Reframing and Extending the Random Probing Expandibility to Make Probing-Secure Compilers Tolerate a Constant Noise

Giuseppe Manzoni
Independent Researcher
giuseppe.manzoni@zelya.org

October 10, 2024

Abstract

In the context of circuits leaking the internal state due to hardware side-channels, the p-random probing model has an adversary who can see the value of each wire with probability p. In this model, for a fixed p, it is possible to reach an arbitrary security by 'expanding' a stateless circuit via iterated compilation, reaching a security of $2^{-\kappa}$ with a polynomial size in κ.

An artifact of the existing proofs of the expansion is that the worst security is assumed for the input circuit. This means that a pre-compiled input circuit loses all the security guarantees of the first compilation. We reframe the expansion, and we prove it as a security reduction from the compiled circuit to the original one. Additionally, we extend it to support a broader range of encodings, and arbitrary probabilistic gates with an arbitrary number of inputs and outputs.

This allows us to prove the following statement: Given a stateless circuit on a field with characteristic ρ, and given a d-probing secure compiler for some integer d, we can produce a circuit with security 2^{-d} against any adversary that sees all wires with a constant SD-noise of $2^{-7.41}/\rho$, at the cost of an additional size factor $\mathcal{O}(\log(d)^3)$.

1 Introduction

Even when a cryptographic algorithm is secure against classical black-box attacks, its implementation could still be vulnerable to side-channel attacks, which make use of some physical leakage (e.g. the running time [Koc96], the power consumption [KJJ99], the electromagnetic radiation [QS01]).

Many types of physical leakage can be modeled using the noisy leakage models [CJRR99, PR13, DDF19, PGMP19]. In particular, we will focus on the noisy models with the following leakage: for every wire with a value $\mathbf{x}$ with distribution X, the adversary can see any leakage $\mathbf{f}(\mathbf{x})$, such that the distribution X and the conditional distribution[1] $X|\mathbf{f}(X)$ are closer than some δ using some metric M (for example, the Euclidean norm in [PR13, PGMP19], the statistical distance in [DDF19, PGMP19, GPRV21], and the Average Relative Error in [PGMP19]). Of those metrics, we will focus the most[2] on the statistical distance because it corresponds to the intuitive concept of the indistinguishability [MT10] between the distributions X and $X|\mathbf{f}(X)$.

As the noisy models tend to be hard to handle when writing proofs, other models have been introduced, like the p-random probing model [ISW03, DDF19, BCP$^+$20] in which each wire leaks the exact value with a given probability p. While this model does not describe any physical attack, it is equivalent to the noisy model. In particular, a compiler that tolerates a leakage of p in the random-probing model is guaranteed to tolerate a noise with leakage of $p/|\mathbb{K}|$ in the noisy model.

In general, when we have calculated the security of a circuit for some given noisy leakage δ, we may find that its guaranteed security is lacking. In this case we can 'compile' it: transform the circuit into a new one that carries out the same function. Usually a compiler will substitute every gate with a small circuit or 'gadget', every wire with n wires or 'shares', and every value will be 'masked' or 'encoded' so that the values in the shares together reveal the original value. This operation usually allows to increase the security at the

[1]For more on this notation and concept see [PR13, DDF19, PGMP19].
[2]For the other metrics the difference is a single factor, see footnote 5.

cost of having a bigger circuit, but this is not guaranteed: if the noise δ associated with the circuit is not enough, then compiling the circuit will only lead to a lower security. For this reason, the ideal compiler is able to guarantee the security of the output circuit for the highest possible tolerated leakage, and with the lowest possible circuit size increase.

The existing literature considers a sequence of compilers with increasing security, and it analyzes the size complexity of the compilation as the security increases. The ideal sequence of compilers tolerates a constant noise (the highest possible) regardless of the required security. The first solution to this problem used expander graphs [Ajt11] and it was followed by a similar compiler sequence [ADF16] which simplified the first and improved it using geometric codes. Then next solution used an expansion strategy of compilers using multi-party computation protocols [AIS18], which had a tolerated leakage in the random probing model of 2^{-26} and complexity of $\mathcal{O}\left(\kappa^{8.2}\right)$ [BCP+20].

The next improvement [BCP+20] tolerated a leakage of $2^{-7.97}$ with a complexity of $\mathcal{O}\left(\kappa^{7.5}\right)$, and did this by introducing the Random Probing Expandability (RPE) property. This expansion works in two steps: first compile a set of gadgets with themselves to obtain bigger and more secure gadgets. This process is repeated until the wanted security level is reached. Lastly, the resulting gadgets are used to compile the input circuit.

This approach was then refined [BRT21] by providing gadgets that reach the limits of what is possible using the RPE property, reaching a better tolerated leakage of $2^{-7.5}$ and lower complexity $\mathcal{O}\left(\kappa^{3.9}\right)$. The same paper also provides a generic compiler sequence, so that the compiler sequence made with gadgets with n shares has a complexity of $\mathcal{O}(\kappa^{e(n)})$ with $e(n) := \frac{\log(3n^2-2n)}{\log(\lfloor(n+1)/2\rfloor)}$, creating a tradeoff between complexity and tolerated leakage. We note that thanks to the expansion, the number of shares of the gadgets is unrelated to the desired security of the final circuit.

A successive article [BRTV21] extended the random probing expansion by allowing a different compiler at different stages of the expansion, and this allows them to solve the tradeoff.[3]

While this research happened, most of the research focused on the t-probing model, where the attacker can place at most t probes, and obtain the exact value contained in those wires. There have been various papers reducing the security of one of the noisy models to that of the probing model. Some [PR13] did so by using refresh gates that do not leak the internal computation. Successive research [DDF19] provided a compiler sequence that given a circuit c and a number of shares n, it produces an output circuit that is both $\lfloor(n-1)/2\rfloor$-probing secure and has a security of $\|c\|\, e^{-n/12}$ for a noise of $\Theta\left(1/(n \cdot |\mathbb{K}|)\right)$, where $\|c\|$ is the circuit size.

There have been improvements to this reduction [PGMP19] to extend it to different metrics, and after we first archived our paper, a new result [BDF24] showed that the $1/|\mathbb{K}|$ factor can be removed at the cost of a noisy leakage that is around the square of the random probing leakage.

1.1 Our Contributions

Our main contribution is to provide a transformation from the probing model to the noisy model such that the resulting circuit tolerates a constant noise. More specifically, if the characteristic of the circuit's field is ρ, then regardless of the desired security guarantee, of the order of the field, or of circuit's size, the resulting circuit will tolerate a SD-noise of $2^{-7.41}/\rho$ which is the state-of-the-art for compilers of circuits with a small characteristic. E.g., for the AES encryption, which works in $\mathbb{F}_{2^8}$, the tolerated noisy leakage is $2^{-8.41}$, but our approach applies to all kinds of circuits. The cost of this transformation is a cube-logarithmic size increase.

At the core of our paper is the introduction of the Random Probing Reducibility (RPR) property, which reduces the security of a compiled circuit to the security of the original one. In particular, if a compiler is f-RPR, its output circuits guarantee the same security for a leakage p as their input circuit does for the leakage $f(p)$. In the useful case of $f(p) < p$, the output circuit guarantees the same security when exposed to a higher leakage.

From the RPR property and from knowing how many gates are used by each gadget, we show how to calculate the tolerated leakage and the exponent of the polylog size increase.

[3]Due to its relevance with our results, we must note that [BRTV21] contains a mistake in its Lemma 9, as the order of the compilers is inverted. In its proof it has an 'i.e.' that does not hold. The core reason (using our notation) is that given two families of gadgets I, O, we have that $CC_O \circ CC_I = CC_{CC_O \circ I}$, which holds due to the associativity of the operation of substituting the leaf of a tree with a sub-tree (i.e. a gate of a circuit with a gadget). Instead that 'i.e.' states $CC_O \circ CC_I = CC_{CC_I \circ O}$.

We solve the question of how to calculate the new property RPR by using the existing RPE property, and we do this by extending the RPE to allow it to cover any kind of gates and gadgets and a much wider range of encodings. This allows us to consider any $\vec{v}$-linear encoding, but more importantly, it allows us to use the field-extension encoding that we need to output a circuit over $\mathbb{F}_p$ from a circuit over $\mathbb{F}_{p^m}$, which we need to reduce the $1/p^m$ factor in the tolerated noisy leakage to a $1/p$ factor. We show that contrary to what one may expect,[4] this transformation should be done first, before the expansion compilation.

In addition, we also show how to calculate the tolerated leakage and the exponent of the polylog size increase directly from the RPE in a way that mirrors [BCP+20]'s results. The difference is that their formalization can not consider any security guarantee for its input circuits, e.g. the probing security. If this guarantee is present, we transform their polynomial size increase into a polylogarithmic one, with the same exponent.

2 Circuit and Security

In this section we will give a definition for what we mean with circuit, compiler and compiler sequence, and we will also report the definition of the various security notions and adversarial models, and the relevant reductions and relationships between them. Lastly, we use those notions to define the main properties of compiler sequences, and we state the main theorem to calculate their asymptotic size increase.

2.1 Basic Notation and Probabilistic Function

We will use $\vec{x}$ for a vector, $\breve{x}$ or $\mathbb{X}$ for a set, $\mathbf{x}$ for a random variable. For functions, these notstions describe their codomain. Similarly to [BCP+20], we will use $[m] := \mathbb{Z} \cap [1, m]$ as the set of indices for a tuple of m values, and we define $\vec{x}|_{\breve{I}}$ as the operation that keeps the elements of $\vec{x}$ that have the indices in $\breve{I}$ while replacing with $\perp$ the elements of $\vec{x}$ whose indices are not in $\breve{I}$.

As we are working with circuits, the most natural definition of 'probabilistic function' is one that returns a new random variable at each invocation, like executing a probabilistic circuit and obtaining a different value each time. As this is not a function, we formally define with 'probabilistic function' $\mathbf{f} : \breve{I} \to \breve{O}$ some deterministic function (here f to differentiate) with domain $\breve{I}$ and codomain the probability distributions over $\breve{O}$. When we define $\mathbf{f}(x) := \dots$ we define the $f(x)$ that calculates the distribution of that expression, and whenever we use $\mathbf{f}(x)$ we mean $\leftarrow f(x)$, which is an anonymous random variable with distribution $f(x)$, independent from all the random variables defined before it and used only there. The intuitive description of 'a function that returns a new random variable at each invocation' holds as long as the expression in definition of $\mathbf{f}(x) := \dots$ has no correlation with any random variable defined outside it, which will always be the case.

2.2 Circuit Type

We can parameterize the type of a circuit based on the values it operates on, and the gates it can contain.

Definition 1. *We define a* ***circuit type*** *as the tuple* $(\mathbb{S}, \mathcal{G})$*:*

- *A finite set* $\mathbb{S}$ *with the values that the circuit operates on.* $|\mathbb{S}| \geq 2$.
- *A finite enumeration* $\mathcal{G}$ *of the description of the gates:*
  - *A deterministic function* $\mathbb{S}^i \times \mathbb{S}^r \to \mathbb{S}^o$ *(where* i, r, o *depend on the gate and are respectively the inputs, randoms, and outputs used by it).*
  - *A probability distributions over* $\mathbb{S}^r$ *that describes the required randoms.*
  - *A boolean value: if the gate's* i *inputs and* r *randoms either all leak (fully leakable gate) or none leak (leakless gate).*

Given a circuit $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$, we will use the following functions describe c's behavior. For a more formal definition see subsection B.1.

[4] E.g. [BRTV21] claims would lead one to believe the inverse, see footnote 3.

Gate:	Function	Random Variable	Which Leakage
Addition	$[a, b], [] \mapsto [a + b]$	$[]$	Fully leakable
Subtraction	$[a, b], [] \mapsto [a - b]$	$[]$	Fully leakable
Copy	$[a], [] \mapsto [a, a]$	$[]$	Fully leakable
Multiplication	$[a, b], [] \mapsto [a \cdot b]$	$[]$	Fully leakable
Random	$[], [r] \mapsto [r]$	$\leftarrow \mathbb{F}_q$	Leakless
Constant-c	$[], [] \mapsto [c]$	$[]$	Leakless

Table 1: Description of the gates of $\mathcal{C}_{\text{std},q}$.

