Reframing And Extending The Random Probing Expandibility To Make Probing-Secure Compilers Tolerate A Constant Noise

Giuseppe Manzoni

Independent Researcher, Bergamo, Italy, giuseppe.manzoni@zelya.org

Abstract. In the context of circuits leaking the internal state, there are various models to analyze what the adversary can see, like the p-random probing model in which the adversary can see the value of each wire with probability p. In this model, for a fixed p, it's 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.
The existing proofs of the expansion work by first compiling the gadgets multiple times, and then by compiling the circuit with the resulting gadgets while assuming the worst from the original circuit. Instead, we reframe the expansion by proving 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 obtain two concrete results: (i) At the cost of an additional size factor $\mathcal{O}(\log(d)^3)$, any d-probing secure compiler can be used to produce stateless circuits with security 2^{-d} against any adversary that sees all wires with a constant SD-noise of $2^{-7.41}/\rho$, where ρ is the characteristic of the circuit's field. (ii) Any n-shares compiler with (t, f)-RPE gadgets needs $t+1$ (which in practice is $\lceil \frac{n}{2} \rceil$) randoms in the random gadget instead of n.

Keywords: Side-Channel Security · Leakage Resilience · Probing Model · Random Probing Model

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 modelled using the noisy leakage models [CJRR99, PR13a, DDF14, PGMP19]. In particular, we'll 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 [PR13a, PGMP19], the statistical distance in [DDF14, PGMP19, GPRV22], and the Average Relative Error in [PGMP19]). Of those metrics, we'll focus the most[2] on the statistical distance as it corresponds to an intuitive concept of the indistinguishability[3] between the distributions X and $X|\mathbf{f}(X)$.

[1] For more on this notation and concept see [PR13a, DDF14, PGMP19].

[2] For the other metrics the difference is a single factor, see footnote 4.

[3] i.e. $\mathrm{SD}\left[\mathbf{x};\mathbf{y}\right] \leq \delta$ iff no adversary $\mathcal{A}$ can distinguish between $\mathbf{x}$ and $\mathbf{y}$ with advantage better than δ, i.e. for all $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathbf{x}) = 1] - \Pr[\mathcal{A}(\mathbf{y}) = 1]| \leq \delta$. See for example [MT10].

Given a circuit and its noisy leakage δ, we can calculate the security of that circuit. In case the guaranteed security is not enough for the intended purpose, we can 'compile' it: transform it into a new circuit 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 cost of having a bigger circuit, but this is not guaranteed: if the circuit doesn't have enough noise (i.e. the leakage δ is too high), then compiling the circuit will only lead to a lower security. For this reason the ideal is a compiler able to guarantee an arbitrary security for the highest possible tolerated leakage, and with the lowest possible circuit size increase.

As the noisy models tend to be hard to handle when writing proofs, other models have been introduced like the the random probing model [ISW03, DDF14, BCP+20] in which each wire leaks the exact value with a given probability p, and leaks nothing with probability $1-p$. While this model doesn't 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 of $p/|\mathbb{K}|$ in the noisy model.

As we anticipated, the ideal compiler is one that can reach an arbitrary security, and do so while tolerating the highest possible leakage, and so the ideal is to tolerate a constant leakage regardless of the desired security. The first compiler we know of that has achieved this was described in [Ajt11]. It used expander graphs and it was followed by [ADF16] which simplified and improved it using geometric codes. [AIS18] provided an even simpler compiler, with (as calculated by [BCP+20]) a tolerated leakage in the random probing model of 2^{-26} and complexity of $\mathcal{O}\left(\kappa^{8.2}\right)$, and this was achieved using an expansion strategy of compilers using multi-party computation protocols. A further improvement was made by [BCP+20] whose compiler tolerated a leakage of $2^{-7.97}$ with complexity of $\mathcal{O}\left(\kappa^{7.5}\right)$ and this by using the Random Probing Expandability (or RPE) property. This expansion works by compiling a set of gadgets with themselves to obtain bigger and more secure gadgets. These are then compiled again and again until the wanted security level is reached, and then they are composed into the output circuit. This approach was then refined in [BRT21] which provided a more in-depth analysis of the RPE property, it calculated a few of its limits, and it provided gadgets that reach them. They provided a few alternative compilers. One with a better tolerated leakage $2^{-7.5}$ and lower complexity $\mathcal{O}\left(\kappa^{3.9}\right)$, while another works for a generic number of shares, with a complexity of $\mathcal{O}(\kappa^{e(n)})$ with $e(n) := \frac{\log(3n^2-2n)}{\log(\lfloor (n+1)/2 \rfloor)}$ where the number of shares and the desired security are unrelated due to the expansion. A successive article [BRTV21] extended the random probing expansion by allowing a different compiler at different stages of the expansion, even if we believe they made a mistake in one of the proofs, and so they inverted the order of the compilers, see footnote 5.

A completely different and widely researched model is 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. Various do so by using refresh gates that don't leak the internal computation, like [PR13b]. In this paper we'll focus on reductions without those leak-free refresh gates, and the first article we know that proved the reduction without them is [DDF14]. They provide a n-shares compiler based on [ISW03, RP10] that produces circuits that are $\lfloor (n-1)/2 \rfloor$-probing secure, and they show that given a circuit c, the compiled circuit has a security of $\|c\| \, e^{-n/12}$ for a noise of $\Theta\left(1/(n \cdot |\mathbb{K}|)\right)$. There have been improvements to this reduction, for example [PGMP19] achieves it by using a different metric. Yet, to our knowledge, there are no results that tolerate a constant noise for an arbitrary security by starting with a generic probing secure compiler.

1.1 Our Contributions

At the core of our paper there is the introduction of the Random Probing Reducibility (or RPR) property, which reduces the security of a compiled circuit to the security of the original one. In particular if a compiler is e-RPR, then any compiled circuit with leakage rate p can be abstracted into a virtual circuit with leakage rate $e(p)$, and the concrete circuit will have the same guaranteed security of the virtual circuit. In other words, given a circuit with a security of $2^{-\kappa}$ for a leakage probability $e(p)$, the compiled circuit has a security of $2^{-\kappa}$ for the leakage probability p. If $e(p) < p$ this guarantees the same security for a higher leakage probability.

With the RPR property we can calculate two out of the three main properties that characterize generic compiler sequences: the security amplification order and tolerated leakage, while the third is the size amplification order. Those three are all we need to calculate the asymptotic size increase necessary to reach a fixed leakage rate as the security level κ grows. More precisely, we analyze a circuit parametric in its guaranteed security level κ, so that c_κ has security $2^{-\kappa}$ for the leakage probability $2^{-p(\kappa)}$, for some function p. If we compile this circuit to obtain one with security $2^{-\kappa}$ for a constant tolerated leakage, then we have a size increase of the factor $\mathcal{O}\left(p(\kappa)^e\right)$ as $\kappa \to \infty$ for some exponent e that depends only on the the compiler sequence. This means that if we compile a fixed circuit c we obtain a complexity of $\mathcal{O}\left(\kappa^e\right)$, like in [BCP+20]. If instead we use an initial circuit c_t that is obtained by a t-probing secure compiler with polynomial complexity, then we obtain a complexity of $\mathcal{O}\left(\|c_t\| \log(t)^e\right)$ for the exact same exponent e. In particular we provide a compiler sequence with $e := 3$, with tolerated leakage rate $2^{-7.41}$.

Additionally we extend the RPE to tolerate more encodings. We provide the 'encoding strength' property that encapsulates the properties of the encoding that are required by the RPE, and 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 $\mathbb{F}_{p^m}$. This is why from a tolerated leakage rate $2^{-7.41}$ we obtain a tolerated noise of $2^{-7.41}/\rho$ instead of $2^{-7.41}/\rho^m$. For the common case of circuits whose fields has characteristic 2, like for the AES encryption, the tolerated noise is $2^{-8.41}$.

The main proof we include for the expansion consists in showing that a compiler made of (t, e)-RPE gadgets is e-RPR. This proof works by composition of sub-circuits: we use a property that is implied by the (t, e)-RPE, that it's preserved by composing two circuit, and such that if all the compiler's output circuits have that property, then the compiler is e-RPR. Yet the RPE is not suited for this job as it was designed to handle the implementation of a gate, not those of a generic circuit, so we need to introduce a new property that we call Extended RPE (or ERPE). As the ERPE needs to handle the implementations of generic probabilistic circuits, it's also capable of handling the implementation of generic probabilistic gates. For this reason we present it as our main expandability property instead of the narrower RPE.

Lastly, to show the concrete usefulness of this new property, we provide a $(t, \cdot)$-ERPE random gadget with only $t+1$ random gates. This means that an n-shares compiler made of (t, e)-RPE gadgets only need $t+1$ randoms in the random gadget, and not n. Thanks to the analysis from [BRT21] the optimal t for the RPE is $t := \left\lfloor \frac{n-1}{2} \right\rfloor$ which means a single step of the expansion converts each random into $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$ randoms instead of n of the existing random gadget that are required by [BCP+20, BRT21].

2 Notation and Fundamental Concepts

We'll use $\vec{x}$ for a vector, $\breve{x}$ for a set, $\mathbf{x}$ for a random variable. For functions, they describe their codomain.

Then we can give a few common/intuitive definitions. We'll indicate with $\Pr[\mathbf{A}]$ the probability of the event $\mathbf{A}$, and with $\mathrm{Is}\,[A]$ the function that returns 1 if the predicate A is true and 0 otherwise. With ':=' we indicate a definition and with '=' equality. Given a set $\breve{S}$ we write $\leftarrow \breve{S}$ to mean an anonymous random variable with the uniform distribution over $\breve{S}$, independent from all the random variables defined before it and used only there. Similarly to [BCP+20], we'll use $[m] := \mathbb{Z} \cap [1, m]$ as the set of indexes for a tuple of m values, $\mathcal{P}([m])$ is the power set of $[m]$. Additionally, we'll sometimes use vector operations on the subsets of $[m]$, by interpreting them like a vector $\{false, true\}^m$ that says if index i is in the set. In particular, we'll use the concatenation of two vectors $\vec{a} \parallel \vec{b}$ also on those sets. Lastly, we use $\mathbb{K}$ for a generic field, $\mathbb{F}_q$ for a field of order q and $\mathbb{S}$ for a set with at least two elements, and we'll note with $\breve{\mathrm{supp}}[\mathbf{x}]$ the support of $\mathbf{x}$ i.e. the possible values that $\mathbf{x}$ can take with probability different from zero. Also, for brevity we'll always use 'monotone' as 'monotone non-strictly increasing' unless specified otherwise, where for sets we'll use the partial order $\subseteq$, while for statements we'll consider $true > false$, and for functions we'll consider $f \leq g$ iff $\forall x.\ f(x) \leq g(x)$.

As the difference is important in various proofs, we'll define both $\vec{x}|_{\breve{I}}$ and $\vec{x}_{\breve{I}}$. We'll use $\vec{x}|_{\breve{I}}$ to return the elements of $\vec{x} \in \mathbb{K}^m$ that have the indexes in $\breve{I} \subseteq [m]$ while keeping track of which elements were selected. This by replacing with $\perp$ the elements of $\vec{x}$ whose indexes are not in $\breve{I}$. Instead we'll use $\vec{x}_{\breve{I}}$ if we're not keeping track of which elements are selected and we want a smaller tuple, the same notation as selecting a single element.

Lastly, 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 $\mathbf{f}(x) := \ldots$ we define the $f(x)$ that calculates the distribution of that expression, and whenever we use $\mathbf{f}(x)$ we mean $\leftarrow f(x)$. The intuitive description holds as long as the expression in definition of $\mathbf{f}(x) := \ldots$ has no correlation with any random variable defined outside it, which will always be the case.

Definition 1 (Partial Order)**.** Given two discrete random variables $\mathbf{a}, \mathbf{b}$, we say $\mathbf{a} \overset{d}{\leq} \mathbf{b}$ iff for all monotone predicates P we have that $\Pr[P(\mathbf{a})] \leq \Pr[P(\mathbf{b})]$.

In particular this is a partial order for the equivalence $\overset{d}{=}$ which asks if they have the same distribution, see Lemma 17, and this partial order is preserved by the parallel composition of arrays as long as the four random variables are independent, see Lemma 18. Both those lemmas are in Subsection B.2.

Definition 2 (Simulatability From A Function)**.** We'll say that 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 $\mathbf{Sim}$ 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 can now provide the common definition of simulatability written in terms of the more general notion introduced above:

Definition 3 (Simulatability)**.** We'll 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 $\breve{I} \subseteq [i]$ if it can be simulated from $(\vec{x} \mapsto \vec{x}_{\breve{I}})$.

Note that the simulatability from the inputs uses an input domain $\mathbb{S}_{\text{in}} \times \ldots \times \mathbb{S}_{\text{in}}$, while the simulatability allows any kind of domain, e.g. any subset of $\mathbb{S}_{\text{in}} \times \ldots \times \mathbb{S}_{\text{in}}$.

Definition 4 (Dependency Function)**.** Given a probabilistic function $\vec{\mathbf{f}} : \mathbb{S}_{\text{in}}^i \to \mathbb{S}_{\text{out}}^o$ we'll indicate with $\breve{\mathrm{Dep}}_{[\mathbf{f}]}$ its dependency function: the minimal function $\breve{Dep} : \mathcal{P}([o]) \to \mathcal{P}([i])$ such that for all subsets of the outputs $\breve{O} \subseteq [o]$ the function $\vec{\mathbf{f}}_{\breve{O}}$ can be simulated using the inputs $\breve{Dep}(\breve{O})$.

Note that the dependency function always exists and it's unique (Lemma 15) and it's monotone (Lemma 16). See Subsection B.1.

3 Circuit and Security

In this section we'll define the circuits that our proofs operate on. In particular, like the previous papers on the expansion [BCP+20, BRT21], they are circuits without memory and we consider only the leakage once the values in the wires become stable, and so we won't consider behaviors like glitches.

We'll also report the definition of the various security notions, 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.

3.1 Circuit Type

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

Definition 5 (Circuit Type)**.** We define a type of circuits as the tuple $(\mathbb{S}, \mathcal{G})$ which is identified by the two values:

- A finite set $\mathbb{S}$ with at least two elements, which describes the values that the circuit operates on.
- A finite enumeration $\mathcal{G}_j$, one for each gate, which contains the following:
  - 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 randoms used by the gate.
  - A boolean: if the gate's i inputs and r randoms either all leak (fully leakable gate) or none leak (leakless gate).

We are aware that the choice of leaking both inputs and the internally generated randoms is unusual, but it's necessary to allow the expansion to work with probabilistic gates. This is because while the random gate can have gadgets with no internal leakage, this is not always the case, and we must have a way to quantify the impact of this leakage. The most straight-forward way to do this is to allow the leakage of the randoms together with the inputs, as the two are independent, and together they completely define the output.

