Simple Refreshing in the Noisy Leakage Model

Stefan Dziembowski[1], Sebastian Faust[2], and Karol Żebrowski[1]

[1] University of Warsaw
[2] TU Darmstadt

Abstract. Masking schemes are a prominent countermeasure against power analysis and work by concealing the values that are produced during the computation through randomness. The randomness is typically injected into the masked algorithm using a so-called *refreshing* scheme, which is placed after each masked operation, and hence is one of the main bottlenecks for designing efficient masking schemes. The main contribution of our work is to investigate the security of a very simple and efficient refreshing scheme and prove its security in the *noisy leakage model* (EUROCRYPT'13). Compared to earlier constructions our refreshing is significantly more efficient and uses only n random values and $< 2n$ operations, where n is the security parameter. In addition we show how our refreshing can be used in more complex masked computation in the presence of noisy leakage. Our results are established using a new methodology for analyzing masking schemes in the noisy leakage model, which may be of independent interest.

1 Introduction

Over the last decade cryptographic research has made tremendous progress in developing solid foundations for cryptography in the presence of side-channel leakage (see, e.g., [19] for a recent overview). The common approach in this area – often referred to as "leakage resilient cryptography" – is to first extend the black-box model to incorporate side-channel leakage, and then to propose countermeasures that are provable secure within this model. The typical leakage model considered in the literature assumes an adversary that obtains some partial knowledge about the internal state of the device. For instance, the adversary may learn a few bits of the intermediate values that are produced by the device during its computation.

One of the countermeasures that significantly benefits from such a formal treatment are masking schemes (see, e.g., [6, 8, 9, 11, 18, 21] and many more). Masking is a frequently used countermeasure against power analysis attacks, which de-correlates the internal computation of a device from the observable leakage (e.g., the power consumption). A core ingredient of any secure masking scheme is a *refreshing* algorithm. At a very high level (we will explain this in much more detail below) the refreshing algorithm introduces new randomness into the masked computation, thereby preventing that an adversary can exploit correlations between different intermediate values of the computation.

Since refreshing schemes are computationally expensive a large body of work has explored how to securely improve their efficiency. Unfortunately, one of the most simple and efficient (in terms of computation and randomness) refreshing schemes due to Rivain and Prouff [22] cannot be proven secure; even worse, it was shown in [5] that a simple – though impractical – attack breaks the scheme in the common threshold probing leakage model [18]. In this work we show – somewhat surprisingly – that the simple refreshing of Rivain and Prouff [22] is secure under noisy leakages [21]. Noisy leakages are considered generally to accurately model physical side-channel leakage, and hence our result implies that the simple refreshing can securely replace more complex and expensive schemes in practice.

1.1 Masking schemes

Ingredients of a masking scheme. One of the most common countermeasures against power analysis attacks are masking schemes. Masking schemes work by randomizing the intermediate values produced during the computation of an algorithm through secret sharing. To this end each sensitive variable x is represented by an encoding $\mathrm{Enc}(x) := (x_1, \ldots, x_n)$ and the corresponding decoding function $\mathrm{Dec}(\cdot)$ recovers $x := \mathrm{Dec}(x_1, \ldots, x_n)$. A simple encoding function uses the additive encoding function, which works by sampling x_i uniformly at random from some finite field $\mathbb{F}$ subject to the constraint that $x := \sum_{i=1}^{n} x_i$. If $\mathbb{F}$ is the binary field, then such a masking scheme is typically called *Boolean masking*.

In addition to an encoding scheme, we need secure algorithms to compute with encoded elements. To this end, the algorithm's computation is typically modeled as an *arithmetic circuit* over a finite field $\mathbb{F}$. In such circuits the wires carry values from $\mathbb{F}$ and the gates perform operations from $\mathbb{F}$. At a high-level the circuit is made out of gates that represent the basic field operations (i.e., addition gate denoted "$\oplus$" and multiplication gate denoted "$\otimes$"). Moreover, it may consist of gates for inversion (i.e., outputting $-x$ on input x), and so-called randomness gates *RND* that take no input and produce an output that is distributed uniformly over $\mathbb{F}$. We often assign unique labels to the wires. Each label can be interpreted as a variable whose value is equal to the value that the corresponding wire carries.

Given a circuit built from these gates, a masking scheme then typically works by replacing each of the above operations by a "masked" version of the gate. For instance, in case of the aforementioned additive encoding scheme $(\mathrm{Enc}, \mathrm{Dec})$ the masked version of the $\oplus$ takes as input two encodings $\mathrm{Enc}(x)$ and $\mathrm{Enc}(y)$ and outputs an encoding $\mathrm{Enc}(z)$, where $\sum_i z_i := \sum_i x_i + \sum_i y_i$. Informally, the masked version of a gate is said to be secure if leakage emitted from the internal computation of the masked version of the gate does not reveal any sensitive information.

Refreshing schemes. A key building block to securely compose multiple masked operations to a complex masked circuit is the *refreshing scheme*. The refreshing scheme takes as input an encoding $\overrightarrow{x}^j := (x_1^j, \ldots, x_n^j) = \mathrm{Enc}(x)$ and outputs

a new encoding $(x_1^{j+1}, \ldots, x_n^{j+1}) = \overrightarrow{x}^{j+1}$ of x. By "new encoding" we mean that this procedure should inject new randomness into the encoding, in such a way that the leakage from the previous encodings should not accumulate. In other words: if we periodically refresh the encodings of x (which leads to a sequence of encodings: $\overrightarrow{x}^0 \mapsto \overrightarrow{x}^1 \mapsto \overrightarrow{x}^2 \mapsto \cdots$) then x should remain secret even if bounded partial information about each $\overrightarrow{x}^j$ leaks to the adversary. The operations of computing $\overrightarrow{x}^{j+1}$ from $\overrightarrow{x}^j$ is also called a *refreshing round*, and a circuit that consists of some number of such rounds (and not other operations) is called a *multi-round refreshing circuit*.

A common approach for securely refreshing additive encodings is to exploit the homomorphism of the underlying encoding with respect to addition:[3] one starts by designing an algorithm that samples $(b_1, \ldots, b_n)$ from the distribution Enc(0), and then, in order to refresh an encoding $(x_1^j, \ldots, x_n^j)$ one adds $(b_1, \ldots, b_n)$ to it. Therefore, the refreshed encoding is equal to $(x_1^j + b_1, \ldots, x_n^j + b_n)$. Observe that after Enc(0) is generated, the refreshing can be done without any further computation, by just adding b_i to every x_i^j. Of course in this approach the whole technical difficulty is to generate the encodings of 0 in a secure way (without relying on any assumptions on leakage-freeness of the encoding generation).

The most simple and efficient refreshing scheme originally introduced in [22] uses the "encoding of 0 approach" mentioned above and works as follows (see also Fig. 1 on page 8). In order to refresh $\overrightarrow{x}^j = (x_1^j, \ldots, x_n^j)$, we first sample $b_1^j, \ldots, b_{n-1}^j$ uniformly at random from $\mathbb{F}$ and set $b_n^j := -b_1^j - \ldots - b_{n-1}^j$. Then, we compute the fresh encoding of x as $(x_1^{j+1}, \ldots, x_n^{j+1}) := (x_1^j + b_1^j, \ldots, x_n^j + b_n^j)$. Notice that besides its simplicity the above refreshing enjoys additional beneficial properties including optimal randomness complexity (only $n-1$ random values are used) and minimal circuit size (only $2n-1$ field operations are required). Somewhat surprisingly this simple refreshing scheme turns out to be insecure in the security model of *threshold probing attacks* introduced in the seminal work of Ishai, Sahai and Wagner [18].

Insecurity of simple refreshing. The standard model to analyze the security of masking schemes is the *t-probing model* [18]. In the t-probing model the adversary can (adaptively) select up to t wires of the internal masked computation and learn the values carried on these wires during computation. While originally it was believed that the simple refreshing from above guarantees security for $t = n - 1$ [22], Coron et al. [10] showed that when it is combined with certain other masked operations (e.g., in a masked AES) the resulting construction can be broken using only $\leq t := n/2 + 1$ probes.

An even more devastating attack against this natural refreshing can be shown in the following setting. Consider a circuit that consists of a sequence of n refreshings of an encoding $\overrightarrow{x}^0$. This may naturally happen in a masked key schedule of the AES algorithm, where the secret key is encoded and after each use for

[3] By this we mean that for every x and y we have $\mathrm{Dec}(\mathrm{Enc}(x) + \mathrm{Enc}(y)) = x + y$, where "+" on the left-hand-side denotes the vector addition.

encrypting/decrypting is refreshed. If for each of these refreshings the adversary can learn 2 values, then a simple attack allows to recover the secret (we describe this attack in more detail below). The attack, however, is rather impossible to carry out in practice. In particular, it requires the adversary to learn for the n consecutive executions of the refreshing scheme specific (different) intermediate values.

The noisy leakage model. The attack against the simple refreshing illustrates that in some sense the probing model is too strong. An alternative model is the so-called *noisy leakage model* of Prouff and Rivain [21]. In the noisy leakage model the leakage is not quantitatively bounded but instead it is assumed that the adversary obtains a "noisy distribution" of each value carried on a wire. The noisy leakage model is believed to model real-world physical leakage accurately, and hence is prominently used in practice to analyze the real-world security of physical devices [12].

In [11] it was shown that the noisy leakage model of [21] can be reduced to the *p-random probing model*. In the p-random probing model we assume that the value carried on each wire is revealed independently with probability p. Since in the p-random probing model the adversary looses control over the choice of wire that he learns, the attack against the simple refreshing ceases to work. This raises the question if the simple and most natural refreshing scheme is secure in the p-random probing model. The main contribution of this paper is to answer this question affirmatively.

1.2 Our contribution

We provide a technical outline of our contributions in Sec. 2 and give in the following only a high-level summary of our results.

Simple refreshing. Our main contribution is to analyze the security of the simple refreshing scheme from [22] in the noisy leakage model. In particular, we show that refreshing an encoding $(x_1, \ldots, x_n)$ is secure even if each wire in the refreshing circuit is revealed with constant probability p. Our result directly implies that refreshing an encoded secret k times (where k may be much larger than the security parameter n) remains secure under noisy leakages for constant noise parameter. Such consecutive use of refreshings naturally appears in many practical settings such as the key schedule of the AES mentioned above, or in general for refreshing the secret key between multiple runs of any cryptographic primitive. Since the simple refreshing is *optimal* in terms of circuit size and randomness complexity our result significantly improves the practicality of the masking countermeasure.

Concretely, the simple refreshing requires $n-1$ random values and uses $2n-1$ addition gates to securely refresh an encoding $(x_1, \ldots, x_n)$ in the random probing model (and hence implying security in the noisy leakage model of [21]). In contrast, the most widely used refreshing scheme from Ishai, Sahai and Wagner [18] requires $(n-1)^2/2$ randoms and $2n^2 + n$ addition gates and has been

proven secure only for $p \approx 1/n$, which is significantly worse than ours.[4] Recently, various works provide asymptotically improved refreshing algorithms. In particular, in [1, 3] it was shown how to build a secure refreshing with circuit size $O(n)$, and randomness complexity $O(n)$ for a constant noise parameter p. While asymptotically these constructions are the same as for the simple refreshing analyzed in our work, from a concrete practical point of view these schemes are very inefficient as they are based on expander graphs.