- $\vec{\mathbf{Rnds}}_c() \in \mathbb{S}^r$; to calculate the randomness needed by the circuit.
- $\vec{c}_{\text{outs}} : \mathbb{S}^i \times \mathbb{S}^r \to \mathbb{S}^o$; to calculate the outputs from the inputs and randoms.
- $\vec{c}_{\text{wires}} : \mathbb{S}^i \times \mathbb{S}^r \to \mathbb{S}^w$; to calculate the inputs and randoms of every fully-leakable gate in c, which are the leakable internal wires of c. We call a circuit leakless if $\vec{c}_{\text{wires}}(\cdot, \cdot) = []$ as it has no leakable wires.
- $\check{\mathbf{Leaking}}_c : [0, 1] \to \mathcal{P}([w])$, where $\mathcal{P}(\cdot)$ denotes the power set; to map the leakage probability of a single wire to the set of the wires that are leaking.
- $\vec{\text{Gates}}(c) \in \mathbb{N}^{|\mathcal{G}|}$; to describe how many gates of a given type are in c.

For simplicity and compactness, we define the following intuitive definitions:

- $\vec{c}_{\text{all}}(\vec{x}, \vec{r}) := \vec{c}_{\text{wires}}(\vec{x}, \vec{r}) \parallel \vec{c}_{\text{outs}}(\vec{x}, \vec{r})$, where $\vec{a} \parallel \vec{b}$ concatenates $\vec{a}$ and $\vec{b}$.
- $\vec{\mathbf{c}}_{\mathbf{outs}}(\vec{x}) := \vec{c}_{\text{outs}}(\vec{x}, \vec{\mathbf{Rnds}}_c())$.
- $\vec{\mathbf{c}}_{\mathbf{wires}}(\vec{x}) := \vec{c}_{\text{wires}}(\vec{x}, \vec{\mathbf{Rnds}}_c())$.
- $\vec{\mathbf{c}}_{\mathbf{all}}(\vec{x}) := \vec{c}_{\text{all}}(\vec{x}, \vec{\mathbf{Rnds}}_c())$.
- $\|c\| := \left\|\vec{\text{Gates}}(c)\right\|_1$; the total number of gates in c, where $\|\cdot\|_1$ is a p-norm.

To give a concrete example, we define the standard circuit type $\mathcal{C}_{\text{std},q} := \mathcal{C}_{\mathbb{F}_q,\mathcal{G}_{\text{std}}}$ as the circuit type over the field of order q and with the standard gates (Table 1).

2.3 Encoding

All the circuits we consider will have an associated encoding.

Definition 2. *We define an **encoding** E as a pair of:*

- *A probabilistic function to obtain a new encoding $E.\vec{\mathbf{Enc}} : \mathbb{D}_\ell \to \mathbb{S}^{\ell'}_{\text{out}}$ with $\mathbb{D}_\ell \subseteq \mathbb{S}^{\ell}_{\text{in}}$ for some tuples (ℓ, ℓ'). Given a set $\breve{U} \subseteq \mathbb{D}_\ell$, we write with $E.\vec{\mathbf{Enc}}[\breve{U}]$ the set of the valid encodings of the elements of $\breve{U}$.*
- *A deterministic function to decode $E.\vec{\text{Dec}} : E.\vec{\mathbf{Enc}}[\mathbb{D}_\ell] \to \mathbb{D}_\ell$ that maps each valid encoding to the value it represents, and it must be such that $E.\vec{\text{Dec}} \circ E.\vec{\mathbf{Enc}}$ is the identity function.*

We can also define the composition of encodings, which is also an encoding: Given a pair of encodings C, D then we denote with $C \circ D$ as $(C \circ D).\vec{\mathbf{Enc}} := C.\vec{\mathbf{Enc}} \circ D.\vec{\mathbf{Enc}}$, $(C \circ D).\vec{\text{Dec}} := D.\vec{\text{Dec}} \circ C.\vec{\text{Dec}}$.

Definition 3. *An n-**shares encoding** is one that has $\ell' = n\ell$ for all $\ell \in \mathbb{N}$, has $\mathbb{D}_\ell := \mathbb{S}^{\ell}_{\text{in}}$, and is compatible with the concatenation of vectors: $E.\vec{\mathbf{Enc}}(\vec{a} \parallel \vec{b}) \overset{d}{=} E.\vec{\mathbf{Enc}}(\vec{a}) \parallel E.\vec{\mathbf{Enc}}(\vec{b})$ and if the number of elements of $\vec{a}$, $\vec{b}$ is a multiple of n, $E.\vec{\text{Dec}}(\vec{a} \parallel \vec{b}) = E.\vec{\text{Dec}}(\vec{a}) \parallel E.\vec{\text{Dec}}(\vec{b})$.*

We now need to report two properties already known in the literature:

Definition 4. *We will say that a probabilistic function* $\vec{\mathbf{f}} : \mathbb{S}_{\text{in}}^{i} \to \mathbb{S}_{\text{out}}^{o}$ ***can be simulated*** *from the input elements* $\breve{I} \subseteq [i]$ *(written* $\breve{I} \xrightarrow{\text{sim}} \vec{\mathbf{f}}$*) if there is a probabilistic function* $\vec{\mathbf{Sim}}$ *such that for all* $\vec{x} \in \mathbb{S}_{\text{in}}^{i}$ *we have that* $\vec{\mathbf{f}}(\vec{x}) \overset{d}{=} \vec{\mathbf{Sim}}(\vec{x}|_{\breve{I}})$ *where* $\overset{d}{=}$ *compares the probability distributions.*

Definition 5. *Given a probabilistic function* $\vec{\mathbf{f}} : \mathbb{S}_{\text{in}}^{i} \to \mathbb{S}_{\text{out}}^{o}$ *we will indicate with* $\breve{\text{Dep}}_{[\vec{\mathbf{f}}]}$ *its* ***dependency function****: the minimal function* $\breve{Dep} : \mathcal{P}([o]) \to \mathcal{P}([i])$ *such that for all* $\breve{O} \subseteq [o]$, $\breve{Dep}(\breve{O}) \xrightarrow{\text{sim}} \vec{\mathbf{f}}|_{\breve{O}}$.

Note that the dependency function always exists, is unique, and is monotone non-strictly increasing.

We can now define the strength of an encoding, which is the property necessary for the expansion to work, and it describes how much security a compilation with this encoding can provide.

Definition 6. *Given an n-shares encoding, it has* ***encoding strength*** k*, with* $k \in \mathbb{N} \cap [0, n)$ *if:*

- *The values of* $\leq k$ *shares provide no information on the decoded value: for all* $\breve{I} \subseteq [n]$ *with* $|\breve{I}| \leq k$, $\breve{\text{Dep}}_{[E.\vec{\mathbf{Enc}}]}(\breve{I}) = \emptyset$.
- *From the value of k shares and their decoded value is possible to uniquely reconstruct the value of the missing shares so that their decoding matches the decoded value.*

An immediate implication is that the dependency function of the encoding is fully determined if an encoding has strength:

Lemma 1. *Given an n-shares encoding E with strength k, then for all* $\breve{I} \subseteq [n]$ *we have that* $\breve{\text{Dep}}_{[E.\vec{\mathbf{Enc}}]}(\breve{I}) = \emptyset$ *iff* $|\breve{I}| \leq k$ *(and* $\breve{\text{Dep}}_{[E.\vec{\mathbf{Enc}}]}(\breve{I}) = [1]$ *iff* $|\breve{I}| > k$*).*

To give some concrete exmples, in this paper we use the following encodings:

- We define the **additive encoding** for some finite field $\mathbb{K}$ as the n-shares encoding with $\mathbb{S}_{\text{in}}, \mathbb{S}_{\text{out}} := \mathbb{K}$, such that for all $\vec{x} \in \mathbb{K}^n$, $\vec{\text{Dec}}(\vec{x}) := \sum_i \vec{x}_i$ and with the encoding function that selects an encoding uniformly between the possible ones. It has encoding strength of $n - 1$.
- We define the P**-field-extension encoding** for some irreducible polynomial $P \in \mathbb{F}_q[x]$ with degree $m \geq 2$ and for some q power of a prime, as the m-shares encoding with $\mathbb{S}_{\text{in}} := \mathbb{F}_{q^m}$, $\mathbb{S}_{\text{out}} := \mathbb{F}_q$, and the following deterministic encoding: as $\mathbb{F}_{q^m}$ can be constructed from $\mathbb{F}_q$ using P, each value of $\mathbb{F}_{q^m}$ can be seen as a polynomial of $\mathbb{F}_q[x]$ with degree $< m$, and so we define $\vec{\mathbf{Enc}}$ as the function that outputs the array of coefficients of the input value, and $\vec{\text{Dec}}$ its inverse. This has encoding strength 0.

The traditional definition of 'correct implementation' is not suitable for probabilistic gates or probabilistic sub-circuits, and so it is not compatible with our proofs. For this reason we provide the following, and more general, definition:

Definition 7. *Given two circuits* v, c *and an n-shares encoding E we say that c is a* ***correct implementation*** *of v for E (or that v is the* ***virtual circuit*** *of c for E) if there is an encoding R for the randoms such that:*

- *The encoding R transforms the random distribution of the virtual circuit into the distribution of the implemented circuit* $\vec{\mathbf{Rnds}}_c() \overset{d}{=} R.\vec{\mathbf{Enc}}(\vec{\mathbf{Rnds}}_v())$
- *For all inputs* $\vec{x} \in E.\vec{\mathbf{Enc}}[\mathbb{S}^i]$ *and all possible random values of the implemented circuit* $\vec{r} \in \vec{\text{supp}}[\vec{\mathbf{Rnds}}_c()]$, *we have that:*
$$\vec{v}_{\text{outs}}(E.\vec{\text{Dec}}(\vec{x}), R.\vec{\text{Dec}}(\vec{r})) = E.\vec{\text{Dec}}(\vec{c}_{\text{outs}}(\vec{x}, \vec{r})) \quad (1)$$

This correctness implies that $\vec{\mathbf{v}}_{\mathbf{outs}} \circ E.\vec{\text{Dec}} \overset{d}{=} E.\vec{\text{Dec}} \circ \vec{\mathbf{c}}_{\mathbf{outs}}$, and for deterministic gates this correctness is equivalent to the more common $\vec{\mathbf{v}}_{\mathbf{outs}} \circ E.\vec{\text{Dec}} = E.\vec{\text{Dec}} \circ \vec{\mathbf{c}}_{\mathbf{outs}}$. Moreover, the property 'a is a correct implementation of b' is transitive, and is preserved by the composition in parallel and in series as long as both implementation and virtual circuit are composed in the same way.

2.4 Circuit Compiler

Definition 8. *We can then define a* ***circuit compiler*** *as a pair of an encoding E from $\mathbb{S}_{\text{in}}$ to $\mathbb{S}_{\text{out}}$ and a compilation function* $\text{CC} : \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}} \to \mathcal{C}_{\mathbb{S}_{\text{out}},\mathcal{G}_{\text{out}}}$ *such that for all circuits $c \in \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}}$, the circuit* $\text{CC}(c)$ *is a correct implementation of c using E.*

From this definition and from the transitiveness of being a correct implementation quickly follows that given a circuit c with an encoding D, and given a compiler (E, CC), the circuit $CC(c)$ has the encoding $E \circ D$.

Like for the encodings, given the two circuit compilers O, I such that the input circuit type of O matches the output circuit type of I, we can define the compiler $O \circ I$ with $(O \circ I).E := O.E \circ I.E$ and $(O \circ I).CC := O.CC \circ I.CC$. This is a circuit compiler as the correctness property is transitive.

Definition 9. *Like in [BCP+20], we define the* ***circuit complexity matrix*** *of a compiler C (if it exists) as the matrix M_C such that for all circuits c,* $\vec{\text{Gates}}(C(c)) = M_C \cdot \vec{\text{Gates}}(c)$.

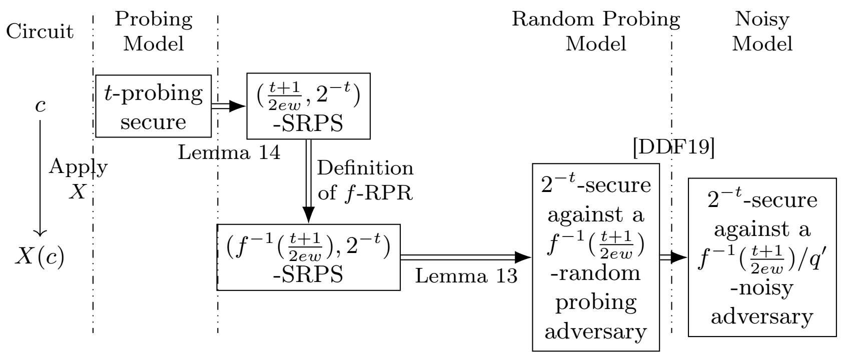
It is immediate that given two compilers O, I then $M_{O \circ I} = M_O M_I$.

