The Random Fault Model

Siemen Dhooghe

imec-COSIC, KU Leuven, Leuven, Belgium

Abstract. In this work, we introduce a more advanced fault adversary inspired from the random probing model, called the random fault model, where the adversary can fault all values in the algorithm but where 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, inner product masking, error detection techniques, error correction techniques, multiplicative tags, and shuffling methods. The results of the security analysis reveal novel insights including: error correction providing little security when faults target more bits; the order between masking and duplication providing a trade-off between side-channel and fault security; and inner product masking and multiplicative masking providing exponential protection in the field size. Moreover, the results also explain the experimental results from CHES 2022 and find weaknesses in the shuffling method from SAMOS 2021.

Keywords: Encoding, Fault Attacks, Masking, Random Probing, Shuffling

1 Introduction

The field of side-channel analysis, following differential power analysis by Kocher *et al.* [19], has made significant progress over the years. Currently, we are capable of practically protecting hardware applications against a significant number of traces using masking. However, such progress did not come without difficulty. Often masking schemes were proposed, broken, and patched. This trail-and-error approach caused for a need for theoretical proofs to guarantee the security of new masking methods. The main adversary in the academic literature is the *probing model* proposed by Ishai *et al.* [18]. 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.* [8], 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.* [14] 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.* [10] 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 inner product 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 [7], 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.* [5] 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 suggested in the literature, let alone used to investigate countermeasures. As such, the question remains how effective certain countermeasures are in practice. Some popular countermeasures include: Boolean masking introduced by Patarin [16] and Chari *et al.* [8]; inner product masking introduced by Balasch *et al.* [4]; error detection methods such as in ParTI [23] and Impeccable Circuits [1]; error correction methods such as in Impeccable Circuits II [24]; multiplicative tags such as in CAPA [22] and M&M [11]; and shuffling such as Rocky [20]. However, none of the above methods for encoding or masking variables 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 masking or encoding techniques: Boolean masking, inner product masking, duplication, masked duplication, and shuffling. While the random probing security of duplication and masking are not novel, they serve as a baseline to compare to other methods. From the analysis, we found the following interesting observations.

- There is a security difference between masking-then-duplicating a variable versus duplicating-then-masking it.
- There is a beneficial trade-off in security between Boolean masking and inner product masking when the field size is larger.
- There are weak inputs for shuffling methods which are avoided when combined with masking.

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 succeed. We then use the random fault model to analyze masking and encoding techniques in two security models: *correctness* and *privacy*. We analyze Boolean masking, inner product masking, error detection methods, error correction methods, multiplicative tags, and shuffling. From the analysis, we found the following interesting observations.

- There is a security difference between masking-then-duplicating a variable versus duplicating-then-masking it.

– We are able to theoretically explain the experimental results of Bartkewitz *et al.* [5] on faulting encoded variables with a different number of parity bits.
– In the correctness model, error correction provides significantly less security than error detection 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 and inner product masking provides an exponential gain in the privacy model.
– Shuffling exhibits several weaknesses when used to secure against fault attacks. As a result, we show that the work by Miteloudi *et al.* [20] has vulnerabilities.

2 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.

3 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. These phases are important in the security models of Sect. 4 and Sect. 6 since the adversaries can not target these parts of the algorithm.

4 Random Probing Model

In this section, we introduce the *random probing model* as originally introduced by Duc *et al.* [14]. More specifically, its adversary and its security model.

4.1 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. In symbols, consider a uniform random secret value X over $\mathbb{F}_2$. An adversary which observes $F(X)$ with F the Bernoulli drawn function over $\mathcal{N}$. Since $\Pr[X = x | F(X) = x] = \varepsilon$, the adversary has gained an ε advantage of guessing X correctly over guessing it randomly. As a result, the function F has given the adversary some information on the variable X.

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 (such as the Hamming weight) of the bit vector which was processed. As a result, the link with practice is weaker when a direct generalization of the random probing model is made.

4.2 Security Model

Consider an algorithm and denote the set of all its variables (excluding the encoding and decoding phases) by V, the set of *ε-random probes* on V is the set $\{F(v) | v \in V, F \overset{Bern(\varepsilon)}{\leftarrow} \mathcal{N}\}$ 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 V which it wants to probe. The oracle then picks the secret k_b, generates its internal randomness, computes the values 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.