While our proofs are more generic, we define $\mathcal{C}_{\mathrm{std},q} := \mathcal{C}_{\mathbb{F}_q,\mathcal{G}_{\mathrm{std}}}$ as the circuit type with the standard gates, from Table 1.

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

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.

We define a circuit $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$ as a stateless composition of gates $\mathcal{G}$, with wires that contain the values in $\mathbb{S}$, and we'll use the following functions describe c's behavior. For a more formal definition see Appendix A.

- $\vec{c}_{\text{outs}} : \mathbb{S}^i \times \mathbb{S}^r \to \mathbb{S}^o$; the deterministic function that calculates the outputs of the circuit from its inputs and randoms.
- $\vec{c}_{\text{wires}} : \mathbb{S}^i \times \mathbb{S}^r \to \mathbb{S}^w$; the deterministic function that calculates the inputs and randoms of every fully-leakable gate in c, which are the leakable internal wires of c.
- $\vec{\mathbf{Rnds}}_c() \in \mathbb{S}^r$; the probabilistic function that calculates the random needed as parameter of the circuit, with the correct distribution.
- $\breve{\mathbf{Leak}}_c : [0,1] \to \mathcal{P}([w])$; the probabilistic function that maps the leakage probability of a single wire to a description of which of the leakable wires are leaking.
- $\vec{\text{Gates}}(c) \in \mathbb{N}^{|\mathcal{G}|}$; the vectorial function that for each gate returns how many gates of that type are in the circuit. This means that $\left\|\vec{\text{Gates}}(c)\right\|_1$ is the total number of gates in c, where $\|\cdot\|_1$ is the p-norm with $p = 1$.

For simplicity and compactness, we'll also define

- $\vec{c}_{\text{all}}(\vec{x}, \vec{r}) := \vec{c}_{\text{wires}}(\vec{x}, \vec{r}) \parallel \vec{c}_{\text{outs}}(\vec{x}, \vec{r})$
- $\vec{\mathbf{c}}_{\mathbf{outs}}(\vec{x}) := \vec{c}_{\text{outs}}(\vec{x}, \vec{\mathbf{Rnds}}_c())$
- $\vec{\mathbf{c}}_{\mathbf{wires}}(\vec{x}) := \vec{c}_{\text{wires}}(\vec{x}, \vec{\mathbf{Rnds}}_c())$
- $\vec{\mathbf{c}}_{\mathbf{all}}(\vec{x}) := \vec{c}_{\text{all}}(\vec{x}, \vec{\mathbf{Rnds}}_c())$
- $\|c\| := \left\|\vec{\text{Gates}}(c)\right\|_1$

And we call a circuit leakless if $\vec{c}_{\text{wires}}(\cdot, \cdot) = []$ as it has no leakable wires. We want to note that the function $\breve{\mathbf{Leak}}_c$ is monotone. More accurately, given any circuit c and any leakage probability $p, p' \in [0,1]$, Lemma 19 shows that that:

$$p \leq p' \implies \breve{\mathbf{Leak}}_c(p) \overset{d}{\leq} \breve{\mathbf{Leak}}_c(p')$$

3.2 Encoding

All the circuits we consider will have an associated encoding.

Definition 6 (Encoding)**.** We define an encoding E as a pair of:

- A probabilistic function to encode $E.\vec{\mathbf{Enc}} : \mathbb{D}_\ell \to \mathbb{S}_{\text{out}}^{\ell'}$ with $\mathbb{D}_\ell \subseteq \mathbb{S}_{\text{in}}^{\ell}$, such that it returns a new encoding for its parameter, for some tuples (ℓ, ℓ'). Given a set $\breve{U} \subseteq \mathbb{D}_\ell$, we write with $E.\vec{\mathbf{Enc}}[\breve{U}]$ the union of the supports of $E.\vec{\mathbf{Enc}}$ for every input in the set $\breve{U}$, i.e. the set of the valid encodings 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: 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 7 (n-shares Encoding)**.** An n-shares encoding is one that has $\ell' = n \cdot \ell$ for fall $\ell \in \mathbb{N}$, has $\mathbb{D}_\ell := \mathbb{S}_{\text{in}}^{\ell}$, and is compatible with the concatenation of vectors: $E.\vec{\textbf{Enc}}(\vec{a} \parallel \vec{b}) \overset{d}{=} E.\vec{\textbf{Enc}}(\vec{a}) \parallel E.\vec{\textbf{Enc}}(\vec{b})$ and $E.\vec{\text{Dec}}(\vec{c} \parallel \vec{d}) = E.\vec{\text{Dec}}(\vec{c}) \parallel E.\vec{\text{Dec}}(\vec{d})$, assuming the the number of elements in $\vec{c}$, $\vec{d}$ is a multiple of n.

Definition 8 (Encoding Strength)**.** Given an n-shares encoding, we say that 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 virtual wires: for all $\breve{S} \subseteq [n]$ with $|\breve{S}| \leq k$, $\breve{\text{Dep}}_{[E.\vec{\textbf{Enc}}]}(\breve{S}) = \emptyset$.
- From the value of k shares and that of the virtual wire is possible to uniquely reconstruct the value of the missing shares so that the decoding matches the virtual wire.

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 $\breve{\text{Dep}}_{[E.\vec{\textbf{Enc}}]}(\breve{I}) = \emptyset$ if $|\breve{I}| \leq k$ and $[1]$ otherwise.*

Note that an n-shares encoding with strength $n-1$ behaves like the additive encoding while one with strength 0 doesn't really do any masking. Also note that [BCP+20] defined the RPE for the additive encoding, but their proofs only need strength $n-1$ to work. Additionally, the RPE can be easily generalized to any encoding strength with little change, see Subsection 4.1.

While the proof of the expansion (Theorem 2) is more generic, in most of this paper we'll use encodings with strength that operate over a finite field. In particular we'll consider the following two encodings:

- For finite field $\mathbb{K}$, given $\vec{v} \in (\mathbb{K}\backslash\{0\})^n$, we define the *$\vec{v}$-linear encoding*, with $\mathbb{S}_{\text{in}}, \mathbb{S}_{\text{out}} := \mathbb{K}$, such that $\forall \vec{x} \in \mathbb{K}^n$. $\vec{\text{Dec}}(\vec{x}) := \vec{v} \cdot \vec{x}$ and with the encoding function that selects an encoding uniformly between the possible ones. They all have encoding strength of $n-1$. This family of encoding includes the *polynomial sharings* of [GPRV22], and the more common *additive encoding* which is the $\vec{1}$-linear encoding, with $\vec{\text{Dec}}(x) = \sum_i x_i$.
- For an irreducible polynomial $P \in \mathbb{F}_q[x]$ with degree $m \geq 2$ and some q power of a prime, we define the *field-extension encoding*, with $\mathbb{S}_{\text{in}} := \mathbb{F}_{q^m}$, $\mathbb{S}_{\text{out}} := \mathbb{F}_q$, m shares, 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 $\vec{\textbf{Enc}}$ as the function that outputs the array of coefficients of the input value. $\vec{\text{Dec}}$ is simply the inverse of $\vec{\textbf{Enc}}$. This has encoding strength of 0.

The traditional definition of 'correct implementation' isn't suitable as we need to maintain a correspondence between the values of the virtual circuit and the values of the correct implementation even when they pass through a probabilistic sub-circuit. Not doing so will also undermine the expansion and the Random Probing Reducibility, as it'd break the abstraction between the virtual circuit and the concrete one.

Definition 9 (Correctness)**.** 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 randoms $\vec{r} \in \operatorname{supp}[\vec{\mathbf{Rnds}}_c()]$ of the implemented circuit,
$$\vec{v}_{\text{outs}}(E.\vec{\text{Dec}}(\vec{x}), R.\vec{\text{Dec}}(\vec{r})) = E.\vec{\text{Dec}}(\vec{c}_{\text{outs}}(\vec{x}, \vec{r}))$$

This correctness implies that $E.\vec{\text{Dec}} \circ \vec{\mathbf{c}}_{\mathbf{outs}} \overset{d}{=} \vec{\mathbf{v}}_{\mathbf{outs}} \circ E.\vec{\text{Dec}}$ (Lemma 20) and for deterministic gates it's equivalent to the more common $E.\vec{\text{Dec}} \circ \vec{\mathbf{c}}_{\mathbf{outs}} = \vec{\mathbf{v}}_{\mathbf{outs}} \circ E.\vec{\text{Dec}}$ (Proposition 6). The property 'a is a correct implementation of b' is transitive (Lemma 21), and being a correct implementation is preserved by the composition in parallel and in series (Lemma 22). See Subsection B.3.

3.3 Circuit compiler

Definition 10 (Circuit compiler)**.** 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 compiled circuit $\text{CC}(c)$ must be a correct implementation of c using E.

From this definition and from Lemma 21 quickly follows that given a circuit c with an encoding D, and given a compiler (E, CC), then $CC(c)$ 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 proven by Lemma 21.

Definition 11 (Circuit Complexity Matrix)**.** Like in [BCP+20], we can define the circuit complexity matrix of a compiler C (if it exists) as the matrix M_C such that for all circuits c, $\vec{\text{Gates}}(C(c)) = M_C \cdot \vec{\text{Gates}}(c)$.

This implies that if M_O, M_I are the circuit complexity matrices of the compilers O, I then $M_{O \circ I} = M_O M_I$.

Definition 12 (Gadgets)**.** Given an encoding E with shares from $\mathbb{S}_{\text{in}}$ to $\mathbb{S}_{\text{out}}$, we can then define the *gadgets* $\vec{G} : \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}} \to \mathcal{C}_{\mathbb{S}_{\text{out}},\mathcal{G}_{\text{out}}}$ as the sequence 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.

With the gadgets we can make a compiler. In particular, given the gadgets $\vec{G}$ with encoding E we can 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, see Lemma 23. The circuit complexity matrix of this compiler exists, and if $\vec{G}$ has ℓ elements we have

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

3.4 Circuit Security

To define the security of a circuit we first describe the adversaries from [DDF14, PGMP19]:

Definition 13 (δ-noisy adversary)**.** 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 $\mathbf{Noise}$ 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.

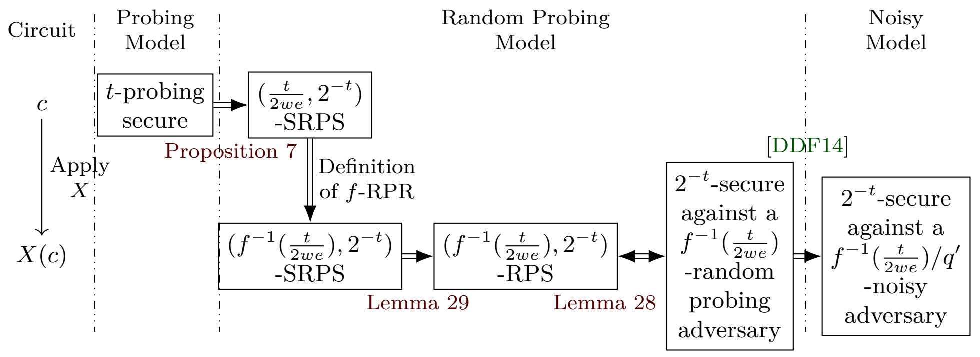

Circuit
Probing Model
Random Probing Model
Noisy Model
c
Apply X
$X(c)$
t-probing secure
$(\frac{t}{2we}, 2^{-t})$ -SRPS
Proposition 7
Definition of f-RPR
$(f^{-1}(\frac{t}{2we}), 2^{-t})$ -SRPS
Lemma 29
$(f^{-1}(\frac{t}{2we}), 2^{-t})$ -RPS
Lemma 28
2^{-t}-secure against a $f^{-1}(\frac{t}{2we})$ -random probing adversary
[DDF14]
2^{-t}-secure against a $f^{-1}(\frac{t}{2we})/q'$ -noisy adversary

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.

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 14 (p-random probing adversary)**.** 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 15 (t-probing adversary)**.** 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 [DDF14], and for every p-random probing adversary there is an equivalent $(p \cdot \frac{|\mathbb{S}|-1}{2})$-noisy adversary [PGMP19].

Definition 16 (Secure Against Adversary)**.** 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'll 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 $\vec{x} \in \mathbb{S}^i_{\text{orig}}$,

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

From the relationship between the adversaries above, if a circuit is ε-secure against δ-noisy adversaries, it's ε-secure against $\frac{2\delta}{|\mathbb{S}|-1}$-random probing adversaries, and if it's ε-secure against p-random probing adversaries, then it's ε-secure against $\frac{p}{|\mathbb{S}|}$-noisy adversaries[4].

We now report the definition of Random Probing Security from [BCP+20]. As it wasn't explicit in that paper, we prove in Lemma 28 that 'c is (p,ε)-RPS' is equivalent to 'c is ε-secure against a p-random probing adversary'.

[4] The $1/|\mathbb{S}|$ factor that originates here and is present throughout the paper is for the SD metric. For the ARE or RE metric it's absent, which means that the field-extension compiler provides no visible advantage. For the EN metric the opposite is true, see [PGMP19]

Definition 17 (RPS)**.** A circuit c is (p, ε)-RPS (Random Probing Secure) if there is a $\vec{\mathbf{Sim}}$ such that for all $\vec{x} \in \mathbb{S}^i_{\text{orig}}$,

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

In this paper we introduce a stronger notion that we'll call Strong Random Probing Security that we need to construct the Random Probing Reducibility. If we instead we construct it with the existing RPS, we could not prove our central Theorem 2.

Definition 18 (SRPS)**.** Given a circuit $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$ with encoding E, we'll 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 upper bounded by ε:

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

We can call p the leakage rate, while ε is essentially an upper bound on the circuit leakage, which is a measure of the guaranteed security, but the higher the ε the lower the security.

As we anticipated, if a circuit is (p, ε)-SRPS, then it's (p, ε)-RPS (see Lemma 29). Additionally, we can rise the ε and lower the p and the circuit remain SRPS (see Lemma 24), and every circuit with $w > 0$ internal wires is $(\varepsilon/w, \varepsilon)$-SRPS for every $\varepsilon \in [0, 1]$ (see Lemma 27).

We also want to note the similarity between our definition of the SRPS and the t-probing security (which can be easily proven equivalent to the 0-security against a t-probing adversary):

Definition 19 (t-Probing Security)**.** Given a circuit c with encoding E, we'll say that c is t-probing secure if $\leq t$ wires don't reveal any information on any inputs:

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

In particular we have the following reduction: if a circuit c with w wires is t-probing secure, then it's $(\frac{t}{2we}, 2^{-t})$-SRPS where e is the mathematical constant, see Proposition 7. 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.

While the RPE of [BCP+20] is a property of a single gadget and its main theorem works by crating bigger and bigger RPE gadgets, we'll provide a property relative to a whole compiler that describes a single step of the compilation and we call it RPR. In particular we defined it so that RPE implies the RPR, see Proposition 1, and we describe its intended interpretation in Figure 2.

