(\mathcal{A}) = (1 - (1 - \varepsilon)^k)^n \leq (k\varepsilon)^n .$$

Duplicate-then-mask. For the second case with $n, k = 2$, we have that

$$\mathrm{Adv}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1]| = 1 - (1 - \varepsilon^2)^2 .$$

Namely, the adversary has a ε^2 advantage to when observing a masking and the adversary has two chances to break it. For n shares and k duplicates, we have a random probing advantage of

$$\mathrm{Adv}(\mathcal{A}) = 1 - (1 - \varepsilon^n)^k \leq k\varepsilon^n .$$

We observe that the duplicate-then-mask method protects better against a random probing adversary compared to the mask-then-duplicate method. We depict the differences between the advantages in Fig. 4.

4.4 Influence of Shuffling

We take a look at shuffling as a countermeasure and assess its security in the random probing model. We then move to the shuffling of masked values.

Consider two values $x, y \in \mathbb{F}_2$. With shuffling, the encoding phase of the algorithm randomly shifts the two values from place. Consider the security model from Sect. 2.3. Since the adversary can choose the two secret inputs, we can take $x = y$ (meaning, $x = y = 0$ or $x = y = 1$ for the two cases of inputs). The advantage of the adversary using random probes (denoted F_0 and F_1 and calling the first operation O_0 and the second O_1) is

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= |\Pr[F_0(O_0) = 1 \vee F_1(O_1) = 1 | X = 1]| \\ &= 1 - (1 - \varepsilon)^2 \leq 2\varepsilon .\end{aligned}$$

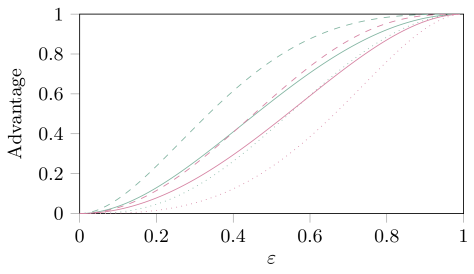

Fig. 4. The random probing advantage of the mask-then-duplicate method (in green) and the duplicate-then-mask method (in red). The full lines depict $(n, k) = (2, 2)$, the dashed lines depict $(n, k) = (2, 3)$, and the dotted lines depict $(n, k) = (3, 2)$.

The above adversary randomly guesses the secret when both random probes return $\perp$ and provides the answer to the probes when a value is returned. The above bound generalizes to $n\varepsilon$ for shuffling n bits.

We observe that the bound of the shuffling method is the same as the bound of an unshuffled n-bit state. As a result, shuffling provides no additional security when the adversary chooses weak inputs.

Shuffled Masking. We consider a state of k-bit n-shared values. For example, for $k = 3$ and $n = 2$, we have a state $(x_0, x_1), (y_0, y_1), (z_0, z_1)$ such that $x = x_0 + x_1$, $y = y_0 + y_1$, $z = z_0 + z_1$ for three bits $x, y, z \in \mathbb{F}_2$. Consider then that the total of nk bits are randomly permuted, meaning that a permutation is uniform randomly drawn for the set of all $\mathbb{F}_2^{nk} \to \mathbb{F}_2^{nk}$ permutations and is applied to the nk bits. We note that practical implementations of shuffling often consider a weaker case where the permutation is drawn from a smaller set, such as cyclical shifts of the shares. The security mentioned in related works such as [2, Section 2.4] is that shuffling these masked values at least improves the side-channel security by a factor k. We will show that in the random probing model, shuffling does not significantly improve the security.

We consider the advantage of random probing all the nk shares in the state. Since the order of the shares is shuffled (from the countermeasure), the values returned to the adversary are also randomized. As a result, the adversary does not know which value belongs to which variable. For example, the adversary can receive the transcript $(1, 0, 0)$ for $n = 2$ and $k = 3$, meaning that the adversary receives three values out of six but does not know which three it received.

We consider an adversary which takes, for the security model, the all-zero secret versus the all-one secret. Meaning that all k sharings are of either the secret zero (in the first case) or the secret all-one (in the second case). The adversary then considers all possible n-sums of the received values. In case the majority of the sums are zero, the adversary decides it is in the first case (with the secrets all equal to zero), otherwise it decides it is in the second case. We calculate the advantage of this adversary.

The probability that the adversary receives exactly i bits from the random probes is $\binom{nk}{i}\varepsilon^i(1-\varepsilon)^{nk-i}$. Given that the adversary has i values, it can calculate a total of $\binom{i}{n}$ n-sums out of the total of $\binom{nk}{n}$ n-sums. Finally, there are a total of k sums which sum to the secret (we call these "correct sums"). The advantage is thus given as follows.

$$\mathrm{Adv}(\mathcal{A}) = \sum_{i=0}^{nk} \binom{nk}{i} \varepsilon^i (1-\varepsilon)^{nk-i} \sum_{j=0}^{k} C(i,j)K(i,j)\,,$$

with $C(i,j)$ the probability to have j correct sums over the total of $\binom{i}{n}$ sums, and $K(i,j)$ the probability to win the game minus the probability to lose the game given j correct sums over a total of $\binom{i}{n}$ sums.

In more detail,

$$C(i,j) = \frac{\binom{a}{j}\binom{b-a}{k-j}}{\binom{b}{k}}\,,$$

with $a = \binom{i}{n}$ and $b = \binom{nk}{n}$ since $C(i,j)$ is the probability of j successes in the hypergeometric distribution. The value

$$K(i,j) = \sum_{\ell=\lceil \frac{a+1}{2} \rceil - j}^{a-j} \frac{1}{2^{a-j}} \binom{a-j}{\ell} - \sum_{\ell=0}^{\lfloor \frac{a}{2} \rfloor - j} \frac{1}{2^{a-j}} \binom{a-j}{\ell}\,,$$

with $a = \binom{i}{n}$ is the probability that the majority of the n-sums are equal to the secret (given that j sums are correct) minus the probability that the majority of the sums equal zero.

