Relaxed Threshold Implementations

Rim Zahmoul, Vincent Grosso

Relaxed Threshold Implementations

Rim Zahmoul*
rim.zahmoul@gmail.com
MSD Santé Animale France
Beaucouzé, France

Vincent Grosso†
vincent.grosso@univ-st-etienne.fr
Laboratoire Hubert Curien
Saint-Étienne, France

ABSTRACT

IoT devices by nature are prone to side-channel attacks. Indeed, it is easy for an attacker to get physical access to such devices. Masking is one of the most widespread side-channel attack countermeasures. Among masking, hardware threshold implementations have the advantage of being resistant in the presence of glitches, while maintaining relatively low share refreshing costs. To this end, threshold implementations rely on three properties that should be satisfied: correctness, uniformity, and non-completeness. Recently, some research has relaxed these properties to present more efficient schemes (e.g. two shares threshold).

In this paper, we study the security that can be achieved by relaxing these properties. We show that, in some cases, we need a local additional property to ensure provable security. This local property allows us to prove the security of a threshold implementation of Friet-P presented in the original paper. Next, thanks to this new property, we show that we can reduce by 25% the required randomness for the threshold implementation of Friet-P. Finally, we show how this property allows us to propose some implementations of 3-share 4-bit s-boxes that were not possible with classical threshold implementation rules.

CCS CONCEPTS

• **Security and privacy → Side-channel analysis and countermeasures**.

KEYWORDS

Side-channel protection; Threshold implementation; Security analysis

ACM Reference Format:
Rim Zahmoul and Vincent Grosso. 2024. Relaxed Threshold Implementations. In *Proceedings of the 21st ACM International Conference on Computing Frontiers Workshops and Special Sessions (CF '24 Companion), May 7–9, 2024, Ischia, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3637543.3654658

1 INTRODUCTION

Many cryptographic implementations have been proven to be vulnerable against side-channel attacks (SCA). However, masking can be employed in order to protect the system's sensitive information from third parties [5, 13, 21].

The typical process in establishing a secure masking scheme is to represent the circuit and its leakages through an abstract leakage model, then prove the security of this construction in that high level of abstraction. In this context, the traditional d-probing model is considered a good example: the adversary obtains the noiseless data by placing d-probes on intermediate wires. This simple model can be compared to the more realistic noisy leakage model [10, 17]. This fact is particularly useful since the application of probing analysis is easier than modeling side-channel leakages in the noisy model.

Thanks to the security reduction from the probing model to the noisy leakage model we can then hope for some level of security. Unfortunately, this reduction relies on some assumptions. In particular, it assumes leakage independence. This means the shares leak independently, however, this may not be true in hardware due to the presence of glitches, defined as an undesired and unexpected pulse in a combinational circuit. Furthermore, glitches cause unnecessary power consumption. Thus, hardware designers also try to diminish the glitches for low energy reasons.

It has been shown that glitches can decrease the security order of masked circuits [14]. Nevertheless, eliminating the glitches while keeping a reasonable implementation cost is not a straightforward task. Therefore, these hardware hazards threaten the security established by masking. In this context, the threshold implementations were proposed as the first method that considers the presence of glitches in masking [16].

Indeed, threshold implementations (TI) do not eliminate this hazard but manage the shares in a way to avoid creating shared functions that are sensitive data dependent in the presence of glitches. In contrast, classical masking methods do not take into consideration the presence of physical defaults like glitches.

In their work, Nikova et al. defined three properties: uniformity, non-completeness, and correctness [16]. These properties guarantee provable security against first-order Differential Power Analysis (DPA) even in the presence of glitches. Besides dealing with glitches, threshold implementations try to reduce the number of random bits needed for refreshing.

Our contribution. In this paper, we are focusing on TI as an efficient side-channel attack countermeasure in the presence of glitches. We also propose a new design for FRIET which represents an Authenticated Encryption (AE) technique. We denoted it as Friet-TI.

To prove the security of this threshold implementation of Friet-P, which was introduced in [22], we propose a new intermediate property that we call independence. Our scheme can solve the non-fulfillment of some important TI properties in certain register layers. Theorem 1 in section 3 shows this important characteristic.

CF '24 Companion, May 7–9, 2024, Ischia, Italy

ACM ISBN 979-8-4007-0492-5/24/05
https://doi.org/10.1145/3637543.3654658

We consolidate different SCA countermeasures in order to prove the security of relaxed TI constructions.

Finally, our method is illustrated through the study and proof of our Friet-TI security. Moreover, we bring the analysis of a 4-bit s-box, as an example of applications of our method.

Paper organization. Section 2 presents the background of different techniques such as the threshold implementations and Friet. Section 3 includes our first main contribution: the analysis of the TI properties and a new intermediate property. Section 4 details our new technique on a real use-case: the Friet-TI. This section also contains accurate security proof for this implementation. In Section 5, we search for more efficient 2-stage 3-share s-boxes that can be efficiently implemented in a relaxed threshold manner.

2 BACKGROUND

In this section, we summarize the threshold implementations and the Friet permutation, on which we will demonstrate the application of our method.

2.1 Threshold implementations

To prevent side-channel attacks (SCA), multiple approaches have been introduced. Among them, masking is one of the most popular due to the good understanding of its security requirements [5, 13]. Proofs of security in masking rely on some assumptions defined by a leakage model [17]. However, the independent leakage assumptions may not be fulfilled in practice. For example, in hardware implementations, glitches may reduce the security order of masked schemes [14]. In this context, due to glitches, a masked circuit may present exploitable leakages that are directly correlated to the unmasked variable.