Definition 10. *Given an encoding E with shares from $\mathbb{S}_{\text{in}}$ to $\mathbb{S}_{\text{out}}$, we can then define the* ***gadgets*** $\vec{G}$ *as the tuple of $|\mathcal{G}_{\text{in}}|$ circuits of type $\mathcal{C}_{\mathbb{S}_{\text{out}},\mathcal{G}_{\text{out}}}$, such that for all $g \in [|\mathcal{G}_{\text{in}}|]$, $\vec{G}_g$ is a correct implementation of g using E.*

Given the gadgets $\vec{G}$ with encoding E we define the function $CC_{\vec{G}}$ that substitutes every gate g with the circuit $\vec{G}_g$. Then the tuple $C := (E, CC_{\vec{G}})$ is a compiler. The circuit complexity matrix of this compiler exists, and if $\vec{G}$ has ℓ elements we have:

$$M_C = \left[\ \vec{\text{Gates}}(\vec{G}_1) \ \middle|\ \vec{\text{Gates}}(\vec{G}_2) \ \middle|\ \cdots \ \middle|\ \vec{\text{Gates}}(\vec{G}_\ell)\ \right] \tag{2}$$

2.5 Security



Circuit
Probing Model
Random Probing Model
Noisy Model
c
t-probing secure
$(\frac{t+1}{2ew}, 2^{-t})$ -SRPS
Lemma 14
Apply X
Definition of f-RPR
[DDF19]
2^{-t}-secure against a $f^{-1}(\frac{t+1}{2ew})$ -random probing adversary
2^{-t}-secure against a $f^{-1}(\frac{t+1}{2ew})/q'$ -noisy adversary
$X(c)$
$(f^{-1}(\frac{t+1}{2ew}), 2^{-t})$ -SRPS
Lemma 13

Figure 1: The properties of a t-probing secure circuit $c : \mathcal{C}_{\text{std},q}$ with $w = \Theta\left(t^2\right)$ wires, and of the result of compiling it with a compiler $X : \mathcal{C}_{\text{std},q} \to \mathcal{C}_{\text{std},q'}$ that is f-RPR.

To define the security of a circuit we first describe the adversaries from [DDF19, PGMP19]:

Definition 11. *Given a $\delta \in [0,1]$ we define a* ***δ-noisy adversary*** *on $\mathbb{S}^\ell$ a machine $\mathcal{A}$ that plays the following game against an oracle that knows $\vec{x} \in \mathbb{S}^\ell$:*

1. *$\mathcal{A}$ specifies a sequence of ℓ functions* **Noise** *such that every* $\mathbf{Noise}_i$ *is δ-noisy, where a function* $\mathbf{f}$ *is δ-noisy if the statistical distance between uniform distribution X and the conditional distribution $X|\mathbf{f}(X)$ is $\leq \delta$. See [PGMP19] for more on what they call δ-SD-noisy functions.*
2. *$\mathcal{A}$ receives* $\mathbf{Noise}_1(\vec{x}_1) \parallel \ldots \parallel \mathbf{Noise}_\ell(\vec{x}_\ell)$ *and outputs some value* $\vec{\mathbf{out}}_{\mathcal{A}}(\vec{x})$.

Definition 12. *Given a $p \in [0,1]$ we define a* ***p-random probing adversary*** *on $\mathbb{S}^\ell$ a machine $\mathcal{A}$ that plays the following game against an oracle that knows $\vec{x} \in \mathbb{S}^\ell$:*

1. $\mathcal{A}$ specifies a sequence $\vec{p} \in [0,p]^\ell$.

2. $\mathcal{A}$ receives $(\mathbf{f}_1 \parallel \ldots \parallel \mathbf{f}_\ell)(\vec{x})$ and outputs some value $\vec{\mathbf{out}}_\mathcal{A}(\vec{x})$, where $\mathbf{f}_i(x)$ returns x with probability $\vec{p}_i$, and it returns $\perp$ with probability $1 - \vec{p}_i$.

Definition 13. *Given a $t \in \mathbb{N}$ we define a **t-probing adversary** on $\mathbb{S}^\ell$ a machine $\mathcal{A}$ that plays the following game against an oracle that knows $\vec{x} \in \mathbb{S}^\ell$:*

1. $\mathcal{A}$ specifies a set of at most t wires: $\breve{W} \subseteq [\ell] : \left|\breve{W}\right| \leq t$.

2. $\mathcal{A}$ receives $\vec{x}|_{\breve{W}}$ and outputs some value $\vec{\mathbf{out}}_\mathcal{A}(\vec{x})$.

Thanks to the existing literature we know that for each δ-noisy adversary there is an equivalent $(\delta \cdot |\mathbb{S}|)$-random probing adversary [DDF19].

Definition 14. *Given a circuit c with encoding E such that the original circuit was defined over $\mathbb{S}_{\text{orig}}$ and with i inputs, and given $\varepsilon \in [0,1]$ we will say that c is **ε-secure against a given type of adversary** if for every adversary $\mathcal{A}$ of that type, there is a random variable $\vec{\mathbf{Sim}}$ such that for all $\vec{x} \in \mathbb{S}^i_{\text{orig}}$,*

$$\mathrm{SD}\left[\vec{\mathbf{Sim}}; \vec{\mathbf{out}}_\mathcal{A}(\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x})))\right] \leq \varepsilon \tag{3}$$

It is immediate that if a circuit is ε-secure against p-random probing adversaries, then it is ε-secure against $\frac{p}{|\mathbb{S}|}$-noisy adversaries.[5]

We now report the definition of Random Probing Security from [BCP+20]. While it was not explicit in that paper, it is easy to prove that 'c is (p,ε)-RPS' is equivalent to 'c is ε-secure against a p-random probing adversary'.

Definition 15. *A circuit c is (p,ε)-**RPS** (Random Probing Secure) if there is a $\vec{\mathbf{Sim}}$ such that for all $\vec{x} \in \mathbb{S}^i_{\text{orig}}$,*

$$\mathrm{SD}\left[\vec{\mathbf{Sim}}; \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{Leaking}}_c(p)}\right] \leq \varepsilon \tag{4}$$

To be able to define the RPR so that we can calculate it using the existing RPE, we introduce a stronger notion that we will call Strong RPS.

Definition 16. *Given a circuit $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$ with encoding E, we will say that c is **(p,ε)-SRPS** (Strong Random Probing Security) with $p, \varepsilon \in [0,1]$ if the probability that the leakage depends on any unmasked inputs is $\leq \varepsilon$:*

$$\Pr\left[\breve{\mathrm{Dep}}_{\left[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}\right]}(\breve{\mathbf{Leaking}}_c(p)) \neq \emptyset\right] \leq \varepsilon \tag{5}$$

If a circuit is (p,ε)-SRPS, then it is (p,ε)-RPS (see Lemma 13). Additionally, we can rise the ε and lower the p and the circuit remain SRPS, and every circuit with $w > 0$ internal wires is $(\varepsilon/w, \varepsilon)$-SRPS for every $\varepsilon \in [0,1]$. We also want to note the similarity between our definition of the SRPS and the t-probing security (which is equivalent to the 0-security against a t-probing adversary):

Definition 17. *Given a circuit c with encoding E, we will say that c is **t-probing secure** if $\leq t$ wires reveal no information on the inputs:*

$$\forall \breve{W} : \left|\breve{W}\right| \leq t.\ \breve{\mathrm{Dep}}_{\left[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}\right]}(\breve{W}) = \emptyset \tag{6}$$

In particular we have the following reduction: if a circuit c with w wires is t-probing secure, then it is $(\frac{t+1}{2ew}, 2^{-t})$-SRPS where e is the mathematical constant, see Lemma 14. If we apply this to the result of the most classic compilers for the t-probing model (which have a circuit size increase of $\Theta\left(t^2\right)$ e.g. [ISW03]) the compiled circuits are $(1/\Theta\left(t\right), 2^{-t})$-SRPS.

Definition 18. *We will say that a compiler $(E, CC) : C_{\text{in}} \to C_{\text{out}}$ is f-**RPR** (Random Probing Reducible) with $f : [0,1] \to [0,1]$ a continuous monotone non-strictly increasing function, if for all circuits $c \in C_{\text{in}}$ and for all $p, \varepsilon \in [0,1]$ we have that 'c is $(f(p), \varepsilon)$-SRPS' implies '$CC(c)$ is (p,ε)-SRPS'.*

[5] This $1/|\mathbb{S}|$ is present throughout the paper, and is only for the SD metric. For the ARE or RE metric it is absent, which means that the field-extension compiler provides no visible advantage. For the EN metric the opposite is true [PGMP19].

We note that the f-RPR is preserved if $f(\cdot)$ is increased. We now report an immediate lemma that proves the composition shown in Figure 2.

Lemma 2. *Given the circuit types $C_{\text{in}}, C_{\text{mid}}, C_{\text{out}}$, given the compiler $I : C_{\text{in}} \to C_{\text{mid}}$ that is f_I-RPR and given a compiler $O : C_{\text{mid}} \to C_{\text{out}}$ that is f_O-RPR, then $O \circ I$ is $(f_I \circ f_O)$-RPR.*

Corollary 1. *By induction with Lemma 2 we obtain that a sequence of compilers C such that C_i is f_i-RPR, then $C_n \circ \ldots \circ C_1$ is $(f_1 \circ \ldots \circ f_n)$-RPR.*

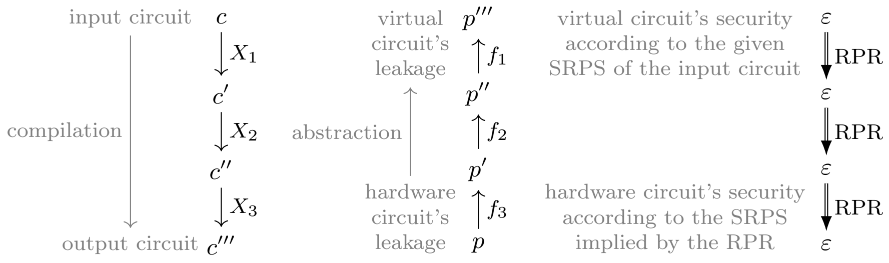

input circuit
compilation
output circuit
virtual circuit's leakage
abstraction
hardware circuit's leakage
virtual circuit's security according to the given SRPS of the input circuit
RPR
hardware circuit's security according to the SRPS implied by the RPR

Figure 2: Visual description of the compilation's security; each compiler X_i is f_i-RPR.

2.6 Compiler Sequences

Definition 19. *A **compiler sequence** is an infinite sequence S of circuit compilers $S_j : C_{\text{in}} \to C_{\text{out}}$ with the circuit types $C_{\text{in}}, C_{\text{out}}$ independent of j.*

For example, the classical expansion [BCP+20] describes the expansion as the compiler sequence $S_i := C^i$ which uses the compiler C in every step of the expansion.

The following properties of compiler sequences are a generalization of the properties of gadgets and compilers found in [BCP+20] so that we can be apply them to any compiler sequence. See subsection 3.4 and Figure 4 for more details on their relationship.

Definition 20. *We say that P is a **tolerated leakage** for a compiler sequence S if there is a sequence f such that S_j is f_j-RPR and $f_m(P) \to 0$.*

In other words, the more the parameter m is increased, the more the leakage of the virtual circuit goes to 0, and this happens for all $p \leq P$ due to the monotonicity of all f.

Definition 21. *We say that λ is a **size amplification order** of a compiler sequence C if the increase in circuit size is $\|M_{C_m}\|_1 = \mathcal{O}(\lambda^m)$.*

Definition 22. *We say that $d > 1$ is a **security amplification order** for a compiler sequence C and a tolerated leakage P if $\log_2 f_m(P) = \Omega(d^m)$ where f is the same as the one in the definition of P.*

From those two amplification orders we can derive how fast the expansion reaches the target leakage rate, this is similar to the exponent of [BCP+20]:

Definition 23. *We call $e := \frac{\log \lambda}{\log d}$ an **expansion exponent** for a compiler sequence C and a tolerated leakage P, where λ is a size amplification order for C and d is an security amplification order for C and P.*

We now consider a circuit parameterized in its security level, and we analyze its asymptotic size when we compile it to reach a constant tolerated leakage.