It is clear that an upper bound for the above advantage is equal to the advantage of an k times n-sharing without shuffling, which is equal to $1-(1-\varepsilon^n)^k$. For $n, k = 2$, we also find that the above advantage is equal to $2\varepsilon^2 - 2\varepsilon^3 + \frac{3}{8}\varepsilon^4$, which is lower than the $2\varepsilon^2 - \varepsilon^4$ advantage without shuffling. This shows that shuffling indeed increases the security of an implementation. However, we see that both advantages have a leading coefficient $2\varepsilon^2$. Meaning that the advantage of shuffling equals the advantage without shuffling plus terms in ε^3 or higher. In words, shuffling only achieves a very insignificant (*i.e.* less than linear) increase in random probing security.

We prove this "less than linear" security gain more formally. First, since $C(i,j) = 0$ for $i < n$, we can write $\mathrm{Adv}(\mathcal{A}) = c\varepsilon^n + \mathcal{O}(\varepsilon^{n+1})$. Second, we find that $c = k$ since $c = \binom{nk}{n}C(n,1)K(n,1)$ with $C(n,1) = \frac{k}{\binom{nk}{n}}$ and $K(n,1) = 1$. As a result, we find that $\mathrm{Adv}(\mathcal{A}) = k\varepsilon^n + \mathcal{O}(\varepsilon^{n+1})$ which can be compared to the advantage of the state without shuffling $1-(1-\varepsilon^n)^k = k\varepsilon^n + \mathcal{O}(\varepsilon^{n+1})$. Thus, we have proven that shuffling can only improve the random probing security by a less-than-linear amount when the adversary chooses weak inputs.

We mention that similar results were found by Bogdanov *et al.* [6] on using higher-order differential computational analysis on white-box implementations using masking and shuffling.

5 Case Studies: Random Fault Correctness

We investigate the random fault security of several popular countermeasures including duplication, error correcting codes, masking, multiplicative tags, and shuffling. Recall from Sect. 3.2 that the random fault model has two security models, namely the correctness and the privacy model. In this section, we evaluate the countermeasures in the correctness model.

To better understand the bounds given in this section, we establish a baseline. Namely, we investigate the correctness security of storing a single bit. The advantage of a random fault adversary changing this single bit value is κ. In case there are m variables (or m queries), then the advantage is $1-(1-\kappa)^m \leq m\kappa$.

5.1 Influence of Masking

Consider masking from Sect. 4.2 where a variable $x \in \mathbb{F}_2$ is split in two parts x^0, x^1 such that $x^0+x^1=x$. Then using a random bitflip fault F with probability κ only on x^0, the advantage of the adversary against an n-masking is still

$$\mathrm{Adv}(\mathcal{A}) = \Pr[F(X^0)+X^1 \neq X^0+X^1] = \kappa\,.$$

However, the adversary can bitflip both shares (denoted F_0 and F_1) to attain the advantage $2\kappa(1-\kappa)$. For n shares, this advantage becomes

$$\mathrm{Adv}(\mathcal{A}) = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2i+1} \kappa^{2i+1}(1-\kappa)^{n-2i-1} = \frac{1}{2}(1-(1-2\kappa)^n) \leq n\kappa\,.$$

We observe that, if $\kappa \leq 1/2$, masking increases the advantage of a faulting adversary in the correctness model over a non-masked alternative.

5.2 Influence of Duplication

We then investigate the effect of duplicating the variable and error checking the duplicates at the end of the computation.

Definition 2 (Duplication). *The n-duplication of a variable $x \in \mathbb{F}_2$ consists of a vector $(x_0, \ldots, x_{n-1}) \in (\mathbb{F}_2)^n$ such that $x_0 = \ldots = x_{n-1}$.*

The above is combined with an error check which verifies if $x_i = x_j$ with $i \neq j$ and aborts the computation if they are not equal.

We calculate the advantage of a random fault adversary injecting a *bitflip* (see Sect. 3) in both duplicates for a two-duplication using random faults (F_0, F_1) with probability κ to occur. We find the following advantage

$$\mathrm{Adv}(\mathcal{A}) = \Pr[F_0(X_0) = F_1(X_1), F_0(X_0) \neq X_0] = \kappa^2\,.$$

This is extended to k-duplication with an advantage of κ^k. As a result, duplication (or any linear code with nontrivial distance) exponentially decreases the advantage of the adversary in the correctness game.

For m-bit k-duplicated variables, the advantage of the adversary is still κ^k if the adversary attacks only one pair of duplicates. In case the adversary attacks all duplicates, the advantage becomes

$$\mathrm{Adv}(\mathcal{A}) = \sum_{i=1}^{m} \binom{m}{i} \kappa^{ik}(1-\kappa)^{k(m-i)} = (\kappa^k + (1-\kappa)^k)^m - (1-\kappa)^{km} ,$$

where the equality comes from the binomial theorem. The above advantage is higher compared to attacking one pair of duplicates in case κ is small.

Specific Codes. We consider the advantage for encodings using different linear codes from the repetition (duplication) code. Consider a value $x \in \mathbb{F}_{2^m}$ encoded as a codeword $c \in \mathcal{C}$ with $\mathcal{C}$ and $[n, m, d]$ code. It is clear that if the adversary faults c to the nearest other codeword, the advantage is κ^d.