TI were proposed as a solution for masking schemes in the presence of glitches [16] that are provable secure against first-order side-channel adversaries. TI adapt the multiparty computation (MPC) technique at the circuit level, with relaxed constraints due to the nature of considered attack [12, 20].

The concept of MPC is carrying the computation by independent parties with no knowledge of the secret value. In TI, these parties are substituted by independent sub-circuits, known as shared functions.

In masking, the Boolean sharing of the secret variable a needs s random shares $(a_1, a_2 \ldots, a_{s+1})$. Thus, a secret share a_{s+1} is computed in a way that $a_{s+1} = a \oplus a_1 \oplus a_2 \oplus \cdots \oplus a_{s+1}$. The goal of a TI is to find shared functions F_i that will allow computing a function F using a partial sharing of the secret variable.

To achieve security against glitches, Nikova et al. [16] proposed the non-completeness property. The main idea of this property is to not use all the shares of the secret in the same sub-circuit implementing a shared function, similarly as in the MPC technique. Thus, a sub-circuit (or party) cannot reveal information about the secret. However, in the case of computing high-degree functions that are used in cryptography, the non-completeness property may not be sufficient. It can be shown that the number of shares to implement a function of algebraic degree α is at least $\alpha + 1$ shares.

To implement a high-degree function efficiently, the literature suggests dividing it into a composition of low-degree functions [4]. Then, the uniformity is a second property required to securely compose masked functions. This property can be outlined when the output of a shared function is indistinguishable from a fresh sharing of the unshared value. This definition is really useful for security in the univariate case, but may be insecure in case of higher-order attacks [19].

In order to define the uniformity, let $\mathsf{F} : \mathbb{F}_{2^m} \mapsto \mathbb{F}_{2^n}$ be a function and F_j with $1 \leq j \leq s+1$ its shared functions. For all random sharing $(a_1, a_2, \ldots, a_{s+1})$ satisfying $\sum_i a_i = a$, the number of tuples of input sharings $\mathcal{I} = \{\mathbf{x}, \mathbf{y}, \mathbf{z}, \ldots\}$ for which $F_j(\mathcal{I}) = a_j$, is equal to $2^{(s-1)(m-n)}$ times the $\#\mathcal{I} \in \mathbb{F}_2^m$ for which $a = \mathsf{F}(x = \sum \mathbf{x}, y = \sum \mathbf{y}, z = \sum \mathbf{z}, \ldots)$. Hence, if F is a permutation on $\mathbb{F}_2^m$, then the functions F_j define together a permutation on $(\mathbb{F}_{2^m})^s$.

While in most existing works, the uniformity definition makes use of the unshared function, in [2, 8] the given description only considers the output sharing. More formally, a sharing $(a_1, a_2, \ldots, a_{s+1})$ is an uniform sharing of a if $\exists\, p^*$ such that $\forall a$ we have :

$$\Pr[(a_1, a_2, \ldots, a_{s+1})|a] = \begin{cases} p^* & \text{if } (a_1, a_2, \ldots, a_{s+1}) \in Sh(a) \\ 0 & \text{otherwise.} \end{cases}$$

$$\sum_{(a_1, a_2, \ldots, a_{s+1}) \in Sh(a)} \Pr[a_1, a_2, \ldots, a_{s+1}] = \Pr[a]. \tag{1}$$

Where $Sh(a)$ denotes the set of valid sharing of a.

Finally, correctness ensures $\mathsf{F}(\sum_i a_i) = \sum_j \mathsf{F}_j(a_1, a_2, \ldots, a_{s+1})$, i.e. the result obtained in shared computation corresponds to the intended computation. Interestingly, this third TI property is necessary for security, as we will see in the next sections.

Different masked circuits have been proposed grounded on the TI concept. In 2012, a TI of all 3×3 and 4×4 s-boxes was proposed in [4], allowing efficient implementation of block-ciphers. A s-box classification was used in order to facilitate the implementation of s-boxes in TI manner. It also introduced a permutation decomposition scheme into a composition of quadratic ones. This decomposition is necessary to reduce the number of required shares (a function of degree t required at least $t + 1$ shares; theorem 1 in [16]). Thus, the TI can be used as a tool to establish secure s-box implementations.

More recently, to make hardware-masked implementations stronger in the presence of physical defaults, a sound, but conservative, property was proposed: the d-glitch immunity [8, 11]. The properties, considered necessary and sufficient for masking, reformulate the probing security concept in hardware implementations using the adversary model introduced in [19]. Every probed wire accords knowledge about its inputs until the final synchronization point to the adversary. By the replacement of each probe with its glitch-extended form, they achieved an information-theoretic condition for masked implementations in the existence of glitches.

2.2 Relaxation of TI

In this section, we look at some schemes inspired by TI. The main idea is to keep inputs of sub-circuit independent of each secret. We call generalization of TI constructions, constructions that deal with secret sharing in hardware but require no extra randomness in their inner computation. It differs from masking as strong masking definition such as NI/SNI, which requires randomness in their inner computation to have provable security.

We consider only first order, as the higher-order case has been shown to be vulnerable to multivariate attacks [15]. One of our

goals is to reduce the number of shares, hence we will consider only 2-share cases.

A first-order sharing of $c = ab$ is constituted by the set of cross-products $a_i b_j$ with $i, j \in \{1, 2\}$. The independence from the sensitive variable for each output share c_i depends on the input sharing independence from a and b. Therefore, non-completeness is a necessary condition. Nonetheless, this solution increases the number of shares.