Theorem 1. *Given a compiler sequence $S : C_{\text{in}} \to C_{\text{out}}$ with tolerated leakage P and expansion exponent e, given a parametric circuit $c_\kappa \in C_{\text{in}}$ that is $(2^{-p(\kappa)}, 2^{-\kappa})$-SRPS for some $p : (0, \infty) \to (0, \infty)$; then there is a function $n : (0, \infty) \to \mathbb{N}$ such that the compiled circuit $c'_\kappa := S_{n(\kappa)}(c_\kappa)$ satisfies the following properties:*

- *For all $\kappa > 0$, the circuit c'_κ is $(P, 2^{-\kappa})$-SRPS.*
- *As $\kappa \to \infty$, $\|c'_\kappa\| = \mathcal{O}(\|c_\kappa\| \, p(\kappa)^e)$.*

The proof of this theorem is in subsection A.3, and it has the following immediate corollary that can be proven like in Figure 3.

Corollary 2. *Given a compiler sequence $S : C_{\text{mid}} \to C_{\text{out}}$ with tolerated leakage P and expansion exponent e, given a compiler sequence $I : C_{\text{in}} \to C_{\text{mid}}$ where I_t is a t-probing secure compiler with polynomial complexity, and given a circuit $c \in C_{\text{in}}$; then there is a function $n : \mathbb{N} \to \mathbb{N}$ such that the compiled circuit $c'_t := S_{n(t)}(I_t(c))$ satisfies the following properties:*

- *For all $t > 0$, the circuit c'_t is 2^{-t}-secure against a $P/|\mathbb{K}_{\text{out}}|$-noisy adversary.*
- *As $t \to \infty$, $\|c'_t\| = \mathcal{O}\left(\|I_t(c)\| \log(t)^e\right)$.*

In other words, with a polylograrithmic size increase, any t-probing secure compiler can be made to create circuits 2^{-t}-secure against a δ-noisy adversary, where δ and the exponent of the logarithm are indepent from the circuit being compiled.

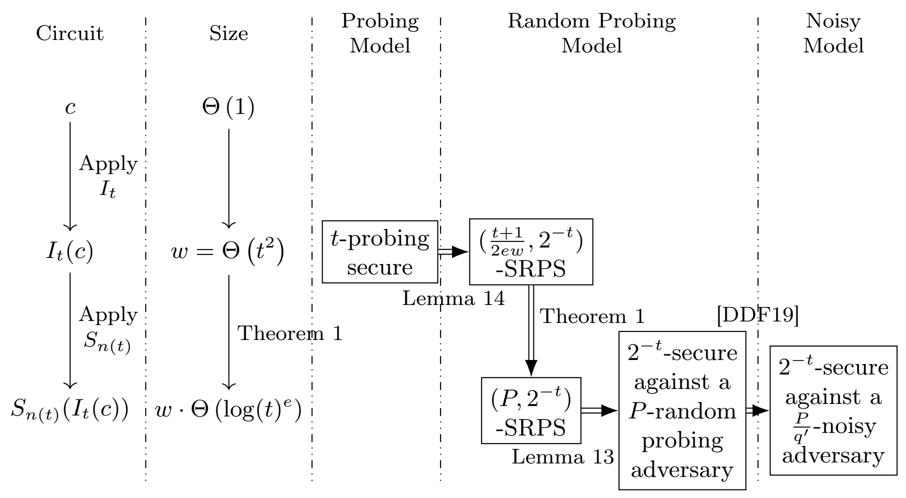


Figure 3: Visual representation of Corollary 2 and its proof.

3 Calculating Those Properties from the RPE.

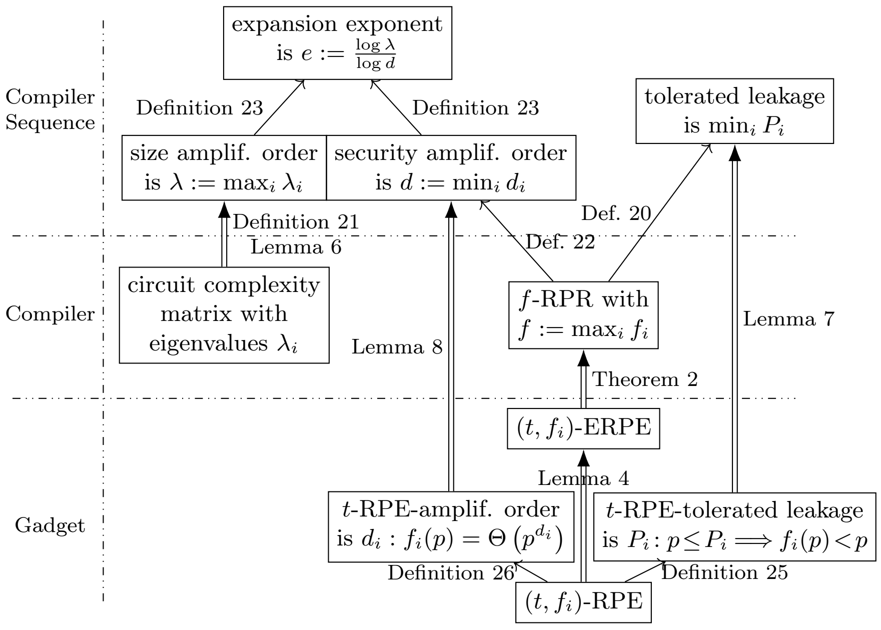
In order to provide the aforementioned results, we need show how to calculate the RPR from the RPE property, so that we can reuse the existing results and tools [BCP+20, BRT21]. In contrast to those papers, we aim to provide a full generic proof for all kind of gates, gadgets, and encodings. To this end, we introduce the Extended RPE property (ERPE) and we show three things: the RPE of [BCP+20] implies the ERPE, if all the gadgets of a compiler are (t, f)-ERPE, the compiler is f-RPR (Theorem 2), and that for the optimal t described in [BRT21], the ERPE allows us to approximately halven the number of random gates in the random gadget.

After this, we consider the classic expansion, and we show how to calculate the security amplification order, the size amplification order and the tolerated leakage from the existing properties [BCP+20] which we also show in Figure 4.

3.1 RPE

The RPE property from [BCP+20] analyzes an n-shares gadget G of a gate g by creating a parallel between the j-th input/output of g and the relative group of shares $\check{shares}_j$ of G.

For compactness, instead of providing multiple definitions like [BCP+20], we provide a single definition valid for any number of inputs and outputs. Also, as their Sim_1^G returns two results, we split it in the two functions O' and I, while their Sim_2^G is implicit in the notion of simulatability. Lastly, for our proof we need to add three requirements on the f of the $(\cdot, f)$-RPE: continuity, monotonicity, and $f : [0, 1] \to [0, 1]$. Those are fulfilled by [BCP+20]'s tool VRAPS.



expansion exponent is $e := \frac{\log \lambda}{\log d}$
Compiler Sequence
Definition 23
Definition 23
tolerated leakage is $\min_i P_i$
size amplif. order is $\lambda := \max_i \lambda_i$
security amplif. order is $d := \min_i d_i$
Definition 21
Lemma 6
Def. 20
Def. 22
Compiler
circuit complexity matrix with eigenvalues λ_i
f-RPR with $f := \max_i f_i$
Lemma 7
Lemma 8
Theorem 2
(t, f_i)-ERPE
Lemma 4
Gadget
t-RPE-amplif. order is $d_i : f_i(p) = \Theta\left(p^{d_i}\right)$
t-RPE-tolerated leakage is $P_i : p \le P_i \implies f_i(p) < p$
Definition 26
Definition 25
(t, f_i)-RPE

Figure 4: The relationships between the various properties for the classical expansion, and how to calculate everything from [BCP+20]'s RPE and circuit complexity matrix.

Definition 24. *Given a gadget $G \in \mathcal{C}_{\mathbb{S}_{\text{out}},\mathcal{G}_{\text{out}}}$ (with w internal wires) a correct implementation for a fully-leakable deterministic gate $g \in \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}}$ (with i inputs and o outputs) using the n-shares additive encoding E, and given a monotone and continuous function $f : [0,1] \to [0,1]$, we say that G is (t, f)-**RPE** (Random Probing Expandability) with $t \in \mathbb{N} \cap [0, n-1]$ if there are:*

- *a function $\breve{O}'$ for the outputs that the simulation is actually providing;*
- *a function $\breve{I}$ for the input dependencies of the simulation;*

such that for all subsets of output wires $\breve{O} \subseteq [n \cdot o]$,

1 *For every subset of the internal wires $\breve{W} \subseteq [w]$ of the gadget, we have that,*[6] $\breve{I}(\breve{O}, \breve{W}) \xrightarrow{\text{sim}} \vec{\mathbf{G}}_{\mathbf{all}}(\cdot)|_{\breve{W} \,\|\, \breve{O}'(\breve{O},\breve{W})}$.

2 *We define the input failure events $\breve{F}(\breve{O}, \breve{W}) \subseteq [i]$ which happen if the simulation needs more than t shares for a given input. More formally, $j \in \breve{F}(\breve{O}, \breve{W})$ if and only if $\left|\breve{I}(\breve{O}, \breve{W}) \cap \breve{shares}_j\right| > t$. Then for every leakage probability $p \in [0,1]$, we have that $\breve{F}(\breve{O}, \breve{\mathbf{Leaking}}_G(p)) \overset{d}{=} \breve{\mathbf{Leaking}}_g(f(p))$.*

3 *For all subsets of internal wires $\breve{W} \subseteq [w]$, for every output wire $j \in [o]$ of the gate, if $\left|\breve{O} \cap \breve{shares}_j\right| \le t$ then $\breve{O}'(\breve{O}, \breve{W}) \cap \breve{shares}_j = \breve{O} \cap \breve{shares}_j$, otherwise $\left|\breve{O}'(\breve{O}, \breve{W}) \cap \breve{shares}_j\right| = n - 1$.*

We define the following two properties roughly equivalent to those from [BCP+20] and which can also be calculated with a tool like VRAPS. The differences are purely formal and they are needed for the proofs.

Definition 25. *We say that P is a t-**RPE-tolerated leakage** of some gadget G if there is an f such that G is (t, f)-RPE, and for all $p \in (0, P]$ we have that $f(p) < p$, which means that compiling with G improves the security of the circuit.*

Definition 26. *We say that d is a t-**RPE-amplification order** of some gadget G if there is an f such that G is (t, f)-RPE, and $f(p) = \Theta\left(p^d\right)$.*

[6]We use the operator for the concatenation of vectors also for set of indices. Given $\breve{a}_j \subseteq [\ell_j]$, $\breve{a}_1 \,\|\, \breve{a}_2 := \breve{a}_1 \cup \{i + \ell_1 : i \in \breve{a}_2\}$

3.2 From the RPE to the ERPE

To show the rationale behind the ERPE and to better explain it, we first provide two intermediate extensions of the RPE, and both are valid expandibility proerties.

To do so, we begin with defining the codomain of $\breve{O}'(\breve{O}, \cdot)$ for some $\breve{O}$, and we extended this to support any encoding with strength. We first define it for a single output:

Definition 27. *Given an n-shares encoding with strength $k < n$, and a $t \in \mathbb{N} \cap [0, k]$, we say that a given $\breve{\breve{s}}$ is* ***a set of t-possible alternative combinations of shares*** *if $\breve{\breve{s}}$ is either the set of all combinations that have exactly k shares (i.e. $\breve{c} \in \breve{\breve{s}} \iff \breve{c} \subseteq [n] \wedge |\breve{c}| = k$) or a set with a single element $\breve{c}$ for some $\breve{c} \subseteq [n]$ with $\leq t$ shares.*

This means that for single-output gates, for every $\breve{O}$ there is a set $\breve{\breve{s}}$ of t-possible alternative combinations of shares such that the codomain of $\breve{O}'(\breve{O}, \cdot)$ is $\breve{\breve{s}}$. More concretely, if $|\breve{O}| \leq t$ then $\breve{\breve{s}} = \{\breve{O}\}$, otherwise $\breve{\breve{s}}$ is independent from $\breve{O}$ and it contains all the combinations with k shares.