New techniques for proving security. At the technical level, our main contribution is to introduce a new technique for proving security in the random probing model. Our main observation is that probing security can be translated into a question of connectivity between nodes in certain graphs. As an example consider the circuit $\widehat{C}$ executing k times the simple refreshing. It can be represented as a grid G with $k+1$ rows and $n+1$ columns, where in each row we have $n+1$ nodes. The edges between the nodes represent intermediate values that are computed during the execution of the circuit. Leakage of a certain sub-set of wires then corresponds to a sub-graph of G, which we call *leakage diagram*.

We then show that if the "leftmost side" and the "rightmost side" of the grid are connected by a path in the leakage diagram, then this leads to an attack that allows to recover the encoded secret that is refreshed by the circuit $\widehat{C}$. On the other hand, and more importantly, if the two sides of the diagram are *not connected* by a path in the leakage diagram, then we show that the adversary does not learn any information about the encoded secret from the leakage. The above can be extended to arbitrary masked arithmetic circuits, in which the graphs representing the circuit are slightly more involved.

The above allows us to cast security against probing leakage as a question about connectivity of nodes within a graph. To show security in the p-random probing model we then need to bound the probability that the random sub-graph of G representing the leakage contains a path that connects the two sides of G. The main challenge is that although in the p-random probing model each wire leaks independently with probability p, in our graph representation certain edges are more likely to be part of the leakage diagram. Even worse, the events of particular edges of G ending up in the leakage diagram are not independent. This significantly complicates our analysis. We believe that the techniques introduced in our paper are of independent interest and provide a novel tool set for analyzing security of masked computation in the random probing model.

Extension to any masked computation. As our last contribution we show how to use the simple refreshing as part of a more complex masked computation. To this end, we study the security of the masking compiler provided by Ishai, Sahai and Wagner [18] when using the simple refreshing described above. Notably, we first show that the simple refreshing can be used to securely compose any affine masked operations. This result is important because it shows for the first

[4] We expect that also the refreshing from [18] is secure for some constant probability p, but we did not analyze its security.

time that the most natural and efficient way to carry out affine computation in the masked domain is secure against noisy leakages. Compared to the standard construction of [18] we save a factor of n in circuit and randomness complexity. Moreover, at the concrete level we make huge practical improvements when compared to the recent works of [1, 3], which use expander graphs and algebraic geometric codes.

Finally, we show that the simple refreshing can also be securely composed with the masked multiplication of [18]. Since the masked multiplication of [18] itself is a composable refreshing [11], this result is maybe not so surprising. Nevertheless, it shows that combining from a complexity point of view optimal masked computation with the ISW masked multiplication results into general masked computation that is secure in the random probing model.[5]

1.3 Other related work

A large amount of work proves different formal security guarantees of masking schemes (see, e.g., [1, 7, 18, 20, 21] and many more), and we only discuss the most relevant work.

Noisy leakage model. As already mentioned most relevant for us is the so-called noisy leakage model introduced in the work of Prouff and Rivain [21] and further refined by Duc et al. [11]. In the later it was also shown that the p-random probing model is closely related to the noisy leakage model. Since both [11, 21] require $p \approx 1/n$, one important goal of research is to improve the noise parameter p. There has recently been significant progress on this. In [1, 3] it was shown how to securely compute in the random probing model for constant p. Further improvements are made in [2, 16], where the later achieves security under a quasi-constant noise for a construction with complexity $O(n \log(n))$ avoiding heavy tools such as expander graphs and AG codes. Another line of work investigates relations between different noisy leakage models [14, 17] and provides tight relations between them. A more practical view on noisy leakage – and in particular a quantitative study of its relation to real-world leakage – was given by Duc et al. [12].

Refreshing schemes and their usage. Refreshing schemes have always been a core ingredient of masking schemes. Their randomness consumption is, however, often the bottleneck for an efficient masked implementation[6]. Hence, an important goal of research is to minimize the overheads resulting from the use of refreshing. There are two main directions to achieve this. First, we may improve the

[5] Recall that for the ISW scheme it is known that $p \approx 1/n$ as otherwise there is an attack against the masked multiplication. Thus, our result requires a similar bound on p in the general case. It is an interesting open question if we can combine the simple refreshing with masked multiplications that are secure for constant p, e.g., the schemes from [1, 3].

[6] Notice that true randomness is hard to generate in practice, and producing securely pseudorandomness is costly as we need to run, e.g., an AES.

refreshing algorithm itself. In particular, in [1, 3] it was shown how to build a secure refreshing with circuits size and randomness complexity $O(n)$ for a constant noise parameter p. While asymptotically optimal from a concrete practical point of view these schemes are very inefficient as they are based on expander graphs. A second direction to improve on the costs for refreshing is to reduce the number of times the refreshing algorithms are used. This approach was taken by several works [6, 8, 9] which develop tools for placing the refreshing algorithm in an efficiency optimizing way without compromising on security. It is an interesting question for future research to develop tools and methods that securely place the simple refreshing within a complex masked circuit.

2 Our approach informally

As a simple example of circuit to present our approach let us consider a circuit $\widehat{C}$ (in the following the "hat notation" will denote masked/transformed circuits) that is a k-round refreshing circuit. This circuit consist of k consecutive subcircuits that we call *refreshing gadgets* $\widehat{R}$, presented in Fig. 1. Note that in addition to the notation from Sect. 1.1 we also use terms c_i^j that denote the partial sums: $c_i^j = b_1^j + \cdots + b_i^j$ (for consistency define c_0^j and c_n^j to be always equal to 0). It is a simple fact that the adversary can learn the encoded secret for $k = n$ even if just 2 wires from each refreshing gadget leak to her (and no additional leakage is given), namely x_{j+1}^j and c_{j+1}^j. We recall this attack in the full version of the paper [15]. Similar attacks for different refreshing schemes have been shown in [5, 13]. This attack strongly relies on the fact that the adversary can *choose* which wires she learns. As discussed in the introduction, in the weaker p-random probing model it is very unlikely that the adversary will be lucky enough to learn x_{j+1}^j and c_{j+1}^j in each round (unless p is close to 1). Of course, the fact that one particular attack does not work, does not immediately imply that the scheme is secure.

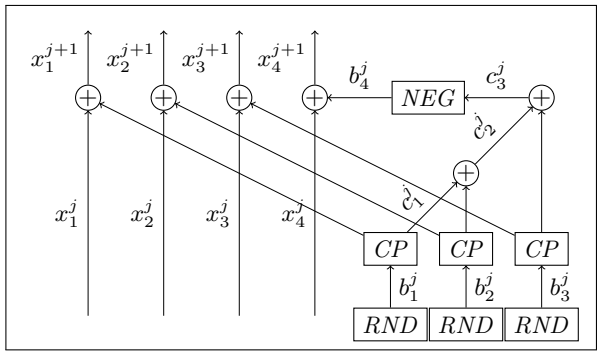
Relaxing the leakage model. As already mentioned in Sect. 1.2 our first main contribution is a formal proof that indeed this simple refreshing procedure is secure in the p-random probing model. Our starting point is the natural question: *can we characterize the leakages which allow the adversary to compute the secret?* We answer this question affirmatively by introducing the notion of *leakage diagrams*, which we explain below (for formal definitions see Sect. 4.4).

Leakage diagrams. Essentially, the leakage diagrams are graphs that can be viewed as abstract representations of the leakage that occurred during the evaluation of a circuit. For a moment let us focus only on leakage diagrams that correspond to k-round refreshing circuits $\widehat{C}$. Let $x_1^0, \ldots, x_n^0$ be some initial encoding of the secret x. In this case the leakage diagram will be a subgraph of a $(n+1) \times (k+1)$ grid G with edges labeled x_i^j and c_i^j as on Fig. 2.

To illustrate how the leakage diagrams are constructed take as an example a 2-round refreshing circuit (with $n = 3$) that is depicted on Fig. (3a). Note

$(b_1^j, \ldots, b_{n-1}^j) \leftarrow \mathbb{F}^{n-1}$
$c_0^j := 0$
for $i = 1, \ldots, n-1$ **do**
  $c_i^j := c_{i-1}^j + b_i^j$
$b_n^j := -c_{n-1}^j$
for $i = 1, \ldots, n$ **do**
  $x_i^{j+1} := x_i^j + b_i^j$

(a) Pseudocode of the simple refreshing gadget $\widehat{R}$.



(b) Corresponding circuit (for $n = 4$).

Fig. 1: The refreshing gadget. The "j" superscript is added for the future reference (e.g. on Fig. (3a)).

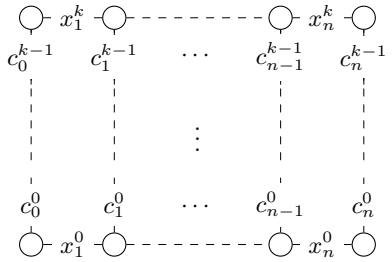
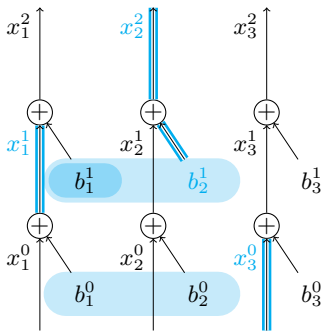


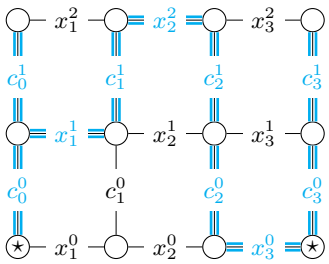
Fig. 2: Graph G corresponding to the k-round refreshing circuit. It has $k+1$ rows. In each jth row (for $j = 0, \ldots, k$) it has $n+1$ vertices connected with edges (there is an edge labeled with "x_i" between the ith and $(i+1)$st vertex). It also has an edge between every pair of ith vertices (for $i = 0, \ldots, n$) in the jth and $j+1$st row. This edge is labeled with "c_i^j".

that this picture omits the part of circuit that is responsible for generating the b_i^j's, and in particular the wires carrying the c_i^j values are missing on it. This is done in order to save space on the picture. Let L be the wires that leaked in the refreshing procedure. Suppose the leaking wires are x_3^0, x_1^1, x_2^3, and b_2^1, which is indicated by double color lines over the corresponding edges on Fig. (3a). We also have to remember about the c_i^j's that were omitted on the figure and can also leak. Recall that every c_i^j is equal to a sum $b_1^j + \cdots + b_i^j$. Hence, the leakage from c_i^j is indicated by a shaded colored region around $b_1^j, \ldots, b_i^j$. Let us assume that c_2^0, c_1^1, and c_2^1 are leaking, and therefore the shaded regions on Fig. (3a) are placed over b_1^1, and the pairs $(b_1^0, b_2^0), (b_1^1, b_2^1)$.

The corresponding leakage diagram is a subgraph of the graph G from Fig. 2 with $k := 2$ and $n := 3$. The leakage diagram $S(L)$ has the same vertices as



(a) A circuit with leaking wires marked with colored double lines. Additionally wires $c_2^0 (= b_1^0 + b_2^0)$, $c_1^1 (= b_1^1)$, and $c_2^1 (= b_1^1 + b_2^1)$ leak, which is indicated by colored shaded areas around "b_1^0 b_2^0", "b_1^1", "b_1^1 b_2^1".