Definition 20 (RPR)**.** We'll say that a compiler $(E, CC) : C_{\text{in}} \to C_{\text{out}}$ is e-RPR (Random Probing Reducible) with $e : [0, 1] \to [0, 1]$ a continuous monotone function, if for all circuits $c \in C_{\text{in}}$ and for all $p, \varepsilon \in [0, 1]$ we have that 'c is $(e(p), \varepsilon)$-SRPS' implies '$CC(c)$ is (p, ε)-SRPS'.

We note that the RPR is preserved if the parameter e in e-RPR is increased, see Lemma 25. We now report a simple lemma that proves the composition shown in Figure 2.

Lemma 2 (RPR of the Composition of Compilers)**.** *Given the circuit types $C_{\text{in}}, C_{\text{mid}}, C_{\text{out}}$, given the compiler $D : C_{\text{in}} \to C_{\text{mid}}$ that is e_D-RPR and given a compiler $C : C_{\text{mid}} \to C_{\text{out}}$ that is e_C-RPR, then $C \circ D$ is $(e_D \circ e_C)$-RPR.*

$$\begin{tikzcd}
\text{output circuit} & & & \text{input circuit} \arrow[lll, "\text{compilation}"'] \\
X_3(X_2(X_1(c))) & X_2(X_1(c)) \arrow[l, "X_3"'] & X_1(c) \arrow[l, "X_2"'] & c \arrow[l, "X_1"'] \\
\text{concrete circuit's leakage} \arrow[rrr, "\text{abstraction}"] & & & \text{virtual circuit's leakage} \\
p \arrow[r, "e_3"] & e_3(p) \arrow[r, "e_2"] & e_2(e_3(p) \arrow[r, "e_1"] & e_1(e_2(e_3(p)))
\end{tikzcd}$$

Figure 2: If each compiler X_i is e_i-RPR, and if we can compile a circuit c into a circuit $c_{\text{out}} := X_3(X_2(X_1(c)))$, then the circuit in the hardware c_{out} with its wires' values and the random probing leakage p can be abstracted into a virtual circuit c whose wires's values are the unencoding of those of c_{out}, and whose random probing leakage is $p_{\text{virt}} := e_1(e_2(e_3(p)))$. The RPR guarantees that if ε is the SRPS security of c with its leakage p_{virt}, then it's also the SRPS security of c_{out} with leakage p. This means that if $e_i(p') < p'$ then adding more compilations decreases p_{virt}, and so it decreases ε, which means higher security.

Proof. We need to prove that given any circuits $c \in C_{\text{in}}$ and any $p, \varepsilon \in [0,1]$ such that c is $(e_D(e_C(p)), \varepsilon)$-SRPS, then $C(D(c))$ is (p, ε)-SRPS.

We can use the definition of 'D is e_D-RPR' with the circuit c, the leakage probability $e_C(p)$ and the circuit leakage ε. Then as c is $(e_D(e_C(p)), \varepsilon)$-SRPS we obtain that $D(c)$ is $(e_C(p), \varepsilon)$-SRPS.

We can use the definition of 'C is e_C-RPR' with the circuit $D(c)$, the leakage probability p and the circuit leakage ε. Then as $D(c)$ is $(e_C(p), \varepsilon)$-SRPS, we obtain that $C(D(c))$ is (p, ε)-SRPS. $\square$

Corollary 1. *By induction with Lemma 2 we obtain that a given a sequence of compilers C such that C_i is e_i-RPR, then $C_n \circ \ldots \circ C_1$ is $(e_1 \circ \ldots \circ e_n)$-RPR*[5]*.*

3.5 Compiler Sequences

In this section we won't limit ourselves to a single expansion strategy, but we'll provide a theorem that describes the asymptotic size of a circuit compiled with C_i as $i \to \infty$, where C is a sequence of compilers, for example the classical expansion of [BCP+20] has $C_i := X^i$ where X the compiler to use for the steps of the expansion.

Definition 21 (Compiler Sequence)**.** We define a compiler sequence as an infinite sequence C of circuit compilers C_j all with the same input and output circuit types and leakages.

This section's theorem can be seen as an extension of the results of [BCP+20], and it uses three properties that are an extension of properties found in [BCP+20] see Subsection 4.5 and Figure 4 for more details on their relationship.

Definition 22 (Tolerated Leakage)**.** We say that P is a tolerated leakage for a compiler sequence C if there is a sequence e such that C_j is e_j-RPR and such that $e_m(P) \to 0$ as $m \to \infty$.

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 e.

[5]As the RPE implies the RPR, this contrasts with [BRTV21]'s Lemma 9, which we believe has a mistake. In particular if we call G^k the gadget of the k-th compilation. Then they define the gadget $G^{(k)} := CC_{k-1} \circ \ldots \circ CC_1(G^k)$ i.e. $G^{(k)} = CC_{G^{(k-1)}}(G^k)$. This 'i.e.' doesn't hold, which undermines their proof. The core reason is that given two groups of gadgets b, b', $CC_b \circ CC_{b'} = CC_{CC_b \circ b'}$, which can be easily proven with the associativity of the operation of substituting a leaf of a tree with a sub-tree. In practice, the right side of the 'i.e.' for $k = 3$ is $G^{(3)} = CC_{CC_{G^1}(G^2)}(G^3)$ which, as explained, means that $G^{(3)} = (CC_{G^1} \circ CC_{G^2})(G^3) = (CC_1 \circ CC_2)(G^3)$. This is not the definition $G^{(3)} = (CC_2 \circ CC_1)(G^3)$.

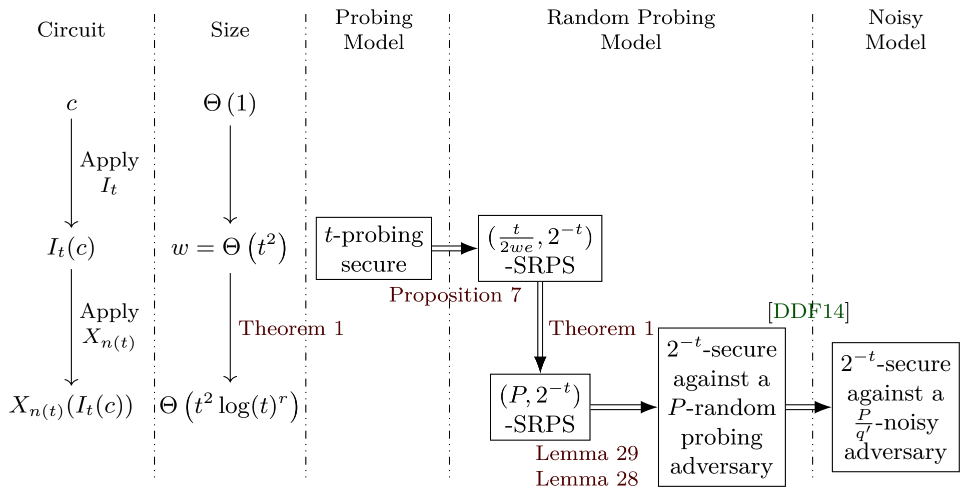


Figure 3: The properties of a circuit $c : \mathcal{C}_{\mathrm{std},q}$ compiled with a compiler sequence $I : \mathcal{C}_{\mathrm{std},q} \to \mathcal{C}_{\mathrm{std},q}$ such that I_t produces circuits that are t-probing secure with complexity $\Theta(t^2)$, and then compiled with a compiler sequence $X : \mathcal{C}_{\mathrm{std},q} \to \mathcal{C}_{\mathrm{std},q'}$ with expansion exponent r with relative tolerated leakage P. The existence of the function n with those characteristics is guaranteed by Theorem 1.

Definition 23 (Size Amplification Order)**.** We say that λ is a size amplification order of a compiler sequence C if the increase in circuit size (measured with the 1-norm of the circuit complexity matrix) is $\|M_{C_m}\|_1 = \mathcal{O}(\lambda^m)$ as $m \to \infty$.

Definition 24 (Security Amplification Order)**.** We say that $d > 1$ is a security amplification order for a compiler sequence C and a tolerated leakage P if $\log_2 e_m(P) = \Omega(d^m)$ as $m \to \infty$ where e is the e from the definition of 'tolerated leakage 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 25 (Expansion Exponent)**.** We say that $e := \frac{\log \lambda}{\log d}$ is 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.

To better understand which characteristics are represented by those definitions, we have the following immediate lemma.

Lemma 3 (Eventual Equivalence)**.** *Given a compiler sequence C with tolerated leakage P, size amplification order λ, security amplification order d relative to P, and expansion exponent erelative to P, then any compiler sequence C' such that $\exists m, m'.\ \forall i \geq 0.\ C_{m+i} = C_{m'+i}$, then C' has tolerated leakage P, size amplification order λ, security amplification order d relative to P, and expansion exponent e relative to 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. This is shown in Figure 3

Theorem 1 (Complexity of Compiler Sequences)**.** *Given a compiler sequence $C : C_{\mathrm{in}} \to C_{\mathrm{out}}$, given a circuit $c_\kappa \in C_{\mathrm{in}}$ parametric in its security level, i.e. such that c_κ is $(2^{-p(\kappa)}, 2^{-\kappa})$-SRPS for some $p : (0, \infty) \to (0, \infty)$; then there is a function $n : (0, \infty) \to \mathbb{N}$ that calculates which compiler of C to use, such that the compiled circuit $c'_\kappa := C_{n(\kappa)}(c_\kappa)$ satisfies the following properties:*

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

This where P is a tolerated leakage of C, and e is a expansion exponent of C and P.

The proof of this theorem is in Subsection C.3, and it has a few immediate consequences. The first follows immediately from Theorem 1 and the security reductions from Subsection 3.4:

Corollary 2. *Theorem 1 is true also with point 1 substituted with:*

1 For all $\kappa > 0$, the circuit c'_κ is $2^{-\kappa}$-secure against a $P/|\mathbb{K}_{\text{out}}|$-noisy adversary.

We can then use a fixed input circuit and use Lemma 27 to obtain the security guarantee for it:

Corollary 3. *If we apply Corollary 2 to a fixed input circuit c, then the compiled circuit size is*

$$\|c'_\kappa\| = \mathcal{O}\left(\kappa^e\right)$$

Note that the expression $\|c'_\kappa\| = \mathcal{O}\left(\kappa^e\right)$ in Corollary 3 is supposed to mirror [BCP+20]'s expression 12: $\|c'_\kappa\| = \mathcal{O}\left(\|c\| \, \kappa^e\right)$. Those are equal because in our case $\|c\|$ is constant by hypothesis, while in theirs we can only assume[6] they use a fixed c.

Lastly, instead of using Corollary 2 with a fixed circuit, we can apply it to the input circuit c_κ obtained as the output of a κ-probing secure compiler. All we need to use is Proposition 7 to obtain an SRPS guarantee from the κ-probing security property.

Corollary 4. *If we apply Corollary 2 to an input circuit c_κ obtained from a κ-probing secure compiler with polynomial complexity, then the compiled circuit size is*

$$\|c'_\kappa\| = \mathcal{O}\left(\|c_\kappa\| \log(\kappa)^e\right)$$

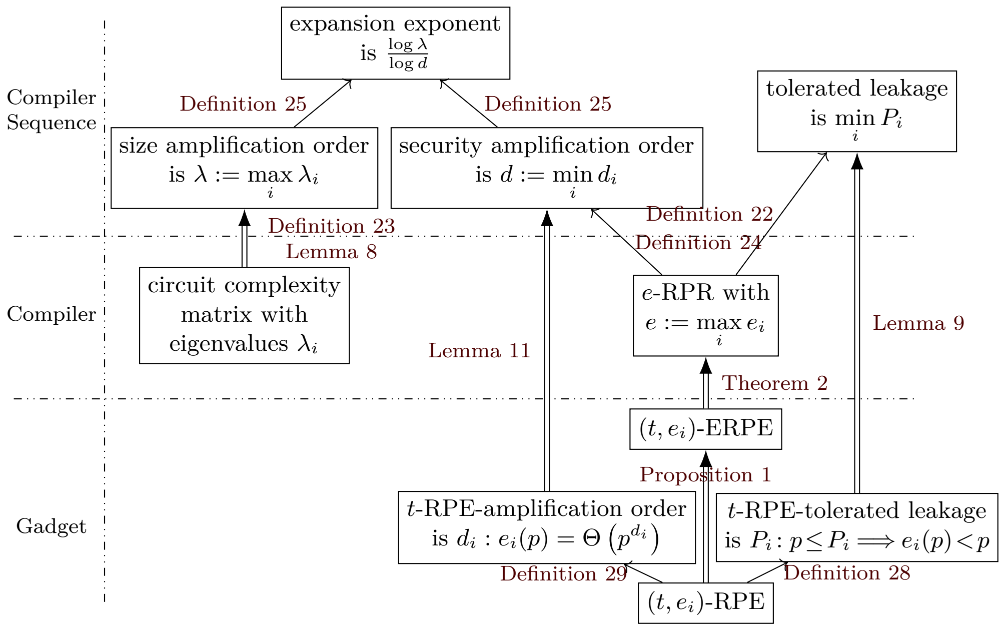
In other words, with a polylograrithmic size increase a t-probing secure compiler can be made to create circuits 2^{-t}-secure against a δ-noisy adversary, where the constant δ and the exponent of the logarithm depend on the compiler sequence.

4 How to Obtain Those Properties.

In order to provide the results mentioned in the abstract, we need show how to calculate the RPR from the RPE property to be able to reuse the existing results and tools from [BCP+20, BRT21]. In contrast to those papers we aim to provide a full proof both for 2-to-1 and 1-to-2 gates, but also for probabilistic gates with no inputs like the random gate, and we aim to do this for any compiler encoding. To this end, we'll introduce the Extended RPE property (or ERPE) and we'll show three things: the RPE of [BCP+20] implies the ERPE, if all the gadgets of a compiler are (t, e)-ERPE, the compiler is e-RPR (Theorem 2), and that for the optimal t described in [BRT21], the ERPE allows us to nearly halven the the number of randoms 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 of the [BCP+20] which we also show in Figure 4. To do this, we first provide a few useful results on how those three properties change when we compose compiler sequences and compilers, which also allows us to go beyond the classic expansion.

[6]They calculate their 'security expansion' using the RPE, and by their Corollary 1, a set of (t, f)-RPE gadgets leads to a $(p, 2f^k(p))$-RPS compiler. By definition that means that given a circuit c, the compiled circuit is $(p, 2\|c\| \, f^k(p))$-RPS. This means that to reach a security level of κ they need to satisfy $2\|c\| \, f^k(p) \leq 2^{-\kappa}$ and not $f^k(p) \leq 2^{-\kappa}$. This undermines their expression 12 if $\|c\| \to \infty$.