Definition 28. *We say that a given $\breve{\breve{s}}$ is* ***a set of t-possible alternative combinations of ℓ inputs (or outputs)*** *if $\breve{\breve{s}} = \breve{\breve{s}}_1 \parallel \ldots \parallel \breve{\breve{s}}_\ell$ where $\breve{\breve{s}}_j$ are all sets of t-possible alternative combinations of shares, and where $\cdot \parallel \cdot$ operator for sets of sets of indices is the cartesian product, except that each pair does not create a tuple but they are concatenated using the $\cdot \parallel \cdot$ operator for sets of indices.*

Now we can now rewrite the RPE with 'for all subsets of output wires $\breve{O} \subseteq [n \cdot o]$' substituted with 'for all sets $\breve{\breve{o}}$ of t-possible alternative combinations of o outputs' and the third point of the RPE replaced by '$\breve{O}'(\breve{\breve{o}}, \cdot)$ has codomain $\breve{\breve{o}}$'. This new definition is equivalent to that of the RPE when we consider gadgets with the additive encoding, but it also works with any other encoding with strength.

Then we need to relax the second point of the RPE, or more specifically that $\breve{F}(\breve{\breve{o}}, \breve{\mathbf{Leaking}}_G(p)) \overset{d}{=} \breve{\mathbf{Leaking}}_g(f(p))$. To do this, we define the following:

Definition 29. *Given two discrete random variables $\mathbf{a}, \mathbf{b}$, we say $\mathbf{a} \overset{d}{\leq} \mathbf{b}$ iff for all monotone non-strictly increasing predicates[7] P we have that* $\Pr[P(\mathbf{a})] \leq \Pr[P(\mathbf{b})]$.

This is a partial order for the equivalence relation $\overset{d}{=}$, and it is such that $\breve{\mathbf{Leaking}}_c(p) \overset{d}{\leq} \breve{\mathbf{Leaking}}_c(p')$ for any leakage probabilities $p \leq p'$ and any circuit c. This can be quickly proven from the following lemma:

Lemma 3. *Given $\vec{\mathbf{v}}, \vec{\mathbf{u}}, \vec{\mathbf{V}}, \vec{\mathbf{U}}$ pairwise independent such that $\vec{\mathbf{v}} \overset{d}{\leq} \vec{\mathbf{V}}$ and $\vec{\mathbf{u}} \overset{d}{\leq} \vec{\mathbf{U}}$ then $\vec{\mathbf{v}} \parallel \vec{\mathbf{u}} \overset{d}{\leq} \vec{\mathbf{V}} \parallel \vec{\mathbf{U}}$.*

We can now generalize the RPE's '$\breve{F}(\breve{\breve{o}}, \breve{\mathbf{Leaking}}_G(p)) \overset{d}{=} \breve{\mathbf{Leaking}}_g(f(p))$' by substituting the '$\overset{d}{=}$' with a '$\overset{d}{\leq}$', and by also requiring that '$\breve{F}(\breve{\breve{o}}, \breve{\mathbf{Leaking}}_G(p)) \overset{d}{=} \breve{F}(\{\emptyset\}, \breve{\mathbf{Leaking}}_G(p))$ to ensure that the distribution of the failing events is independent from $\breve{\breve{o}}$.

Lastly, to make the ERPE support probabilistic gates, we need to generalize the simulatability property to allow simulation from a function instead of from a set of inputs:

Definition 30. *A probabilistic function $\mathbf{f} : \mathbb{S}_{\text{in}} \to \mathbb{S}_{\text{out}}$* ***can be simulated*** *using some probabilistic function $\mathbf{g}$ with domain $\mathbb{S}_{\text{in}}$ if there is a probabilistic function* **Sim** *such that for all $x \in \mathbb{S}_{\text{in}}$ we have that $\mathbf{f}(x) \overset{d}{=} \mathbf{Sim}(\mathbf{g}(x))$.*

To say that $\mathbf{f}$ can be simulated using $\mathbf{g}$, we could write that $\mathbf{g} \xrightarrow{\text{sim}} \mathbf{f}$, or that for all $x \in \mathbb{S}_{\text{in}}$, $\mathbf{g}(x) \xrightarrow{\text{sim}} \mathbf{f}(x)$. We want to note that the existing simulatability property $\breve{I} \xrightarrow{\text{sim}} \vec{\mathbf{f}}$ is equivalent to $(\vec{x} \mapsto \vec{x}|_{\breve{I}}) \xrightarrow{\text{sim}} \vec{\mathbf{f}}$.

Putting this all together we obtain the ERPE:

Definition 31. *Given a generic circuit $c \in \mathcal{C}_{\mathbb{S}_{\text{out}}, \mathcal{G}_{\text{out}}}$ (with w internal wires) a correct implementation of a circuit $v \in \mathcal{C}_{\mathbb{S}_{\text{in}}, \mathcal{G}_{\text{in}}}$ (with i inputs, o outputs) using the n-shares encoding E with strength k and the encoding of the randoms R, given a continuous and monotone function $f : [0, 1] \to [0, 1]$, and a $t \in [0, k] \cap \mathbb{N}$, we say that c is (t, f)-**ERPE** (Extended Random Probing Expandability) of v if there is:*

- *a function $\breve{O}_c$ for the outputs actually being simulated;*

[7]We use $true > false$ so that if P is monotone increasing, so is $x \mapsto \Pr[P(x)]$.

- *a function $\breve{\breve{i}}_c$ for the dependency to a set of t-possible alternative combinations of i inputs required for the simulation;*
- *a function $\breve{W}'_c$ for the dependencies on the wires of v;*

such that for all sets $\breve{\breve{o}}$ of t-possible alternative combinations of o outputs,

1. *For every subset of the internal wires $\breve{W} \subseteq [w]$, for all the possible inputs actually provided $\breve{I} \in \breve{\breve{i}}_c(\breve{o}, \breve{W})$, the outputs actually simulated $\breve{O}_c(\breve{o}, \breve{W}, \breve{I})$ and the wires $\breve{W}$ can be simulated from the inputs $\breve{I}$ and virtual wires $\breve{W}'_c(\breve{o}, \breve{W})$. More formally, for all inputs $\vec{x} \in E.\vec{\mathbf{Enc}}[\mathbb{S}^i_{\text{in}}]$ and for all the virtual randoms with probability greater than zero $\vec{r'} \in \breve{\text{supp}}[\vec{\mathbf{Rnds}}_v()]$,*

$$\vec{x}|_{\breve{I}} \parallel \vec{v}_{\text{wires}}(E.\vec{\text{Dec}}(\vec{x}), \vec{r'})|_{\breve{W}'_c(\breve{o},\breve{W})} \xrightarrow{\text{sim}} \vec{c}_{\text{all}}(\vec{x}, R.\vec{\mathbf{Enc}}(\vec{r'}))|_{\breve{W} \parallel \breve{O}_c(\breve{o},\breve{W},\breve{I})} \tag{7}$$

2. *For all leakage rates $p \in [0,1]$, $\breve{W}'_c(\{\emptyset\}, \breve{\mathbf{Leaking}}_c(p)) \overset{d}{\leq} \breve{\mathbf{Leaking}}_v(f(p))$ and $\breve{W}'_c(\breve{o}, \breve{\mathbf{Leaking}}_c(p)) \overset{d}{=} \breve{W}'_c(\{\emptyset\}, \breve{\mathbf{Leaking}}_c(p))$.*
3. *For every subset of the internal wires $\breve{W} \subseteq [w]$, for all the possible inputs actually provided $\breve{I} \in \breve{\breve{i}}_c(\breve{o}, \breve{W})$, the outputs actually simulated must be $\breve{O}_c(\breve{o}, \breve{W}, \breve{I}) \in \breve{o}$.*

It is possible to see how the second and third points of the ERPE are very close to the second and third point of the RPE extended earlier in this section. The difference in the second point is that the RPE uses only failure events of the inputs, while the ERPE uses the set of virtual leaking wires, which includes all the internal wires of v, including those that are not inputs.

The ERPE has two more differences compared to the RPE. The first is that the input dependency function no longer returns all the inputs necessary for the simulation, but some of the information must be taken from the virtual wires. E.G. I can not contain all the shares of a wire. Also, notice how the simulation must work for all possible combinations of inputs present in the set of t-possible alternatives. This is because when circuits are combined in series, the chosen combination is decided by the input circuit, and the output circuit needs to be able to work with all t-possible combinations.

Lastly, we have the most obvious difference, which is the randoms. The encoding of the randoms was already used in the definition of correct implementation to create a link between the randoms of the virtual circuit and that of the implementation, and here it is used in the same way.

To conclude, we have the following immediate proposition, which can be proven by restricting the ERPE to the gadgets considered by the RPE, and by following the previous explanations:

Lemma 4. *The (t, f)-RPE implies the (t, f)-ERPE.*

Lastly, note that the (t, f)-ERPE is preserved by increasing f (as long as f remains continuous, monotone, etc). This can be shown using the aformentioned properties of $\overset{d}{\leq}$ and $\breve{\mathbf{Leaking}}_c(\cdot)$.

3.3 Obtaining the RPR

We will now enunciate the main theorem for the expansion, its proof is in subsection B.2.

Theorem 2. *Given the gadgets $\vec{G}$ for an n-shares encoding E with strength, such that for all input gates g the gadget $\vec{G}_g$ is (t, f)-ERPE of g, then the compiler $(E, CC_{\vec{G}})$ is f-RPR.*

The random gadget with n randoms from [BCP+20, BRT21] is not RPE, but with the ERPE we have the immediate and more general lemma:

Lemma 5. *For the n-shares additive encoding, given a $t \in [0, n-1] \cap \mathbb{N}$, we consider the gadget obtained by a parallel of n gagets, of which $> t$ are random gates and the remaining are constant-0 gates. This gadget is (t, f)-ERPE for the random gate, for any continuous monotone $f : [0,1] \to [0,1]$.*

With this we can define the compiler sequences from [BCP+20]:

Definition 32. *We call a* ***classic compiler****, a compiler $\mathcal{C}_{\text{std},q} \to \mathcal{C}_{\text{std},q}$ with additive encoding such that its random gadget is made of a parallel of n random gates, and such that its constant-c gadgets are made by a parallel of n constant gates.*

With this we can obtain the following result that links our paper with [BCP$^+$20]. This result can be derived from the last two propositions, that the f-ERPE is preserved by rising f, and Theorem 2:

Theorem 3. *Given a classic compiler such that its addition, subtractions, copy, multiplicaion gadgets are respectively (t,f_{add})-RPE, (t,f_{sub})-RPE, (t,f_{copy})-RPE, (t,f_{mul})-RPE, then that compiler is f-RPR with $f := \max_i f_i$.*

3.4 Classic Expansion

As we are using a different formalization, in this section we will analyze the classical expansion strategy [BCP$^+$20, BRT21], and we show that we can calculate the size amplification order, security amplification order and tolerated leakage from the properties of their gadgets in a way that mirrors existing results.

Lemma 6. *Given a compiler sequence $C_n := X^n$, with diagonalizable circuit complexity matrices M_X, then the highest module of any eigenvalue of M_X is a size amplification order.*

Lemma 7. *Given a compiler sequence $C_m := X^m$ such that X is a classic compiler, given a t such that the addition, multiplication, copy, subtraction gates have a t-RPE-tolerated leakage of respectively P_{add}, P_{mul}, P_{copy}, P_{sub}, then $P := \min_i P_i$ is a tolerated leakage for C.*

Lemma 8. *Given a compiler sequence $C_m := X^m$ such that X is a classic compiler, given a t such that the addition, multiplication, copy, subtraction gates have a t-RPE-amplification order of respectively d_{add}, d_{mul}, d_{copy}, d_{sub}, then $d := \min_i d_i$ is a security amplification order for C for any valid tolerated leakage.*

4 Building the Main Compiler Sequence

In this section we will provide the compiler sequence which can be used to compile the output of a t-probing secure compiler (over a field with characteristic ρ) to make it tolerate a constant noise of $2^{-7.41}/\rho$ with a cube-logarithmic size increase. To do that we first need to define one compiler and a few properties of the composition of compiler sequences.