For a different attack, the adversary bitflips each bit of c where the advantage is the probability they form a codeword. In particular, if $\kappa = 0.5$, one gets a random m-bit fault for which the advantage is $\frac{2^m-1}{2^n}$. This result is, for example, given by Schneider *et al.* [21] where it is called the "fault coverage" of the code. For more accurate results when $\kappa \neq 0.5$, the specific advantage of the adversary depends on the actual code that is used. We provide some examples.

Consider the $[m+1, m, 2]$ parity code (*i.e.* $(x[0], ..., x[m-1], c)$ with $c = \sum_{i=0}^{m-1} x[i]$). The advantage of a random fault adversary is

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= \sum_{i=1}^{\lfloor \frac{m+1}{2} \rfloor} \binom{m+1}{2i} \kappa^{2i}(1-\kappa)^{m-2i+1} \\ &= \frac{1}{2}(1 + (1-2\kappa)^{m+1}) - (1-\kappa)^{m+1} .\end{aligned}$$

For other examples, we need the weight distribution of the codes we are investigating.

– For the $[7, 4, 3]$ Hamming code, the weight distribution, ranging from zero to seven ,of the codewords is $[1, 0, 0, 7, 7, 0, 0, 1]$. As a result, the advantage is $7\kappa^3(1-\kappa)^4 + 7\kappa^4(1-\kappa)^3 + \kappa^7$.
– Similarly, the weight distribution of the $[8, 4, 4]$ extended Hamming code, ranging from zero to eight, is $[1, 0, 0, 0, 14, 0, 0, 0, 1]$. Thus, the advantage is $14\kappa^4(1-\kappa)^4 + \kappa^8$.

The difference in advantages between the codes is shown in the first graph of Fig. 5. From this figure, we find that the number of parity bits have a significant effect on the advantage of a random fault adversary and that not only the minimal distance of the code matters. Note that the "kink" in the graphs is given by the difference of advantages of different attacks.

In the work by Bartkewitz *et al.* [3], experiments were performed by faulting only the message bits in an implementation (leaving the parity bits unaltered).

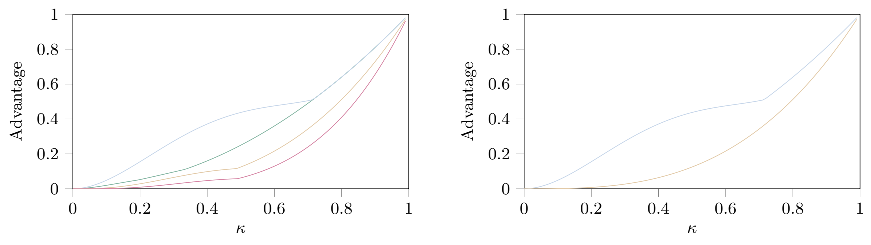

Fig. 5. The advantage of a random fault adversary against encoded values on the left and on the right when only the message bits are attacked. Blue depicts the $[5, 4, 2]$ code, green $[8, 4, 2]$, yellow $[7, 4, 3]$, and red $[8, 4, 4]$. For the right figure, the $[8, 4, 2]$ and $[8, 4, 4]$ codes have advantage zero.

Assuming that a κ-random fault in the message bits was injected (and that indeed the parity bits were unaffected), the result of the advantage for the different codes investigated by Bartkewitz *et al.* is given in the second graph of Fig. 5. The difference between the codes becomes more significant when faulting only the message bits versus faulting all bits in the codeword.

5.3 Influence of Triplication

Consider the duplication method from before, but with a minimum of three duplicates (x_0, x_1, x_2). Instead of using error detection where the algorithm can abort, we correct the errors using a majority voting.

We consider an adversary which bitflips two out of three duplicates of a single bit. This adversary has the following advantage

$$\mathrm{Adv}(\mathcal{A}) = \Pr[F_0(X_0) = F_1(X_1), F_0(X_0) \neq X_0] = \kappa^2 \,.$$

For k duplicates, the advantage would be $\kappa^{\lceil k/2 \rceil}$.

We extend the above analysis by considering m bits. When each variable is duplicated and an error detection method is used, the advantage of the adversary is $(\kappa^2+(1-\kappa)^2)^m-(1-\kappa)^{2m}$ when attacking all variables, and κ^2 when attacking one pair of duplicates. However, with error correction, the advantage against an m-bit three-duplicate correction method becomes

$$\mathrm{Adv}(\mathcal{A}) = 1 - (1 - \kappa^2)^m \,,$$

as the adversary can re-try the attack with each variable and win when one of the m variables is error-corrected to the wrong output. The advantage is depicted in Fig. 6. We find that triplication performs significantly worse when faults can target a large state size compared to duplication. We note that the combination of Boolean masking with triplication would not improve the advantage.

5.4 Influence of Masked Duplication

Recall the two methods of both masking and duplicating variables from Sect. 4.3.

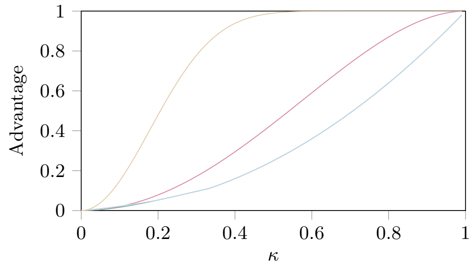

Fig. 6. The advantage against error correction (with three duplicates) is shown in red (for $m = 2$) and yellow ($m = 16$). The advantage against error detection (with two duplicates) is shown in green (for $m = 2$) and blue ($m = 16$).