(b) The corresponding leakage diagram. We show how the adversary can compute the sum of edges x_1^0, x_2^0, and x_3^0. The leftmost and the rightmost vertices of the row containing these edges are marked with "$\circledast$".

Fig. 3: A leaking circuit and its corresponding leakage diagram.

G, but it has only a subset of its edges. Informally, the labels on the edges of $S(L)$ are variables that suffice to fully reconstruct the leakage from the circuit. More precisely: given these values one can compute the same leakage information that the adversary received. Going back to our example: the leakage diagram corresponding to the leakage presented on Fig. (3a) is depicted on Fig. (3b), on which the members of $S(L)$ are marked with double colored lines. The set $S(L)$ is created according to the following rules. First, we add to $S(L)$ all the edges labeled x_i^j and c_i^j if the corresponding wires are in L. For this reason $S(L)$ on Fig. (3b) contains $x_3^0, x_1^1, x_2^2, c_2^0, c_1^1$, and c_2^1. Handling leaking b_i^j's is slightly less natural, since graph G does not contain edges labeled with the b_i^j's. To deal with this, we make use of the fact that every b_i^j can be computed from c_i^j and c_{i-1}^j (as $b_i^j = c_i^j - c_{i-1}^j$). Hence, for every b_i^j from L we simply add c_i^j and c_{i-1}^j to $S(L)$. For this reason we add c_1^1 and c_2^1 to L (as b_2^1 is in L). This approach works, since, as mentioned above, the edges in $S(L)$ should suffice to fully reconstruct L. Note that in some sense we are "giving out too much" in the leakage diagram (as c_i^j and c_{i-1}^j cannot be uniquely determined from b_i^j). Fortunately, this "looseness" does not cost us much in terms of parameters, while at the same time it greatly simplifies our proofs. Finally, we add to $S(L)$ all the edges labeled with c_0^j and c_n^j (i.e.: the leftmost and the rightmost columns in G). We can do it since these edges are always equal to 0 and hence the adversary knows them "for free".

What the adversary can learn from a leakage diagram. The ultimate goal of the adversary is to gain some information about the encoded secret. To achieve this it is enough that she learns the sum of all the x_i^j's from some row of the diagram.

We now show how in case of leakage from Fig. 3 the adversary can compute $x_1^0 + x_2^0 + x_3^0$ from the values that belong to the $S(L)$ (i.e. those that are marked with double colored lines on Fig. (3b)). Using the facts that $x_i^{j+1} = x_i^j + b_i^j$ and $c_{i+1}^j = c_i^j + b_{i+1}^j$ several times we have:

$$x_1^0 + x_2^0 + x_3^0 = (x_1^1 - b_1^0) + (x_2^1 - b_2^0) + x_3^0 = x_1^1 + x_2^1 - (b_1^0 + b_2^0) + x_3^0 = \\ x_1^1 + (x_2^2 - b_2^1) - c_2^0 + x_3^0 = x_1^1 + x_2^2 + c_1^1 - c_2^1 - c_2^0 + x_3^0$$

where all the variables on the right hand side belong to $S(L)$. It is easy to see that the reason why the adversary is able to compute $x_1^0 + x_2^0 + x_3^0$ is that the leftmost and the rightmost nodes in the row containing edges labeled with variables were connected. These nodes are indicated with the "$\circledast$" symbol on Fig. (3b).

Since the leftmost and the rightmost columns always belong to the leakage diagram, thus in general a similar computation is possible when these two columns are connected. Our first key observation is that if these columns are *not* connected, then the secret x remains secure. We state this fact below in the form of a following informal lemma.

Informal Lemma 1 *Consider a multi-round refreshing circuit. Let L be the set of leaking wires. Let E denote the event that the leftmost and the rightmost columns of $S(L)$ are connected. If E did not occur then the adversary gains no information about the secret.*

This informal lemma is formalized as Claim 5 (in the full version of this paper [15]), where it is also stated in a more general form, covering the case of more complicated circuits (i.e. those that perform some operations in addition to refreshing). The rest of this section is organized as follows. In Sect. 2.1 we outline the main ideas behind the proof on Informal Lemma 1, in Sect. 2.2 we sketch the proof of the upper bound on the probability of E. This, together with the Informal Lemma 1 shows the security of our multi-round refreshing construction. Then in Sect. 2.3 we describe how these ideas can be generalized to arbitrary circuits. Besides of presenting the intuitions behind our formal proof, the goal of this part is also to introduce some more terminology that is useful later (e.g.: the "modification vectors"). In the sequel we use the following convention: if G is a labeled graph such that the labels on its edges are unique, then we sometimes say "edge λ" as a shortcut for "edge *labeled* with λ". The same convention applies to circuits and wires.

2.1 Proof sketch of Informal Lemma 1

Here we present the main ideas behind the proof of Informal Lemma 1. Consider a k-round refreshing circuit $\widehat{C}$ that takes as input a secret shared over n wires. For two arbitrary field elements $x^0, x^1 \in \mathbb{F}$ consider experiments of applying $\widehat{C}$ to their random encodings. In the proof we consider a fixed set L of leaking wires in $\widehat{C}$. Assume that event E did not occur, i.e., the leftmost and the rightmost columns of the leakage diagram are disconnected. To prove Informal Lemma 1,

it is enough to show that for the distributions of the values of wires in L are identical in both experiments (following the standard approach in cryptography this formally captures the fact that the adversary "gains no information about the secret"). We do it using a hybrid argument. Namely, we consider a sequence of experiments denoted $\mathrm{Exp}_A^0, \mathrm{Exp}_B^0, \mathrm{Exp}_C, \mathrm{Exp}_B^1$, and Exp_A^1 (see below), such that: (a) Exp_A^ℓ (for $\ell = 0, 1$) is equal to the original experiment in which x^ℓ is refreshed k times, and (b) the view of the adversary is identical for each pair of consecutive experiments on this list (and hence it is identical for all of them).

$\mathbf{Exp}_A^\ell$:

Sample $\overrightarrow{x}^{0,\ell} \leftarrow \mathrm{Enc}(x^\ell)$.
For $j = 0$ to $k-1$ do:
1. sample $\overrightarrow{b}^{j,\ell} \leftarrow \mathrm{Enc}(0)$,
2. let $\overrightarrow{c}^{j,\ell} := f(\overrightarrow{b}^{j,\ell})$,
3. let $\overrightarrow{x}^{j+1,\ell} := \overrightarrow{x}^{j,\ell} + \overrightarrow{b}^{j,\ell}$,

$\mathbf{Exp}_B^\ell$:

Sample $\overrightarrow{x}^{0,\ell} \leftarrow \mathrm{Enc}(x^\ell)$.
For $j = 0$ to $k-1$ do:
1. sample $\overrightarrow{x}^{j+1,\ell} \leftarrow \mathrm{Enc}(x^\ell)$,
2. let $\overrightarrow{b}^{j,\ell} := \overrightarrow{x}^{j+1,\ell} - \overrightarrow{x}^{j,\ell}$,
3. let $\overrightarrow{c}^{j,\ell} := f(\overrightarrow{b}^{j,\ell})$.

$\mathbf{Exp}_C$:

Sample $\overrightarrow{x}^{0,1} \leftarrow \mathrm{Enc}(x^0) + (x^1 - x^0) \cdot \overrightarrow{m}^0$.
For $j = 0$ to $k-1$ do:
1. sample $\overrightarrow{x}^{j+1,1} \leftarrow \mathrm{Enc}(x^0) + (x^1 - x^0) \cdot \overrightarrow{m}^{j+1}$,
2. let $\overrightarrow{b}^{j,1} := \overrightarrow{x}^{j+1,1} - \overrightarrow{x}^{j,1}$,
3. let $\overrightarrow{c}^{j,1} := f(\overrightarrow{b}^{j,1})$.

Fig. 4: The sequence of experiments.

Extending the notation from the pseudocode given in Fig. (1a), we will add for future reference to the procedure that refreshes a secret x^ℓ (with $\ell \in \{0, 1\}$) a superscript "ℓ" to all the labels, i.e., denote $\overrightarrow{x}^{j,\ell} := (x_1^{j,\ell}, \ldots, x_n^{j,\ell})$, $\overrightarrow{b}^{j,\ell} := (b_1^{j,\ell}, \ldots, b_n^{j,\ell})$ and $\overrightarrow{c}^{j,\ell} := (c_1^{j,\ell}, \ldots, c_n^{j,\ell})$. Note that all the operations in the refreshing circuit are linear, and in terms of linear algebra this experiment (repeated k times) can be described as Exp_A^ℓ on Fig. 4, where f is a linear function defined as $f(\overrightarrow{b}^{j,\ell}) = (b_1^{j,\ell}, b_1^{j,\ell} + b_2^{j,\ell}, \ldots, b_1^{j,\ell} + \cdots + b_n^{j,\ell})$. It is easy to see that the experiment Exp_B^ℓ depicted on Fig. 4 (where the $\overrightarrow{x}^{j,\ell}$'s are chosen *first*, and then $\overrightarrow{b}^{j,\ell}$ is computed as their difference) has the same distribution of the variables as Exp_A^ℓ. To finish the proof of the Informal Lemma 1 we need to construct an experiment Exp_C, such that the view of the adversary in experiments Exp_B^0, Exp_C and Exp_B^1 is identical. Our approach to this is as follows. Based on the leakage diagram $S(L)$ (and *independently* from the choice of the $x_i^{j,\ell}$'s) we construct carefully crafted vectors $\overrightarrow{m}^0, \ldots, \overrightarrow{m}^k \in \{-1, 0, 1\}^n$ that we call *basic modification vectors* such that for every j we have that $m_1^j + \cdots + m_n^j = 1$ (where $(m_1^j, \ldots, m_n^j) = \overrightarrow{m}^j$). These vectors have to satisfy also some other conditions (that we define in the full version). See Fig. 5 for an example. The modification of Exp_B^ℓ is denoted Exp_C and presented on Fig 4.