To illustrate this affirmation, we draw the following example for first-order TI with 2 shares. We consider the sharing of the function $d = a \oplus bc$:

$$d_1 = a_1 \oplus b_1 c_1 \qquad d_2 = b_1 c_2 \qquad d_3 = a_2 \oplus b_2 c_2 \qquad d_4 = b_2 c_1.$$

Then if we consider $e_1 = d_1 \oplus d_2$ and $e_2 = d_3 \oplus d_4$, the generated sharing will be uniform. But it requires 2 cycles as we need to store the partial results d_i. Different examples were presented in [2], showing that the required number of shares is influenced by the function's representation and the degree of the s-box.

Also, the cost of TI strongly depends on the number of input and output shares, both in terms of gating counting and latency. Thus, it the important to afford a minimum number of required shares for TI (see Lemma 6 in [2]). On the other hand, to attain first-order DPA security of quadratic permutations, Bilgin et al. proposed the use of two input shares [3]. Based on these findings, the authors proved the existence of at least a single permutation in all quadratic 3-bit and 4-bit classes that admit a uniform first-order TI.

From the previous works, one can conclude that decreasing the number of shares has different advantages but also some important drawbacks, especially in verifying the TI properties. Thus, in this manuscript, we establish a new local intermediate property to prove the security of the two-share decomposition.

2.3 Friet

An Authenticated Encryption (AE) scheme denoted as Friet was introduced by Simon et al. in [22]. This scheme is based on a permutation mode called Friet-PC. Embedding Friet-PC in another permutation results in the creation of Friet-P, which was primarily designed to resist against fault attacks.

We study in this work the Friet-TI which is the threshold implementation of Friet-P. Friet-P was proposed with a first-order protected version Friet-TI, we analyze the security of this version and propose improvement. Our technique is, naturally, applicable to a large variety of permutations. The security of the proposed scheme is proved by the local independent property, introduced in section 3. Then, the Friet-TI is detailed in section 4 with its security proof.

3 SECURITY OF RELAXED TI

There exists some generalization of TI that does not strictly follow the TI rules. And in this case, special attention should be put on the security arguments. We call these schemes relaxed TI, and their security is not based on extra randomness as opposed to most of the known masking schemes.

We first analyze in more detail the role of every TI property. In particular, we emphasize the crucial role of correctness in secure implementations. Next, we present a property to prove security when correctness is not fulfilled.

3.1 Which properties of TI are necessary for security

When we look at the generalization of TI, we remark that they do not fulfill the three properties of classical TI. We first analyze what weakness may appear, if only a subset of these properties is applied.

- *Non-completeness.* If non-completeness is neglected, considering a worst-case scenario as proposed in [1] and no assumptions on the implementation, security cannot be guaranteed. Also, if we consider the robust probing model in the presence of glitches [8, 11, 18], no security proofs can be furnished.
- *Uniformity.* As discussed in [8], uniformity is not necessary in relaxed TI. However, a non-uniform output requires a larger number of shares.
- *Correctness.* The last property is correctness, while this property is clearly required for the whole circuit, it can be dropped locally. For example, the 2-share TI [11, 19] of Toffoli gate. For $(a, b, c) \mapsto (a, b, c \oplus ab)$ the 2 stages 2-share TI is given by :

$$s_1 = (a_1', b_1', c_1') = ([[a_1]], [[b_1]], [[c_1 \oplus (a_1 b_1)] \oplus (a_1 b_2)])$$
$$s_2 = (a_2', b_2', c_2') = ([[a_2]], [[b_2]], [[c_2 \oplus (a_2 b_2)] \oplus (a_2 b_1)]).$$

Where the [ ] corresponds to register storage. The shared implementation is also described in Figure 1. Hence, this implementation needs two clock cycles to compute the shared function.

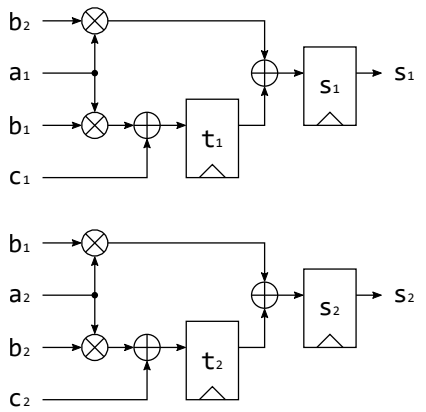

Figure 1: Example of 2-share TI use as an example in the literature e.g. [6].

If we look at the results after one cycle we have:

$$([a_1], [b_1], [c_1 \oplus (a_1 b_1)])$$
$$([a_2], [b_2], [c_2 \oplus (a_2 b_2)]).$$

This does not correspond to a deterministic function for the unshared values. For example, if we look at sharing $(1, 1, 1)$

in table 1, the value $t = t_1 \oplus t_2$ is not constant, which means non-correctness.

Table 1: Intermediate result of third bit of Toffoli gate.

a_1	a_2	b_1	b_2	c_1	c_2	t_1	t_2
1	0	1	0	1	0	0	0
1	0	1	0	0	1	1	1
1	0	0	1	1	0	1	0
1	0	0	1	0	1	0	1
0	1	1	0	1	0	1	0
0	1	1	0	0	1	0	1
0	1	0	1	1	0	1	1
0	1	0	1	0	1	0	0

In a shared function, intermediate layers where the output does not correspond to a correct sharing will be called non-correct layers. In contrast, the output layer of shared functions will be called a correct layer.