4.1 Field-Extension Compiler

First we analyze two common types of gates for encodings with strength 0:

Lemma 9. *Any leakless implementation of a leakless gate for a strength-0 encoding is $(0,f)$-ERPE for any continuous and monotone function $f : [0,1] \to [0,1]$.*

Lemma 10. *Given a gadget G (with w internal wires) for the fully-leakable deterministic gate g (with i inputs) that is correct for an n-shares encoding E with strength 0, then G is $(0,f)$-ERPE, with $f(x) := \min\{1, (w \cdot x)^{1/i}\}$.*

We can now present a simple field-extension compiler, which is f-RPR with $f(p) = \Theta\left(kp^{1/2}\right)$, and the 1-norm of its circuit complexity matrix is $\Theta\left(k^2\right)$.

Compiler 1. *We define the field-extension compiler as a compiler $\mathcal{C}_{\text{std},q^k} \to \mathcal{C}_{\text{std},q}$ with k shares, with the field-extension encoding, which creates an extension field $\mathbb{F}_{q^k}$ from a field $\mathbb{F}_q$ for some prime q by using some irreducible polynomial $P \in \mathbb{F}_q[x]$ with degree $k > 0$.*

This extension compiler follows the usual way to build a field of order q^k from one of order q using the same irreducible polynomial P. The biggest gadget is the multiplication, which is quadratic in k.

4.2 Composition of Compiler Sequences

Before providing the main compiler sequence, we will first analyze how composing compiler sequences alters their properties.

Lemma 11. *Given a compiler sequence C and given a compiler I that is f-RPR for some $f(x) = x^{\Omega(1)}$ as $x \to 0$, then $\{C_n \circ I\}_{n=1}^{\infty}$, when meaningful, has all the tolerated leakage, security amplification order, size amplification order and expansion exponent of C.*

Lemma 12. *Given a compiler sequence I and a compiler sequence O (with respectively a security amplification order of d_I, d_O for a tolerated leakage of P_I, P_O) then there is a k such that*[8] *$\{O_k \circ I_n\}_{n=1}^{\infty}$ has a security amplification order of d_I relative to a tolerated leakage of P_O.*

In other words, it is the last compilers that define the tolerated leakage, so they should be the last trasformation applied to a circuit.

4.3 Main Compiler Sequence

We define the main compiler sequence of this article as follows:

Compiler Sequence 1. *For every q that is a power of a prime, and given a fixed $m \in \mathbb{N}$, we define $X_t := H^m \circ C^t \circ E$, where H is the [BRT21]'s 3-shares compiler with a refresh at the input of this compiler's copy gadget (it has a tolerated leakage of $2^{-7.41}$) E is the field-extension compiler we described, C is the parametric compiler from [BRT21] instantiated with 21 shares (it has a expansion exponent of 3).*

The value of m represents how many expansions of the high-tolerance compiler H are necessary to make the virtual leakage applied to C lower than H's tolerated leakage. Otherwise the expansion of C would worsen the security as t increases.

We do not provide the value for m given q because this compiler sequence is only meant to be indicative and to show the asymptotical results. Any real implementation should not use an output sequence made of m identical compilers H; instead it should gradually shift the tolerated leakage/expansion exponent tradeoff to minimize the concrete complexity [BRTV21]. It should also use a better field-extension compiler.

Finally, we provide the last theorem to obtain our main result, which is immediate from the properties described until now:

Theorem 4. *For every prime power q there is an $m \in \mathbb{N}$ such that X has expansion exponent 3 relative to a tolerated leakage $2^{-7.41}$.*

5 Conclusions

In this paper we have provided a more general and formalized framework to handle the random probing model and the expansion strategy, and we have shown a useful result that could not be obtained with the existing formalization.

By combining Theorem 4 and Corollary 2 we show how given a parametric circuit c_t which is t-probing secure, we can transform it into a circuit that is 2^{-t} secure against a $2^{-7.41}$-noisy adversry for a $\mathcal{O}\left(\log(t)^3\right)$ size increase.

Lastly, we hope the generic nature of our definitions and proofs and the new capabilites that our formalization introduces will help ease future research in the random probing model.

A From Probing Security to Constant Noise

We will prove the lemmas and theorems that are needed for the Corollary 2, which shows that with a polylograrithmic size increase, any t-probing secure compiler can be made to create circuits 2^{-t}-secure against a δ-noisy adversary for some constant δ.

[8][BRTV21] says $\{O_n \circ I_k\}_{n=1}^{\infty}$ instead of $\{O_k \circ I_n\}_{n=1}^{\infty}$, see footnote 3.

A.1 Lemma 13

Lemma 13 (SRPS to RPS)**.** *If a circuit is (p,ε)-SRPS, then it is (p,ε)-RPS.*

Proof. If a circuit c with n-share encoding E with $n\cdot i$ inputs and original field $\mathbb{S}_{\text{orig}}$ is (p,ε)-SRPS then we have the following:[9]

$$\begin{aligned}
&1-\varepsilon\\
\leq&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\Pr\Big[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{W})=\emptyset\Big]\\
=&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\mathrm{Is}\Big[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{W})=\emptyset\Big]\\
=&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\mathrm{Is}\Big[\exists\vec{\mathbf{Sim}}.\,\forall\vec{x}\in\mathbb{S}^i_{\text{orig}}.\ \vec{\mathbf{Sim}}|_{\breve{W}}\overset{d}{=}\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\Big]\\
=&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\max_{\vec{\mathbf{Sim}}}\min_{\vec{x}\in\mathbb{S}^i_{\text{orig}}}\mathrm{Is}\Big[\vec{\mathbf{Sim}}|_{\breve{W}}\overset{d}{=}\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\Big]\\
\leq&\max_{\vec{\mathbf{Sim}}'}\min_{\vec{x}\in\mathbb{S}^i_{\text{orig}}}\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\mathrm{Is}\Big[\vec{\mathbf{Sim}}'(\breve{W})|_{\breve{W}}\overset{d}{=}\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\Big]
\end{aligned}$$

This means that there is a $\vec{\mathbf{Sim}}'$ such that for every $\vec{x}\in\mathbb{S}^i_{\text{orig}}$,

$$1-\varepsilon\leq\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\mathrm{Is}\Big[\vec{\mathbf{Sim}}'(\breve{W})|_{\breve{W}}\overset{d}{=}\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\Big]$$

Let us analyze this inequality:

$$\begin{aligned}
\varepsilon\geq&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\mathrm{Is}\Big[\neg(\vec{\mathbf{Sim}}'(\breve{W})|_{\breve{W}}\overset{d}{=}\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}})\Big]\\
\geq&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{Leaking}}_c(p)\Big]\mathrm{SD}\ \Big[\vec{\mathbf{Sim}}(\breve{W})|_{\breve{W}};\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\Big]
\end{aligned}$$

Let us define $\breve{\mathbf{L}}:=\breve{\mathbf{Leaking}}_c(p)$ and $\vec{\mathbf{Sim}}'(\breve{W}):=\vec{\mathbf{Sim}}(\breve{W})|_{\breve{W}}$. Then,

$$\begin{aligned}
=&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{L}}\Big]\mathrm{SD}\ \Big[\vec{\mathbf{Sim}}'(\breve{W});\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\Big]\\
=&\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{L}}\Big]\frac{1}{2}\sum_{\vec{y}}\Big|\Pr\Big[\vec{\mathbf{Sim}}'(\breve{W})=\vec{y}\Big]-\Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}=\vec{y}\Big]\Big|\\
=&\frac{1}{2}\sum_{\vec{y}}\sum_{\breve{W}}\Pr\Big[\breve{W}=\breve{\mathbf{L}}\Big]\Big|\Pr\Big[\vec{\mathbf{Sim}}'(\breve{W})=\vec{y}\Big]-\Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}=\vec{y}\Big]\Big|\\
=&\frac{1}{2}\sum_{\vec{y}}\Big|\Pr\Big[\vec{\mathbf{Sim}}'(\breve{\mathbf{L}})=\vec{y}\Big]-\Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{L}}}=\vec{y}\Big]\Big|\\
=&\mathrm{SD}\ \Big[\vec{\mathbf{Sim}}'(\breve{\mathbf{Leaking}}_c(p));\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{Leaking}}_c(p)}\Big]
\end{aligned}$$

This immediately implies the definition of 'c is (p,ε)-RPS'. □

A.2 Lemma 14

As we need to obtain the SRPS property from the probing security, we can not use the result from [DDF19], which instead obtains the weaker property of security in the random probing model. For this reason we prove the following proposition:

[9]We define $\mathrm{Is}[A]:=\Pr[A]$ but limited to deterministic statements.

Lemma 14. *If a circuit c with w wires is t-probing secure, then it is $(\frac{t+1}{2ew}, 2^{-t})$-SRPS where e is the mathematical constant.*

Proof. We can prove this with the following passages:

$$\begin{aligned}
&\Pr\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leaking}}_c(p)) \neq \emptyset\right] \\
&= \sum_{\breve{W}\subseteq[w]} \Pr\left[\breve{W} = \breve{\mathbf{Leaking}}_c(p)\right] \mathrm{Is}\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{W}) \neq \emptyset\right]
\end{aligned}$$

By definition of 'secure in the t-probing model', $|\breve{W}| \leq t$ guarantees no dependency. We also use Stirling's approximation, which is also a lower bound.

$$\leq \sum_{\substack{\breve{W}\subseteq[w] \\ |\breve{W}|>t}} p^{|\breve{W}|} \leq \sum_{l:=t+1}^{w} \binom{w}{l} p^l \leq \sum_{l:=t+1}^{w} \frac{w^l}{l!} p^l \leq \sum_{l:=t+1}^{w} \frac{(pw)^l}{\sqrt{2\pi l}(\frac{l}{e})^l} \leq \sum_{l:=t+1}^{w} (\frac{epw}{t+1})^l$$

The thesis quickly follows by choosing $p := \frac{t+1}{2ew}$. □

A.3 Theorem 1

Theorem 1: Given a compiler sequence $S : C_{\text{in}} \to C_{\text{out}}$ with tolerated leakage P and expansion exponent e, given a parametric circuit $c_\kappa \in C_{\text{in}}$ that is $(2^{-p(\kappa)}, 2^{-\kappa})$-SRPS for some $p : (0,\infty) \to (0,\infty)$; then there is a function $n : (0,\infty) \to \mathbb{N}$ such that the compiled circuit $c'_\kappa := S_{n(\kappa)}(c_\kappa)$ satisfies the following properties:

- For all $\kappa > 0$, the circuit c'_κ is $(P, 2^{-\kappa})$-SRPS.
- As $\kappa \to \infty$, $\|c'_\kappa\| = \mathcal{O}\left(\|c_\kappa\|\, p(\kappa)^e\right)$.

Proof. Let us call d the security amplification order and λ the size amplification order that lead to the expansion exponent e. Then from the definition of d, there is a sequence of functions f such that S_m is f_m-RPR and $\log_2 f_m(P) = \Omega\left(d^m\right)$ as $m \to \infty$. In other words there is a $b > 0$ such that:

$$\liminf_{m\to\infty} \frac{|\log_2 f_m(P)|}{d^m} = 2b \tag{8}$$

This means that there is an n' such that for every $m \geq n'$ we have that $f_m(P) \leq 2^{-bd^m}$. We can now choose $n(\kappa) := \max\{n', \log_d \frac{p(\kappa)}{b}\}$, and it is easy to proove that $f_{n(\kappa)}(P) \leq 2^{-p(\kappa)}$. This means that as c_κ is $(2^{-p(\kappa)}, 2^{-\kappa})$-SRPS, then it is also $(f_{n(\kappa)}(P), 2^{-\kappa})$-SRPS. We can then apply the definition of RPR to '$S_{n(\kappa)}$ is $f_{n(\kappa)}$-RPR' to obtain that c'_κ is $(P, 2^{-\kappa})$-SRPS, which satisfies point 1 of the theorem.

Then by definition of size amplification order, the definition of circuit complexity matrix, of circuit size and of p-norm, $\|c'_\kappa\| = \mathcal{O}\left(\|c_\kappa\|\, \lambda^{n(\kappa)}\right)$. Also, as $n(\kappa) = \frac{\log p(\kappa)}{\log d} + \Theta(1)$ then $\lambda^{n(\kappa)} = \Theta\left(p(\kappa)^e\right)$. Putting them together we obtain that $\|c'_\kappa\| = \mathcal{O}\left(\|c_\kappa\|\, p(\kappa)^e\right)$ which proves the second point of the theorem. □

B Expansion

In this appendix we will prove our main theorem for the expansion: Theorem 2. To do this we first need to give a concrete definition of what is a circuit.