Two claims (discussed extensively in the full version of this paper [15]) that allow the proof to go through are that (1) the joint distribution of the variables $\overrightarrow{x}^{j,1}$, $\overrightarrow{b}^{j,1}$, and $\overrightarrow{c}^{j,1}$ in Exp_C is the same as in Exp_B^1, and (2) the view of the adversary are distributed identically in Exp_C and in Exp_B^0. What remains is to show how the basic modification vectors are constructed. Let LS be the connected component of $S(L)$ that contains its leftmost column. By assumption, E did not occur so LS does *not* contain the rightmost column of $S(L)$. This makes it possible to construct the basic modification vectors with desired properties. For each j construct $\overrightarrow{m}^j = (m_1^j, \ldots, m_n^j)$ according to the following rules: (i) if

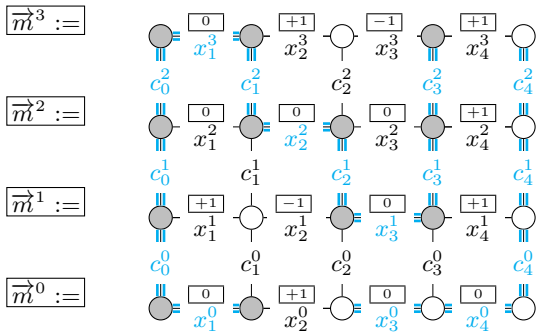


Fig. 5: The example of the leakage diagram with leakage indicated with double colored lines. The nodes of the connected component LS (containing the leftmost column) are indicated with gray color. The modification vectors $\overrightarrow{m}^j$ and their coordinates are placed in boxes (e.g.: $\overrightarrow{m}^0 := (0, +1, 0, 0)$).

the left node of the edge "x_i^j" *does* belong to LS and its right node *does not* belong to LS, then let m_i^j be equal to $+1$, (ii) if the left node of the edge "x_i^j" *does not* belong to LS and its right node *does belong* to LS, then let m_i^j be equal to -1, and (iii) let all the other m_i^j's be equal to 0. An example of how the basic modification vectors are constructed is presented on Fig. 5 (these vectors and their coordinates are marked there with numbers in boxes). As it turns out (see Lemma 6 in the full version of this paper [15] for a generalization of this statement) these rules guarantee that the requirement that "$m_1^j + \cdots + m_n^j = 1$", and all other necessary conditions, are satisfied.

2.2 Bounding the probability of E

To show how we derive a bound on the probability of E we take a closer look at how, from the probabilistic point of view, the leakage diagram is constructed (see p. 9). By definition, it is a subgraph of a graph G from Fig. (2). Recall that in our experiment every wire of the circuit $\widehat{C}$ leaks independently at random

with probability p. The leakage diagram $S(L)$ corresponding to leakage L is a random subgraph of G.

Let us now analyze the distribution of $S(L)$. It is easy to see that every edge "x_i^j" is added to $S(L)$ independently with probability p. Unfortunately, the situation is slightly more complicated when it comes to the c_i^j's. Recall that c_i^j's can be added to $S(L)$ for three reasons. The first (trivial) reason is that $i = 0$ or $i = n$. The second reason is that the wire "c_i^j" leaks in $\widehat{C}$ (i.e.: it belongs to L). The third reason is that the wire b_i^j or b_{i+1}^j leaks in $\widehat{C}$. Because of this, the events $\{$"c_i^j belongs to $S(L)$"$\}_{i,j}$ are *not* independent, and the probability of each of them may *not* equal to p.[7]

Let us look at the "non-trivial" edges in $S(L)$, i.e., the x_i^j's and the c_i^j's such that $i \in \{1, \ldots, n-1\}$. Let $\mathcal{U}$ be the variable equal to the set of non-trivial edges in $S(L)$. To make the analysis of the leakage diagram simpler it will be very useful to eliminate the dependencies between the "$c_i^j \in \mathcal{U}$" events. We do it by defining another random variable $\mathcal{Q}$ (that takes the same values as $\mathcal{U}$), and that has the following properties.

1. It is "more generous to the adversary", i.e., for every set $\mathcal{C}$ of the edges we have that
$$\Pr[\mathcal{C} \subset \mathcal{Q}] \geq \Pr[\mathcal{C} \subset \mathcal{U}] \tag{1}$$
(we will also say that the distribution of Q *covers the distribution of* $\mathcal{U}$, see Def. 1 on p. 17), and
2. The events $\{v \in \mathcal{Q}\}$ (where v is a non-trivial edge) are independent and have equal probability. Denote this probability q, and say that $\mathcal{Q}$ has a *standard distribution* (see Def. 2 on p. 17).

Now, consider an experiment $\mathrm{Exp}_{\mathcal{Q}}$ of constructing a leakage diagram when the "$c_{i,j}$" and "$x_{i,j}$" edges are chosen according to $\mathcal{Q}$. More precisely: let the edges in the leakage diagram be sampled independently according to the following rules: the $\{c_0^j\}$'s and $\{c_n^j\}$'s are chosen with probability 1, and the remaining $\{c_i^j\}$'s are chosen with probability q. It is easy to see that, thanks to Eq. (1), the probability of E in $\mathrm{Exp}_{\mathcal{Q}}$ is at least as high as in the probability in the original experiment. Hence, to give a bound on the probability of E it suffices to bound the probability of this probability in $\mathrm{Exp}_{\mathcal{Q}}$. Thanks to the independence of the events $\{c_i^j \in \mathcal{Q}\}_{i,j} \cup \{x_i^j \in \mathcal{Q}\}_{i,j}$ bounding the probability of E in $\mathrm{Exp}_{\mathcal{Q}}$ becomes a straightforward probability-theoretic exercise. For the details on how it is done see full version of this paper [15].

2.3 Generalizations to arbitrary circuits

As mentioned in Sect. 1.2, our final main contribution is a circuit compiler that uses the simple refreshing together with gadgets that perform the field

[7] For example: it is easy to see that if we know that $c_i^j \in S(L)$ then the event "c_{i+1}^j belongs to $S(L)$" becomes more likely (because leakage of b_{i+1}^j is more likely).

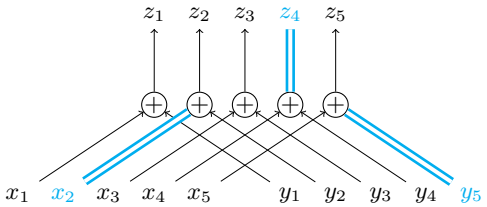
operations. We follow the standard method of constructing compilers in a "gate-by-gate" fashion (see, e.g., [18], and the follow up work). A compiler takes as input a circuit C (for simplicity assume it has no randomness gates) and produces as output a transformed circuit $\widehat{C}$ (that contains randomness gates RND). More concretely a wire carrying x in C gets transformed into a *bundle* of n wires carrying a random encoding of x. Every gate Γ in C is transformed into a "masked gate" $\widehat{\Gamma}$. For example, an addition gadget will have $2n$ inputs for n-share encodings of two values a and b, and n output wires that will carry some encoding of $a+b$. The masked input gates simply encode the secret (they have one input and n outputs). The masked output gates decode the secret (they have n inputs and one outputs). These two gadgets are assumed to be leak-free. They are also called: *input encoder* $\widehat{I}$ and *output decoder* $\widehat{O}$, respectively. For technical reasons, in our construction we insert the refreshing gadgets between the connected gadgets.

The main challenge in extending our ideas to such general circuits is that we need to take into account the leakage from wires of the individual gadgets, and represent them in the leakage diagram. We do it in such a way that unless an event E occurs, we are guaranteed that the adversary gained no information about the secret input. By the "event E" we mean a generalization of the event E (from the previous sections) to more complicated leakage diagrams. More concretely (see Sect. 4.1 for details) our approach is to represent each gadget $\widehat{\Gamma}$ in the graph G with a path $N_0^{\widehat{\Gamma}} - \cdots - N_n^{\widehat{\Gamma}}$ of length n and to "project" the leaking wires of the given gadget onto the edges of the path. Technically, this is done be defining, for every gadget $\widehat{\Gamma}$, a leakage *projection function* (see Sect. 4.3) that describes how a leakage from an internal wire is mapped on the path.

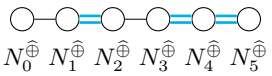
A projection function P, by definition, takes as argument a leaking wire w in a gadget $\widehat{\Gamma}$, and returns a subset of $[n]$ (usually of size 1 except for some wires in the multiplication gadget). We can refer to a projection of a set of wires in $\widehat{\Gamma}$ defined in a natural way as $P(\{w_1,\ldots,w_l\}) := P(w_1)\cup\cdots\cup P(w_l)$. One of the requirements that we impose on the function P is the following: every set of probes $\{w_1,\ldots,w_l\}$ (regardless of its size) from $\widehat{\Gamma}$ can be simulated knowing only input shares of indices in the projection $P(\{w_1,\ldots,w_l\})$ within each input bundle. Notice that it makes our definition of the gadget security similar in spirit to the existing definitions for the t-probing leakage model, like d-non-interference. One of the differences is that we care not only about the number of input shares that suffice to simulate the leakage, but also take into account their indices in a particular input bundle. Having a leakage projection function P defined for a gadget $\widehat{\Gamma}$, we will represent a leakage from that gadget in the leakage diagram as a subset of the edges from the path in G: $N_0^{\widehat{\Gamma}} - \cdots - N_n^{\widehat{\Gamma}}$. The positions (with the edge $N_0^{\widehat{\Gamma}} - N_1^{\widehat{\Gamma}}$ being the 1st one) of these edges in the path are taken from the set $P(\{w_1,\ldots,w_l\})$, when the wires $w_1,\ldots,w_l$ are leaking. This way we can "project" any given leakage from a gadget onto the path of length n in the leakage diagram.

As an example consider the addition gadget "$\widehat{\oplus}$" that computes an encoding $\overrightarrow{z}$ of $z=x+y$ as $\overrightarrow{z} := \overrightarrow{x} + \overrightarrow{y}$ (where $\overrightarrow{x}$ and $\overrightarrow{y}$ are encodings of x and y,

respectively). The leakage projection function $P_{\widehat{\oplus}}$ for this gadget is defined as follows. Each input wire that is on ith position in the input bundle is projected onto the set $\{i\}$, i.e., $P_{\widehat{\oplus}}(x_i) = \{i\}$ and $P_{\widehat{\oplus}}(y_i) = \{i\}$. Moreover, projection of the output wires is defined similarly, namely $P_{\widehat{\oplus}}(z_i) = \{i\}$. It is easy to see that with such projection function the above mentioned simulation requirement is satisfied. For example, the leakage illustrated on Fig. (6a) can be simulated knowing 3 input shares from each input bundle, namely x_2, x_4, x_5 and y_2, y_4, y_5. On the leakage diagram we represent this particular leakage from the addition gadget with 3 edges, as illustrated on Fig. (6b). Note that the addition gate is simple, and hence the projection function for it is rather straightforward. The projection function for a multiplication gadget is more involved (see Sect. 4.2).



(a) An example of leakage from the addition gadget "$\widehat{\oplus}$" (marked with double colored lines).



(b) The corresponding "projected" leakage in the leakage diagram (marked with double colored lines).

Fig. 6: Leakage from an addition gadget and the corresponding "projected" leakage. This is a valid projection, since it is enough to know x_2, x_4, x_5 and y_2, y_4, y_5 to simulate the leakage.

Having the projections of leakages for individual gadgets defined, we can generalize the idea of a leakage diagram $S(L)$ presented in previous sections from simple sequential k-round refreshing circuits to arbitrary private circuits built according to our construction. Recall that we insert a refreshing gadget between each pair of connected gadgets. The leakage from each individual gadget is projected onto a respective path in the leakage diagram, and the leakage from the remaining wires, i.e., wires used to generate encodings Enc(0) between two gadgets is "projected" onto the edges connecting the respective paths (analogue of the edges c_i^j's from previous sections). See Sect. 4.4 for the details. Overall, we obtain a graph that is similar to the leakage diagrams from the previous sections, but it is more general. In case of an example depicted on Fig. (7) the leakage from the gadget $\widehat{\Gamma}_1$ induces a projection set $\{3\}$. This fact is represented by including the edge $N_2^{\widehat{\Gamma}_1} - N_3^{\widehat{\Gamma}_1}$ into the leakage diagram.