Mask-then-duplicate. We consider the first case where each share is duplicated. If the adversary only bitflips one pair of duplicates, the advantage is κ^2 (or κ^k for k duplicates). In case the adversary bitflips all values (denoting the random faults F_0, F_1, F_2, F_3), the advantage for $n, k = 2$ is

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= \Pr[F_0(X_0^0) = F_1(X_1^0), F_2(X_0^1) = F_3(X_1^1), F_0(X_0^0) + F_2(X_0^1) \neq X] \\ &= 2\kappa^2(1-\kappa)^2 \,.\end{aligned}$$

Given that the probability to break the correctness of a k-duplication is κ^k and the probability to leave each duplicate unchanged is $(1-\kappa)^k$, the advantage with n shares and k duplicates becomes

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} \binom{n}{2i+1} \kappa^{k(2i+1)}(1-\kappa)^{k(n-2i-1)} \\ &= \frac{1}{2}((1-\kappa)^k + \kappa^k)^n - ((1-\kappa)^k - \kappa^k)^n \,.\end{aligned}$$

Duplicate-then-mask. Recall that for the second case, the variable is first duplicated and then each duplicate is shared separately. When attacking only one share per duplicate, the advantage is κ^k. When attacking all variables (denoting the random faults F_0, F_1, F_2, F_3), for $n, k = 2$, the advantage becomes

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= \Pr[F_0(X_0^0) + F_1(X_0^1) = F_2(X_2^0) + F_3(X_3^1), F_0(X_0^0) + F_1(X_0^1) \neq X] \\ &= 4\kappa^2(1-\kappa)^2 \,.\end{aligned}$$

For n shares and k duplicates, the probability to change the correctness of an n-sharing is $\frac{1}{2}(1-(1-2\kappa)^n)$, so we have

$$\mathrm{Adv}(\mathcal{A}) = 2^{-k}(1-(1-2\kappa)^n)^k \,.$$

We observe that for small parameters κ, the mask-then-duplicate method provides more security as opposed to the duplicate-then-mask method. This is depicted for small variables n, k in Fig. 7.

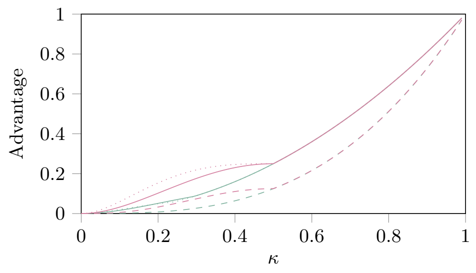

Fig. 7. The random fault advantage in the correctness model of the mask-then-duplicate method (in green) and the duplicate-then-mask method (in red). The full lines depict $(n, k) = (2, 2)$, the dashed lines depict $(n, k) = (2, 3)$, and the dotted lines depict $(n, k) = (3, 2)$.

5.5 Influence of Multiplicative Tags

We can encode variables against fault attacks by multiplying the duplicate with a random value. We call this a multiplicative tag.

Definition 3 (Multiplicative Tag). *A multiplicative tag of $x \in \mathbb{F}_{2^m}$ is a value $\alpha x \in \mathbb{F}_{2^m}$ with $\alpha \in \mathbb{F}_{2^m}$ chosen uniformly random with each query.*

Consider the encoding of $x \in \mathbb{F}_{2^m}$ with a multiplicative tag $(x, \alpha x) \in (\mathbb{F}_{2^m})^2$ and $\alpha \in \mathbb{F}_{2^m}$ chosen randomly with every query. Error detection is performed by taking the message x, multiplying it with the tag α, and verifying it against the duplicate αx. We investigate the security of this method in the correctness game with a random fault adversary.

We consider a first adversary which changes a single bit of x assuming $\alpha = 0$. Consider F a random fault flipping the first bit of x. Then, the advantage is

$$\mathrm{Adv}(\mathcal{A}) = \Pr[F(X_0) \neq X_0, \alpha = 0] = 2^{-m}\kappa\,.$$

For a second adversary, we take $x = 1$ and fault both x and αx with set-to-zero faults. We number the bits of x by $(x[0], ..., x[m-1])$ and the bits of αx by $(\alpha x[0], ..., \alpha x[m-1])$. Then, the advantage of the adversary applying random faults $F_0, ..., F_m$ against an m-bit multiplicative tag is given as follows

$$\begin{aligned}\mathrm{Adv}(\mathcal{A}) &= \Pr[F_0(X[0]) = 0, F_1(\alpha X[0]) = 0, ..., F_m(\alpha X[m-1]) = 0] \\ &= \kappa \left(\frac{1+\kappa}{2}\right)^m .\end{aligned}$$

While, in this work, we are not able to provide bounds for general adversaries, we observe that multiplicative tags provide a promising countermeasure against faults compared to using linear codes from Sect. 5.2.

5.6 Influence of Shuffling

Consider the shuffling countermeasure from Sect. 4.4. Without duplication, the adversary can bitflip each value for the same advantage as in the non-shuffled case. Since there is no detection step, as long as one fault hits, the adversary wins.

In case duplication is used on top of the shuffling, similar to the weak input attack in Sect. 4.4, there are weak inputs in the correctness model. Namely, for two two-duplicated bits $(x_0, y_0), (x_1, y_1)$, pick the secret $(0, 1)$. By applying a set-to-zero fault on all values, only one variable can change in its value providing the same advantage as in the non-shuffled case where the adversary only targets one pair of duplicates. Moreover, recall from Sect. 5.2 that when k is small, the best attack of the adversary is to fault all duplicates. Such an attack has the same probability to break correctness when shuffling the duplicates. As a result, for small κ, shuffling does not improve security in the random fault model (independent of how the shuffling is done). Together with the weak inputs when attacking only one pair of duplicates, we conclude that we cannot find a non-trivial upper bound on the security of shuffling in the correctness model.