One major issue when dealing with non-correct layers is that we cannot use the definition of uniformity defined earlier [16]. Hopefully, some definitions of uniformity get rid of the need for correctness by considering only the intermediate sharing. We recall the definition given in [8] for sharing uniformity in equation (1).

In this definition, one considers only the current value and not the input as opposites as in [16]. However, as previously discussed in [4], special attention needs to be taken in the second stage. Figure 2 shows that relaxing the TI definitions may result in an insecure implementation. We give the output for every layer:

$$a_1 = z_1 \oplus z_2 \qquad a_2 = z_2 \qquad a_3 = z_3$$
$$b_1 = a_1 \oplus a_3 = z \qquad b_2 = a_2 = z_2 \qquad b_3 = a_3 = z_3$$
$$c_1 = b_1 \oplus b_3 = z_1 \oplus z_2 \qquad c_2 = b_2 = z_2 \qquad c_3 = b_3 = z_3$$

It is easy to verify that the input and the output of the circuit are identical. Thus we have a correct identity considering the full circuit, but if we look at the first layer we see that two sharings of 1 can give different intermediate results, this is what we call the lack of correctness; e.g. $z_1 = 1, z_2 = 0, z_3 = 0$ gives $a_1 = 1, a_2 = 0, a_3 = 0$ and $z_1 = 0, z_2 = 1, z_3 = 0$ gives $a_1 = 1, a_2 = 1, a_3 = 0$.

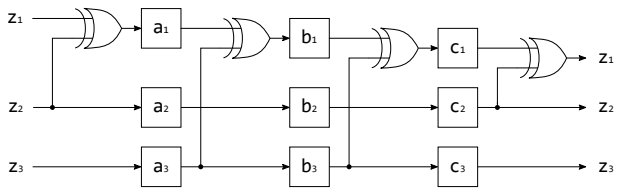

Figure 2: Identity shared with four stages and correctness only for the last stage.

The decomposition consists of three non-correct layers and a final correct layer. One can easily check that in every step we have uniformity as we have a bijection over the shares. We can also verify that the output is the same as the input so we have correctness in the last stage. However, we can remark that b_1 corresponds to the unshared input $b_1 = z = z_1 \oplus z_2 \oplus z_3$.

By analyzing the TI properties, we have noticed their major impact on the establishment of a secure implementation. Meanwhile, different issues may appear when one of these properties is not fulfilled. We will focus in the next section on finding some solutions for these weaknesses.

3.2 A new intermediate property: independence

To avoid the issue described in the section 3.1, we suggest checking independence, instead of non-completeness. We recall that the three TI properties imply independence over the input. But as discussed in the two-share TI, we can have independence without having the three TI properties at the same time. Besides, it is in general computationally easier to check non-completeness.

We recall that the independence between two random variables X and Y can be efficiently checked using the following equality:

$$\forall y, \Pr[X = x | Y = y] = \Pr[X = x].$$

As we will consider a circuit with some incorrect layers we first define an incorrect layer.

Definition 3.1. A combinational layer corresponds to the combinational logic between two register layers. Figure 3 has two combinational layers, one constituted of the sub-functions f_1, f_2, f_3, and the second one constituted of sub-functions g_1, g_2, g_3. We always consider a circuit that starts with a correct sharing, i.e. $x = x_1 \oplus \cdots \oplus x_n$. For a combinational layer that outputs $a_1, \ldots, a_m$, if there exists a deterministic function f such that the $f(x) = a_1 \oplus \cdots \oplus a_m$ for every x and every sharing of x, we call this layer that outputs the a_i share a correct layer. If such a function does not exist we call it an incorrect layer.

In Figure 2, the three first AND layers (i.e. outputting a_i, b_i, c_i) are incorrect layers while the last one is a correct layer (with inputs c_i and output z_i).

Theorem 3.2. *A relaxed TI circuit with $k \geq 1$ correct layers, since at least the last one must be a correct layer and n incorrect layers, is first-order secure if the following conditions are satisfied:*

- *the TI circuit corresponds to a bijection.*
- *for every correct layer output, we have a uniform and correct sharing.*
  For every a_i we have $x = x_1 \oplus \cdots \oplus x_n$.
- *for every layer, we have independence of $f(x)$, the shared function of the shares, from the unshared output of the previous or next correct layer.*

Proof. As the whole circuit corresponds to a bijection F every correct layer corresponds to a bijection. Indeed, if $F = F^0 \circ \cdots \circ F^k$ is a bijection then every F^i is a bijection.

As we check independence for every layer from the unshared output of a correct layer, first-order information does not leak. Due to the bijection between the sensitive unshared input and the unshared output of a correct layer, we can conclude the independence

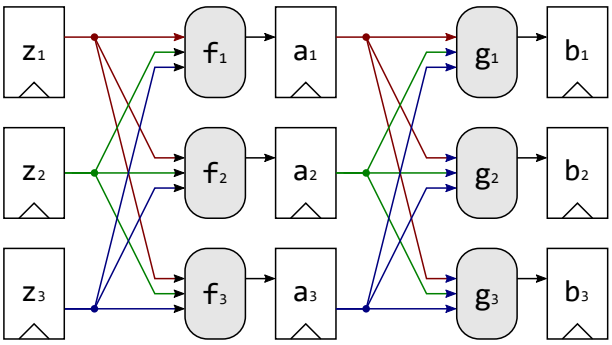

Figure 3: Scheme of circuit with correct and not correct layers. The stored value (z_1, z_2, z_3) is a correct sharing of z, the stored value (b_1, b_2, b_3) is a correct sharing of $b = \mathsf{F}(z)$, but the stored value (a_1, a_2, a_3) does not correspond to a correct value. Thus for every g_i function, we check that the inputs are independent of z.

of this sensitive unshared input value from the input of a shared function.