A crucial property of such leakage diagrams is that the generalization of the Informal Lemma 1 still holds: the notion of the leftmost and the rightmost column are generalized to the leftmost and the rightmost *sides* (respectively).

On Fig. (7) the leftmost side is a graph consisting of nodes $N_0^{\widehat{\Gamma}_1}, N_0^{\widehat{\Gamma}_2}, N_0^{\widehat{\Gamma}_3}, N_0^{\widehat{\Gamma}_4}$, and $N_0^{\widehat{\Gamma}_5}$, while the rightmost one consists of nodes $N_3^{\widehat{\Gamma}_1}, N_3^{\widehat{\Gamma}_2}, N_3^{\widehat{\Gamma}_3}, N_3^{\widehat{\Gamma}_4}$, and $N_3^{\widehat{\Gamma}_5}$. We now define the event E as: "the leftmost and the rightmost sides are connected". For example E does *not* hold for the diagram on Fig. (7). To make it easier to verify this fact, we indicate (with gray color) the nodes connected with the leftmost side.

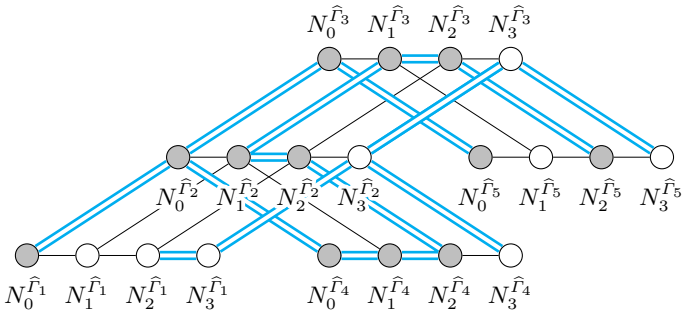


Fig. 7: An example of a leakage diagram for a transformed circuit $\widehat{C}$ with 5 gadgets. The nodes connected with the leftmost side are marked in gray.

When using the leakage projection functions we encounter the following problem that is similar to the "lack of independency problem" described in Sect. 2.2. Namely, it may happen that the events different edges become part of the projected set are *not* independent (this is, e.g., the case for the multiplication gadget in Sect. 4.2). We handle this problem in a similar way as before (see points 1 and 2 on page 13). That is: we define a "more generous" *leakage projection distribution* that (1) "covers" the original distribution, and (2) is "standard" (see the aforementioned points for the definition). Let q be the parameter denoting the probability in the standard distribution. This parameter, of course, depends on the probability p with which a wire leaks. A function that describes this dependence is called *projection probability function*. Every gadget in our construction comes with such a function. See Sect. 4.1 for a formalization of these notions.

Our construction is modular and works for different implementations of the addition and multiplication gadgets, assuming that they come with the leakage projection that satisfies certain conditions (see Thm. 1 on p. 25). We show (see Sect. 4.2) that the standard gadgets from the literature (including the ISW multiplication gadget [18]) satisfy this condition. Note that the construction and reasoning regarding the refreshing circuit presented in previous sections are special case of the construction and the security proof for the general arithmetic circuit. Indeed, we can treat each bundle between refreshing gadgets as an "identity gadget" (see Sect. 4.2).

Organization of the rest of the paper. In the next two sections we describe the technical details of the ideas outlined above. Sect. 3 consists of formal definitions, and Sect. 4 contains the details of our constructions. Due to the lack of space, some parts of these sections are moved to the full version of this paper [15].

3 Formal definitions

We start by presenting formal definitions of some notions that were introduced informally in Sect. 2. Let us start with introducing some standard notation. In the sequel $[n]$ denotes the set $\{1, 2, \ldots, n\}$. We write $x \leftarrow \mathcal{X}$ when the element x is chosen uniformly at random from the finite set $\mathcal{X}$. A circuit C is *affine* if it does not use product gates. A *statistical distance* between two random variables X_0 and X_1 (distributed over some set $\mathcal{X}$) is defined as $\Delta(X_0; X_1) := 1/2 \cdot \sum_{x \in \mathcal{X}} |\Pr[X_0 = x] - \Pr[X_1 = x]|$. If $\Delta(X_0; X_1) \leq \epsilon$ then we say that X_0 *and* X_1 *are* ϵ*-close*. We assume a fixed security parameter n, i.e. every wire in C will be represented by a bundle of n wires in $\widehat{C}$.

Assumptions about the circuit. For syntactic purposes we introduce special input encoding I gate and output decoding O gate used in original circuit C that simply implement the identity function, but will be transformed to $\widehat{I}$ and $\widehat{O}$ gadgets in $\widehat{C}$ (see Sect. 2.3). Gate I is required at every input wire of the circuit C that will be a subject to our compiler, and similarly O gate is required at every output gate of C. We call such circuits satisfying that requirement *complete*. However, in our proofs we consider also transformations of circuits that do not use gates I on the input wires and O on the output wires. Such circuits will be called *incomplete*. For incomplete circuit C we denote by its *completion* a circuit C with added gates I at every input and O at every output.

We also assume that the original C is deterministic, i.e. it has no randomness gates. This can be done without loss of generality, as the randomness can be provided to C as an additional input.

Partial order of the distributions over subsets. We now provide formal definition of what it means that one probability distribution "covers" another one. The motivation and the intuition behind this concept were described in Sect. 2.2.

Definition 1. *Consider a fixed finite set A and its power set $P(A)$. Let D_1 and D_2 be some probability distributions over $P(A)$. We will say that distribution D_2* covers *distribution D_1 if it is possible to obtain D_2 from D_1 by a sequence of the following operations on a distribution D:*

1. *Pick two subsets satisfying $A_1 \subset A_2 \subset A$.*
2. *Pick a real value $0 < d < D(A_1)$.*
3. *Subtract d from $D(A_1)$ and add d to $D(A_2)$.*

It is clear from the definition above that the relation of covering is indeed a partial order on the probability distributions. We will write $D_2 \geq D_1$ to denote the coverage relationship of the distributions (when it is clear from the context over which power set these distributions are). As already mentioned in Sect. 2.2, one specific distribution over a power set $P(A)$ that we will consider is *a standard distribution* $D_p(A)$ where $0 < p < 1$.

Definition 2. *Let A be a finite set and let $0 < p < 1$. We define a random subset S of A as follows: any element of A belongs to S with probability p, independently. We call the distribution over $P(A)$ determined by the random subset S* a standard distribution $D_p(A)$.

For a random variable X with a domain $P(A)$ we denote by $\mathcal{D}(X)$ the probability distribution over $P(A)$ generated by that variable. For two random variables X, Y with the same domain $P(A)$ we will say that Y covers X if $\mathcal{D}(Y)$ covers $\mathcal{D}(X)$.

3.1 Security definitions

In this section we present the formal definitions of soundness and privacy of a circuit transformation. Soundness is defined as follows.

Definition 3. *We say that transformation $\widehat{C}$ of k-input complete circuit C is* sound *if it preserves the functionality of C, that is*

$$\widehat{C}(\overrightarrow{x}) = C(\overrightarrow{x})$$

for every input $\overrightarrow{x}$ of length k. In case of incomplete circuit C, we say that its transformation is sound if transformation of its completion is sound.

To reason about privacy we consider the following experiment.

Definition 4. *For a fixed circuit C with k input wires, its input $\overrightarrow{x} = (x_1, \ldots, x_k)$ and probability p we define an experiment $Leak(C, \overrightarrow{x}, p)$ that outputs an adversarial view as follows:*

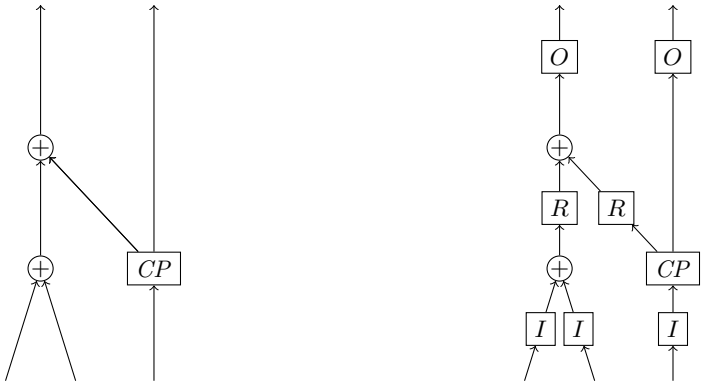
1. *Transformed circuit $\widehat{C}$ is fed with $(x_1, \ldots, x_k)$ resulting with some assignment of the wires of $\widehat{C}$.*
   In case when C is incomplete, the i-th input wire bundle of transformed circuit $\widehat{C}$ is fed with an encoding of respective input value x_i, chosen uniformly at random.
2. *Each wire of $\widehat{C}$ leaks independently with probability p. Note that in case of complete circuit C input and output wires do not leak, as part of $\widehat{I}$ and $\widehat{O}$ gadgets.*
3. Output: *(LW: set of leaking wires in $\widehat{C}$, A: values assigned to the leaking wires in LW during the circuit evaluation).*

We are now ready to define privacy of a circuit transformation.

Definition 5. *We say that transformation $\widehat{C}$ of circuit C is (p, ϵ)-private if leakage in experiment $Leak(C, \overrightarrow{x}, p)$ can be simulated up to ϵ statistical distance, for any input $\overrightarrow{x}$. More precisely, there exist a simulation algorithm that, not knowing input $\overrightarrow{x}$, outputs a random variable that is ϵ-close to the actual output of $Leak(C, \overrightarrow{x}, p)$.*

4 Technical details of the circuit transformation

Let us now present the technical details of the ideas outlined in Sect. 2, i.e., our construction of the transformed circuit $\widehat{C}$ together with a proof of the privacy. For syntactic purposes we introduce a special single-input single-output refreshing gate R that acts as an identity function, similarly to I and O gates, but can be placed anywhere in the circuit C. The general transformation of the original circuit C consists of two phases. We start with the *preprocessing phase*. In this phase, if circuit C is incomplete then we add I gate to every input wire and O to every output wire. Moreover, we add refreshing gate R on every wire of C that connects any two gates, except for I and O (see Fig. 8). We call the resulting circuit C'. We then proceed to the *actual transformation phase* in which each wire in C' carrying value x is replaced with a bundle of n wires that carry an encoding of x. Each gate Γ in C' is replaced with a respective gadget subcircuit $\widehat{\Gamma}$ that operates on the encodings. Below we give a detailed description of the gadget subcircuits.



(a) An original circuit C (b) Circuit C' after preprocessing phase

Fig. 8: Example of the preprocessing phase of the transformation.

We say that two regular gadgets (not refreshing gadgets) $\widehat{\Gamma}_1$ and $\widehat{\Gamma}_2$ in $\widehat{C}$ are *connected* if there is a refreshing gadget between them. More precisely, if there is a refreshing gadget $\widehat{R}$ that takes as input the output bundle of $\widehat{\Gamma}_1$, and outputs the input bundle to $\widehat{\Gamma}_2$.

4.1 General gadget description

In this section we give a general definition of a gadget and the required properties. Every gadget used in our construction, except for the refreshing gadget $\widehat{R}$, satisfies the given definition.