6 Case Studies: Random Fault Privacy

In Sect. 5, we investigated the correctness security of several countermeasures. In this section, we investigate the privacy security (from Sect. 3.3). Recall that the privacy model is only relevant to countermeasures which can abort the computation. Therefore, we do not investigate countermeasures such as masking.

6.1 Influence of Duplication

Consider the duplication method from Sect. 5.2 (the advantage is similar when using multiplicative tags from Sect. 5.5). A privacy adversary (taking $x = 0$ and $x = 1$) faulting one variable to zero with probability κ has an advantage

$$\mathrm{Adv}(\mathcal{A}) = |\ \Pr[\mathcal{A}^{\mathcal{O}^0} = \perp] - \Pr[\mathcal{A}^{\mathcal{O}^1} = \perp]\ | = \kappa\,.$$

This is because, when $x = 0$, the countermeasure can never abort or, when $x = 1$, it aborts with probability κ. The bound for m-bit variables (or viewing the variable m times) is $1 - (1 - \kappa)^m \leq m\kappa$. Similar to probing the same variable multiple times, it is advantageous to fault the same variable multiple times causing a "horizontal"-like attack to be possible. We provide examples of such attacks on a mask-then-duplicated multiplier in Appendix C.

In case the adversary faults all k duplicates to zero, the advantage becomes

$$\mathrm{Adv}(\mathcal{A}) = \sum_{i=1}^{k-1} \binom{k}{i} \kappa^i (1 - \kappa)^{k-i} = 1 - \kappa^k - (1 - \kappa)^k\,.$$

6.2 Influence of Masked Duplication

Similar to Sect. 5.4, we consider a variable which is both masked and duplicated.

Mask-then-duplicate. Consider a masked and encoded value $(x_0^0, x_0^1, x_1^0, x_1^1)$ where the shares are duplicated. When faulting both x_0^0 and x_0^1 to zero, we get

$$\mathrm{Adv}(\mathcal{A}) = |\ \Pr[\mathcal{A}^{\mathcal{O}^0} = \perp] - \Pr[\mathcal{A}^{\mathcal{O}^1} = \perp]\ | = \kappa - (1-(1-\kappa)^2)/2 = \kappa^2/2\,.$$

When considering n shares, the probability to change at least one share out of n (due to a set fault) when the sharing has secret zero κ_0 or secret one κ_1 is

$$\kappa_0 = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} 2^{1-n}\binom{n}{2i}(1-(1-\kappa)^{2i}), \quad \kappa_1 = \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} 2^{1-n}\binom{n}{2i+1}(1-(1-\kappa)^{2i+1})\,.$$

The advantage of the above attack is $|\,\kappa_0 - \kappa_1\,| = 2^{1-n}\kappa^n$.

When faulting $(x_0^0, x_0^1, x_1^0, x_1^1)$ all to zero, the advantage becomes $2\kappa^2(1-\kappa)^2$. For small κ, the advantage is higher when faulting all duplicates and shares. For the bound when faulting all k duplicates and n shares is

$$\mathrm{Adv}(\mathcal{A}) = 2^{1-n}(1-(1-\kappa)^k - \kappa^k)^n,$$

since the advantage when faulting a k-duplication is $1-(1-\kappa)^k - \kappa^k$.

Duplicate-then-mask. Consider the masking $(x_0^0, x_1^1, x_2^0, x_3^1)$ such that $x_2^0 + x_3^1 = x_0^0 + x_1^1$. When faulting x_0^0 and x_1^1 both to zero, we have an advantage

$$\mathrm{Adv}(\mathcal{A}) = |\ \Pr[\mathcal{A}^{\mathcal{O}^0} = \perp] - \Pr[\mathcal{A}^{\mathcal{O}^1} = \perp]\ | = \kappa^2\,.$$

For n shares, this attack generalizes to the advantage κ^n. When faulting all bits $(x_0^0, x_1^1, x_2^0, x_3^1)$ to zero, the advantage is $2\kappa^2(1-\kappa)^2$. When investigating the advantage for n shares and k duplicates, when faulting all shares to zero, the probability to change the value of the secret when it is equal to zero is

$$\begin{aligned}\kappa_0 &= \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} 2^{1-n}\binom{n}{2i}\left(\sum_{j=0}^{i-1}\binom{2i}{2j+1}\kappa^{2j+1}(1-\kappa)^{2i-2j-1}\right)\\ &= \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} 2^{-n}\binom{n}{2i}(1-(1-2\kappa)^{2i}) = 1/2(1-(1-\kappa)^n - \kappa^n)\,,\end{aligned}$$

where the equalities are derived from the binomial expansion theorem. Similarly, the probability to change the secret of a sharing of one is

$$\begin{aligned}\kappa_1 &= \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} 2^{1-n}\binom{n}{2i+1}\left(\sum_{j=0}^{i}\binom{2i+1}{2j+1}\kappa^{2j+1}(1-\kappa)^{2i-2j}\right)\\ &= 1/2(1-(1-\kappa)^n + \kappa^n)\,.\end{aligned}$$

Then, when faulting all shares in the duplication, the probability to abort for secret zero is $1 - \kappa_0^k - (1 - \kappa_0)^k$. Similarly, for secret one the probability is $1 - \kappa_1^k - (1 - \kappa_1)^k$. Thus, the advantage against n shares and k duplicates is

$$\mathrm{Adv}(\mathcal{A}) = |\, \kappa_0^k - \kappa_1^k + (1 - \kappa_0)^k - (1 - \kappa_1)^k \,| \,.$$

We observe that the mask-then-duplicate method scales better for higher parameters n and the duplicate-then-mask method scales better for higher parameters k (with the mask-then-duplicate method performing better for equal parameters n, k) as depicted in Fig. 8.