Let x be the unshared input of the full circuit, z be the unshared value of a correct layer and y is the inputs of a subcircuit if we have $\Pr[Z = z|Y = y] = \Pr[Z = z]$ then $\Pr[X = x|Y = y] = \Pr[X = x]$ due to the existence of a bijection. Thus $\Pr[X \cap Z] = \Pr[X = x] = \Pr[Z = z]$. □

We recall the results from [4]: if we want to share a permutation, then the shared functions define together a permutation over the shares. This allows us to efficiently check uniformity when we look at permutation with a large state, e.g. more than 32 bits.

Using this property we can prove the security of the two shares decomposition proposed in [11, 19]. It is easy to check the independence between $\{t_1, a_1, b_2\}$ or $\{t_2, a_2, b_1\}$ and $\{a, b, c\}$.

The difference between this property and the classical TI properties is that this new one considers only results between two register layers. The property needs to go back several layers to find a correct register layer output, which may cause an upstream of several register layers.

We conclude that the main disadvantage of this technique is that if we have to go back too far away, due to diffusion in cryptographic permutation, the computer-aided analysis of the statistical independence may be too expensive.

Indeed, if too many variables are considered, the computational cost can become prohibitive. What sounds reassuring is that in most designs we can rely on the correct layer. That's because, in most 2-shares TI, there is often a correct layer; e.g. in the Toffoli gate case, we have a correct layer after every second layer of the register.

Next, we applied the local independent property to prove the security of Friet implementation and found some sharing 3-share sharing of 4-bit s-box with odd parity.

4 FRIET IMPLEMENTATION

To maintain the secrecy of data in the presence of fault injections and first-order side-channel attacks, [22] proposed an authenticated encryption method based on permutations. This scheme is composed of a 384-bit state which is divided into three limbs denoted by a, b, and c. Each limb is 128-bit wide and is updated in 16 rounds. The permutation used in Friet, which is called Friet-P, is 512-bit wide, divided into 4 limbs: a, b, c, and d. The Friet-P representation is given in figure 2 of the paper [22]. It is depicted by 6 steps:

- δ_i: A round constant addition rc_i, $(a, b, c, d) \leftarrow (a, b, c \oplus rc_i, d \oplus rc_i)$
- τ_1 and τ_2: Two non-native limb transpositions $\tau_1 : (a, b, c, d) \leftarrow (d, c, a, b)$, $\tau_2 : (a, b, c, d) \leftarrow (a, d, c, b)$
- μ_1 and μ_1: Two mixing steps $\mu_1 : (a, b, c, d) \leftarrow (a, b \oplus (c <<< 1), c, d \oplus (c <<< 1))$, $\mu_2 : (a, b, c, d) \leftarrow (a, b, c \oplus (b <<< 80), d \oplus (b <<< 80))$
- ξ_i: A non-linear step $(a, b, c, d) \leftarrow (a \oplus ((b <<< 36) \vee (c <<< 67)), b, c, d \oplus ((b <<< 36) \vee (c <<< 67)))$

In this manuscript, we study a Friet design based on share masking and threshold implementation, which we call the new architecture Friet-TI. It adopts two shares masking depicted by 0 and 1 subscripts, effectively copying every limb. The round function is divided into four blocks and each block handles four limbs. Two α blocks work on two shares independently and cover the linear steps μ_1 and μ_2. The round constant is added at only one side. The associated nonlinear step ξ receives only one share. The specificity of the two other blocks β is determined by the nonlinear step ξ that receives two shares, a single share per limb for each β block. To avoid shares recombination, the computation of blocks including their internal variables should be separated from each other while implementing the Friet-TI.

We give the definition of the set of operations α and β up to the wire crossing at the end of the round. α : add round constant $(a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1) \leftarrow (a_0, b_0, c_0 \oplus rc_i, d_0 \oplus rc_i, a_1, b_1, c_1, d_1)$, the linear steps $\mu_1 : (a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1) \leftarrow (a_0, b_0 \oplus a_0 <<< 1, c_0 \oplus a_0 <<< 1, d_0, a_1, b_1 \oplus a_1 <<< 1, c_1 \oplus a_1 <<< 1, d_1)$ and $\mu_2 : (a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1) \leftarrow (a_0 \oplus c_0 <<< 80, b_0 \oplus c_0 <<< 80, c_0, d_0, a_1 \oplus c_1 <<< 80, b_1 \oplus c_1 <<< 80, c_1, d_1)$ and the beginning of the non linear layer $(a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1) \leftarrow (a_0, b_0, c_0 \oplus ((a_0 <<< 67) \vee (b_0 <<< 36)), d_0 \oplus ((a_0 <<< 67) \vee (b_0 <<< 36)), a_1, b_1, c_1 \oplus ((a_1 <<< 67) \vee (b_1 <<< 36)), d_1 \oplus ((a_1 <<< 67) \vee (b_1 <<< 36)))$. Finally β contains the second part of the non-linear layer $(a_0, b_0, c_0, d_0, a_1, b_1, c_1, d_1) \leftarrow (a_0, b_0, c_0 \oplus ((a_1 <<< 67) \vee (b_0 <<< 36)), d_0 \oplus ((a_1 <<< 67) \vee (b_0 <<< 36)), a_1, b_1, c_1 \oplus ((a_0 <<< 67) \vee (b_1 <<< 36)), d_1 \oplus ((a_0 <<< 67) \vee (b_1 <<< 36)))$.