Input and output wires of the gadget. Let us consider a gate Γ in circuit C of $0 \leq i \leq 2$ inputs and $1 \leq o \leq 2$ outputs excluding the case $(i, o) = (2, 2)$, for example Γ might be a sum gate $\oplus$ or a product gate $\otimes$. A respective gadget $\widehat{\Gamma}$ will have i input wire bundles and o output wire bundles, that is $i \cdot n$ inputs and $o \cdot n$ outputs in total. We will denote with $IN_k^b(\widehat{\Gamma})$ the k-th wire of its b-th input bundle and with $OUT_k^b(\widehat{\Gamma})$ the k wire of its b-th output bundle. We denote with $IN_k(\widehat{\Gamma})$ all the input wires of index k in its input bundle. More precisely, $IN_k(\widehat{\Gamma}) := \{IN_k^b(\widehat{\Gamma}) | 1 \leq b \leq i\}$. Similarly, we define $OUT_k(\widehat{\Gamma})$ as $OUT_k(\widehat{\Gamma}) := \{OUT_k^b(\widehat{\Gamma}) | 1 \leq b \leq o\}$. Moreover, we use $IN^b(\widehat{\Gamma})$ to denote the b-th input bundle of $\widehat{\Gamma}$ and OUT^b to denote the b-th output bundle. That is, $IN^b(\widehat{\Gamma}) := \{IN_k^b(\widehat{\Gamma}) | 1 \leq k \leq n\}$, and $OUT^b(\widehat{\Gamma}) = \{OUT_k^b(\widehat{\Gamma}) | 1 \leq k \leq n\}$. Let $g : \mathbb{F}^i \to \mathbb{F}^o$ be the function computed by the gate Γ. The gadget $\widehat{\Gamma}$ should implement the same functionality as Γ. More precisely, if $g(x_1, \ldots, x_i) = (y_1, \ldots, y_o)$ then for *any* encoding $(\overrightarrow{x_1}, \ldots, \overrightarrow{x_i})$ of $(x_1, \ldots, x_i)$ fed to $\widehat{\Gamma}$ as input, it outputs *some* encoding $(\overrightarrow{y_1}, \ldots, \overrightarrow{y_o})$ of $(y_1, \ldots, y_o)$.

Leakage projections. We now define the "leakage projections" already informally discussed in Sect. 2.3. Every gadget comes with *a leakage projection function* P that takes as input a leaking wire w in $\widehat{\Gamma}$ and outputs an associated subset $P(w)$ of $[n]$, usually an one-element subset. We can refer to *the projection set* of a subset W of wires in $\widehat{\Gamma}$ defined as $P(W) = \bigcup_{w \in W} P(w)$. We require the following properties of the projection P. Firstly, for any subset LG of leaking wires in $\widehat{\Gamma}$, it is enough to know the values carried by wires of the indices $\in P(LG)$ from every input bundle, i.e. the wires in $\{IN_k^b(\widehat{\Gamma}) | 1 \leq b \leq i, k \in P(LG)\}$ to simulate the leakage from $\widehat{\Gamma}$ perfectly (without knowing the values of the other input wires). Secondly, for every output wire w in $\widehat{\Gamma}$ that is k-th wire in any output bundle, i.e. $w \in OUT_k(\widehat{\Gamma})$, we have $P(w) = \{k\}$.

Consider an experiment where each of the wires in the gadget $\widehat{\Gamma}$ leaks independently with probability p. Let us call a set of leaking wires LR. Induced projection of the leakage $P(LR)$ defines a probability distribution over the subsets of $[n]$. We denote this *leakage projection distribution* with $D_p(\widehat{\Gamma})$. In the security proof it will be convenient to consider only the gadgets $\widehat{\Gamma}$ with the following property: if every wire of $\widehat{\Gamma}$ leaks independently with probability p then projection of the leakage contains every number $i \in [n]$ with some probability q *independently*. However it is not the case, e.g. for the product gadget. For that reason, we introduce *a projection probability function* describing a particular gadget. Essentially, it expresses with what probability do we need to add every particular number $\in [n]$ to the projection in order to make these events

independent. More precisely, we will say that a function $f : [0,1] \to \mathbb{R}$ is *a projection probability function* for a gadget $\widehat{\Gamma}$ if the leakage projection distribution $D_p(\widehat{\Gamma})$ is covered by the standard distribution $D_{f(p)}([n])$ (as in Def. 1). Note that the function f may depend on the security parameter n, like in the case of product gadget $\widehat{\otimes}$.

4.2 The gadgets used in our construction

In this section we present all the gadgets used in our construction.

ISW product gadget. As the product gadget $\widehat{\otimes}$ in our construction we use the gadget proposed in [18]. Here we recall their scheme and prove that it satisfies the general gadget definition.

1. **Input:** 2 bundles $\overrightarrow{x} = (x_1, \dots x_n)$ and $\overrightarrow{y} = (y_1, \dots y_n)$
2. For $1 \le i < j \le n$ sample $z_{i,j} \leftarrow \mathbb{F}$
3. For $1 \le i < j \le n$ compute $z_{j,i} = (z_{i,j} \oplus x_i \otimes y_j) \oplus x_j \otimes y_i$
4. Compute the output encoding $(t_1, \dots t_n)$ as $t_i = x_i \otimes y_i \oplus \bigoplus_{j \neq i} z_{i,j}$
5. *Output:* a bundle $(t_1, \dots, t_n)$

We define the projection function P for this gadget as follows: For every wire w of the form x_i, y_i, $x_i \otimes y_i$, $z_{i,j}$ (for any $j \neq i$) or a sum of values of the above form (with t_i as a special case), $P(w) = \{i\}$. For the remaining wires w, which are of the form $x_i \otimes y_j$ or $z_{i,j} \oplus x_i \otimes y_j$, we define $P(w) = \{i, j\}$. The following lemmas are proven in the full version of this paper [15].

Lemma 1. *The ISW product gadget with its projection function satisfies a general gadget description (given in the Section 4.1) for the multiplication function* $g(x, y) = x \cdot y$.

Lemma 2. *The function* $f(p) = n(8p + \sqrt{3p})$ *is a projection probability function for the ISW product gadget.*

Other gadgets. We already described the addition gadget $\widehat{\oplus}$ in Sect. 2.3. Besides of this, we use a *copy gadget* $\widehat{CP}$ that takes one input bundle $\overrightarrow{x} = (x_1, \dots, x_n)$. Then it applies the copy gate CP to each wire $x_1, \dots, x_n$ obtaining n respective pairs $(y_1, z_1), \dots, (y_n, z_n)$. We define two output bundles as $\overrightarrow{y} = (y_1, \dots, y_n)$ and $\overrightarrow{z} = (z_1, \dots, z_n)$. The *negation gadget* $\widehat{NEG}$. takes one input bundle $\overrightarrow{x} = (x_1, \dots, x_n)$ and it applies the negation gate NEG to each of n wires $x_1, \dots, x_n$ obtaining wires $y_1, \dots, y_n$. We define the output bundle of the gadget as $\overrightarrow{y} = (y_1, \dots, y_n)$. *Constant gadget* $\widehat{Const_\alpha}$ has zero input bundles and one output bundle carrying n constant values: $(\alpha, 0, \dots, 0)$. Finally, the *Identity gadget* $\widehat{ID}$ is a special gadget has one input and one output bundle, and simply outputs the input.

Properties of gadgets other than ISW product gadget. It is clear that all the gadgets described above correctly implement the desired functions. Also, it is easy to see that for each gadget the leakage projection function P can be defined as follows: for any input or output wire w in the gadget, define $P(w) = \{i\}$, where i is an index of w in its (input or output) bundle. Clearly for each of these gadgets the function $f(p) = 3p$ is a projection probability function. For the gadgets $\widehat{Const_\alpha}$ and $\widehat{ID}$ even smaller function $f(p) = p$ is a projection probability function. We omit the proofs in these cases, as they are very straightforward.

4.3 Refreshing gadget properties

In this section we describe properties of the refreshing gadget $\widehat{R}$. (see Fig. (1a) on p. 8) that are crucial to the security of the construction, and are used in the privacy proof. Below, by *refreshing bundle* $B_{\widehat{R}}$ we mean the wires that are used to generate the fresh encoding Enc(0) in the refreshing gadget $\widehat{R}$, i.e., wires carrying $b_1^j, \ldots, b_n^j$ and $c_1^j, \ldots, c_{n-1}^j$ on Fig. 1.

Refreshing bundle leakage projection. Consider a refreshing bundle $B_{\widehat{R}}$. Suppose that LR is a set of leaking wires in $B_{\widehat{R}}$. We define a subset $S(LR)$ of $\{0, \ldots, n\}$ representing the leakage LR as follows: we start with the set $S = \{0, n\}$. For every wire of the form $c_k^j = b_1^j \oplus b_2^j \oplus \cdots \oplus b_k^j$ in LR, where $1 \leq k < n$, add k to S. For every wire of the form b_k^j in LR , where $1 < k \leq n$, add k and $k-1$ to S.

One may think of the function $S(\cdot)$ as an analogue of the leakage projection function (introduced in Section 4.1) in case of a refreshing gadget. The difference is, however, that $S(\cdot)$ codomain size is $n+1$ instead of n, and that 2 elements (0 and n) belong to $S(LR)$ "by default".

Leakage projection coverage. Here we show a random subset of $\{0, \ldots, n\}$ that covers the projection of the refreshing bundle leakage. Let us define a random subset R_q of $\{0, \ldots, n\}$ as follows: R_q contains 0 and n with probability 1, and for any other number $i \in \{0, \ldots, n\}$ R_q contains i with probability q, independently. The proof of the following lemma appears in the full version of this paper [15].

Lemma 3. *Let LR be a subset of leaking wires of a refreshing bundle $B_{\widehat{R}}$ when each wire leaks independently with probability p. Then the random subset $S(LR) \subset \{0, \ldots, n\}$ is covered by $R_{p+2\sqrt{3p}}$.*

Leakage diagrams. The main technical concept of this work is a *leakage diagram* (already introduced informally in Sect. 2). Consider a transformed circuit $\widehat{C}$, as described in previous sections. Suppose that LW is the set of leaking wires in $\widehat{C}$. The leakage diagram is a representation of the set LW. As explained in Sect. 2, in the security proof the leakage diagram is used to determine whether the leakage compromises the secret or not. This is because of the property that if the *leftmost* and *rightmost* sides of the leakage diagram are disconnected then the privacy is preserved.