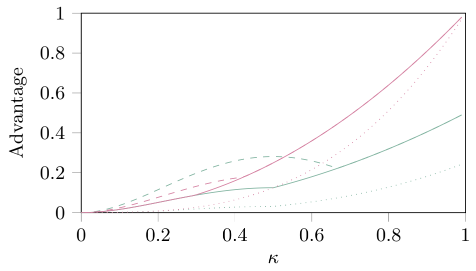

Fig. 8. The privacy advantage of the mask-then-duplicate method (in green) and the duplicate-then-mask method (in red). The full lines depict $(n, k) = (2, 2)$, the dashed lines depict $(n, k) = (2, 3)$, and the dotted lines depict $(n, k) = (3, 2)$.

6.3 Influence of Shuffling

Consider the shuffling method from Sect. 4.4 but with duplicated values. Similar to Sect. 4.4 and Sect. 5.6, there are weak inputs in the privacy model. Namely, when shuffling $(x, y) \in \mathbb{F}_2^2$ and taking the two secrets $x = y$, shuffling becomes obsolete as the same values are shuffled. Moreover, the same attack described in Sect. 6.1 still applies. Namely, to attack all duplicates with a set-to-zero fault between the all-zero and all-one secrets. As a result, shuffling with duplication does not improve the random fault security in the privacy model. Moreover, when using a masking countermeasure, for small parameters κ the best attack is to fault all shares and duplicates with the same fault. The advantage of this attack would not change when shuffling the values.

Together with the weaknesses found in the correctness model in Sect. 5.6, we conclude that shuffling against fault attacks exhibits several weaknesses. The vulnerabilities found in this work directly apply to the Rocky countermeasure [19] which we show is weak in both the correctness and privacy models of the random fault model for certain parameters κ and for certain weak inputs. In addition with the weaknesses found for shuffling in the random probing model from Sect. 4.4, we do not believe shuffling can provide a significant improvement for security against either random probing or random fault adversaries.

7 Conclusion

In this work, we proposed a new fault adversary model called *the random fault model* which is a model inspired from the known random probing model. The goal of the work was to investigate and compare different countermeasures and observe which can promise good security against fault attacks. To make this comparison properly, we also investigated the countermeasures in the random probing model.

Most results in the random probing model are intuitive. Masking provides exponential protection in the number of shares and duplication linearly decreases security in the number of duplicates. However, we did observe a difference between the security of first duplicating a variable and masking each duplicate versus masking a variable and duplicating each share which, to the best of our knowledge, had not been investigated before. One surprising result was that both shuffling or shuffling masked algorithms does not significantly increase the security in the random probing model. This result is quite significant since it holds no matter how the shuffling is performed.

In the random fault model, we found that encoding techniques such as duplication exponentially improves the security in the minimal distance of the code. But we also observed that the number of parity bits has an influence to the countermeasure's security which provides a theoretical explanation for the practical results by Bartkewitz *et al.* [3]. Interestingly enough, we also find that triplication methods (error correction) are significantly less secure than duplication methods (error detection) when the adversary faults several bits. Since there is a lot of work on error correction methods, it remains a question whether these works can provide any good security in a formal security model and we are left with the open question of investigating their security in practice. Moving to masking, we showed that masking reduces the security of a countermeasure against fault attacks and, similar to results in the random probing model, that there is a trade-off in security between duplicate-then-masking a variable versus masking-then-duplicating one. Finally, similar to the results in the random probing model, we show that shuffling does not improve a countermeasure's security against fault attacks which implies that the countermeasure Rocky [19] does not currently provide a theoretical foundation for its security.

While we investigated several countermeasures in both the random probing and random fault model, we leave some open problems for future work.

– We investigated the random probing and random fault security of countermeasures, but we leave the combined security analysis from Appendix A.
– The application of the random fault model to masked or encoded operations such as multipliers following the observations from Appendix C.
– The random probing or random fault analysis of other masking techniques such as multiplicative masking, arithmetic masking, or prime field masking.

Acknowledgements. A special thanks to Vincent Rijmen for the helpful discussions. This work was supported by CyberSecurity Research Flanders with reference number VR20192203.