The previous issue can be solved in hardware by applying combinatorial logic in hardwiring the four blocks and separating the α and β layers by inserting registers. On the other hand, the execution of the 4 blocks should be done serially and each limb's shares are separated. Figure 4 shows the Friet-TI hardware architecture. Through this figure, We can see the independence of the variables (e_0, f_0, g_0, h_1, e_1, f_1, g_1 and h_0) from the main inputs.

4.1 Security proof

The security proof of the Friet-TI can be achieved thanks to theorem 3.2. First, we can see that Friet-TI defines a permutation over the shares. If we reverse the figure 4, we see that a_0 can be computed from d'_0, c'_0, b'_0, c'_1, and this can be done for every input share. Thus, we have a permutation over the shares.

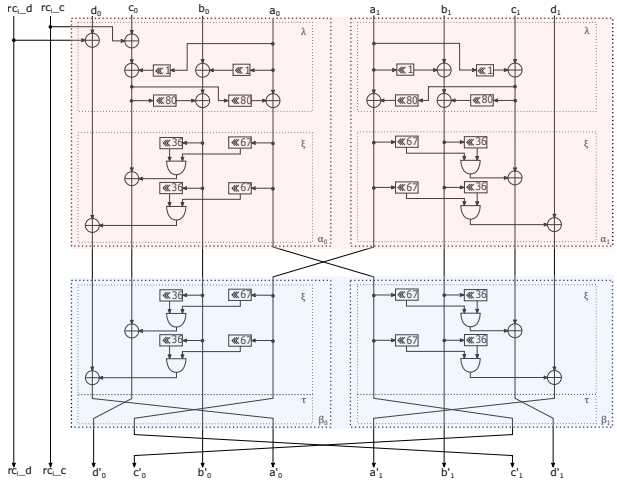

Figure 4: Schematic representation of the 2-share TI of Friet. The red rectangle corresponds to the operation in α, the blue rectangle corresponds to the operation n β.

As stated in [4], if we share a permutation, defining a permutation over the shares, then we have a uniform sharing. Hence, the output sharing is uniform. The next step is to check the independence with the input for every shared function. We can divide the design into four shared functions. The two first ones correspond to the linear layer (δ, τ, μ) and the first part of the non-linear layer (ξ). This corresponds, in the figure 4, to the part taking as input the variables a, b, c, d and with the intermediate results e, g, f, h as outputs. Here, the independence can be checked with the classical TI property of non-completeness.

Non-completeness can be used here due to the existence of a uniform and correct sharing at the input of every round. The uniformity can be checked thanks to the permutation over the shares.

The last two shared functions correspond to the last part of the non-linear operation ξ. They take as input the partial results e, f, g, h, and output the round results a', b', c', d'. If we look in a bitwise manner, we found that every bit depends on at most three bits of the unshared input of the round. Thus, statistical independence can be efficiently checked, equation (1). We use the fact that the round's output is a uniform sharing, as it has been shown in the security proof, to have first-order security. [1]

4.2 More efficient sharing over code

Based on the remark in the footnote and the description of the code used in Friet, we observe that if we apply the code over the shares, we do not have a uniform sharing. In other words

$$(a_0, a_1, b_0, b_1, c_0, c_1, a_0 \oplus b_0 \oplus c_0, a_1 \oplus b_1 \oplus c_1) \tag{2}$$

is not an uniform sharing of $(a, b, c, d = a \oplus b \oplus c)$. In order to be uniform, d must be shared with fresh randomness. However, if we look at only 3 variables out of 4 a, b, c, d, we can see that the corresponding sub-sharing of (2) is uniform.

Thus if we can define shared functions that take shares from at most 3 of the literals, then non-completeness over the literals and the independence seen in the previous version can prove the first order security when we use a sharing as described in (2). In that version, we need just 3×128 bits of randomness to protect Friet against first-order side-channel instead of 4×128.

As discussed in the Friet paper [22], the Friet-TI protect against SIFA [9], following the protection developed in [7]. The Friet-TI with reduced randomness has a similar way of protection against SIFA as the one with full randomness.

The main point of the analysis is the non-completeness over the literals that are required by code-abiding permutation for fault protection. This property can also be used to save some randomness in TI.

[1] In general the bitwise study is not recommended in TI analysis as the concatenation of two uniform sharing may not be uniform; e.g. if (a_1, a_2, a_3) is a uniform sharing of a,$(a_1, a_2, a_3, a_1, a_2, a_3)$ is not a uniform sharing of (a, a), the two sharings of a must be independent. Here, we checked the global uniformity thanks to the permutation over the sharings.

5 SEARCH FOR MORE EFFICIENT 2 STAGES 3-SHARE S-BOXES

Our method can help to implement more efficiently some block cipher in relaxed TI. In [4], it has been shown that only 4-bit s-boxes in the alternative group can be implemented in a TI manner with 3 shares. All quadratic 4-bit s-boxes belong to the alternative group, thus only functions in this subgroup can be implemented with three shares.

We show that it is possible with relaxed TI to obtain a 3-share decomposition of cubic s-boxes that do not belong to the alternative group. Our example only required 2 stages that are optimal in relaxed TI for a degree 3 function.