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

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

5 Case Studies: Random Probing Model

We apply the security model introduced in Sect. 4 to several popular countermeasures including duplication, Boolean masking, inner product masking, and shuffling.

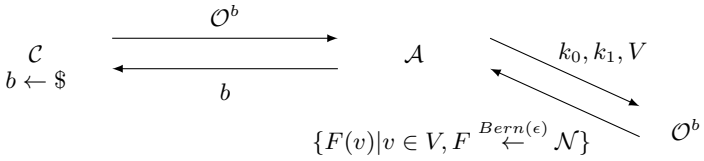

Fig. 1. The random probing leakage model.

5.1 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 the horizontal attack [10] to be effective.

5.2 Influence of Masking

Masking was independently introduced by Goubin and Patarin [16] and Chari *et al.* [8] 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 $(x^0, \ldots, x^{n-1}) \in (\mathbb{F}_2)^n$ 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 two independent random variables X^0 and X^1 and two independent random probing functions F_0 and F_1 with probability ε to observe the value, then

$$\begin{aligned} \Pr[X^0 \oplus X^1 = x \,|\, F_0(X^0) = y^0, F_1(X^1) = y^1] &= 1/2(1 - \varepsilon^2) + \varepsilon^2 \\ &= 1/2 + \varepsilon^2/2\,. \end{aligned}$$

Namely, the probability that $X^0 \oplus X^1 = x$ is $1/2$ (guessing randomly) unless both random probes return a value which happens with probability ε^2. Similarly,

for n shares, the probability is $1/2$ unless all random probes return a value which happens with probability ε^n.

With the above observation, we can consider an algorithm which stores a single n-shared bit $x^0 + ... + x^{n-1} = x$. The oracle $\mathcal{O}^x$ randomly shares x into $x^0 + ... + x^{n-1}$ every query and simply stores the shares. The advantage of the adversary making one query is then

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

That means the advantage is exponential in the number of shares.

5.3 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 [12, 17].
- 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 [11, 22, 23].

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. 2.

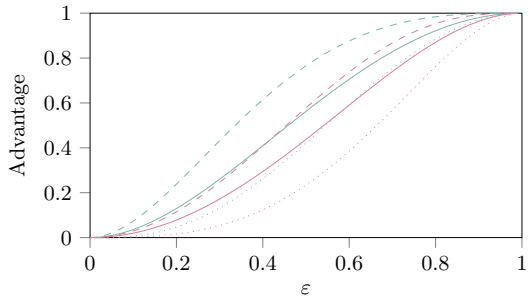

Fig. 2. The random probing advantage of the mask-then-duplicate method (in green) and the duplicate-then-mask method (in red). The full lines depict $(n, k) = (2, 2)$, the dashed lines depict $(n, k) = (2, 3)$, and the dotted lines depict $(n, k) = (3, 2)$.

5.4 Influence of Inner Product Masking

We consider inner product masking from Balasch *et al.* [3] as defined in Definition 2. We note that we consider the multiplicative mask (ℓ) to be secret. In case ℓ is public, such as in the work by Balasch *et al.* [4], the security of the countermeasure comes back to using linear codes as noted by Pousier *et al.* [21]. In that case, the security analysis of inner product masking is similar to using Boolean masking from Sect. 5.2.

Definition 2 (Inner product masking [3]). *The n-shared inner product masking of a variable $x \in \mathbb{F}_{2^m}$ consists of a vector $(x^0, \ldots, x^{n-1}) \in (\mathbb{F}_{2^m})^n$ such that $x = x^0 + \sum_{i=1}^{n-1} \ell_i x^i$. The variables $(x^1, \ldots, x^{n-1})$ and $\ell_i \neq 0$ are chosen uniformly random from $\mathbb{F}_{2^m}$ with each query.*

We consider the two-shared case where a variable $x \in \mathbb{F}_{2^m}$ is masked to x^0, x^1 such that $x^0 + \ell x^1 = x$ with $\ell \in \mathbb{F}_{2^m}/\{0\}$ randomly chosen for each query. Recall from Sect. 4 that we assume random probes over $\mathbb{F}_2$, thus they can only view one bit at a time. By random probing ℓ using the random probes $F_0, ..., F_{m-1}$, the adversary has the following probability to guess the value ℓ,

$$\Pr[L = \ell | F_0(L[0]), ..., F_{m-1}(L[m-1])] = \sum_{i=0}^{m} \frac{(2\varepsilon)^i (1-\varepsilon)^{m-i}}{2^m - 1} \leq \frac{(1+\varepsilon)^m}{2^m - 1},$$

where, for the inequality, we used the binomial expansion theorem.