References

1. Aghaie, A., Moradi, A., Rasoolzadeh, S., Shahmirzadi, A.R., Schellenberg, F., Schneider, T.: Impeccable circuits. IEEE Trans. Computers **69**(3), 361–376 (2020). https://doi.org/10.1109/TC.2019.2948617, https://doi.org/10.1109/TC.2019.2948617
2. Azouaoui, M., Bronchain, O., Grosso, V., Papagiannopoulos, K., Standaert, F.: Bitslice masking and improved shuffling: How and when to mix them in software? IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(2), 140–165 (2022)
3. Bartkewitz, T., Bettendorf, S., Moos, T., Moradi, A., Schellenberg, F.: Beware of insufficient redundancy an experimental evaluation of code-based FI countermeasures. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(3), 438–462 (2022)
4. Battistello, A., Coron, J., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: CHES. Lecture Notes in Computer Science, vol. 9813, pp. 23–39. Springer (2016)
5. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO'97. Lecture Notes in Computer Science, vol. 1294, pp. 513–525. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 1997). https://doi.org/10.1007/BFb0052259
6. Bogdanov, A., Rivain, M., Vejre, P.S., Wang, J.: Higher-order DCA against standard side-channel countermeasures. In: Polian, I., Stöttinger, M. (eds.) Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11421, pp. 118–141. Springer (2019). https://doi.org/10.1007/978-3-030-16350-1_8, https://doi.org/10.1007/978-3-030-16350-1_8
7. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_26
8. Clavier, C.: Secret external encodings do not prevent transient fault analysis. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2007. Lecture Notes in Computer Science, vol. 4727, pp. 181–194. Springer, Heidelberg, Germany, Vienna, Austria (Sep 10–13, 2007). https://doi.org/10.1007/978-3-540-74735-2_13
9. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 10: 12th International Conference on Information and Communication Security. Lecture Notes in Computer Science, vol. 6476, pp. 46–61. Springer, Heidelberg, Germany, Barcelona, Spain (Dec 15–17, 2010)
10. De Meyer, L., Arribas, V., Nikova, S., Nikov, V., Rijmen, V.: M&M: Masks and macs against physical attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(1), 25–50 (2018). https://doi.org/10.13154/tches.v2019.i1.25-50, https://tches.iacr.org/index.php/TCHES/article/view/7333
11. Dhooghe, S., Nikova, S.: My gadget just cares for me - how NINA can prove security against combined attacks. In: Jarecki, S. (ed.) Topics in Cryptology – CT-RSA 2020. Lecture Notes in Computer Science, vol. 12006, pp. 35–55. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 24–28, 2020). https://doi.org/10.1007/978-3-030-40186-3_3

12. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: SIFA: Exploiting ineffective fault inductions on symmetric cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 547–572 (2018). https://doi.org/10.13154/tches.v2018.i3.547-572, https://tches.iacr.org/index.php/TCHES/article/view/7286
13. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 423–440. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_24
14. Fuhr, T., Jaulmes, É., Lomné, V., Thillard, A.: Fault attacks on AES with faulty ciphertexts only. In: Fischer, W., Schmidt, J. (eds.) 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013. pp. 108–118. IEEE Computer Society (2013). https://doi.org/10.1109/FDTC.2013.18, https://doi.org/10.1109/FDTC.2013.18
15. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Çetin Kaya., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES'99. Lecture Notes in Computer Science, vol. 1717, pp. 158–172. Springer, Heidelberg, Germany, Worcester, Massachusetts, USA (Aug 12–13, 1999). https://doi.org/10.1007/3-540-48059-5_15
16. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 308–327. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006). https://doi.org/10.1007/11761679_19
17. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003). https://doi.org/10.1007/978-3-540-45146-4_27
18. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_25
19. Miteloudi, K., Batina, L., Daemen, J., Mentens, N.: ROCKY: rotation countermeasure for the protection of keys and other sensitive data. In: Orailoglu, A., Jung, M., Reichenbach, M. (eds.) Embedded Computer Systems: Architectures, Modeling, and Simulation - 21st International Conference, SAMOS 2021, Virtual Event, July 4-8, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13227, pp. 288–299. Springer (2021). https://doi.org/10.1007/978-3-031-04580-6_19, https://doi.org/10.1007/978-3-031-04580-6_19
20. Reparaz, O., Meyer, L.D., Bilgin, B., Arribas, V., Nikova, S., Nikov, V., Smart, N.P.: CAPA: the spirit of beaver against physical attacks. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10991, pp. 121–151. Springer (2018). https://doi.org/10.1007/978-3-319-96884-1_5, https://doi.org/10.1007/978-3-319-96884-1_5

21. Schneider, T., Moradi, A., Güneysu, T.: Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9815, pp. 302–332. Springer (2016)
22. Shahmirzadi, A.R., Rasoolzadeh, S., Moradi, A.: Impeccable circuits II. In: 57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020. pp. 1–6. IEEE (2020). https://doi.org/10.1109/DAC18072.2020.9218615, https://doi.org/10.1109/DAC18072.2020.9218615

A The Random Combined Model

We consider the security model where the random probing and random fault adversaries are combined, meaning an adversary that can both fault and probe the algorithm. Although, in this work, we do not analyze the security of any countermeasures against the combined adversary, we wish to provide the combined model to support future work.

A.1 Random Combined Adversary

The random combined adversary consists of the natural combination of the random probe and random fault adversary. Namely, this is an adversary which can place a random probe and a random fault on each variable in an algorithm. However, we note that the random probe and the fault act independent of each other. Meaning, while a random probe has an ε probability to view a variable, the random fault has a κ probability to be applied, and the values ε and κ do not influence each other. For consistency's sake, we consider the random probe is applied first and the random fault second. Meaning that the random probe observes the unaltered value in the algorithm before the fault is applied (placing a probe on the next operation using the variable would view the altered value).

To remain in line with the random probing and random fault model, we consider that an adversary has to place all its random probes and random faults before the algorithm is run and the adversary receives the results from the random probes when the algorithm is finished. When considering implementations closer to reality, adaptive adversaries, which can place faults depending on what previous probes returned, could be considered.

A.2 Security Model: Correctness

Similar to the random fault model, the combined model also consists of two security games, a correctness game and a privacy game. We explain the correctness game first.

Since the adversary has to place all random faults before the random probes return any information, the correctness game for a random combined adversary is the same as for a random fault adversary. As a result, the advantage is the same as with the correctness game of the random fault model.