Using classes notion and notation from [4], we were interested in the s-box $C_1^4 = \{0x0, 0x1, 0x2\ , 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB, 0xC, 0xD, 0xF, 0xE\}$. The class C_1^4 is a class of functions belonging to odd permutations, thus no 3-share decomposition has been found so far.

We recall the algebraic normal form of this 4-bit s-box C_1^4 : $(x, w, v, u) \mapsto (x, w, v, u \oplus (x \vee w \vee v))$. There is only one non-linear term that corresponds to the product $x \vee w \vee v$. Special attention needs to be paid when XORing share's cross-product of this term.

Using the local independence property we can have the following decomposition for 3-share in equation (3). It is easy to check that $((s_0, v_0, w_0, x_0), (s_1, v_1, w_1, x_1), (s_2, v_2, w_2, x_2))$ is a correct and uniform sharing of the s-box. The uniformity comes from the fact we define a bijection over the shares.

In the proposed decomposition, we have decomposed the shares t_i with i in $\{0, 5\}$ into 2 stages 3 shares x_j, w_j, v_j and u_j with j in $\{0, 2\}$. The independence for the t_i can be checked thanks to non-completeness. While for the s_i we check the independence with the input u, v, w, x by verifying equation (1). Finding these 3 shares s-boxes proves that the security of threshold implementations based on relaxed TI can be achieved using different s-boxes cryptographic algorithms.

$$
\begin{aligned}
t_0 =& u_0 \oplus (x_0 \wedge w_0 \wedge v_0) \oplus (x_0 \wedge w_0 \wedge v_1) \oplus (x_0 \wedge w_1 \wedge v_0) \\
&\oplus (x_0 \wedge w_1 \wedge v_1) \oplus (x_1 \wedge w_0 \wedge v_0) \oplus (x_1 \wedge w_0 \wedge v_1) \\
&\oplus (x_1 \wedge w_1 \wedge v_0) \\
t_3 =& (x_0 \wedge w_1 \wedge v_2) \oplus (x_0 \wedge w_2 \wedge v_1) \\
t_1 =& u_2 \oplus (x_2 \wedge w_2 \wedge v_2) \oplus (x_2 \wedge w_2 \wedge v_0) \oplus (x_2 \wedge w_0 \wedge v_2) \\
&\oplus (x_2 \wedge w_0 \wedge v_0) \oplus (x_0 \wedge w_2 \wedge v_2) \oplus (x_0 \wedge w_2 \wedge v_0) \\
&\oplus (x_0 \wedge w_0 \wedge v_2) \\
t_4 =& (x_1 \wedge w_2 \wedge v_0) \oplus (x_1 \wedge w_0 \wedge v_2) \qquad (3) \\
t_2 =& u_1 \oplus (x_1 \wedge w_1 \wedge v_1) \oplus (x_1 \wedge w_1 \wedge v_2) \oplus (x_1 \wedge w_2 \wedge v_1) \\
&\oplus (x_1 \wedge w_2 \wedge v_2) \oplus (x_2 \wedge w_1 \wedge v_1) \oplus (x_2 \wedge w_1 \wedge v_2) \\
&\oplus (x_2 \wedge w_2 \wedge v_1) \\
t_5 =& (x_2 \wedge w_1 \wedge v_0) \oplus (x_2 \wedge w_0 \wedge v_1) \\
\hline
s_0 =& t_0 \oplus t_3 \qquad s_1 = t_1 \oplus t_4 \qquad s_2 = t_2 \oplus t_5
\end{aligned}
$$

All 2-shares decomposition of this class that we found required 4 stages without randomness or using 2 stages and additional randomness. Thus we have the classical trade-off of 2-share vs 3-share implementation between the number of cycles vs. size and randomness requirement.

One can use four shares and compute the output of the s-box in one cycle according to the result published in [4] and not extra randomness. Another option is to use the solution presented in this paper, using 3 shares and two cycles without extra randomness.

Finally, there exist some 2-share implementations that either require 4 cycles and no randomness or 2 cycles but with additional randomness. These results are summarized in the table 2. All these solutions offer different trade-offs for designers according to the parameters they want to optimize: size, latency, and randomness requirements.

Table 2: Comparison of different first order masked implementations of C_1^4

Version	# shares	# cycle	extra rand.
[4]	4	1	no
this paper	3	2	no
2-share TI based on [2]	2	4	no
[11]	2	2	yes

Since there is no correctness for the intermediate results, it is not mandatory to apply the decomposition into quadratic permutation used in [4]. Thus, the use of this permutation cannot be done. Looking at the non-complete Boolean function that we can compose, for 3-share of 4 inputs, we have $(2^8)^2$ functions that can be composed and used in parallel.

6 CONCLUSION

In this work, we have looked at a relaxation of threshold implementation as a countermeasure against first-order side-channel attacks in the presence of glitches. We demonstrated that when correctness is not fulfilled at every register stage, we need another property that replaces non-completeness.

This new property of independence needs to rely on correctness and uniformity in some layers in order to be computationally efficient. We illustrate the power of this approach by studying and proving the security of Friet-TI, we show how to reduce the randomness requirement. Finally, we apply our method to exhibit some new relaxed TI of 4-bit s-boxes. The method offers to designer some new trade-offs in terms of implementation concerning the number of cycles vs the size of the circuit.

Our method allows us to prove the security of some relaxation of TI, it also opens the direction for a new design that can be efficiently protected against side-channel attacks. However, since our method can be computationally expensive, we rely on the correct and uniform layers to limit the computational costs.