Given that the adversary can correctly guess ℓ, we consider the worst-case scenario where an m-bit two-share Boolean masking is left for which the advantage is upper bounded by $m\varepsilon^2$. When the adversary guesses ℓ wrongly, the advantage is zero. This gives us the total advantage

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

The above advantage can be compared to the advantage of two-share Boolean masking $m\varepsilon^2$ showing that inner product masking gives a significant improvement.

5.5 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. 4. 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

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

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.

We observe that the bound of the shuffling method is the same as the bound of a two-bit two-shared masking. As a result, shuffling provides no additional security when the adversary chooses weak inputs.

Shuffled Masking. We remove the weak input scenario's for shuffling by applying masking on top of the shuffling. Note that, when naively shuffling masked values (for example shuffling the order of masked S-boxes), shuffling is still vulnerable to weak inputs. Instead, we consider shuffling only one share of each masking. For example, consider the case of masked values (x^0, x^1) and (y^0, y^1) where we shuffle the values x^1 and y^1 randomly following a cyclical shift $z \in \mathbb{F}_2$. We refer to the work by Azouaoui *et al.* [2] on methods to compute on masked and shuffled data.

We consider the case where the adversary randomly guesses z and then attacks the masking. Half of the time, the adversary guesses wrong leading to a zero advantage. In case the adversary guesses z correctly, the advantage is that of a two-bit two-shared masking $(1-(1-\varepsilon^2)^2)$. Thus, the advantage of a random probing adversary against shuffling two-bit two-shared value is

$$\mathrm{Adv}(\mathcal{A}) = \frac{1}{2}(1 - (1-\varepsilon^2)^2)\,.$$

The above has twice the security over a non-shuffled method. However, the adversary can also probe the value z which slightly increases the advantage.

Consider that the adversary also probes the value z, the advantage of the adversary when shuffling m n-shared values equals the advantage of the sharing without shuffling times the probability the adversary can guess the value z after random probing it. Consider k the largest integer such that $m \geq 2^k$, then the advantage is

$$\mathrm{Adv}(\mathcal{A}) = (1 - (1-\varepsilon^n)^m)\Big(\sum_{i=0}^{k} \frac{1}{2^{k-i}}\varepsilon^i(1-\varepsilon)^{k-i}\Big)\Big(\varepsilon + \frac{2^k}{m}(1-\varepsilon)\Big)\,,$$

which, if $m = 2^k$, is upper bounded by $m\varepsilon^n \left(\frac{1+\varepsilon}{2}\right)^{\log(m)}$.

Shuffling could achieve better bounds when also masking the value z; by shuffling all-but-one of the shares randomly (when $n > 2$); or by using more general permutations instead of only cyclical shifts such as explained in the work by Azouaoui *et al.* [2]. We leave the random probing analysis of these cases as future work.

6 Random Fault Model

We propose a novel adversary model inspired by the random probing model where the adversary can fault the entire state but there is a limited probability for the fault to occur (following the mechanisms of statistical fault analysis [15]).

We note for clarity that the adversary can **precisely target** the faults (over bits), but the probability for the fault to occur is random.

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 an adversary can make over bits, we consider three different functions.

- *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. 7 and Sect. 8, we use the above function names to indicate which fault is injected.

We only consider fault over bits and not over larger fields (though the generalization is easily made). Similar to the random probing model, when considering random faults over larger fields, the distribution of the applied faults in practice might significantly differ from an all-or-nothing metric causing a weaker link to practice when the random fault model is naively generalized over larger fields.

6.1 Correctness

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

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. 3. We require, for the algorithm to be useful, that it outputs a correct result in case no faults were present.

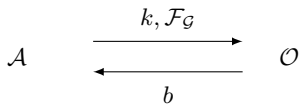

Fig. 3. The correctness game of the random fault model.

The advantage of the adversary is the probability of the oracle outputting zero after a single query.