We first define the leakage diagram as a subgraph of $G = G(\widehat{C})$ - a graph associated with the transformed circuit $\widehat{C}$. The leakage diagram inherits all nodes of G and some of its edges, depending on the set of leaking wires LW. The exact construction of graph $G(\widehat{C})$ and the leakage diagram are described in the following paragraphs. Let C be any circuit and $\widehat{C}$ its transformation as described in Section 4. We define an associated undirected graph $G = G(\widehat{C})$ as follows. For each general gadget $\widehat{\Gamma}$ in $\widehat{C}$ (every gadget except the refreshing gadgets) $G(\widehat{C})$ contains *a crosswise path* of length n, where n is the security parameter of the construction. We denote the nodes of this path $N_0^{\widehat{\Gamma}}, \ldots, N_n^{\widehat{\Gamma}}$. Moreover, for every pair $\widehat{\Gamma}_1, \widehat{\Gamma}_2$ of connected gadgets in $\widehat{C}$, we add to the graph $G(\widehat{C})$ a *vertical matching* consisting of the following $n+1$ edges: $(N_0^{\widehat{\Gamma}_1}, N_0^{\widehat{\Gamma}_2}), \ldots, (N_n^{\widehat{\Gamma}_1}, N_n^{\widehat{\Gamma}_2})$. We call all the nodes of the form $N_0^{\widehat{\Gamma}}$, for some gadget $\widehat{\Gamma}$ in $\widehat{C}$, together with the edges between these nodes *a leftmost of* G. Analogically, we define *a rightmost of* G as all the nodes of the form $N_n^{\widehat{\Gamma}}$ with all the edges between them. The construction of $G(\widehat{C})$ can be naturally decomposed into separate subsets of edges - its crosswise paths and vertical matchings. We will call it a *decomposition* of $G(\widehat{C})$.

While the computation is executed on circuit $\widehat{C}$ some wires will leak the carried values. Let LW denote the set of all the leaking wires. We will be representing this set with *a leakage diagram* H - a subgraph of $G(\widehat{C})$. The leakage diagram inherits all the nodes from $G(\widehat{C})$ and some of its edges as in the following construction. Each leaking wire $w \in LW$ that belongs to some general gadget $\widehat{\Gamma}$ is *projected* onto the respective crosswise path in G. More precisely, if $P_{\widehat{\Gamma}}$ is leakage projection function for the gadget $\widehat{\Gamma}$ then we add to the leakage diagram H the edges in the crosswise path of order in $P_{\widehat{\Gamma}}(w)$, i.e. edges $\{(N_{i-1}^{\widehat{\Gamma}}, N_i^{\widehat{\Gamma}}) | i \in P_{\widehat{\Gamma}}(w)\}$.

The rest of the leaking wires in the set LW are part of some refreshing bundle $B_{\widehat{R}}$, where the refreshing gadget $\widehat{R}$ connects some gadgets $\widehat{\Gamma}_1$ and $\widehat{\Gamma}_2$. Let $LR \subset LW$ be a set of leaking wires in this refreshing bundle. It is represented in the leakage diagram H by the subset of the vertical matching between two respective crosswise paths, namely $\{(N_i^{\widehat{\Gamma}_1}, N_i^{\widehat{\Gamma}_2}) | i \in S(LR)\}$. An example of a leakage diagram is illustrated on Fig. 7.

Modification vectors. In the security proof we use a sequence of hybrid experiments that produce exactly the same leakage. One of the hybrids requires to assign every gadget in $\widehat{C}$ with a *basic modification vector*. They were already informally introduced in Sect. 2. Let us now present their formal definition. A *basic modification vector* is a vector $\overrightarrow{m} = (m_1, \ldots, m_n)$ of length n whose coordinates are in the set $\{-1, 0, 1\}$ and additionally $\sum_{i=1}^{n} m_i = 1$. We assign a gadget $\widehat{\Gamma}$ with the basic modification vector $\overrightarrow{m}^{\widehat{\Gamma}}$ based on the leakage diagram H. Let LS be the connected component of the leftmost side of H. Let I be the set of nodes indices from $\{N_0^{\widehat{\Gamma}}, \ldots, N_n^{\widehat{\Gamma}}\}$ that belong to LS. Now, based on the set I we assign the modification vector $\overrightarrow{m}^{\widehat{\Gamma}}$ according to the following rule: the

i-th coordinate of $\overrightarrow{m}^{\widehat{\Gamma}}$ equals 1 if $i-1 \in I$ and $i \notin I$, equals -1 if $i-1 \notin I$ and $i \in I$, and equals 0 in other cases.

We generalize the definition of a basic modification vector to a *modification vector*. We will say that a vector $\overrightarrow{w}$ is *a modification vector* if it can be written in the form $\overrightarrow{w} = v \cdot \overrightarrow{m}$ for some scalar value $v \in \mathbb{F}$ and a basic modification vector $\overrightarrow{m}$. Moreover, we will say that a modification vector $\overrightarrow{m} = (m_1, \ldots, m_n)$ is *disjoint* with a set $A \subset [n]$ if $m_a = 0$ for all $a \in A$. Let S be a subset of $\{0, \ldots, n\}$ and let $\overrightarrow{m}^1, \overrightarrow{m}^2$ be any modification vectors of length n. We will say that $\overrightarrow{m}^1$ and $\overrightarrow{m}^2$ are *indistinguishable* under S if for every $k \in S$ we have that $\sum_{i=1}^{k} m_i^1 = \sum_{i=1}^{k} m_i^2$.

Leakage and extended leakage from a gadget In this section we give the formal definitions of leakages from a gadget. Here, we consider only gadgets other than refreshing gadget.

Extended leakage. In order to express the desired property of a gadget we define a random variable that we call *extended leakage*. It is a leakage from a subset of wires in $\widehat{\Gamma}$ together with values carried by *all* the output wires of $\widehat{\Gamma}$, including the non-leaking wires (a more restrictive definition that does not include these wires is given in the full version of this paper [15].

Definition 6. *Let $\widehat{\Gamma}$ be a gadget with i input bundles and o output bundles and let LG be a subset of its wires. We define a function $\mathrm{ExtLeak}_{\widehat{\Gamma}}^{LG}(\overrightarrow{x_1}, \ldots, \overrightarrow{x_i})$ as the output of the following experiment:*

1. *The gadget $\widehat{\Gamma}$ is fed with input $(\overrightarrow{x_1}, \ldots, \overrightarrow{x_i})$ resulting with some assignment of the wires of $\widehat{\Gamma}$.*
2. *Let $\overrightarrow{y}_1, \ldots, \overrightarrow{y}_o$ be the produced output of $\widehat{\Gamma}$.*
3. Output: *(values assigned to wires in LG, values assigned to* all *the output wires $\overrightarrow{y_1}, \ldots, \overrightarrow{y_o}$).*

Extended leakage shiftability. Recall that in Sect. 2.1 one of the main technical tricks was to show that the experiments Exp_C and Exp_B^0 are indistinguishable from the point of view of the adversary. This was done by showing that the vectors encoding the secret can be "shifted" (i.e. a certain vector can be added to it) in way that is not noticeable to the adversary. This idea is formalized and generalized to gadgets below.

Definition 7. *Let $\overrightarrow{v_1}, \ldots \overrightarrow{v_k}$ and $\overrightarrow{m}$ be vectors of the same length. and let $T = (T_1, \ldots, T_k)$ be a sequence of k field elements. We define a $\mathrm{shift}_{\overrightarrow{m}}^{T}(\overrightarrow{v_1}, \ldots, \overrightarrow{v_k})$ as follows: it is a sequence of vectors $\overrightarrow{w_1}, \ldots, \overrightarrow{w_k}$, with $\overrightarrow{w_j}$ being a modified vector $\overrightarrow{v_j}$:*

$$\overrightarrow{w_j} = \overrightarrow{v_j} + T_j \cdot \overrightarrow{m}.$$

Also, when applicable, we treat values $v_1, \ldots, v_l$ as vectors of length 1. Then we assume a default basic modification vector (1) and write $\mathrm{shift}^{T}(v_1, \ldots, v_k)$ instead of $\mathrm{shift}_{(1)}^{T}(v_1, \ldots, v_k)$.

Recall that the informal description in Sect. 2 was simplified, since it was focusing on the multi-round refreshing circuits only. Making this idea work for arbitrary circuits requires some extra work. In particular, we need to ensure that nothing goes wrong in the (non-refreshing) gadgets if their input is shifted. Let LG denote a fixed subset of leaking wires in the gadget $\widehat{\Gamma}$. Informally speaking, *the extended leakage shiftability property* says that shifting the value of the wires of index $w \notin P(LG)$ in the input bundles of $\widehat{\Gamma}$ results in shifting the extended leakage only on the index w in the output bundles. This is formalized below.

Definition 8. *Let $\widehat{\Gamma}$ be a gadget with i input bundles and o output bundles implementing a function g, and let P be its leakage projection function. We say that a pair $(\widehat{\Gamma}, P)$ satisfies an* extended leakage shiftability *property if the following holds: Let $x_1, \ldots, x_i$ be any input to g and suppose that $g(shift^S(x_1, \ldots, x_i)) = shift^T(g(x_1, \ldots, x_i))$ for some sequences S and T of lengths i and o, respectively. For any fixed encodings $\overrightarrow{x_1}, \ldots, \overrightarrow{x_i}$ of $x_1, \ldots, x_i$, any subset of leaking wires LG and any basic modification vector $\overrightarrow{m}$ that is disjoint with the set $P(LG)$ we have*

$$ExtLeak^{LG}_{\widehat{\Gamma}}(shift^S_{\overrightarrow{m}}(\overrightarrow{x_1}, \ldots, \overrightarrow{x_i})) = shift^T_{\overrightarrow{m}}(ExtLeak^{LG}_{\widehat{\Gamma}}(\overrightarrow{x_1}, \ldots, \overrightarrow{x_i})).$$

Here, when the function shift is applied to the output of the ExtLeak experiment, it is applied only to the second part of the experiment output i.e. values assigned to the output bundles of a $\widehat{\Gamma}$.

Based on the following lemma, whose proof appears in the full version of this paper [15], every gadget used in our construction satisfies the extended leakage shiftability property.

Lemma 4. *Every general gadget $\widehat{\Gamma}$ with its leakage projection function, as described in Section 4.1, satisfies the extended leakage shiftability property.*

In the proof of Thm. 1 we also use a concept of *refreshed gadget reconstruction* that is presented in the full version of this paper [15].

4.4 Privacy of the construction

Here we present and prove a central theorem of our work.

Theorem 1. *Let C be any arithmetic circuit and $\widehat{C}$ its transformation as described in Section 4. Assume that for all gadgets used in $\widehat{C}$ the projection probability functions are upper-bounded by a function $q : [0, 1] \to \mathbb{R}$, which also upper-bounds the function $f(p) = p + 2\sqrt{3p}$. Then $\widehat{C}$ is sound implementation of C and $\widehat{C}$ is $(p, |C| \cdot (4q(p))^n)$-private for any probability p.*

This theorem is proven along the lines of the intuitions presented in Sect. 2. Due to space limitation we give only a proof overview. Below we present in a formal way some tools that are used in the proof (and that were already informally discussed in Sect. 2). These tools are used in the proof of Thm. 1, which appears in the full version of this paper [15].

Proof overview. To prove the privacy of our construction, we will show that any two inputs X_1, X_2 to circuit $\widehat{C}$ induce leakages that are close in terms of statistical distance. We compare these two leakages conditioned on the set of leaking wires being some *fixed* set LW. Let H be a leakage diagram induced by LW. We show that if the left and right sides of the graph H are not connected then the two leakages are actually identical. To this end, we use a hybrid argument, with the set of leaking wires being fixed to LW. We define a sequence of experiments, called hybrids, and show that every two consecutive experiments produce identical output. Here we briefly describe them:

Hybrid_1 (this corresponds to experiment Exp_A^0 in Sect. 2.1): simply outputs the leakage when $\widehat{C}$ is fed with X_1.