Compiler Sequence
Compiler
Gadget
expansion exponent is $\frac{\log\lambda}{\log d}$
Definition 25
Definition 25
tolerated leakage is $\min_i P_i$
size amplification order is $\lambda := \max_i \lambda_i$
security amplification order is $d := \min_i d_i$
Definition 23
Lemma 8
Definition 22
Definition 24
circuit complexity matrix with eigenvalues λ_i
e-RPR with $e := \max_i e_i$
Lemma 11
Lemma 9
Theorem 2
(t, e_i)-ERPE
Proposition 1
t-RPE-amplification order is $d_i : e_i(p) = \Theta\left(p^{d_i}\right)$
t-RPE-tolerated leakage is $P_i : p \le P_i \Longrightarrow e_i(p) < p$
Definition 29
Definition 28
(t, e_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.

4.1 RPE

Before reporting the definition of RPE from [BCP+20], we'll first give a helper definition of 'RPE (t, k)-normalization' that captures the relationship between J and J' as used in [BCP+20]'s RPE. In particular, the RPE uses the RPE $(t, n-1)$-normalization for n shares, as the additive encoding has strength $n-1$. As noted in Subsection 3.2, the RPE can be extended to any encoding with strength k, and this by using the RPE (t, k)-normalization instead of the RPE $(t, n-1)$-normalization.

Definition 26 (RPE (t, k)-normalization)**.** Calling n the number of shares, given $\breve{J} \subseteq [n \cdot o]$, we'll say that a $\breve{J}' \subseteq [n \cdot o]$ is an RPE (t, k)-normalization of $\breve{J}$ (with $t \le k < n$) if all the blocks of shares with $\le t$ shares are the same between $\breve{J}$ and $\breve{J}'$, while each block with with $> t$ shares is substituted with some set of k shares. Formally (with $\breve{s}(i)$ the set of all the shares of the virtual wire i) we need that for all virtual wires $i \in [o]$:

$$\breve{J}' \cap \breve{s}(i) = \begin{cases} \breve{J} \cap \breve{s}(i) & \text{if } \left|\breve{J} \cap \breve{s}(i)\right| \le t \\ \breve{z} \text{ for some } \breve{z} : |\breve{z}| = k & \text{otherwise} \end{cases}$$

We'll now provide the definition of RPE from [BCP+20], but extended to any i-to-o gate. We have split their Sim_1^G (which returns two results) in the two functions O' and I, while their Sim_2^G is implicit in the notion of simulatability. We also extend it to support any encoding with strength. Lastly, we add three requirements on the e of the e-RPE that are only present for the formal proof: continuity, monotonicity, and $e : [0,1] \to [0,1]$.

Definition 27 (RPE)**.** 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 encoding E with strength k, and given a monotone and continuous function $e : [0,1] \to [0,1]$, we say that G is (t, e)-RPE (Random Probing Expandability) with $t \in \mathbb{N} \cap [0, k]$ if there are the deterministic functions $\breve{O}', \breve{I}$ such that for all subsets of output wires $\breve{O} \subseteq [n \cdot o]$, 0 We define the input failure events $\breve{F}(\breve{O},\breve{W}) \subseteq [i]$, and the failure of an input happens if the simulation needs more than t shares. More formally, for $j \in [i]$

$$j \in \breve{F}(\breve{O},\breve{W}) \iff \left|\breve{I}(\breve{O},\breve{W}) \cap \breve{s}(j)\right| > t$$

where $\breve{I}(\cdot,\cdot) \subseteq [n \cdot i]$ are the input dependencies, and $\breve{s}(j)$ are the shares of j.

1 For every subset of the internal wires $\breve{W} \subseteq [w]$, the function $\vec{x} \mapsto \vec{\mathbf{G}}_{\mathbf{all}}(\vec{x})_{\breve{W} \,\|\, \breve{O}'(\breve{O},\breve{W})}$ can be simulated using the inputs $\breve{I}(\breve{O},\breve{W})$.

2 For all subsets of internal wires $\breve{W} \subseteq [w]$, the outputs $\breve{O}'(\breve{O},\breve{W})$ that the simulation is actually providing are an RPE (t,k)-normalization of $\breve{O}$.

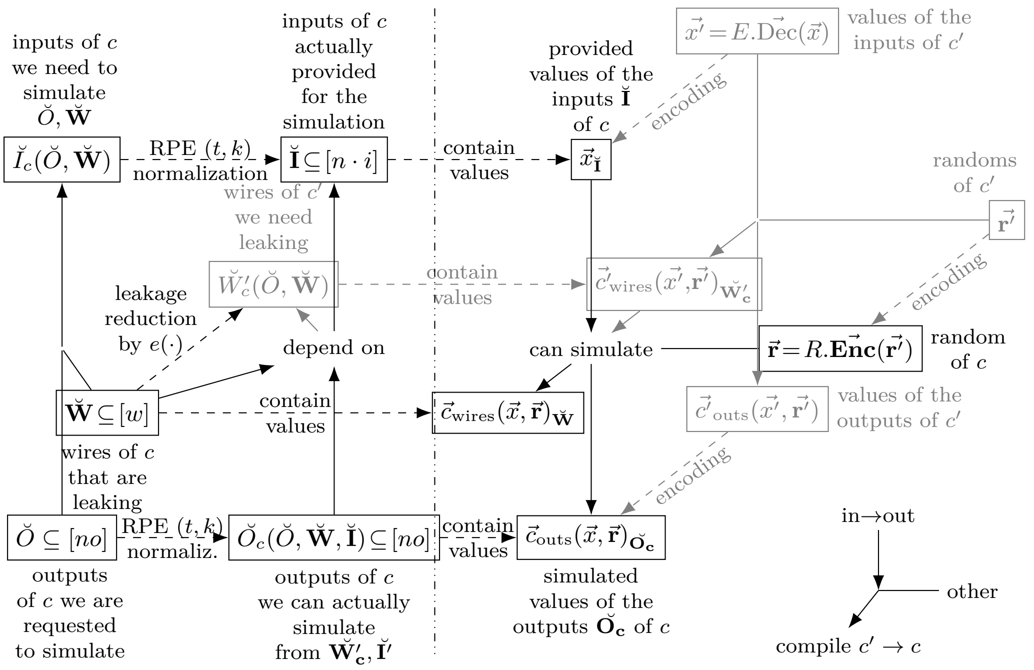
3 For every leakage probability vector $p \in [0,1]$, the failure events must have the probability distribution $\breve{F}(\breve{O}, \breve{\mathbf{Leak}}_G(p)) \overset{d}{=} \breve{\mathbf{Leak}}_g(e(p))$

We define the following two properties roughly equivalent to those from [BCP+20] and which can also be calculated with a tool like VRAPS:

Definition 28 (RPE-Tolerated Leakage)**.** We say that P is a t-RPE-tolerated leakage of some gadget G if there is an e such that G is (t,e)-RPE, and for all $p \in (0,P]$ we have that $e(p) < p$.

Definition 29 (RPE-amplification order)**.** We say that d is a t-RPE-amplification order of some gadget G if there is an e such that G is (t,e)-RPE, and $e(p) = \Theta\left(p^d\right)$.

4.2 ERPE



inputs of c we need to simulate $\breve{O}, \breve{\mathbf{W}}$
$\breve{I}_c(\breve{O},\breve{\mathbf{W}})$
RPE (t,k) normalization
inputs of c actually provided for the simulation
$\breve{\mathbf{I}} \subseteq [n \cdot i]$
contain values
provided values of the inputs $\breve{\mathbf{I}}$ of c
$\vec{x}_{\breve{\mathbf{I}}}$
$\vec{x'} = E.\vec{\mathrm{Dec}}(\vec{x})$
values of the inputs of c'
encoding
randoms of c'
$\vec{\mathbf{r}'}$
wires of c' we need leaking
$\breve{W}'_c(\breve{O},\breve{\mathbf{W}})$
contain values
$\vec{c'}_{\mathrm{wires}}(\vec{x'},\vec{\mathbf{r}'})_{\breve{\mathbf{W}}'_{\mathbf{c}}}$
encoding
leakage reduction by $e(\cdot)$
depend on
can simulate
$\vec{\mathbf{r}} = R.\vec{\mathbf{Enc}}(\vec{\mathbf{r}'})$
random of c
$\breve{\mathbf{W}} \subseteq [w]$
contain values
$\vec{c}_{\mathrm{wires}}(\vec{x},\vec{\mathbf{r}})_{\breve{\mathbf{W}}}$
$\vec{c'}_{\mathrm{outs}}(\vec{x'},\vec{\mathbf{r}'})$
values of the outputs of c'
wires of c that are leaking
encoding
in→out
$\breve{O} \subseteq [no]$
RPE (t,k) normaliz.
$\breve{O}_c(\breve{O},\breve{\mathbf{W}},\breve{\mathbf{I}}) \subseteq [no]$
contain values
$\vec{c}_{\mathrm{outs}}(\vec{x},\vec{\mathbf{r}})_{\breve{\mathbf{O}}_{\mathbf{c}}}$
other
outputs of c we are requested to simulate
outputs of c we can actually simulate from $\breve{\mathbf{W}}'_{\mathbf{c}}, \breve{\mathbf{I}}'$
simulated values of the outputs $\breve{\mathbf{O}}_{\mathbf{c}}$ of c
compile $c' \to c$

Figure 5: The structure of the ERPE property. On the left side the wires, and on the right side their values. In gray the entities of the virtual circuit c'.

As we anticipated in the introduction, instead of the RPE we use the more generic ERPE, which allows us to handle generic probabilistic gates. Also note that the ERPE is defined for generic circuits and not only gates, and it's defined so that it's preserved by composition.

Definition 30 (ERPE)**.** 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 $c' \in \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}}$ (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 $e : [0,1] \to [0,1]$, and a $t \in [0,k] \cap \mathbb{N}$, we say that c is (t,e)-ERPE (Extended Random Probing Expandability) of c' if there is a function for the input dependencies $\breve{I}_c$, a function for the outputs actually simulated $\breve{O}_c$, and a function for the dependencies on the virtual wires $\breve{W}'_c$ such that

1. For all leakage rates $p \in [0,1]$, for all subsets of output $\breve{O} \in [n \cdot o]$, the distribution of virtual wires is the same for all $\breve{O}$ (i.e. $\breve{W}'_c(\breve{O}, \breve{\mathbf{Leak}}_c(p)) \overset{d}{=} \breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p)))$ and it's upper bounded by the leakage $e(p)$ (i.e. $\breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'}(e(p)))$.
2. For every subset of the internal wires $\breve{W} \subseteq [w]$, for all possible combinations of outputs to simulate $\breve{O} \in [n \cdot o]$, for all the possible inputs actually provided $\breve{I}$ that are an RPE (t,k)-normalization of $\breve{I}_c(\breve{O}, \breve{W})$, the outputs actually simulated must be $\breve{O}_c(\breve{O}, \breve{W}, \breve{I})$ an RPE (t,k)-normalization of $\breve{O}$, and they together with $\breve{W}$ can 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 $\vec{r'} \in \breve{\text{supp}}[\vec{\mathbf{Rnds}}_{c'}()]$

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

In Figure 5 we represent a compact scheme of the relationships within the ERPE, in particular, the composition in series is equivalent to stacking two copies of this scheme on top of each other, while in case of multiple compilations we stack them in depth, roughly by substituting the exact calculation of $\vec{c'}_{\text{all}}$ with the other simulator.

We will now enunciate the main theorem for the expansion, its proof is in Appendix D

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

We will show in Lemma 26 that the (t,e)-ERPE is preserved by increasing e.

A special consideration can be given to leakless gadgets for leakless gates, as the (t,e)-ERPE property only uses the input-output behavior of the gadget and it's independent from e. From the definition of ERPE we have the following immediate necessary and sufficient condition:

Lemma 4 (Leakless ERPE)**.** *Given a leakless circuit $c \in \mathcal{C}_{\mathbb{S}_{\text{out}},\mathcal{G}_{\text{out}}}$ a correct implementation of a leakless circuit $c' \in \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}}$ using the n-shares encoding E with strength k and the encoding of the randoms R, where c' has i inputs, o outputs, given a continuous and monotone function $e : [0,1] \to [0,1]$, and a $t \in [0,k] \cap \mathbb{N}$, then c is (t,e)-ERPE of c' iff for all combinations of the required output wires $\breve{O} \in [n \cdot o]$, there is a combination of possible inputs $\breve{I} \in [n \cdot i]$ such that for all the inputs actually provided $\breve{I}'$ an RPE (t,k)-normalization of $\breve{I}$ there is a set of outputs actually provided $\breve{O}'$ an RPE (t,k)-normalization of $\breve{O}$ such for all inputs $\vec{x} \in E.\vec{\mathbf{Enc}}[\mathbb{S}^i_{\text{in}}]$ and for all virtual randoms $\vec{r'} \in \breve{\text{supp}}[\vec{\mathbf{Rnds}}_{c'}()]$ we must have that $\vec{x}_{\breve{I}'} \xrightarrow{\text{sim}} \vec{c}_{\text{outs}}(\vec{x}, R.\vec{\mathbf{Enc}}(\vec{r'}))_{\breve{O}'}$.*

This lemma has the following immediate implication:

Corollary 5 (Constant Gate)**.** *Any n-shares leakless implementation for an encoding of strength k of a leakless constant gate is (t,e)-ERPE for any $t \in [0,k] \cap \mathbb{N}$, and for any continuous and monotone function $e : [0,1] \to [0,1]$.*

4.3 From RPE to RPR

To use the results from [BCP+20]'s framework we need to calculate the RPR from the RPE. To do this we prove that the random gadget from [BCP+20] is ERPE, and that (t, e)-RPE implies the (t, e)-ERPE. For this last implication we first define a weaker version of the RPE property that we need in a proof later:

Definition 31 (wRPE)**.** We define the (t, e)-wRPE (Weak RPE) as property that is identical to the RPE except that the item 3 has the $\overset{d}{\leq}$ instead of the $\overset{d}{=}$, and that the probability distribution of the failure events $\breve{F}(\breve{O}, \mathbf{Leak}_G(p))$ is the same for all $\breve{O}$.

Then we have the following implications that we'll prove in Subsection E.1.

Proposition 1. *The (t, e)-RPE implies the (t, e)-wRPE which implies the (t, e)-ERPE.*

Instead of showing that the random gadget with n randoms from [BCP+20, BRT21] is $(t, \cdot)$-ERPE, we will prove in Subsection E.1 the more general proposition:

Proposition 2 (Additive Random Gadgets)**.** *For the n-shares additive encoding, given a $t \in [0, n-1] \cap \mathbb{N}$, and any continuous monotone $e : [0, 1] \to [0, 1]$, we consider the gadget obtained by a parallel of at least $t+1$ random gates, while the remaining shares are obtained from constant-0 gates. This gadget is (t, e)-ERPE for the random gate.*

With this we can define a slight extension to the compiler sequences from [BCP+20]:

Definition 32 (classic compiler)**.** 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 one described in Proposition 2, and such that its constant-c gadgets are leakless.