Future work. For our analysis, we need to rely on correct and uniform layers that are stored in registers. One open question is whether we need to synchronize the signal at this step. This required analyzing the independence with the input of a shared function with some unstable signal.

Our method can be relaxed to other s-boxes in order to offer to the implementer several possibilities and trade-offs. However, as we do not consider the correct intermediate results, the search space for full s-box decomposition is very large. Some optimization of the search space is necessary in order to look at all 4-bit s-box classes.

REFERENCES

[1] Guido Bertoni, Marco Martinoli, and Maria Chiara Molteni. 2017. A Methodology for the Characterisation of Leakages in Combinatorial Logic. *J. Hardware and Systems Security* 1, 3 (2017), 269–281. https://doi.org/10.1007/s41635-017-0015-0

[2] Begül Bilgin. 2015. *Threshold implementations : as countermeasure against higher-order differential power analysis.* Ph. D. Dissertation. University of Twente, Enschede, Netherlands. http://purl.utwente.nl/publications/95796

[3] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. 2014. Higher-Order Threshold Implementations. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 8874)*, Palash Sarkar and Tetsu Iwata (Eds.). Springer, 326–343. https://doi.org/10.1007/978-3-662-45608-8_18

[4] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. 2012. Threshold Implementations of All 3 ×3 and 4 ×4 S-Boxes. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7428)*, Emmanuel Prouff and Patrick Schaumont (Eds.). Springer, 76–91. https://doi.org/10.1007/978-3-642-33027-8_5

[5] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. 1999. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings (Lecture Notes in Computer Science, Vol. 1666)*, Michael J. Wiener (Ed.). Springer, 398–412. https://doi.org/10.1007/3-540-48405-1_26

[6] Cong Chen, Mohammad Farmani, and Thomas Eisenbarth. 2016. A Tale of Two Shares: Why Two-Share Threshold Implementation Seems Worthwhile - and Why It Is Not. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10031)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.). 819–843. https://doi.org/10.1007/978-3-662-53887-6_30

[7] Joan Daemen, Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Florian Mendel, and Robert Primas. 2019. Protecting against Statistical Ineffective Fault Attacks. *IACR Cryptol. ePrint Arch.* 2019 (2019), 536. https://eprint.iacr.org/2019/536

[8] Lauren De Meyer, Begül Bilgin, and Oscar Reparaz. 2019. Consolidating Security Notions in Hardware Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 3 (2019), 119–147. https://doi.org/10.13154/tches.v2019.i3.119-147

[9] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. 2018. SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 3 (2018), 547–572. https://doi.org/10.13154/tches.v2018.i3.547-572

[10] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. 2019. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. *J. Cryptology* 32, 1 (2019), 151–177. https://doi.org/10.1007/s00145-018-9284-1

[11] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. 2018. Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 3 (2018), 89–120. https://doi.org/10.13154/tches.v2018.i3.89-120

[12] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. 2015. Masking vs. Multiparty Computation: How Large is the Gap for AES? *IACR Cryptol. ePrint Arch.* 2015 (2015), 492. http://eprint.iacr.org/2015/492

[13] Yuval Ishai, Amit Sahai, and David A. Wagner. 2003. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2729)*, Dan Boneh (Ed.). Springer, 463–481. https://doi.org/10.1007/978-3-540-45146-4_27

[14] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. 2005. Side-Channel Leakage of Masked CMOS Gates. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3376)*, Alfred Menezes (Ed.). Springer, 351–365. https://doi.org/10.1007/978-3-540-30574-3_24

[15] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. 2019. Glitch-Resistant Masking Revisited or Why Proofs in the Robust Probing Model are Needed. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 2 (2019), 256–292. https://doi.org/10.13154/tches.v2019.i2.256-292

[16] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. 2006. Threshold Implementations Against Side-Channel Attacks and Glitches. In *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4307)*, Peng Ning, Sihan Qing, and Ninghui Li (Eds.). Springer, 529–545. https://doi.org/10.1007/11935308_38

[17] Emmanuel Prouff and Matthieu Rivain. 2013. Masking against Side-Channel Attacks: A Formal Security Proof. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7881)*, Thomas Johansson and Phong Q. Nguyen (Eds.). Springer, 142–159. https://doi.org/10.1007/978-3-642-38348-9_9

[18] Emmanuel Prouff and Thomas Roche. 2011. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6917)*, Bart Preneel and Tsuyoshi Takagi (Eds.). Springer, 63–78. https://doi.org/10.1007/978-3-642-23951-9_5

[19] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. 2015. Consolidating Masking Schemes. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9215)*, Rosario Gennaro and Matthew Robshaw (Eds.). Springer, 764–783. https://doi.org/10.1007/978-3-662-47989-6_37

[20] Oscar Reparaz, Lauren De Meyer, Begül Bilgin, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Nigel P. Smart. 2018. CAPA: The Spirit of Beaver Against Physical Attacks. In *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)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, 121–151. https://doi.org/10.1007/978-3-319-96884-1_5

[21] Matthieu Rivain and Emmanuel Prouff. 2010. Provably Secure Higher-Order Masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6225)*, Stefan Mangard and François-Xavier Standaert (Eds.). Springer, 413–427. https://doi.org/10.1007/978-3-642-15031-9_28

[22] Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel. 2020. Friet: An Authenticated Encryption Scheme with Built-in Fault Detection. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12105)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, 581–611. https://doi.org/10.1007/978-3-030-45721-1_21