B.1 Definition of Circuit

We believe the best way to describe the type of circuits we consider is the following, as it simplifies our proofs:

Definition 33. *We will define a **circuit** $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$ as either:*

- *An index of a gate in $\mathcal{G}$.*

- *Parallel composition of two smaller circuits $c := c' \parallel c''$.*
- *Serial composition of two smaller circuits $c := c_o \circ c_i$, as long as the number of inputs of c_o matches that of the outputs of c_i.*
- *The identity circuit, which outputs its single input.*
- *The swap circuit, which swaps its two inputs.*

Given a circuit $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$, we can now give the formal definitions of the functions described in subsection 2.2:

- If c is the g-th gate, then $\vec{c}_{\text{outs}}$ is the function that defines the gate. If it is leakless then $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := []$ otherwise $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := \vec{x} \parallel \vec{r}$, $\vec{\textbf{Rnds}}_c()$ draws a value from the probability distribution in the description of the gate. Also, $\vec{\text{Gates}}(c)_g := 1$ and $\vec{\text{Gates}}(c)_{\neq g} := \vec{0}$. Lastly, given $\mathbf{v} := \breve{\textbf{Leaking}}_c(p)$, all the events $j \in \mathbf{v}$ have intependent probability p.
- If c is an identity circuit, then $\vec{c}_{\text{outs}}([a]) = [a]$, $\vec{\textbf{Rnds}}_c() := []$, $\vec{\text{Gates}}(c) := \vec{0}$. $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := []$, $\breve{\textbf{Leaking}}_c(p) := \emptyset$.
- If c is a swap circuit, then $\vec{c}_{\text{outs}}([a,b]) = [b,a]$, $\vec{\textbf{Rnds}}_c() := []$, $\vec{\text{Gates}}(c) := \vec{0}$. $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := []$, $\breve{\textbf{Leaking}}_c(p) := \emptyset$.
- If c is a parallel $c' \parallel c''$, then $\vec{\textbf{Rnds}}_c() := \vec{\textbf{Rnds}}_{c'}() \parallel \vec{\textbf{Rnds}}_{c''}()$, $\vec{\text{Gates}}(c) := \vec{\text{Gates}}(c') + \vec{\text{Gates}}(c'')$, $\breve{\textbf{Leaking}}_c(p) := \breve{\textbf{Leaking}}_{c'}(p) \parallel \breve{\textbf{Leaking}}_{c''}(p)$, and

$$\vec{c}_{\text{outs}}(\vec{x'} \parallel \vec{x''}, \vec{r'} \parallel \vec{r''}) := \vec{c'}_{\text{outs}}(\vec{x'},\vec{r'}) \parallel \vec{c''}_{\text{outs}}(\vec{x''},\vec{r''})$$
$$\vec{c}_{\text{wires}}(\vec{x'} \parallel \vec{x''}, \vec{r'} \parallel \vec{r''}) := \vec{c'}_{\text{wires}}(\vec{x'},\vec{r'}) \parallel \vec{c''}_{\text{wires}}(\vec{x''},\vec{r''})$$

- If c is a series $o \circ i$, then $\vec{\textbf{Rnds}}_c() := \vec{\textbf{Rnds}}_i() \parallel \vec{\textbf{Rnds}}_o()$, $\vec{\text{Gates}}(c) := \vec{\text{Gates}}(i) + \vec{\text{Gates}}(o)$, $\breve{\textbf{Leaking}}_c(p) := \breve{\textbf{Leaking}}_i(p) \parallel \breve{\textbf{Leaking}}_o(p)$, and

$$\vec{c}_{\text{outs}}(\vec{x}, \vec{r_i} \parallel \vec{r_o}) := \vec{o}_{\text{outs}}(\vec{i}_{\text{outs}}(\vec{x},\vec{r_i}),\vec{r_o})$$
$$\vec{c}_{\text{wires}}(\vec{x}, \vec{r_i} \parallel \vec{r_o}) := \vec{i}_{\text{wires}}(\vec{x},\vec{r_i}) \parallel \vec{o}_{\text{wires}}(\vec{i}_{\text{outs}}(\vec{x},\vec{r_i}),\vec{r_o})$$

B.2 Theorem 2

Theorem 2: Given the gadgets $\vec{G}$ correct for an n-shares encoding E, and such that for all gates $g \in \mathcal{G}_{\text{in}}$ the gadget $\vec{G}_g$ is (t,f)-ERPE of g, then the compiler $(E, CC_{\vec{G}})$ is f-RPR.

Proof. This proof is divided in two parts. First we prove by induction over the definition of circuit that all compiled circuits are (t,f)-ERPE. Then we show that if this property holds then the compiler is f-RPR. We break this proof into the following lemmas. □

Lemma 15. *Any correct and leakless n-shares implementation G of the identity circuit and of the swap circuit for an encoding with strength k is (t,f)-ERPE for any $t \in [0,k] \cap \mathbb{N}$ and for any monotone and continuous $f : [0,1] \to [0,1]$.*

Proof. It follows immediately from applying the definition of ERPE. □

Lemma 16. *Given a compiler (E, CC_b) and two circuits v_1, v_2 such that $c_1 := CC_b(v_1)$ is (t,f)-ERPE of v_1 and $c_2 := CC_b(v_2)$ is (t,f)-ERPE of v_2, then $c_1 \parallel c_2$ is (t,f)-ERPE of $v_1 \parallel v_2$.*

Proof. We will call $c = c_1 \parallel c_2$, $v = v_1 \parallel v_2$, and we prove that c is (t,f)-ERPE of v by using the definition. In particular, we choose:

$$\breve{O}_c(\breve{o}_1 \parallel \breve{o}_2, \breve{W}_1 \parallel \breve{W}_2, \breve{I}'_1 \parallel \breve{I}'_2) := \breve{O}_{c_1}(\breve{o}_1, \breve{W}_1, \breve{I}'_1) \parallel \breve{O}_{c_2}(\breve{o}_2, \breve{W}_2, \breve{I}'_2)$$
$$\breve{\breve{i}}_c(\breve{o}_1 \parallel \breve{o}_2, \breve{W}_1 \parallel \breve{W}_2) := \breve{\breve{i}}_{c_1}(\breve{o}_1, \breve{W}_1) \parallel \breve{\breve{i}}_{c_2}(\breve{o}_2, \breve{W}_2)$$
$$\breve{W}'_c(\breve{o}_1 \parallel \breve{o}_2, \breve{W}_1 \parallel \breve{W}_2) := \breve{W}'_{c_1}(\breve{o}_1, \breve{W}_1) \parallel \breve{W}'_{c_2}(\breve{o}_2, \breve{W}_2)$$

The three properties of the ERPE can be proven by subsituting all the definitions with their values, and then use the respective property of the sub-circuits. The only thing worth notice is that property 2 requires Lemma 3. □

Lemma 17. *Given a compiler (E, CC_b) and two circuits c'_i, c'_o such that $c_i := CC_b(c'_i)$ is (t,f)-ERPE of c'_i and $c_o := CC_b(c'_o)$ is (t,f)-ERPE of c'_o, then $c_o \circ c_i$ is (t,f)-ERPE of $c'_o \circ c'_i$.*

Proof. We will call $c := c_o \circ c_i$, $c' := c'_o \circ c'_i$, and we prove that c is (t,f)-ERPE of c' by using the definition. In particular, we choose:

$$\breve{O}_c(\breve{\breve{o}}, \breve{W}_i \parallel \breve{W}_o, \breve{I}) := \breve{O}_{c_o}(\breve{\breve{o}}, \breve{W}_o, \breve{O}_{c_i}(\breve{\breve{i}}_{c_o}(\breve{\breve{o}}, \breve{W}_o), \breve{W}_i, \breve{I}))$$
$$\breve{\breve{i}}_c(\breve{\breve{o}}, \breve{W}_i \parallel \breve{W}_o) := \breve{\breve{i}}_{c_i}(\breve{\breve{i}}_{c_o}(\breve{\breve{o}}, \breve{W}_o), \breve{W}_i)$$
$$\breve{W}'_c(\breve{\breve{o}}, \breve{W}_i \parallel \breve{W}_o) := \breve{W}'_{c_i}(\breve{\breve{i}}_{c_o}(\breve{\breve{o}}, \breve{W}_o), \breve{W}_i) \parallel \breve{W}'_{c_o}(\breve{\breve{o}}, \breve{W}_o)$$

Then it is immediate that $\breve{\breve{i}}_c(\cdot,\cdot)$ is a set of t-possible alternative combinations of i_c inputs. We can now prove the three properties of the definition of ERPE: for all sets $\breve{\breve{o}}$ of t-possible alternative combinations of o outputs,

1 For every $\breve{W}_i \subseteq [w_i]$, $\breve{W}_o \subseteq [w_o]$, we define $\breve{\breve{m}} := \breve{\breve{i}}_{c_o}(\breve{\breve{o}}, \breve{W}_o)$, $\breve{\breve{i}} := \breve{\breve{i}}_{c_i}(\breve{\breve{m}}, \breve{W}_i)$

   For every $\breve{I} \in \breve{\breve{i}}$ we define $\breve{M} := \breve{O}_{c_i}(\breve{\breve{m}}, \breve{W}_i, \breve{I})$ and $\breve{O} := \breve{O}_{c_o}(\breve{\breve{o}}, \breve{W}_o, \breve{M})$

   For every $\vec{r'_i} \in \breve{\text{supp}}[\vec{\mathbf{Rnds}}_{c'_i}()]$, we define $\vec{\mathbf{r_i}} = R_{c_i}.\vec{\mathbf{Enc}}(\vec{r'_i}))$.

   For every $\vec{r'_o} \in \breve{\text{supp}}[\vec{\mathbf{Rnds}}_{c'_o}()]$ we define $\vec{\mathbf{r_o}} = R_{c_o}.\vec{\mathbf{Enc}}(\vec{r'_o}))$.

   For every $\vec{x} \in E.\vec{\mathbf{Enc}}[\mathbb{S}^i_{\text{in}}]$ we define $\vec{x'} := E.\vec{\text{Dec}}(\vec{x})$ and

$$\begin{aligned} \vec{\mathbf{y}} &:= (\vec{c_i})_{\text{outs}}(\vec{x}, \vec{\mathbf{r_i}}) & \vec{\mathbf{z}} &:= (\vec{c_o})_{\text{outs}}(\vec{\mathbf{y}}, \vec{\mathbf{r_o}}) \\ \vec{y'} &:= (\vec{c'_i})_{\text{outs}}(\vec{x'}, \vec{r'_i}) & \vec{z'} &:= (\vec{c'_o})_{\text{outs}}(\vec{y'}, \vec{r'_o}) \\ \vec{\mathbf{w_i}} &:= (\vec{c_i})_{\text{wires}}(\vec{x}, \vec{\mathbf{r_i}}) & \vec{\mathbf{w_o}} &:= (\vec{c_o})_{\text{wires}}(\vec{\mathbf{y}}, \vec{\mathbf{r_o}}) \\ \vec{w'_i} &:= (\vec{c'_i})_{\text{wires}}(\vec{x'}, \vec{r'_i}) & \vec{w'_o} &:= (\vec{c'_o})_{\text{wires}}(\vec{y'}, \vec{r'_o}) \end{aligned}$$

   We need to prove:

$$\vec{x}_{\breve{I}} \parallel \vec{w'_i}|_{\breve{W}'_{c_i}(\breve{\breve{m}}, \breve{W}_i)} \parallel \vec{w'_o}|_{\breve{W}'_{c_o}(\breve{\breve{o}}, \breve{W}_o)} \xrightarrow{\text{sim}} \vec{\mathbf{w_i}}|_{\breve{W}_i} \parallel \vec{\mathbf{w_o}}|_{\breve{W}_o} \parallel \vec{\mathbf{z}}|_{\breve{O}} \tag{9}$$

   We can use the property 1 of the ERPE of c_i and we obtain:

$$\vec{x}_{\breve{I}} \parallel \vec{w'_i}|_{\breve{W}'_{c_i}(\breve{\breve{m}}, \breve{W}_i)} \xrightarrow{\text{sim}} \vec{\mathbf{y}}|_{\breve{M}} \parallel \vec{\mathbf{w_i}}|_{W_i} \tag{10}$$

   Using the property 3 of the ERPE of c_i we have that $\breve{M} \in \breve{\breve{m}}$. Using the definition of correctness, we obtain that $\vec{y'} = E.\vec{\text{Dec}}(\vec{\mathbf{y}})$. Together, those allow us to use the property 3 of the ERPE of c_o, and we obtain:

$$\vec{\mathbf{y}}_{\breve{M}} \parallel \vec{w'_o}|_{\breve{W}'_{c_o}(\breve{\breve{o}}, \breve{W}_o)} \xrightarrow{\text{sim}} \vec{\mathbf{z}}_{\breve{O}} \parallel \vec{\mathbf{w_o}}|_{\breve{W}_o} \tag{11}$$

   Combining (10) and (11) we obtain the required (9).