Hybrid_2 (this corresponds to experiment Exp_B^0): in this experiment each gadget in $\widehat{C}$ is evaluated separately, and the assignment of the refreshing bundles between the gadgets are derived from there. To this end, we consider the evaluation of the original circuit C when fed with X_1. If a particular wire w in C, which is an input to a gate Γ, is assigned with a value v then the respective input bundle in the gadget $\widehat{\Gamma}$ in $\widehat{C}$ is fed with a freshly chosen random encoding $\overrightarrow{v} \leftarrow \mathrm{Enc}(v)$. Then each gadget in $\widehat{C}$ is evaluated accordingly to the chosen inputs. This determines the assignment of all the refreshing bundles in $\widehat{C}$. The output of the experiment consists of the values assigned to wires in LW.

Hybrid_3 (this corresponds to experiment Exp_C): this experiment is the same as Experiment 2, except for the random vectors that are assigned to the input bundles of each individual gadget. Here, after choosing a random encoding $\overrightarrow{v} \leftarrow \mathrm{Enc}(v)$ just as in Experiment 2, we shift it by carefully chosen modification vector $\overrightarrow{m}$. As a result, we feed the particular input bundle with $\overrightarrow{v} + \overrightarrow{m}$. The modification vector for the input bundles of each gadget is constructed based on inputs X_1 and X_2, and the leakage diagram H. At this point we use the fact that the left and right sides of the leakage diagram H are not connected. The details of the construction for modification vectors are given in the Sect. 4.3.

Based on the properties of the refreshed gadgets subcircuits in $\widehat{C}$ and taking into account the construction of the modification vectors, we argue that shifting values that are fed to each gadget actually does not change the leakage. Hence this experiment outputs the same random variable as Experiment 2.

Hybrid_4 (this corresponds to experiment Exp_B^1): this experiment is analogous to the Experiment 2, with input X_2 instead of X_1. We argue that the random vectors assigned to the input bundles of each individual gadget in are actually the same in this experiment and in Experiment 3. Hence, the two experiments produce identical outputs.

Hybrid_5 (this corresponds to experiment Exp_A^1) : this experiment is analogous to the Experiment 1, with input X_2 instead of X_1. Also the transition between Experiment 4 and this experiment is analogous to the transition for Experiments 1 and 2.

The hybrid argument above essentially shows that unless the left and right sides of the leakage diagram H are connected, the leakage is the same independently of the input X fed to the transformed circuit $\widehat{C}$. Now, to complete the

privacy proof, it is enough to upper-bound the probability of the left and right sides of H being connected. This is a pure probability theory exercise, given that $q(p)$ upper-bounds the leakage projection function of used gadgets which means that each edge will be included to the leakage diagram independently with probability at most $q(p)$.

4.5 Concrete results

In this section we present the concrete results implied by Theorem 1. These are immediate consequences of the theorem. For affine circuits we obtain the following.

Proposition 1. *Assume that a circuit C is an affine circuit. Our transformation $\widehat{C}$, as described in Section 4, is $(p, |C| \cdot (4p + 8\sqrt{3p})^n)$-private for any probability p.*

Proof. As stated in the Section 4.2, for every gadget used in $\widehat{C}$ its projection probability function is upper-bounded by $3p$ and hence by $p + 2\sqrt{3p}$. Thus, the Proposition is a consequence of the Theorem 1 for the function $q(p) = p + 2\sqrt{3p}$. □

For the general circuits we have the following.

Proposition 2. *Assume that a circuit C is an arithmetic circuit. Our transformation $\widehat{C}$, as described in Section 4, is $(p, |C| \cdot (32np + 4n\sqrt{3p})^n)$-private for any probability p.*

Proof. From the Section 4.2 we conclude that for every gadget used in $\widehat{C}$ its projection probability function is upper-bounded by $n(8p + \sqrt{3p})$. Assuming $n \geq 2$, this function also upper-bounds $p + 2\sqrt{3p}$. Thus, the Proposition is a consequence of the Theorem 1 for the function $q(p) = n(8p + \sqrt{3p})$. □

Finally, let us state the result for the multi-round simple refreshing circuits.

Proposition 3. *Consider a k-round refreshing circuit (see Sect. 2). This circuit is $(p, k \cdot (4p + 8\sqrt{3p})^n)$-private for any probability p.*

Proof. As stated in the Section 4.2, the projection probability function of the identity gadgets $\widehat{ID}$ used in the circuit equals p and hence is upper-bounded by $p + 2\sqrt{3p}$. Thus, the Proposition is a consequence of the Theorem 1 for the function $q(p) = p + 2\sqrt{3p}$. □

5 Conclusion

In this work we introduce a new method to analyze the security of masking schemes in the noisy leakage model of Prouff and Rivain [21]. Our approach enables us to show the security of a simple refreshing scheme which is optimal in terms of randomness complexity (it requires only $n-1$ random values), and uses

a small number of arithmetic operations. Our results are achieved by introducing a new technique for analyzing masked circuits against noisy leakages, which is of independent interest.

We believe that our results are of practical importance to the analysis of side-channel resistant masking schemes. The reason for this are twofold. First, our refreshing scheme is very simple and efficient, and reduces the overheads of the masking countermeasure significantly – in particular, for certain types of computation. For example in the case of a secure key update mechanism as used in any cryptocraphic scheme, we can reduce randomness and circuit complexity from $O(n^2)$ using ISW-like refreshing to $O(n)$, where the asymptotic in the later is with nearly optimal constants. Second, while in [5] it was shown how to construct a very simple refreshing scheme (similar to the one used in our work), the security analysis was in a more restricted model (the bounded moment model), and carried out only for small n. In our case, the analysis works for any n and in the standard noisy model that is well accepted in practice.

Interesting questions for future research include to extend our analysis to other masking schemes [4], to explore the tightness of our bounds and to verify our results experimentally in practice (e.g., by providing simulations on the practical resistance of the countermeasure and its efficiency).

Acknowledgements The authors thank Sonia Belaïd and the anonymous reviewers for their constructive comments. Sebastian Faust received funding from the German Federal Ministery of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity (CRISP). Additionally, he received funding from the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG) and by the VeriSec project 16KIS0634 from the Federal Ministry of Education and Research (BMBF). Stefan Dziembowski and Karol Żebrowski received funding from the Foundation for Polish Science (grant agreement TEAM/2016-1/4) co-financed with the support of the EU Smart Growth Operational Programme (PO IR).

References

[1] Miklós Ajtai. "Secure computation with information leaking to an adversary". In: *43rd Annual ACM Symposium on Theory of Computing*. ACM Press, 2011, pp. 715–724.

[2] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. "Private Circuits: A Modular Approach". In: *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*. 2018, pp. 427–455.

[3] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. "Circuit Compilers with $O(1/\log(n))$ Leakage Rate". In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Vol. 9666. Lecture Notes in Computer Science. Springer, Heidelberg, 2016, pp. 586–615.

[4] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. "Theory and Practice of a Leakage Resilient Masking Scheme". In: *Advances in Cryptology – ASIACRYPT 2012*. Vol. 7658. Lecture Notes in Computer Science. Springer, Heidelberg, 2012, pp. 758–775.

[5] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. "Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model". In: *Advances in Cryptology – EUROCRYPT 2017, Part I*. Vol. 10210. Lecture Notes in Computer Science. Springer, Heidelberg, 2017, pp. 535–566.

[6] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. "Strong Non-Interference and Type-Directed Higher-Order Masking". In: *ACM CCS 16: 23rd Conference on Computer and Communications Security*. ACM Press, 2016, pp. 116–129.

[7] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. "Verified Proofs of Higher-Order Masking". In: *Advances in Cryptology – EUROCRYPT 2015, Part I*. Vol. 9056. Lecture Notes in Computer Science. Springer, Heidelberg, 2015, pp. 457–485.

[8] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. "Tight Private Circuits: Achieving Probing Security with the Least Refreshing". In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*. 2018, pp. 343–372. URL: https://doi.org/10.1007/978-3-030-03329-3_12.

[9] Jean-Sébastien Coron. "Formal Verification of Side-Channel Countermeasures via Elementary Circuit Transformations". In: *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*. 2018, pp. 65–82. URL: https://doi.org/10.1007/978-3-319-93387-0_4.

[10] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. "Higher-Order Side Channel Security and Mask Refreshing". In: *Fast Software Encryption – FSE 2013*. Vol. 8424. Lecture Notes in Computer Science. Springer, Heidelberg, 2014, pp. 410–424.

[11] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage". In: *Advances in Cryptology – EUROCRYPT 2014*. Vol. 8441. Lecture Notes in Computer Science. Springer, Heidelberg, 2014, pp. 423–440.

[12] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. "Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device". In: *Advances in Cryptology – EUROCRYPT 2015, Part I*. Vol. 9056. Lecture Notes in Computer Science. Springer, Heidelberg, 2015, pp. 401–429.

[13] Stefan Dziembowski and Sebastian Faust. "Leakage-Resilient Cryptography from the Inner-Product Extractor". In: *Advances in Cryptology – ASI-*

ACRYPT 2011. Vol. 7073. Lecture Notes in Computer Science. Springer, Heidelberg, 2011, pp. 702–721.

[14] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. "Noisy Leakage Revisited". In: *Advances in Cryptology – EUROCRYPT 2015, Part II*. Vol. 9057. Lecture Notes in Computer Science. Springer, Heidelberg, 2015, pp. 159–188.

[15] Stefan Dziembowski, Sebastian Faust, and Karol Żebrowski. *Simple Refreshing in the Noisy Leakage Model*. Cryptology ePrint Archive. extended version of this paper. 2019.

[16] Dahmun Goudarzi, Antoine Joux, and Matthieu Rivain. "How to Securely Compute with Noisy Leakage in Quasilinear Complexity". In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*. 2018, pp. 547–574. URL: https://doi.org/10.1007/978-3-030-03329-3_19.

[17] Dahmun Goudarzi, Ange Martinelli, Alain Passelègue, and Thomas Prest. *Unifying Leakage Models on a Rényi Day*. Cryptology ePrint Archive, Report 2019/138. https://eprint.iacr.org/2019/138. 2019.

[18] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *Advances in Cryptology – CRYPTO 2003*. Vol. 2729. Lecture Notes in Computer Science. Springer, Heidelberg, 2003, pp. 463–481.

[19] Yael Tauman Kalai and Leonid Reyzin. "A Survey of Leakage-Resilient Cryptography". In: *IACR Cryptology ePrint Archive* 2019 (2019), p. 302. URL: https://eprint.iacr.org/2019/302.

[20] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. "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*. 2006, pp. 529–545. URL: https://doi.org/10.1007/11935308_38.

[21] Emmanuel Prouff and Matthieu Rivain. "Masking against Side-Channel Attacks: A Formal Security Proof". In: *Advances in Cryptology – EUROCRYPT 2013*. Vol. 7881. Lecture Notes in Computer Science. Springer, Heidelberg, 2013, pp. 142–159.

[22] Matthieu Rivain and Emmanuel Prouff. "Provably Secure Higher-Order Masking of AES". In: *Cryptographic Hardware and Embedded Systems – CHES 2010*. Vol. 6225. Lecture Notes in Computer Science. Springer, Heidelberg, 2010, pp. 413–427.