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

6.2 Privacy

The privacy game of the *random fault model* is the bounded query, left-right security game represented in Fig. 4. 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) $\mathcal{G} = \{g_i \mid i \in V\}$. 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 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. Examples of fault attacks in this model include Clavier's ineffective faults [9] and statistical ineffective faults by Dobraunig *et al.* [13].

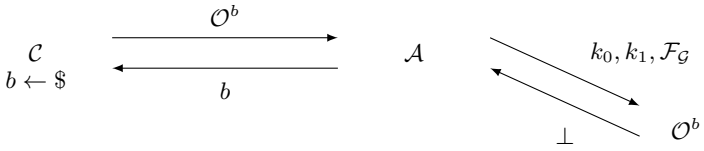

Fig. 4. The privacy game of the random fault model.

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

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

In case the algorithm does not have an abort signal (such as with masking or error correction), the privacy model does not apply to the method. The protection of these cases is solely determined by the correctness model.

7 Case Studies: Correctness

We go over several countermeasures and evaluate their security in the correctness model from Sect. 6.1.

We first establish a baseline. When storing a single bit, the advantage of a random fault adversary changing this value is κ. In case there are m variables (or m queries), then the advantage is $1-(1-\kappa)^m \leq m\kappa$.

7.1 Influence of Masking

Consider masking from Sect. 5.2 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.

7.2 Influence of Duplication

We then investigate the effect of duplicating the variable and error checking the duplicates at the end of the computation (aborting if they are not equal).

Definition 3 (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 $x_0 = \ldots = x_{n-1}$.*

We calculate the advantage of a random fault adversary injecting a *bitflip* (see Sect. 6) 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 better 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. The advantage is the probability that the n bitflips form one of the 2^m codewords of $\mathcal{C}$. In particular, if $\kappa = 0.5$, you get 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.* [23] 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 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 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.* [5], experiments were performed by faulting only the message bits in an implementation (leaving the parity bits unaltered).

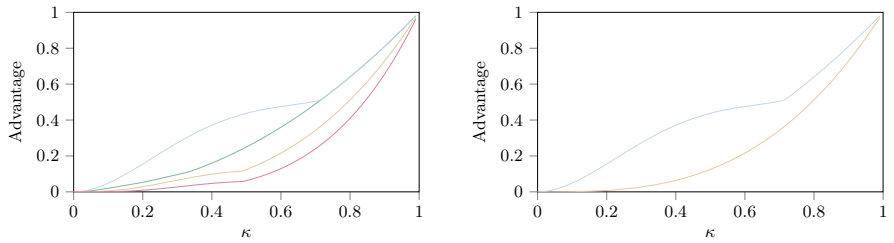

Fig. 5. The advantage of a random fault adversary against encoded values on the left and on the right when only the message bits are attacked. Blue depicts the $[5, 4, 2]$ code, green $[8, 4, 2]$, yellow $[7, 4, 3]$, and red $[8, 4, 4]$. For the right figure, the $[8, 4, 2]$ and $[8, 4, 4]$ codes have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

7.3 Influence of Error Correction

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. 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 variables. 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 wins in case one of the m variables is error-corrected to the wrong output. The advantage for parameters m is given in Fig. 6. From this graph, we find that error correction performs significantly worse when faults can target a large state size compared to duplication. We also note that the combination of Boolean masking or inner product masking with error correction would not improve the advantage.

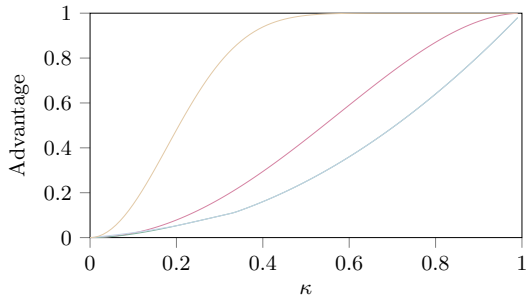

Fig. 6. The advantage against error correction (with three duplicates) is shown in red (for $m = 2$) and yellow ($m = 16$). The advantage against error detection (with two duplicates) is shown in green (for $m = 2$) and blue ($m = 16$).

7.4 Influence of Masked Duplication

Recall the two methods of both masking and duplicating variables from Sect. 5.3.

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 $(1-\kappa)^k$, we can generalize for n shares and k duplicates. Namely, the advantage with n shares and k duplicates becomes

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

The above bound is derived from the masking and duplication advantages.