With this we can obtain the following result that links our paper with [BCP+20], which can be derived from the last two propositions, Lemma 26, and Theorem 2:

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

4.4 Composition of Compiler Sequences

We'll first analyze the size amplification order by adding a compiler before and after the main compiler sequence. Note that any composition of compilers is itself a compiler. The proofs of this section are in Subsection E.2.

Lemma 5 (Size Amplification Order)**.** *Given a compiler sequence C and given two compilers I, O such that we can define $C'_m := O \circ C_m \circ I$ then C' has all the size amplification order of C.*

Then we have a lemma for how adding a pre-compiler affects the security amplification order of a given compiler sequence.

Lemma 6 (Single Input Compiler)**.** *Given a compiler sequence C and given a compiler I such that we can define $C'_n := C_n \circ I$ and such that I is $x^{\Omega(1)}$-RPR then C' has all the tolerated leakage, security amplification order, size amplification order and expansion exponent of C.*

Lastly, we give a lemma for the security amplification order, in case we add a post-compiler, which was taken from a compiler sequence so that we can obtain the following useful property.

Lemma 7 (Fixed Output Compiler Sequence)**.** *Given a compiler sequence I and a compiler sequence O (with respectively a tolerated leakage of P_i, P_o; a security amplification order of d_i, d_o for the aforementioned tolerated leakage) then there is a k such that*[7] $C_n := O_k \circ I_n$ *has a security amplification order of d_i relative to a tolerated leakage of P_o.*

4.5 Classic Expansion

In this section we'll analyze the classical expansion strategy from [BCP+20, BRT21], where the compiler sequence is $C_n := X^n$, and we calculate the three properties of compiler sequences from the properties of their gadgets. The proofs will be in Subsection E.3.

The following lemma shows that [BCP+20]'s $N_{\max}$ (which they define as the highest module of any eigenvalue of the circuit complexity matrix M_X) is a size amplification order.

Lemma 8 (Size Amplification Order)**.** *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.*

We can then analyze the tolerated leakage, which is the minimum RPE-tolerated leakage of all gadgets. This akin to [BRT21].

Lemma 9 (Tolerated Leakage)**.** *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.*

We can then show that for these families of compilers the security amplification order is independent of the tolerated leakage, like it's the case in [BRT21].

Lemma 10 (Universality of security amplification order)**.** *Given a compiler sequence $C_n := X^n$ if d is a security amplification order for some tolerated leakage, then it's a security amplification order for any tolerated leakage.*

Lastly, we can show that the overall security amplification order is the minimal RPE-amplification order of all the compilers. This aking to [BRT21].

Lemma 11 (Security Amplification Order)**.** *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 tolerated leakage.*

5 Main Compiler Sequence

In this section we'll provide Compiler Sequence 2 which compiles 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 two more compilers.

5.1 Field-Extension compiler

First we'll give a pair of useful lemmas for gadgets whose encoding has strength 0. We report an immediate consequence of Lemma 4 in case of a leakless gate implemented by a leakless gadget:

[7]In [BRTV21] they use $C_n := O_n \circ I_k$ instead of $C_n := O_k \circ I_n$, due to what we believe to be an error in their Lemma 9, see footnote 5.

Proposition 3. *Any correct leakless implementation of a leakless gate for a strength* 0 *encoding is* $(0, e)$*-ERPE for any continuous and monotone function* $e : [0, 1] \to [0, 1]$.

Then we can analyze a fully-leakable deterministic gate, proven in Subsection E.4.

Lemma 12. *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* $(0, e)$*-ERPE, with* $e(x) := \min\{1, (w \cdot x)^{1/i}\}$.

The Field-Extension compiler we'll present here is similar to the method presented in [GJR18] to reduce the order of the field, the main difference is that they obtain $p' = \Theta\left(\frac{p}{k \log k}\right)$. The lack of the square for the p is due to the different leakage model, as they consider one where the inputs are bundled together: they either are all leaking or all not leaking.

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 $\mathbb{F}_q$ 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.

We can implement the addition, subtraction, copy, random gadgets using k gates of the same types (one per coefficient of the polynomials). The constant-c gadget also uses k constant gates, but the contstants are obtaineded using the field-extension encoding of c.

The last gadget is the multiplication, which is made of a multiplication of the two inputs polynomials followed by the rest of the division by P, and it's widely known that this can be implemented using $\Theta(k \log k)$ gates by using the fast Fourier transform.

Proof. The correctness of these gadgets follows from the definition of the field-extension encoding and the properties of extension fields. The random and constant-c gadgets are leakless as they're made of leakless gates. □

Proposition 4. *The field-extension compiler* $\mathcal{C}_{\text{std},q^k} \to \mathcal{C}_{\text{std},q}$ *is* e*-RPR with* $e(p) := \min\{1, (w \cdot p)^{1/2}\}$ *where* $w = \Theta(k \log k)$ *is the number of wires in the multiplication gadget, and the 1-norm of its circuit complexity matrix is* $\Theta(k \log k)$.

Proof. The field-extension compiler's encoding has strength 0. This means that every gadget of this compiler is $(0, e)$-ERPE thanks to Lemma 12 and Lemma 26 for the fully leakable gates, and Proposition 3 for the leakless ones. Then we know that this compiler is e-RPR from Theorem 2 as all gadgets are $(0, e)$-ERPE. The size complexity is $\|M\|_1 := \Theta(k \log k)$ as the complexity matrix has finite elements and the highest element is proportional to the number of wires w, which is $\Theta(k \log k)$. □

5.2 High Tolerated Leakage Compiler

Compiler 2. The high-tolerance compiler is a compiler $\mathcal{C}_{\text{std},q} \to \mathcal{C}_{\text{std},q}$ with 3 shares, and additive encoding.

The random gadget is $[] \mapsto [\leftarrow \mathbb{F}_q, \leftarrow \mathbb{F}_q, 0]$, the constant-$c$ gadget is $[] \mapsto [c, 0, 0]$. It uses the addition, subtraction and multiplication gadgets from [BRT21], while the copy gadget has an additional refresh circuit on the inputs.

We'll first report the refresh circuit of [BRT21]'s compiler this is based on. We'll call this $\vec{\mathbf{ref}}(\vec{a})$. This calculates $\vec{\mathbf{c}}$ with randoms $\vec{\mathbf{r}} :=\leftarrow \mathbb{F}_q^2$:

$$\vec{\mathbf{c}}_1 := \vec{a}_1 + \vec{\mathbf{r}}_1$$
$$\vec{\mathbf{c}}_2 := \vec{a}_2 + \vec{\mathbf{r}}_2$$

$$\vec{\mathbf{c}}_3 := \vec{a}_3 - (\vec{\mathbf{r}}_1 + \vec{\mathbf{r}}_2)$$

We report [BRT21]'s addition gadget, which is $\vec{a}, \vec{b} \mapsto \vec{\mathbf{ref}}(\vec{a}) + \vec{\mathbf{ref}}(\vec{b})$, while the subtraction gadget is $\vec{a}, \vec{b} \mapsto \vec{\mathbf{ref}}(\vec{a}) - \vec{\mathbf{ref}}(\vec{b})$. The multiplication gadget is $\vec{a}, \vec{b} \mapsto \vec{\mathbf{c}}$ with $\vec{\mathbf{c}}$ calculated with

$$\begin{aligned}
&\forall i \in [3].\ \vec{\mathbf{b'}}_{i,\cdot} := \vec{\mathbf{ref}}(\vec{b}) \\
&\vec{\mathbf{a'}} := \vec{\mathbf{ref}}(\vec{a}) \\
&\forall i \in [3].\ \vec{\mathbf{c}}_i := \left(\sum_k (\vec{\mathbf{a'}}_i \cdot \vec{\mathbf{b'}}_{i,k} + \vec{\mathbf{r}}_{i,k})\right) - \left(\sum_k \vec{\mathbf{r}}_{k,i}\right)
\end{aligned}$$

Which uses 4 refresh circuits and 9 additional randoms.

The difference is the copy gadget, which is $\vec{a} \mapsto \vec{\mathbf{ref}}(\vec{\mathbf{a'}}), \vec{\mathbf{ref}}(\vec{\mathbf{a'}})$ with $\vec{\mathbf{a'}} := \vec{\mathbf{ref}}(\vec{a})$. This means it has an additional refresh on the input.

Compiler Sequence 1. The high-tolerance compiler sequence is the sequence $C_m := X^m$ where X is the high-tolerance compiler Compiler 2. In other words it's the classical expansion of the high-tolerance compiler, expanded m times.

Proposition 5. *The Compiler Sequence 1 has expansion exponent* 4.09 *and tolerated leakage* $2^{-7.41}$.

Proof. In Table 2 we report the 1-RPE-tolerated leakage and the 1-RPE-amplification order of the various gadgets of Compiler 2 as calculated by VRAPS. Then Lemma 9 and Lemma 11 state that their minimal values are the tolerated leakage and the security amplification order of Compiler Sequence 1.

Table 2: *log*$_2$ of RPE-tolerated leakage, and the 1-RPE-amplification order for the RPE with $t := 1$, calculated using the VRAPS tool

Gadget	*log*$_2$ of 1-RPE-tolerated leakage	1-RPE-amplification order
Addition	-4.75	2
Subtraction	-4.75	2
Multiplication	$\mathbf{-7.41}$	2
Copy	-4.95	2

In particular, the tolerated leakage is $2^{-7.41}$, which can be compared to [BRT21]'s $2^{-7.50}$, while the security amplification order is 2 which is the same as [BRT21].

The circuit complexity matrix with the gates in the order (addition, subtraction, copy, multiplication, random, constant-0, constant-c with $c \neq 0$) is

$$M = \begin{bmatrix} 9 & 6 & 9 & 35 & 0 & 0 & 0 \\ 2 & 5 & 3 & 5 & 0 & 0 & 0 \\ 4 & 4 & 9 & 29 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 & 0 \\ 4 & 4 & 6 & 17 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Is diagonalizable with the eigenvalues 17,3,3,9,2,3,1, and so by Lemma 8 the size amplification order is 17.

By using both amplification orders we can calculate the expansion exponent, which is $e := \frac{\log 17}{\log 2} = 4.09$, which can be compared to [BRT21]'s 3.9.

The differences in the results of this compiler sequence and [BRT21]'s are all due to the copy gadget. □

5.3 Main Compiler Sequence

Compiler Sequence 2 (Main Compiler Sequence)**.** For every q that is a power of a prime, we define the main compiler sequence of this article as

$$X_t := \begin{cases} H_t \circ E & \text{if } t \leq m \\ H_m \circ C_{t-m} \circ E & \text{if } t \geq m \end{cases}$$

where H is the high-tolerance compiler sequence Compiler Sequence 1, E is the field-extension compiler of Compiler 1, and where C_t is the t-th iteration of the classical expansion of the parametric compiler from [BRT21] instantiated with 21 shares, except that we use our random gadget with 11 randoms instead of 21, and where $m \in \mathbb{N}$ is the minimal value such that X has expansion exponent 3 relative to a tolerated leakage $2^{-7.41}$.

The value of m represents how many expansions of the high-tolerance compiler of H are necessary to make the virtual leakage applied to C lower than its tolerated leakage. If instead it was higher, then 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. Any real implementation should not use a sequence H made of m identical compilers, but instead it should use an optimized sequence of different compilers to lower the compiled circuit size as much as possible. For example by using compilers with an intemediate number of shares between 3 and 21. We believe this optimization is beyond the scope of this work.

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

Proof. We first analyze C. We report that [BRT21]'s parametric compiler has size amplification order of $3n^2 - 2n$ for n shares, and $\lfloor (n-1)/2 \rfloor$-RPE-amplification order of $\lfloor (n+1)/2 \rfloor$. As we're using $n = 21$ this means it has a size amplification order of 1281 (which means 1331 is also a size amplification order) and a 10-RPE-amplification order of 11. From Lemma 11 we know that 11 is a security amplification order of C.

For the security amplification order of C and the tolerated leakage of H from Proposition 5, we can use Lemma 7 to show that there is a m such that $(H_m \circ C_t)_t$ has the security amplification order 11 for the tolerated leakage $2^{-7.41}$. We can then apply Lemma 6 to $X'_t := H_m \circ C_t \circ E$ and obtain that it has the security amplification order 11 for the tolerated leakage $2^{-7.41}$.

As m is a constant, we can use Lemma 5 to show that X' has the size amplification order of 1331. Then by defintion, we have that X' has a expansion exponent of 3 relative to a tolerated leakage of $2^{-7.41}$.

As we can remove the first elements from X' and X to obtain the same sequences, we can lastly use Lemma 3 to prove that X (with the chosen m) has a expansion exponent of 3 relative to a tolerated leakage of $2^{-7.41}$. $\square$

6 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 two useful results that could not be obtained with the existing formalization, e.g. of [BCP+20].

First, we have opened the way to analyze any non-deterministic gate, and we give the common example of the random gate. We use this result in our Compiler Sequence 2: every step of the sub-compiler C turns all the randoms gates into 11 random gates instead of 21, and C could have multiple steps that further amplify this result.

Second, we show how given a parametric circuit c_t which is t-probing secure, there is an $n(t)$ such that the compiler $X_{n(t)}$ (with X Compiler Sequence 2) will make $X_{n(t)}(c_t)$ be 2^{-t} secure for some constant noise and for a $\mathcal{O}\left(\log(t)^3\right)$ size increase. Note that this $n(t)$ depends both on the size of c, and on the exact compiler sequence X.

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 and thus improve the link between probing security and noisy leakage models.

A Definition of Circuit

The usual way to describe the circuit, for example in [BCP⁺20], is to define them as a graph where the nodes are the gates and the links the wires. Yet a normal graph doesn't track where each wire is connected, and so it's only meaningful for commutative gates with identical outputs. Additionally, we have the restriction that each input and output has a single wire linked to it which we'd need to codify into the graph. We need to ask that the graph is acyclic, we need two additional gate types for the circuit's overall inputs and outputs, and we need a way to assign to those additional nodes a label to distinguish which input or output of the overall circuit they represent.

We believe a more suitable structure for this specific kind of circuits, which also simplifies our proofs, is a tree that describes the circuits iteratively:

Definition 33 (Circuit)**.** We'll 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.

Where the identity and swap circuits are just for reshuffling the wires, to ensure that this formalism can represent all the circuits that can be defined as a graph of gates.