A.3 Security Model: Privacy

The privacy game of the random combined model is similar to the random probing model and the privacy game of the random fault model. We depict the game in Figure 9. The game consists of a challenger picking a random bit b, the challenger then creates an oracle $\mathcal{O}^b$ from the algorithm and provides this to the adversary $\mathcal{A}$. This adversary is computationally unbounded, but it is bounded in the number of queries to the oracle. The adversary provides two secrets k_0, k_1 together with a set of variables of the algorithm $\mathcal{V}_p$ to probe and a set of functions (excluding the encoding and decoding phases) $\mathcal{G} = \{g_i \mid i \in \mathcal{V}_f\}$ to fault. The oracle then picks the secret k_b, generates its internal randomness, computes the algorithm one variable at a time, samples a random probe on the values V from the variables $\mathcal{V}_p$, from $Bern(\varepsilon)$, to view them and (in case the adversary specified so) applies a random fault, from $Bern(\kappa)$. The oracle returns the state of the abort signal of the algorithm together with the values returned by the random probes. After q queries, the adversary returns the bit b which was chosen by the challenger.

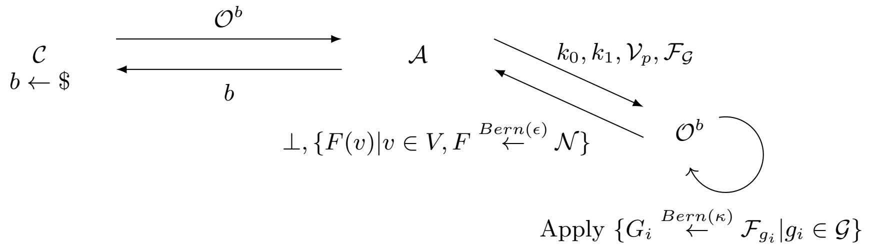
$\mathcal{C}$
$b \leftarrow \$$
$\mathcal{O}^b$
b
$\mathcal{A}$
$k_0, k_1, \mathcal{V}_p, \mathcal{F}_{\mathcal{G}}$
$\perp, \{F(v) | v \in V, F \overset{Bern(\epsilon)}{\leftarrow} \mathcal{N}\}$
$\mathcal{O}^b$
Apply $\{G_i \overset{Bern(\kappa)}{\leftarrow} \mathcal{F}_{g_i} | g_i \in \mathcal{G}\}$

Fig. 9. The privacy game of the random combined model.

The advantage of $\mathcal{A}$ is defined as

$$\mathrm{Adv}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1]|\,.$$

B Random Fault Reduction

In this appendix, we show that a random fault adversary reduces to a threshold fault adversary. We recall that a k-threshold fault adversary on a string $\mathbb{F}_2^\ell$ can choose up to $k \le \ell$ variables to apply faults (which apply with 100% certainty). The proof follows the one from the reduction of the random probing adversary to a threshold probing adversary from Duc *et al.* [13].

Theorem 1. *Let $\mathcal{A}$ be a κ-random fault adversary on $\mathbb{F}_2^\ell$. Then, there exists a $\lceil 2\kappa\ell - 1 \rceil$-threshold fault adversary $\mathcal{S}$ on $\mathbb{F}_2^\ell$ such that $\forall (x_1, ..., x_\ell) \in \mathbb{F}_2^\ell$*

$$\boldsymbol{out}_{\mathcal{A}}(x_1, ..., x_\ell) \overset{d}{=} \boldsymbol{out}_{\mathcal{S}}(x_1, ..., x_\ell)\,,$$

with $\boldsymbol{out}_{\mathcal{A}}$ *the state* $(x_1, ..., x_\ell)$ *after applying the faults from* $\mathcal{A}$ *(equiv.* $\mathcal{S}$*) and* $\overset{d}{=}$ *denoting the equality between two distributions. The above equality holds as long as* $\mathcal{A}$ *applies at most* $\lceil 2\kappa\ell - 1 \rceil$ *faults which happens with probability at least* $1 - \exp(-\frac{\kappa\ell}{3})$*.*

The proof immediately follows from the Chernoff bound.

C "Horizontal" Statistical Ineffective Fault Attacks

In this appendix, we analyze the security of a masked and duplicated multiplier. We observe that we can enhance the advantage of a random fault adversary by attacking multiple inputs over faulting a single variable.

Consider a masked-then-duplicated AND where the shares (a^0, a^1, b^0, b^1) of the bits a, b are multiplied to form the cross products $(a^0b^0, a^0b^1, a^1b^0, a^1b^1)$. This operation is performed twice with the second time over the duplicates such that the cross products can be verified for error detection. This example of a multiplier is not uncommon, for example it follows the design by Dhooghe and Nikova [11].

We provide some advantages in the privacy model considering different attack vectors. These attacks are all examples of statistical ineffective fault attacks by Dobraunig *et al.* [12].

- When attacking only one share (e.g. a^0) with a bitflip, the advantage of the adversary between the secrets $(a, b) = (0, 0)$ and $(1, 1)$ is $\kappa/2$.
- A bitflip of both shares of a single secret (e.g. both a^0, a^1) gives the adversary a $\frac{\kappa(2-\kappa)}{2}$ advantage.
- Set-to-zero faults of both shares of a single secret (e.g. both a^0, a^1) provides an advantage $\frac{\kappa(2+\kappa)}{4}$.
- A bitflip to all input shares provides an advantage $2\kappa - 5\kappa^2 + 5\kappa^3 - 3/2\kappa^4$.

From the above attacks' success rates, we observe that it is beneficial to attack multiple bits (for example, by widening the spot of a laser). These improved attacks can be compared to the fault variant of the horizontal attack on the ISW multiplier as observed by Battistello *et al.* [4]. Instead of probing several cross products for an improved advantage, we fault the inputs to trip several abort signals on the cross products for an improved advantage. A full analysis of the random fault advantage of masked and encoded multipliers is left for future work.