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 $\frac{1}{2}(1-(1-2\kappa)^n)$. As a result, breaking the correctness of a k-duplicated n-sharing is

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

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.

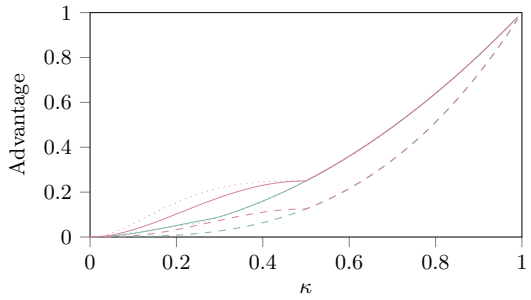

Fig. 7. The random fault advantage in the correctness model of the mask-then-duplicate method (in green) and the duplicate-then-mask method (in red). The full lines depict $(n, k) = (2, 2)$, the dashed lines depict $(n, k) = (2, 3)$, and the dotted lines depict $(n, k) = (3, 2)$.

7.5 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 4 (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 with the hope that $\alpha = 0$. Consider F a random fault flipping the first bit of the value. 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

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

7.6 Influence of Shuffling

Consider the shuffling countermeasure from Sect. 5.5. Similar to the weak input attack in Sect. 5.5, there are weak inputs in the correctness model. Namely, for two bits $(x, y) \in \mathbb{F}_2^2$, the adversary can still bitflip each value for the same advantage as in the non-shuffled case.

In case duplication is used on top of the shuffling, there are still weak inputs. 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. 7.2 that when k is small, the best attack of the adversary is to fault all duplicates. However, when shuffling the duplicates, such an attack has the same probability to break correctness. 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 can not find a non-trivial upper bound on the security of shuffling in the correctness model.

8 Case Studies: Privacy

We go over several countermeasures and evaluate their security in the privacy model from Sect. 6.2. Recall that the privacy model only applies to countermeasures which can abort the computation.

8.1 Influence of Duplication

Consider the duplication method from Sect. 7.2 (the advantage is similar when using multiplicative tags from Sect. 7.5). 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 A.

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

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

8.2 Influence of Masked Duplication

Similar to Sect. 7.4, 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

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

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 $\mid \kappa_0 - \kappa_1 \mid = 2^{1-n}\kappa^n$.

When faulting $(x_0^0, x_0^1, x_1^0, x_1^1)$ all to zero, the advantage becomes $2\kappa^2(1-\kappa)^2$. Similar to the bounds in Sect. 7.2, for small values 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

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

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.

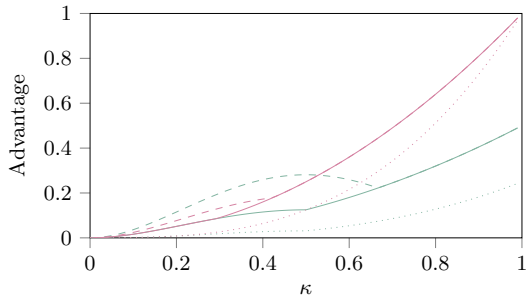

Fig. 8. The privacy advantage of the mask-then-duplicate method (in green) and the duplicate-then-mask method (in red). The full lines depict $(n,k)=(2,2)$, the dashed lines depict $(n,k)=(2,3)$, and the dotted lines depict $(n,k)=(3,2)$.

8.3 Influence of Inner Product Masking

To have the field size m take an exponential role in the advantage in the privacy game, we use inner product masking from Sect. 5.4.

We consider the mask-then-duplicate approach with inner product masking which shares a value $x \in \mathbb{F}_{2^m}$ to $(x_0^0, x_0^1, x_1^0, x_1^1)$ with $x = x_0^0 + \ell x_0^1 \in \mathbb{F}_{2^m}$ and with $x_0^0 = x_1^0$ and $x_0^1 = x_1^1$. Since the privacy analysis of this case is more complicated, we provide an upper bound on the advantage.

Similar to the analysis in Sect. 5.4, we make the assumption that the adversary can fault ℓ and be provided information whether the m-bit value changed or not. Consider an adversary placing m (set-to-zero) random faults on ℓ, in case

the adversary receives the information whether ℓ changed, then the adversary learns at most one bit of ℓ. As a result, the probability of the random fault adversary guessing ℓ is upper bounded by $\frac{2}{2^m-1}$.