Given a circuit $c \in \mathcal{C}_{\mathbb{S},\mathcal{G}}$, we can now give the formal definitions of $\vec{c}_{\text{outs}}$, $\vec{c}_{\text{wires}}$, $\vec{\mathbf{Rnds}}_c()$, $\check{\mathbf{Leak}}_c$ and $\vec{\text{Gates}}(c)$, and we'll do so iteratively based on c:

- If c is the g-th gate, then $\vec{c}_{\text{outs}}$ is the function that defines the gate. If it's leakless then $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := []$ otherwise $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := \vec{x} \parallel \vec{r}$, $\vec{\mathbf{Rnds}}_c()$ draws a value from the probability distribution in the description of the gate. We can then define $\vec{\text{Gates}}(c)_g := 1$ and $\vec{\text{Gates}}(c)_{\neq g} := \vec{0}$. Lastly, given $\mathbf{v}$ a result of $\check{\mathbf{Leak}}_c(p)$ we have that for all j the probability of $j \in \mathbf{v}$ is p and they are independent.
- If c is an identity circuit, then $\vec{c}_{\text{outs}}([a]) = [a]$, $\vec{\mathbf{Rnds}}_c() := []$, $\vec{\text{Gates}}(c) := \vec{0}$. $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := []$, $\check{\mathbf{Leak}}_c(p) := \emptyset$.
- If c is a swap circuit, then $\vec{c}_{\text{outs}}([a,b]) = [b,a]$, $\vec{\mathbf{Rnds}}_c() := []$, $\vec{\text{Gates}}(c) := \vec{0}$. $\vec{c}_{\text{wires}}(\vec{x},\vec{r}) := []$, $\check{\mathbf{Leak}}_c(p) := \emptyset$.
- If c is a parallel $c' \parallel c''$, then $\vec{\mathbf{Rnds}}_c() := \vec{\mathbf{Rnds}}_{c'}() \parallel \vec{\mathbf{Rnds}}_{c''}()$, $\vec{\text{Gates}}(c) := \vec{\text{Gates}}(c') + \vec{\text{Gates}}(c'')$, $\check{\mathbf{Leak}}_c(p) := \check{\mathbf{Leak}}_{c'}(p) \parallel \check{\mathbf{Leak}}_{c''}(p)$, 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 $c'' \circ c'$, then $\vec{\mathbf{Rnds}}_c() := \vec{\mathbf{Rnds}}_{c'}() \parallel \vec{\mathbf{Rnds}}_{c''}()$, $\vec{\text{Gates}}(c) := \vec{\text{Gates}}(c') + \vec{\text{Gates}}(c'')$, $\breve{\mathbf{Leak}}_c(p) := \breve{\mathbf{Leak}}_{c'}(p) \parallel \breve{\mathbf{Leak}}_{c''}(p)$, and

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

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

B Lemmas for Fundamental Concepts

We'll prove here various lemmas for the properties of the simulatability, of the dependency functions, of the partial order of distributions, of our definition of correctness and the monotonicity of a few security definitions.

B.1 Simulatability and Dependency Functions

Lemma 13 (Simulatability from the Inputs)**.** *The probabilistic vectorial function $\vec{\mathbf{f}} : \mathbb{S}_{\text{in}}^i \to \mathbb{S}_{\text{out}}^o$ can be simulated using the input elements $\breve{I} \subseteq [i]$ iff for all $\vec{x}, \vec{y} \in \mathbb{S}_{\text{in}}^i$, $\vec{y}_{\breve{I}} = \vec{x}_{\breve{I}}$ implies $\vec{\mathbf{f}}(\vec{x}) \overset{d}{=} \vec{\mathbf{f}}(\vec{y})$.*

Proof. By definition, $\vec{\mathbf{f}}$ can be simulated using the input elements $\breve{I}$ iff $\vec{\mathbf{f}}$ can be simulated from $(\vec{x} \mapsto \vec{x}_{\breve{I}})$, i.e. iff there is a $\mathbf{Sim} : \mathbb{S}_{\text{in}}^{|\breve{I}|} \to \mathbb{S}_{\text{out}}^o$ such that $\vec{x} \in \mathbb{S}_{\text{in}}^i$, $\vec{\mathbf{f}}(x) \overset{d}{=} \mathbf{Sim}(\vec{x}_{\breve{I}})$.

This implies that for every $\vec{x}, \vec{y} \in \mathbb{S}_{\text{in}}^i$, $\vec{y}_{\breve{I}} = \vec{x}_{\breve{I}} \implies \vec{\mathbf{f}}(\vec{x}) \overset{d}{=} \mathbf{Sim}(\vec{x}_{\breve{I}}) \overset{d}{=} \mathbf{Sim}(\vec{y}_{\breve{I}}) \overset{d}{=} \vec{\mathbf{f}}(\vec{y})$.

For the opposite implication we have that $\vec{x}, \vec{y} \in \mathbb{S}_{\text{in}}^i$, $\vec{y}_{\breve{I}} = \vec{x}_{\breve{I}} \implies \vec{\mathbf{f}}(\vec{x}) \overset{d}{=} \vec{\mathbf{f}}(\vec{y})$. This implies that for every $\vec{c} \in \mathbb{S}_{\text{in}}^{|\breve{I}|}$ all $\vec{\mathbf{f}}(\vec{x})$ with $\vec{x}_{\breve{I}} = \vec{c}$ have all the same probability distribution. This means that there is a $\mathbf{Sim}$ such that $\vec{x} \in \mathbb{S}_{\text{in}}^i$, $\vec{\mathbf{f}}(x) \overset{d}{=} \mathbf{Sim}(\vec{x}_{\breve{I}})$. ☐

Lemma 14 (Multiple Simulatability)**.** *Given a probabilistic vectorial function $\vec{\mathbf{f}} : \mathbb{S}_{\text{in}}^i \to \mathbb{S}_{\text{out}}^o$, if it can be simulated from the inputs $\breve{I}$, and it can also be simulated from the inputs $\breve{I}'$, then it can be simulated from the inputs $\breve{I} \cap \breve{I}'$.*

Proof. This was proven in [BBP+16] as their Lemma 7.5. We could prove it using the equivalence relationship $\vec{x} \sim_{\breve{I}} \vec{y} \iff \vec{x}_{\breve{I}} = \vec{y}_{\breve{I}}$, and so from Lemma 13 we have that $\vec{\mathbf{f}}$ can simulated from the inputs I iff $\vec{x} \sim_{\breve{I}} \vec{y} \implies \vec{\mathbf{f}}(\vec{x}) \overset{d}{=} \vec{\mathbf{f}}(\vec{y})$. Then it's a matter of showing that $\vec{y} \sim_{\breve{I} \cap \breve{I}'} \vec{y'}$ means that there are $\vec{x}, \vec{x'}, \vec{m}$ such that $\vec{y} \sim_{\breve{I}} \vec{x} \sim_{\breve{I}'} \vec{m} \sim_{\breve{I}'} \vec{x'} \sim_{\breve{I}} \vec{y'}$. Note that this only works because the domain $\mathbb{S}_{\text{in}}^i$ allows every element to take values independent from the values of the other elements. ☐

Lemma 15 (Dependency Function)**.** *Given a probabilistic vectorial function $f : \mathbb{S}_{\text{in}}^i \to \mathbb{S}_{\text{out}}^o$, its dependency function always exist and is unique.*

Proof. The dependency function exists if for every subset of the outputs there is the minimal subset of inputs that allow to simulate those outputs. Thanks to Lemma 14 those subsets of the inputs create a finite meet-semilattice and so there is a single global minimum. ☐

Lemma 16 (Monotonicity)**.** *All the dependency functions are monotone.*

Proof. Given a $\vec{\mathbf{f}} : \mathbb{S}_{\text{in}}^i \to \mathbb{S}_{\text{out}}^o$, we define $\breve{Dep}(\breve{O}) := \bigcap_{\breve{O}' \supseteq \breve{O}} \breve{\text{Dep}}_{[\vec{\mathbf{f}}]}(\breve{O}')$ and by construction we have that $\breve{Dep}$ is monotone and $\breve{Dep}(\breve{O}) \subseteq \breve{\text{Dep}}_{[\vec{\mathbf{f}}]}(\breve{O})$. Also, $\vec{\mathbf{f}}_{\breve{O}}$ can be simulated using the inputs $\breve{Dep}(\breve{O})$. This is because it can be simulated using the inputs $\breve{\text{Dep}}_{[\vec{\mathbf{f}}]}(\breve{O}')$ for any $\breve{O}' \supseteq \breve{O}$ by simulating more outputs and discarding the ones not needed. Then we can apply Lemma 14 on every possible $\breve{\text{Dep}}_{[\vec{\mathbf{f}}]}(\breve{O}')$ to see that it can be simulated from the inputs $\breve{Dep}(\breve{O})$. This means that $\breve{Dep} \supseteq \breve{\text{Dep}}_{[\vec{\mathbf{f}}]}$ as the latter is minimal by definition. All this means $\breve{Dep} = \breve{\text{Dep}}_{[\vec{\mathbf{f}}]}$ and that they are monotone. ☐

B.2 Partial Order of Distributions

Lemma 17 (Partial Order)**.** *The relationship $\overset{d}{\leq}$ is a partial order for the equivalence $\overset{d}{=}$.*

Proof. The reflexivity is immediately proven, as $\mathbf{a} \overset{d}{\leq} \mathbf{a}$ iff for all monotone P, $\Pr[P(\mathbf{a})] \leq \Pr[P(\mathbf{a})]$ which is obviously true.

The transitivity is just as obvious, as given a $\mathbf{a}, \mathbf{b}, \mathbf{c}, P$ we have that $\Pr[P(\mathbf{a})] \leq \Pr[P(\mathbf{b})]$ and $\Pr[P(\mathbf{b})] \leq \Pr[P(\mathbf{c})]$ imply $\Pr[P(\mathbf{a})] \leq \Pr[P(\mathbf{c})]$.

For the antisymmetry, we have that for all monotone P $\Pr[P(\mathbf{a})] \leq \Pr[P(\mathbf{b})]$ and for all monotone P, $\Pr[P(\mathbf{b})] \leq \Pr[P(\mathbf{a})]$, mean that for all monotone P, $\Pr[P(\mathbf{a})] = \Pr[P(\mathbf{b})]$. As the random variables $\mathbf{a}, \mathbf{b}$ are discrete, there is a sequence of P such that P_1 is always false, and P_{i+1} was true for all the elements of P_i plus one. All this implies that for all x, $\Pr[\mathbf{a} = x] = \Pr[\mathbf{b} = x]$ and so $\mathbf{a} \overset{d}{=} \mathbf{b}$. ☐

Lemma 18 (Parallel Composition)**.** *If $\vec{\mathbf{v}} \overset{d}{\leq} \vec{\mathbf{V}}$ and $\vec{\mathbf{u}} \overset{d}{\leq} \vec{\mathbf{U}}$, with $\vec{\mathbf{v}}, \vec{\mathbf{u}}, \vec{\mathbf{V}}, \vec{\mathbf{U}}$ pair-wise independent, then $\vec{\mathbf{v}} \parallel \vec{\mathbf{u}} \overset{d}{\leq} \vec{\mathbf{V}} \parallel \vec{\mathbf{U}}$*

Proof. To prove this, we'll prove it in two stages: $\vec{\mathbf{v}} \parallel \vec{\mathbf{u}} \overset{d}{\leq} \vec{\mathbf{v}} \parallel \vec{\mathbf{U}}$ and $\vec{\mathbf{v}} \parallel \vec{\mathbf{U}} \overset{d}{\leq} \vec{\mathbf{V}} \parallel \vec{\mathbf{U}}$. We'll only write the proof of the first as the other one is basically identical.

So we prove that given a generic monotone P, $\Pr[P(\vec{\mathbf{v}} \parallel \vec{\mathbf{u}})] \leq \Pr\left[P(\vec{\mathbf{v}} \parallel \vec{\mathbf{U}})\right]$

Let's call $P_{\vec{x}}(\vec{y}) := P(\vec{x} \parallel \vec{y})$. This is monotone, and so by hypothesis

$$\forall \vec{x}.\ \Pr[P_{\vec{x}}(\vec{\mathbf{u}})] \leq \Pr\left[P_{\vec{x}}(\vec{\mathbf{U}})\right]$$

This implies that $\sum_{\vec{x}} \Pr[\vec{x} = \vec{\mathbf{v}}]\Pr[P_{\vec{x}}(\vec{\mathbf{u}})] \leq \sum_{\vec{x}} \Pr[\vec{x} = \vec{\mathbf{v}}]\Pr\left[P_{\vec{x}}(\vec{\mathbf{U}})\right]$

As $\vec{\mathbf{v}}$ is independent with $\vec{\mathbf{u}}$ and $\vec{\mathbf{U}}$, we have that

$$\sum_{\vec{x}} \Pr[\vec{x} = \vec{\mathbf{v}} \wedge P(\vec{x} \parallel \vec{\mathbf{u}})] \leq \sum_{\vec{x}} \Pr\left[\vec{x} = \vec{\mathbf{v}} \wedge P(\vec{x} \parallel \vec{\mathbf{U}})\right]$$

And so that $\Pr[P(\vec{\mathbf{v}} \parallel \vec{\mathbf{u}})] \leq \Pr\left[P(\vec{\mathbf{v}} \parallel \vec{\mathbf{U}})\right]$ ☐

Lemma 19 (Leaking Wires)**.** *For every circuit c and for every leakage rates $p, p' \in [0,1]$,*

$$p \leq p' \implies \breve{\mathbf{Leak}}_c(p) \overset{d}{\leq} \breve{\mathbf{Leak}}_c(p')$$

Proof. First of all, if c has no leakable wire, then $\breve{\mathbf{Leak}}_c(p) := []$ making the thesis true by definition of $\overset{d}{\leq}$. Then let's first consider $\breve{\mathbf{v}}(p) \in \{\emptyset, [1]\}$ a probabilistic function that returns a random variable with Bernoulli distribution $\Pr[\breve{\mathbf{v}}(p) = [1]] = p$.