2 We need to prove two things. The first one is that $\breve{W}'_c(\breve{\breve{o}}, \breve{\mathbf{Leaking}}_{c'}(p)) \overset{d}{=} \breve{W}'_c(\{\emptyset\}, \breve{\mathbf{Leaking}}_{c'}(p))$. We will prove this by showing that for every $\breve{\breve{o}}$,

$$\breve{W}'_c(\breve{\breve{o}}, \breve{\mathbf{Leaking}}_c(p)) \overset{d}{=} \breve{W}'_{c_i}(\{\emptyset\}, \breve{\mathbf{Leaking}}_{c_i}(p)) \parallel \breve{W}'_{c_o}(\{\emptyset\}, \breve{\mathbf{Leaking}}_{c_o}(p)) \tag{12}$$

For every $\breve{w}_i', \breve{w}_o'$, let $\breve{\mathbf{L}}_{\mathbf{o}} := \mathbf{Lea\breve{k}ing}_{c_o}(p)$, then

$$
\begin{aligned}
&\Pr\Big[\breve{W}_c'(\breve{o}, \mathbf{Lea\breve{k}ing}_c(p)) = \breve{w}_i' \parallel \breve{w}_o'\Big] \\
=&\Pr\Big[\breve{W}_{c_i}'(\breve{\breve{i}}_{c_o'}(\breve{o}, \breve{\mathbf{L}}_{\mathbf{o}}), \mathbf{Lea\breve{k}ing}_{c_i}(p)) \parallel \breve{W}_{c_o}'(\breve{o}, \breve{\mathbf{L}}_{\mathbf{o}}) = \breve{w}_i' \parallel \breve{w}_o'\Big] \\
=&\sum_{\breve{L}_o} \Pr\Big[\breve{W}_{c_i}'(\breve{\breve{i}}_{c_o}(\breve{o},\breve{L}_o),\mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i' \wedge \breve{W}_{c_o}'(\breve{o}, \breve{L}_o) = \breve{w}_o' \wedge \breve{\mathbf{L}}_{\mathbf{o}} = \breve{L}_o\Big] \\
=&\sum_{\breve{L}_o} \Pr\Big[\breve{W}_{c_i}'(\breve{\breve{i}}_{c_o}(\breve{o},\breve{L}_o),\mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i'\Big] \Pr\Big[\breve{W}_{c_o}'(\breve{o},\breve{L}_o) = \breve{w}_o' \wedge \breve{\mathbf{L}}_{\mathbf{o}} = \breve{L}_o\Big]
\end{aligned}
$$

For the property 2 of the ERPE of c_i

$$
\begin{aligned}
=&\sum_{\breve{L}_o} \Pr\Big[\breve{W}_{c_i}'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i'\Big] \Pr\Big[\breve{W}_{c_o}'(\breve{o}, \breve{L}_o) = \breve{w}_o' \wedge \breve{\mathbf{L}}_{\mathbf{o}} = L_o\Big] \\
=&\Pr\Big[\breve{W}_{c_i}'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i'\Big] \sum_{\breve{L}_o} \Pr\Big[\breve{W}_{c_o}'(\breve{o}, \breve{L}_o) = \breve{w}_o' \wedge \breve{\mathbf{L}}_{\mathbf{o}} = L_o\Big] \\
=&\Pr\Big[\breve{W}_{c_i}'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i'\Big] \Pr\Big[\breve{W}_{c_o}'(\breve{o}, \breve{\mathbf{L}}_{\mathbf{o}}) = \breve{w}_o'\Big] \\
=&\Pr\Big[\breve{W}_{c_i}'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i'\Big] \Pr\Big[\breve{W}_{c_o}'(\breve{o}, \mathbf{Lea\breve{k}ing}_{c_o'}(p)) = \breve{w}_o'\Big]
\end{aligned}
$$

For the property 2 of the ERPE of c_o

$$
\begin{aligned}
=&\Pr\Big[\breve{W}_{c_i}'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_{c_i}(p)) = \breve{w}_i'\Big] \Pr\Big[\breve{W}_{c_o}'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_{c_o}(p)) = \breve{w}_o'\Big] \\
=&\Pr\Big[\breve{W}_{c_i}'(\emptyset, \mathbf{Lea\breve{k}ing}_{c_i}(p)) \parallel \breve{W}_{c_o}'(\emptyset, \mathbf{Lea\breve{k}ing}_{c_o}(p)) = \breve{w}_i' \parallel \breve{w}_o'\Big]
\end{aligned}
$$

The second thing we need to prove is that for all leakage rates $p \in [0, 1]$,

$$\breve{W}_c'(\{\emptyset\}, \mathbf{Lea\breve{k}ing}_c(p)) \overset{d}{\leq} \mathbf{Lea\breve{k}ing}_c(e(p)) \tag{13}$$

This is immediate, by using (12), Lemma 3, and the property 2 of the ERPE of c_o and c_i.

3 Is immediate by using property 3 of the sub-circuits.

□

Lemma 18. *Given a circuit v with a n-shares encoding V (with w internal wires), the compiled circuit c (with encoding $C := E \circ V$, i inputs, o outputs) that is (t, f)-ERPE of v, and given $\breve{I}_c$,$\breve{W}_c'$,$\breve{O}_c$ from the definition of ERPE, then:*

$$\forall \breve{W} \subseteq [w].\ \mathrm{D\breve{e}p}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ C.\mathbf{E\vec{n}c}]}(\breve{W}) \subseteq \mathrm{D\breve{e}p}_{[\vec{\mathbf{v}}_{\mathbf{wires}} \circ V.\mathbf{E\vec{n}c}]}(\breve{W}_c'(\{\emptyset\}, \breve{W})) \tag{14}$$

Proof. We know from the item 3 of the ERPE with $\breve{o} := \{\emptyset\}$, that for all $\breve{I} \in \breve{\breve{i}}_c(\{\emptyset\}, \breve{W})$, for all $\vec{x} \in E.\mathbf{E\vec{n}c}[\mathbb{S}_{\text{in}}^i]$ and for all $\vec{r'} \in \mathrm{su\breve{p}p}[\mathbf{R\vec{n}ds}_v()]$,

$$\vec{x}_{\breve{I}} \parallel \vec{v}_{\text{wires}}(E.\mathrm{D\vec{e}c}(\vec{x}), \vec{r'})|_{\breve{W}_c'(\{\emptyset\}, \breve{W})} \xrightarrow{\text{sim}} \vec{c}_{\text{wires}}(\vec{x}, R.\mathbf{E\vec{n}c}(\vec{r'}))|_{\breve{W}} \tag{15}$$

As this is true for all $\vec{r'} \in \mathrm{su\breve{p}p}[\mathbf{R\vec{n}ds}_v()]$, we can assign $\vec{\mathbf{r'}} := \mathbf{R\vec{n}ds}_v()$, and $\forall \vec{y} \in \mathbb{S}_{\text{in}}^i$ we can choose $\vec{\mathbf{x}} := E.\mathbf{E\vec{n}c}(\vec{y})$ and obtain:

$$E.\mathbf{E\vec{n}c}(\vec{y})|_{\breve{I}} \parallel \vec{\mathbf{v}}_{\mathbf{wires}}(\vec{y})|_{\breve{W}_c'(\{\emptyset\}, \breve{W})} \xrightarrow{\text{sim}} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\mathbf{E\vec{n}c}(\vec{y}))|_{\breve{W}} \tag{16}$$

As E has strength k, we have that $\mathrm{D\breve{e}p}_{[E.\mathbf{E\vec{n}c}]}(\breve{I}) = \emptyset$. As v has encoding V,

$$(\vec{\mathbf{v}}_{\mathbf{wires}} \circ V.\mathbf{E\vec{n}c})|_{\breve{W}_c'(\emptyset, \breve{W})} \xrightarrow{\text{sim}} (\vec{\mathbf{c}}_{\mathbf{wires}} \circ C.\mathbf{E\vec{n}c})|_{\breve{W}} \tag{17}$$

The thesis follows from the dependency function being minimal. □

Lemma 19. *Given a compiler (E, CC) such that for all circuits $v \in C_{\text{in}}$ the compiled circuit $CC(v)$ is (t, f)-ERPE of v, then (E, CC) is f-RPR.*

Proof. Let us call V the encoding of v, $C := E \circ V$ the encoding of $c := CC(v)$, and let us use $\breve{W}'_c$ from the definition of ERPE. To say that (E, CC) is f-RPR is to say that given a circuit v, if v is $(f(p), \varepsilon)$-SRPS, then c is (p, ε)-SRPS.

So we know by hypothesis that:

$$\Pr\left[\breve{\mathrm{Dep}}_{\left[\vec{\mathbf{v}}_{\mathbf{wires}} \circ V.\vec{\mathbf{Enc}}\right]}(\breve{\mathbf{Leaking}}_v(f(p))) \neq \emptyset\right] \leq \varepsilon \tag{18}$$

The item 2 of the ERPE says that $\breve{W}'_c(\{\emptyset\}, \breve{\mathbf{Leaking}}_c(p)) \overset{d}{\leq} \breve{\mathbf{Leaking}}_v(f(p))$ and as $\breve{W} \mapsto \breve{\mathrm{Dep}}_{\left[\vec{\mathbf{v}}_{\mathbf{wires}} \circ V.\vec{\mathbf{Enc}}\right]}(\breve{W}) \neq \emptyset$ is monotone, we can apply the definition of $\overset{d}{\leq}$ and obtain:

$$\Pr\left[\breve{\mathrm{Dep}}_{\left[\vec{\mathbf{v}}_{\mathbf{wires}} \circ V.\vec{\mathbf{Enc}}\right]}(\breve{W}'_c(\{\emptyset\}, \breve{\mathbf{Leaking}}_c(p))) \neq \emptyset\right] \leq \varepsilon \tag{19}$$

The thesis that c is (p, ε)-SRPS follows from using Lemma 18. □

References

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/\log(n))$ leakage rate. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 586–615, Vienna, Austria, May 8–12, 2016. Springer, Berlin, Heidelberg, Germany.

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[Ajt11] Miklos Ajtai. Secure computation with information leaking to an adversary. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 715–724, New York, NY, USA, 2011. Association for Computing Machinery.

[BCP$^+$20] Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Abdul Rahman Taleb. Random probing security: Verification, composition, expansion and new constructions. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 339–368, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[BDF24] Gianluca Brian, Stefan Dziembowski, and Sebastian Faust. From random probing to noisy leakages without field-size dependence. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part IV*, volume 14654 of *Lecture Notes in Computer Science*, pages 345–374, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[BRT21] Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 313–343, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

[BRTV21] Sonia Belaïd, Matthieu Rivain, Abdul Rahman Taleb, and Damien Vergnaud. Dynamic random probing expansion with quasi linear asymptotic complexity. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 157–188, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Heidelberg, Germany.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. *Journal of Cryptology*, 32(1):151–177, January 2019.

[GPRV21] Dahmun Goudarzi, Thomas Prest, Matthieu Rivain, and Damien Vergnaud. Probing security through input-output separation and revisited quasilinear masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):599–640, 2021.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Berlin, Heidelberg, Germany.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Heidelberg, Germany.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Berlin, Heidelberg, Germany.

[MT10] Ueli M. Maurer and Stefano Tessaro. A hardcore lemma for computational indistinguishability: Security amplification for arbitrarily weak PRGs with optimal stretch. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 237–254, Zurich, Switzerland, February 9–11, 2010. Springer, Berlin, Heidelberg, Germany.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a Rényi day. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159, Athens, Greece, May 26–30, 2013. Springer, Berlin, Heidelberg, Germany.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas Jensen, editors, *Smart Card Programming and Security*, pages 200–210, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.