Given that the adversary can guess ℓ, we assume the worst-case where an m-bit two-share two-duplicate Boolean masking is left for which the advantage is upper bounded by $2m\kappa^2(1-\kappa)^2$ following Sect. 8.2. In total, the advantage of a random fault adversary against a two-share inner product masking with m bits in the privacy model is

$$\mathrm{Adv}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{O}^0} = \perp] - \Pr[\mathcal{A}^{\mathcal{O}^1} = \perp]| \leq \frac{4m}{2^m-1}\kappa^2(1-\kappa)^2\,.$$

Combining inner product masking and multiplicative tags allows us to bound the advantage in both the correctness and privacy models exponentially by the field size compared to regular duplication and Boolean masking.

8.4 Influence of Shuffling

Consider the shuffling method from Sect. 5.5 but with duplicated values. Similar to Sect. 5.5 and Sect. 7.6, 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. 8.1 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 also using masking 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. 7.6, we conclude that shuffling against fault attacks exhibits several weaknesses. The vulnerabilities found in this work directly apply to the Rocky countermeasure [20] which we show is weak in both the correctness and privacy models for certain parameters κ and for certain weak inputs. We believe the addition of masking could circumvent the weak inputs (similar to Sect. 5.5). However, a more in-depth analysis is required to clarify the efficiency-security trade-off.

9 Conclusion

In this work, we investigated the random probing security of Boolean masking, inner product masking, duplication, masked duplication, and shuffling. We find the following novel observations.

- There is a security difference between first masking and then duplicating a variable versus first duplicating and then masking it.
- We find that inner product masking provides an exponential random probing security in terms of the field size.

– We find weak inputs for shuffling methods and find a mitigation when combined with masking.

We then propose a new fault adversary called the *random fault model* where the adversary can fault all variables in the algorithm but each fault only has a limited probability to occur. We propose the *correctness* and *privacy* security models. These models are used to analyze the security of masking, error detection methods, error correction methods, multiplicative tags, inner product masking, and shuffling methods. We find the following novel observations.

– There is a difference in security between codes with a different minimal distance and a different number of parity bits as experimentally noted by Bartkewitz *et al.* [5].
– There is a security difference between first masking and then duplicating a variable versus first duplicating and then masking it.
– Error correction provides little security when a fault targets several bits.
– Similar to horizontal attacks which are captured by the random probing model, the random fault model captures statistical ineffective faults (security in the privacy model) targeting multiple values.
– Multiplicative tags are able to provide strong security when working over a large field size in the correctness model. Similarly, inner product masking provides strong security in the privacy model over large field sizes.
– Shuffling methods exhibit several weaknesses as a countermeasure against faults. As a result, we observe that the work by Miteloudi *et al.* [20] provides no additional security in the random fault model.

We leave several open problems for future work.

– We investigated the random probing and random fault security of countermeasures, but we leave the combined security as future work.
– We observed that shuffling methods exhibit weaknesses against random fault attacks and we leave its fix for future work.
– The application of the random fault model to masked or encoded operations such as multipliers.

Acknowledgements. Thank you to Vincent Rijmen for the helpful discussions. Siemen Dhooghe is supported by a PhD Fellowship from the Research Foundation – Flanders (FWO).

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. Balasch, J., Faust, S., Gierlichs, B.: Inner product masking revisited. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 9056, pp. 486–510. Springer (2015)
4. Balasch, J., Faust, S., Gierlichs, B., Paglialonga, C., Standaert, F.X.: Consolidating inner product masking. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 724–754. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70694-8_25
5. 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)
6. 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)
7. 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
8. 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
9. 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
10. 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)
11. 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
12. 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
13. 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
14. 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
15. 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
16. 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
17. 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
18. 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
19. 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
20. 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
21. Poussier, R., Guo, Q., Standaert, F., Carlet, C., Guilley, S.: Connecting and improving direct sum masking and inner product masking. In: CARDIS. Lecture Notes in Computer Science, vol. 10728, pp. 123–141. Springer (2017)
22. 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
23. 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)
24. 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 "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.

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 [12].

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.* [13].

- 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 attacks on the ISW multiplier as observed by Battistello *et al.* [6]. Instead of probing several cross products for an improved advantage, we fault the inputs to trip several abort signals on the cross products.