The possible monotone predicates for $\breve{\mathbf{v}}$ are $P(\cdot) := true$, $P(\cdot) := false$, $P(\breve{a}) := (\breve{a} = [1])$. It's immediate to prove that if $p \leq p'$ then for all monotone predicates P,

$$\Pr[P(\breve{\mathbf{v}}(p))] \leq \Pr[P(\breve{\mathbf{v}}(p'))]$$

This means that $p \leq p' \implies \breve{\mathbf{v}}(p) \overset{d}{\leq} \breve{\mathbf{v}}(p')$. As $\breve{\mathbf{Leak}}_c(p) \overset{d}{=} \breve{\mathbf{v}}(p) \parallel \ldots \parallel \breve{\mathbf{v}}(p)$, we can prove the lemma by using Lemma 18. ☐

B.3 Correctness

Lemma 20 (Necessary Condition for Correctness)**.** *If C is a correct implementation of c for E, then for every encoded input $\vec{x} \in E.\vec{\mathbf{Enc}}[\mathbb{S}^i]$*

$$E.\vec{\mathrm{Dec}}(\vec{\mathbf{C}}_{\mathbf{outs}}(\vec{x})) \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{outs}}(E.\vec{\mathrm{Dec}}(\vec{x}))$$

Proof. By definition $\vec{\mathbf{Rnds}}_C() \overset{d}{=} \vec{\mathbf{Rnds}}_C()$, and this is preserved if we apply equal functions one on each side of the $\overset{d}{=}$, so

$$\vec{c}_{\mathrm{outs}}(E.\vec{\mathrm{Dec}}(\vec{x}), R.\vec{\mathrm{Dec}}(\vec{\mathbf{Rnds}}_C())) \overset{d}{=} E.\vec{\mathrm{Dec}}(\vec{C}_{\mathrm{outs}}(\vec{x}, \vec{\mathbf{Rnds}}_C()))$$

As by hypothesis $\vec{\mathbf{Rnds}}_C() \overset{d}{=} R.\vec{\mathbf{Enc}}(\vec{\mathbf{Rnds}}_c())$, we can apply $R.\vec{\mathrm{Dec}}$ to each side and obtain that $R.\vec{\mathrm{Dec}}(\vec{\mathbf{Rnds}}_C()) \overset{d}{=} \vec{\mathbf{Rnds}}_c()$, so the previous expression is equivalent to

$$\vec{\mathbf{c}}_{\mathbf{outs}}(E.\vec{\mathrm{Dec}}(\vec{x})) \overset{d}{=} E.\vec{\mathrm{Dec}}(\vec{\mathbf{C}}_{\mathbf{outs}}(\vec{x}))$$

□

Proposition 6 (Correctness for Deterministic)**.** *Given a deterministic c, 'C is a correct implementation of c for E' is equivalent to: for every encoded input $\vec{x} \in E.\vec{\mathbf{Enc}}[\mathbb{S}^i]$*

$$E.\vec{\mathrm{Dec}}(\vec{\mathbf{C}}_{\mathbf{outs}}(\vec{x})) = \vec{\mathbf{c}}_{\mathbf{outs}}(E.\vec{\mathrm{Dec}}(\vec{x}))$$

Proof. For Lemma 20, the correctness implies

$$E.\vec{\mathrm{Dec}}(\vec{\mathbf{C}}_{\mathbf{outs}}(\vec{x})) \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{outs}}(E.\vec{\mathrm{Dec}}(\vec{x}))$$

As $\vec{\mathbf{c}}_{\mathbf{outs}}$ is deterministic by hypothesis, and because $\vec{\mathbf{C}}_{\mathbf{outs}}$ has a discrete probability distribution, the $\overset{d}{=}$ implies the $=$.

For the opposite implication, we have by hypothesis that c is deterministic, so we can rewrite the hypothesis as: for all encoded inputs $\vec{x}$

$$\vec{c}_{\mathrm{outs}}(E.\vec{\mathrm{Dec}}(\vec{x}), []) = E.\vec{\mathrm{Dec}}(\vec{C}_{\mathrm{outs}}(\vec{x}), \vec{\mathbf{Rnds}}_C())$$

In other words, for all encoded inputs $\vec{x}$ for all the possible values $\vec{r}$ of the randoms of the compiled circuit

$$\vec{c}_{\mathrm{outs}}(E.\vec{\mathrm{Dec}}(\vec{x}), []) = E.\vec{\mathrm{Dec}}(\vec{C}_{\mathrm{outs}}(\vec{x}, \vec{r}))$$

We can prove the definition of correct implementation by using any encoding for the randoms such that $\vec{\mathbf{Enc}}([]) = \vec{\mathbf{Rnds}}_C()$. □

Lemma 21 (Transitiveness of the correctness)**.** *If c'' is a correct implementation of c' (using E'), and c' is a correct implementation of c (using E), then c'' is a correct implementation of c (using $E' \circ E$).*

Proof. Let's call R' the encoding of the randoms relative to E', same for R and E, then to prove the point 1 of the definition of correct implementation we have:

$$\vec{\mathbf{Rnds}}_{c''}() \overset{d}{=} R'.\vec{\mathbf{Enc}}(\vec{\mathbf{Rnds}}_{c'}()) \overset{d}{=} R'.\vec{\mathbf{Enc}}(R.\vec{\mathbf{Enc}}(\vec{\mathbf{Rnds}}_c())) \overset{d}{=} (R' \circ R).\vec{\mathbf{Enc}}(\vec{\mathbf{Rnds}}_c())$$

We can then report the point 2 of the definition of 'c' is a correct implementation of c' applied with the values $\vec{x} := E'.\vec{\mathrm{Dec}}(\vec{x})$ and $\vec{r} := R'.\vec{\mathrm{Dec}}(\vec{r})$:

$$\vec{c}_{\mathrm{outs}}(E.\vec{\mathrm{Dec}}(E'.\vec{\mathrm{Dec}}(\vec{x})), R.\vec{\mathrm{Dec}}(R'.\vec{\mathrm{Dec}}(\vec{r}))) = E.\vec{\mathrm{Dec}}(\vec{c'}_{\mathrm{outs}}(E'.\vec{\mathrm{Dec}}(\vec{x}), R'.\vec{\mathrm{Dec}}(\vec{r})))$$

And we can apply $E.\vec{\text{Dec}}$ to both sides of the point 2 of the definition of 'c'' is a correct implementation of c'':

$$E.\vec{\text{Dec}}(\vec{c'}_{\text{outs}}(E'.\vec{\text{Dec}}(\vec{x}), R'.\vec{\text{Dec}}(\vec{r}))) = E.\vec{\text{Dec}}(E'.\vec{\text{Dec}}(\vec{c''}_{\text{outs}}(\vec{x}, \vec{r})))$$

As $=$ is transitive, they imply the point 2 of the definition of correct implementation. ☐

Lemma 22 (The composition retains the correctness)**.** *If c', d' are correct implementations of respectively c, d, then $c' \parallel d'$ is a correct implementation of $c \parallel d$, and $c' \circ d'$ is a correct implementation of $c \circ d$, assuming the last composition is meaningful.*

Proof. The first point of the definition of correctness can be proven by the definition of $\vec{\textbf{Rnds}}()$ for series and parallel and by the encoding of the randoms for the series and parallel being obtained by a parallel of the two encodings of the randoms.

The second point can be proven quickly by writing one side of the equation, write it in terms of the two sub-circuits, split the encoding functions in the parallel of the two encodings, apply the second point of the two correctness hypotheses for the sub-circuits, and then write it again in terms of the overall circuit. ☐

Lemma 23 (Gadget-based compiler)**.** *Given the gadgets $\vec{G}$ defined with the n-shares encoding E, then $(E, CC_{\vec{G}})$ is a compiler, where the compilation function $CC_{\vec{G}} : \mathcal{C}_{\mathbb{S}_{\text{in}},\mathcal{G}_{\text{in}}} \to \mathcal{C}_{\mathbb{S}_{\text{out}},\mathcal{G}_{\text{out}}}$ obtained by substituting every gate g with the relative circuit $\vec{G}_g$. More formally,*

$$CC_{\vec{G}}(c) = \begin{cases} \vec{G}_c & \text{if } c \text{ a gate of } \mathcal{G}_{\text{in}} \\ CC_{\vec{G}}(c') \parallel CC_{\vec{G}}(c'') & \text{if } c = c' \parallel c'' \text{ for any } c', c'' \\ CC_{\vec{G}}(c_o) \circ CC_{\vec{G}}(c_i) & \text{if } c = c_o \circ c_i \text{ for any } c_o, c_i \\ i' & \text{if } c \text{ is the identity circuit} \\ s' & \text{if } c \text{ is the swap circuit} \end{cases}$$

where the identity gadget i' is obtained by composing in parallel n identity circuits, and the swap gadget s' is any gadget made of identity and swap circuits to link the inputs i with the outputs $i + n$ and vice-versa.

Proof. We can prove that $(E, CC_{\vec{G}})$ is a compiler inductively. For the base case, the $\vec{G}_c$ are correct by definition of gadgets, while the identity and swap gadget are correct as their circuit is deterministic and $E.\vec{\text{Dec}}$ is preserved, see Proposition 6. For the induction step, both types of composition preserve the correctness, as stated in Lemma 22. ☐

B.4 Monotonicity of security definitions

Lemma 24 (Monotone SRPS)**.** *If a circuit c is (p, ε)-SRPS then given any $p', \varepsilon \in [0, 1]$ with $p' \leq p$, $\varepsilon' \geq \varepsilon$ the circuit c is (p', ε')-SRPS.*

Proof. We can first define the monotone predicate (monotone due to composition of monotone functions remaining monotone, and the dependency function is monotone for Lemma 16) $P(W) := \check{\text{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(W) \neq \emptyset$.

Then (p, ε)-SRPS iff $\Pr\left[P(\check{\textbf{Leak}}_c(p))\right] \leq \varepsilon$ and by Lemma 19 we obtain that

$$\Pr\left[P(\check{\textbf{Leak}}_c(p'))\right] \leq \Pr\left[P(\check{\textbf{Leak}}_c(p))\right] \leq \varepsilon \leq \varepsilon'$$

☐

Lemma 25 (Monotone RPR)**.** *Given a compiler C that is e-RPR, and given a monotone continuous $e' \geq e$, then C is also e'-RPR.*

Proof. We need to prove that C is also e'-RPR. This means that given a generic $p, \varepsilon \in [0,1]$ and a generic circuit c, we have additional hypothesis that c is $(e'(p), \varepsilon)$-SRPS and we need to prove that $C(c)$ is (p, ε)-SRPS.

As $e(p) \leq e'(p)$ and c is $(e'(p), \varepsilon)$-SRPS, then by Lemma 24 we obtain that c is $(e(p), \varepsilon)$-SRPS, which means we can use the hypothesis that C that is e-RPR to obtain that $C(c)$ is (p, ε)-SRPS. ☐

Lemma 26 (Monotone ERPE)**.** *Given two circuits c, c' such that c is (t, e)-ERPE of c', then for all continuous monotone $e' : [0,1] \to [0,1]$ with $e' \geq e$, the circuit c is (t, e')-ERPE of c'*

Proof. All the conditions of the ERPE are untouched by changing e with e' except the continuity, the monotonicity and the $[0,1] \to [0,1]$ (which are preserved by hypothesis), and that $\breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'}(e(p))$. As for all p we have that $e(p) \leq e'(p)$, then we can use Lemma 19 which guarantees that $\breve{\mathbf{Leak}}_g(e(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_g(e'(p))$ and this proves the thesis as $\overset{d}{\leq}$ is a partial order, and so it's transitive. ☐

Lemma 27. *Given a circuit c with $w \geq 1$ internal wires, then $\forall \varepsilon \in [0,1]$, c is $(\varepsilon/w, \varepsilon)$-SRPS.*

Proof. By definition of SRPS we need to show that

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

As $\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(\emptyset) = \emptyset$, we have that $\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(\breve{w}) \neq \emptyset \implies \breve{w} \neq \emptyset$, which implies $\Pr\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leak}}_c(\varepsilon/w)) \neq \emptyset\right] \leq \Pr\left[\breve{\mathbf{Leak}}_c(\varepsilon/w) \neq \emptyset\right] = 1 - (1 - \varepsilon/w)^w$

So it's sufficient to show that $1 - (1 - \varepsilon/w)^w \leq \varepsilon$. As $w \geq 1$, $\varepsilon/w \in [0,1]$ we can use Bernoulli's inequality, which states that $\varepsilon/w \cdot w \geq 1 - (1 - \varepsilon/w)^w$. ☐

C From Probing Security To Constant Noise With Polylog Size Increase

We'll prove the lemmas and theorems that are needed for the Corollary 4, 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.

C.1 Basic Lemmas on the Random Probing Security

As [BCP+20] gives a definition of random probing security without an explicit connection to existing concepts like the random probing adversary of [DDF14], we state the following:

Lemma 28 (Security against random probing adversary)**.** *A circuit c is (p, ε)-RPS if and only if it's ε-secure against a p-random probing adversary.*

Proof. Note that the definition of (p, ε)-RPS can be rewritten as

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

where $\mathcal{A}'$ is the p-random probing adversary that specifies the maximum leakage p for all wires, and that returns the value of the leaking wires without altering them.

From the definition of ε-secure against an adversary, and for the expression just written, this lemma can be proven by the following: for all p-random probing adversary $\mathcal{A}$ there is a $\vec{\mathbf{Sim}}_{\mathbf{A}}$ such that for all $\vec{x} \in \mathbb{S}^i_{\text{orig}}$

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

We can prove this by first noticing that for every p-random probing adversaries $\mathcal{A}$ there is a $\vec{\mathbf{f}}_{\mathcal{A}}$ such that $\vec{\mathbf{out}}_{\mathcal{A}} = \vec{\mathbf{f}}_{\mathcal{A}} \circ \vec{\mathbf{out}}_{\mathcal{A}'}$, as $\mathcal{A}'$ uses the highest possible leakage rate and returns the leaking wires as is. For this reason $\vec{\mathbf{f}}_{\mathcal{A}}$ has to first lower the probability of each wires from p to the various $p_i \leq p$ chosen by $\mathcal{A}$ (this by keeping each leaking value with probability p_i/p, and discarding it with probability $1 - p_i/p$), and then apply the same function that $\mathcal{A}$ does on the vector of leaked values.

This means that we can choose $\vec{\mathbf{Sim}}_{\mathbf{A}} := \vec{\mathbf{f}}_{\mathcal{A}}(\vec{\mathbf{Sim}})$. From here the lemma follows from a well-known property of the SD (which we can find for example in [HU05]): for all $\mathbf{A}, \mathbf{B}, \mathbf{f}$ we have that $\text{SD}\left[\mathbf{f}(\mathbf{A}); \mathbf{f}(\mathbf{B})\right] \leq \text{SD}\left[\mathbf{A}; \mathbf{B}\right]$. $\square$

Related to this lemma we have the following reduction.

Lemma 29 (SRPS to RPS)**.** *If a circuit is (p, ε)-SRPS, then it's (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 by definition

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

$$\iff \sum_{\breve{W}} \Pr\left[\breve{W} = \breve{\mathbf{Leak}}_c(p)\right] \Pr\left[\breve{\text{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(\breve{W}) = \emptyset\right] \geq 1 - \varepsilon$$

We also have that

$$\begin{aligned}\Pr\left[\breve{\text{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(\breve{W}) = \emptyset\right] &= \text{Is}\left[\breve{\text{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ E.\vec{\mathbf{Enc}}]}(\breve{W}) = \emptyset\right] \\ &= \text{Is}\left[\exists \vec{\mathbf{Sim}}.\ \forall \vec{x} \in \mathbb{S}^i_{\text{orig}}.\ \vec{\mathbf{Sim}} \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))_{\breve{W}}\right] \\ &= \max_{\vec{\mathbf{Sim}}} \min_{\vec{x} \in \mathbb{S}^i_{\text{orig}}} \text{Is}\left[\vec{\mathbf{Sim}} \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right]\end{aligned}$$

If we substitute this into the statement before the last we obtain

$$\iff \sum_{\breve{W}} \Pr\left[\breve{W} = \breve{\mathbf{Leak}}_c(p)\right] \max_{\vec{\mathbf{Sim}}} \min_{\vec{x} \in \mathbb{S}^i_{\text{orig}}} \text{Is}\left[\vec{\mathbf{Sim}} \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right] \geq 1 - \varepsilon$$

We can bring the maximization over $\vec{\mathbf{Sim}}$ out of the sum by making it a maximization over a function $\vec{\mathbf{Sim}}'$:

$$\iff \max_{\vec{\mathbf{Sim}}'} \sum_{\breve{W}} \Pr\left[\breve{W} = \breve{\mathbf{Leak}}_c(p)\right] \min_{\vec{x} \in \mathbb{S}^i_{\text{orig}}} \text{Is}\left[\vec{\mathbf{Sim}}'(\breve{W}) \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right] \geq 1 - \varepsilon$$

As $\sum \min \leq \min \sum$, the last statement implies

$$\max_{\vec{\mathbf{Sim}}'} \min_{\vec{x} \in \mathbb{S}^i_{\text{orig}}} \sum_{\breve{W}} \Pr\left[\breve{W} = \breve{\mathbf{Leak}}_c(p)\right] \text{Is}\left[\vec{\mathbf{Sim}}'(\breve{W}) \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right] \geq 1 - \varepsilon$$

Which is equivalent to

$$\exists \vec{\mathbf{Sim}}'.\ \forall \vec{x} \in \mathbb{S}^i_{\text{orig}}.\ \sum_{\breve{W}} \Pr\left[\breve{W} = \breve{\mathbf{Leak}}_c(p)\right] \text{Is}\left[\vec{\mathbf{Sim}}'(\breve{W}) \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right] \geq 1 - \varepsilon$$

That is equivalent to saying that $\exists \vec{\mathbf{Sim}}'$ such that $\vec{\mathbf{Sim}}'(\breve{W})$ returns some vector $\vec{\cdot}|_{\breve{W}}$, and such that $\forall \vec{x} \in \mathbb{S}^i_{\text{orig}}$

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

As the content of the $\text{Is}[\dots]$ is a deterministic predicate that compares probability distributions, the last statement is equivalent to saying that $\exists \vec{\mathbf{Sim}}'$ such that $\vec{\mathbf{Sim}}'(\breve{W})$ returns some vector $\vec{\cdot}|_{\breve{W}}$, and such that $\forall \vec{x} \in \mathbb{S}^i_{\text{orig}}$

$$\sum_{\breve{W}} \Pr\Big[\breve{W} = \breve{\mathbf{Leak}}_c(p)\Big] \text{Is}\left[\neg(\vec{\mathbf{Sim}}'(\breve{W}) \overset{d}{=} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}})\right] \leq \varepsilon$$

As $\text{SD}[\mathbf{A};\mathbf{B}] \leq \text{Is}\left[\neg(\mathbf{A} \overset{d}{=} \mathbf{B})\right]$ can be quickly proven by separating the two possible values of the $\text{Is}[\cdot]$, the previous statement implies: $\exists \vec{\mathbf{Sim}}$ such that $\vec{\mathbf{Sim}}(\breve{W})$ returns some vector $\vec{\cdot}|_{\breve{W}}$, and such that $\forall \vec{x} \in \mathbb{S}^i_{\text{orig}}$

$$\sum_{\breve{W}} \Pr\Big[\breve{W} = \breve{\mathbf{Leak}}_c(p)\Big] \text{SD}\left[\vec{\mathbf{Sim}}(\breve{W}); \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right] \leq \varepsilon$$

Let's consider the left side of this inequality

$$\sum_{\breve{W}} \Pr\Big[\breve{W} = \breve{\mathbf{Leak}}_c(p)\Big] \text{SD}\left[\vec{\mathbf{Sim}}(\breve{W}); \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right]$$

Let's define the random variable $\breve{\mathbf{L}} := \breve{\mathbf{Leak}}_c(p)$

$$= \sum_{\breve{W}} \Pr\Big[\breve{W} = \breve{\mathbf{L}}\Big] \text{SD}\left[\vec{\mathbf{Sim}}(\breve{W}); \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}\right]$$

By definition of Statistical Distance,

$$= \sum_{\breve{W}} \Pr\Big[\breve{W} = \breve{\mathbf{L}}\Big] \frac{1}{2} \sum_{\vec{y} \in (\mathbb{S} \cup \{\perp\})^{ni}} \left| \Pr\Big[\vec{\mathbf{Sim}}(\breve{W}) = \vec{y}\Big] - \Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}} = \vec{y}\Big] \right|$$

As both $\vec{\mathbf{Sim}}(\breve{W})$ and $\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{W}}$ return a vector $\vec{\cdot}|_{\breve{W}}$, we can ignore the other elements as they are all $\perp$

$$= \sum_{\breve{W}} \Pr\Big[\breve{W} = \breve{\mathbf{L}}\Big] \frac{1}{2} \sum_{\vec{y} \in \mathbb{S}^{|\breve{W}|}} \left| \Pr\Big[\vec{\mathbf{Sim}}(\breve{W})_{\breve{W}} = \vec{y}\Big] - \Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))_{\breve{W}} = \vec{y}\Big] \right|$$

$$= \frac{1}{2} \sum_{\breve{W}} \sum_{\vec{y} \in \mathbb{S}^{|\breve{W}|}} \left| \Pr\Big[\breve{W} = \breve{\mathbf{L}}\Big] \Pr\Big[\vec{\mathbf{Sim}}(\breve{W})_{\breve{W}} = \vec{y}\Big] - \Pr\Big[\breve{W} = \breve{\mathbf{L}}\Big] \Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))_{\breve{W}} = \vec{y}\Big] \right|$$

$$= \frac{1}{2} \sum_{\breve{W}} \sum_{\vec{y} \in \mathbb{S}^{|\breve{W}|}} \left| \Pr\Big[\breve{W} = \breve{\mathbf{L}} \wedge \vec{\mathbf{Sim}}(\breve{W})_{\breve{W}} = \vec{y}\Big] - \Pr\Big[\breve{W} = \breve{\mathbf{L}} \wedge \vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))_{\breve{W}} = \vec{y}\Big] \right|$$

$$= \frac{1}{2} \sum_{\breve{W}} \sum_{\vec{y} \in \mathbb{S}^{|\breve{W}|}} \left| \Pr\Big[(\vec{\mathbf{Sim}}(\breve{\mathbf{L}})_{\breve{\mathbf{L}}}, \breve{\mathbf{L}}) = (\vec{y}, \breve{W})\Big] - \Pr\Big[(\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))_{\breve{\mathbf{L}}}, \breve{\mathbf{L}}) = (\vec{y}, \breve{W})\Big] \right|$$

As given $\vec{z}, \breve{w}$, $(\vec{z}_{\breve{w}}, \breve{w})$ is bijective with $\vec{z}|_{\breve{w}}$,

$$= \frac{1}{2} \sum_{\vec{y} \in (\mathbb{S} \cup \{\perp\})^{ni}} \left| \Pr\Big[\vec{\mathbf{Sim}}(\breve{\mathbf{L}})|_{\breve{\mathbf{L}}} = \vec{y}\Big] - \Pr\Big[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{L}}} = \vec{y}\Big] \right|$$

$$
\begin{aligned}
=&\frac{1}{2}\sum_{\vec{y}\in(\mathbb{S}\cup\{\perp\})^{ni}}\left|\Pr\left[\vec{\mathbf{Sim}}(\breve{\mathbf{L}})=\vec{y}\right]-\Pr\left[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{L}}}=\vec{y}\right]\right|\\
=&\frac{1}{2}\sum_{\vec{y}\in(\mathbb{S}\cup\{\perp\})^{ni}}\left|\Pr\left[\vec{\mathbf{Sim}}(\breve{\mathbf{Leak}}_c(p))=\vec{y}\right]-\Pr\left[\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{Leak}}_c(p)}=\vec{y}\right]\right|\\
=&\mathrm{SD}\left[\vec{\mathbf{Sim}}(\breve{\mathbf{Leak}}_c(p));\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{Leak}}_c(p)}\right]
\end{aligned}
$$

This means that $\exists\vec{\mathbf{Sim}}'.\ \forall\vec{x}\in\mathbb{S}^i_{\text{orig}}.\ \mathrm{SD}\left[\vec{\mathbf{Sim}}';\vec{\mathbf{c}}_{\mathbf{wires}}(E.\vec{\mathbf{Enc}}(\vec{x}))|_{\breve{\mathbf{Leak}}_c(p)}\right]\leq\varepsilon$, which is the definition of 'c is (p,ε)-RPS'. □

C.2 Proposition 7: Probing Model to SRPS

As we need to obtain the SRPS property from the probing security, we can not use the result from [DDF14], which instead obtains the weaker property of security in the random probing model. For this reason we prove the following proposition:

Proposition 7 (Probing Model to SRPS)**.** *If a circuit c with w wires is t-probing secure, then it's $(\frac{t}{2we},2^{-t})$-SRPS where e is the mathematical constant ≈ 2.718.*

Proof. We can prove this with the following passages:

$$
\begin{aligned}
&\Pr\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leak}}_c(p))\neq\emptyset\right]\\
=&\sum_{\breve{W}\subseteq[w]}\Pr\left[\breve{W}=\breve{\mathbf{Leak}}_c(p)\right]\mathrm{Is}\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{W})\neq\emptyset\right]\\
=&\sum_{\breve{W}\subseteq[w]}p^{|\breve{W}|}(1-p)^{w-|\breve{W}|}\mathrm{Is}\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{W})\neq\emptyset\right]
\end{aligned}
$$

By definition of 'secure in the t-probing model', $|\breve{W}|\leq t$ guarantees no dependency, so

$$
\begin{aligned}
\leq&\sum_{\breve{W}\subseteq[w]}p^{|\breve{W}|}(1-p)^{w-|\breve{W}|}\mathrm{Is}\left[\left|\breve{W}\right|>t\right]=\sum_{\substack{\breve{W}\subseteq[w]\\ |\breve{W}|>t}}p^{|\breve{W}|}(1-p)^{w-|\breve{W}|}\\
=&\sum_{l:=t+1}^{w}\binom{w}{l}p^l(1-p)^{w-l}\leq\sum_{l:=t+1}^{w}\binom{w}{l}p^l\leq\sum_{l:=t+1}^{w}\frac{w^l}{l!}p^l
\end{aligned}
$$

Using Stirling's approximation, which is known to be a lower bound of $l!$

$$
\leq\sum_{l:=t+1}^{w}\frac{(pw)^l}{\sqrt{2\pi l}(\frac{l}{e})^l}\leq\sum_{l:=t+1}^{w}(\frac{epw}{l})^l\leq\sum_{l:=t+1}^{w}(\frac{epw}{t})^l
$$

If we choose a $p<\frac{t}{we}$,

$$
\Pr\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leak}}_c(p))\neq\emptyset\right]\leq\sum_{l:=t+1}^{w}(\frac{epw}{t})^l\leq\sum_{l:=t+1}^{\infty}(\frac{epw}{t})^l=\frac{(\frac{epw}{t})^{t+1}}{1-\frac{epw}{t}}
$$

If we choose $p:=\frac{t}{2we}$

$$
\Pr\left[\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}}\circ E.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leak}}_c(\frac{t}{2we}))\neq\emptyset\right]\leq\frac{(\frac{epw}{t})^{t+1}}{1-\frac{epw}{t}}=\frac{0.5^{t+1}}{1-0.5}=2^{-t}
$$

This by definition means that c is $(\frac{t}{2we},2^{-t})$-SRPS □

C.3 Theorem 1: Complexity of Compiler Sequences

Theorem 1: Given a compiler sequence $C : C_{\text{in}} \to C_{\text{out}}$, given a circuit $c_\kappa \in C_{\text{in}}$ parametric in its security level, i.e. such that c_κ is $(2^{-p(\kappa)}, 2^{-\kappa})$-SRPS for some $p : (0, \infty) \to (0, \infty)$; then there is a function $n : (0, \infty) \to \mathbb{N}$ that calculates which compiler in C to use, such that the compiled circuit $c'_\kappa := C_{n(\kappa)}(c_\kappa)$ satisfies the following properties:

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

This where P is a tolerated leakage of C, and e is a expansion exponent of C, P.

Proof. Let's 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 e such that C_m is e_m-RPR and $\log_2 e_m(P) = \Omega\left(d^m\right)$ as $m \to \infty$. In other words there is a $b < 0$ such that

$$\liminf_{m\to\infty} \frac{|\log_2 e_m(P)|}{d^m} = -2b > 0$$

As $\log_2 e_m(P) \le 0$, this means that $\limsup\limits_{m\to\infty} \frac{\log_2 e_m(P)}{d^m} = 2b < b$

And so eventually $\log_2 e_m(P) \le bd^m$, i.e. eventually $e_m(P) \le 2^{d^m b}$. More precisely this means that there is an n' such that for every $m \ge n'$ we have that $e_m(P) \le 2^{d^m b}$.

We can now choose

$$n(\kappa) := \max\{n', \log_d \frac{p(\kappa)}{-b}\}$$

As $n(\kappa) \ge n'$ we have that $e_{n(\kappa)}(P) \le 2^{d^{n(\kappa)} b}$ Additionally, $2^{d^{n(\kappa)} b} \le 2^{-p(\kappa)}$ as

$$d^{n(\kappa)} b \le d^{\log_d \frac{p(\kappa)}{-b}} b = \frac{p(\kappa)}{-b} b = -p(\kappa)$$

This means that as c_κ is $(2^{-p(\kappa)}, 2^{-\kappa})$-SRPS, then by Lemma 24 the circuit c_κ is $(e_{n(\kappa)}(P), 2^{-\kappa})$-SRPS as $e_{n(\kappa)}(P) \le 2^{-p(\kappa)}$.

We can then apply the definition of RPR from '$C_{n(\kappa)}$ is $e_{n(\kappa)}$-RPR' to obtain that $c'_\kappa := C_{n(\kappa)}(c_\kappa)$ 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,

$$\|c'_\kappa\| \le \left\|M_{C_{n(\kappa)}}\right\|_1 \|c_\kappa\| = \mathcal{O}\left(\|c_\kappa\| \, \lambda^{n(\kappa)}\right)$$

As $n(\kappa) = \max\{n', \log_d \frac{p(\kappa)}{-b}\} = \max\{n', \frac{\log p(\kappa)}{\log d} - \frac{\log -b}{\log d}\} = \frac{\log p(\kappa)}{\log d} + \Theta\left(1\right)$ then

$$\lambda^{n(\kappa)} = \lambda^{\frac{\log p(\kappa)}{\log d} + \Theta(1)} = \Theta\left(\lambda^{\frac{\log p(\kappa)}{\log d}}\right) = \Theta\left(p(\kappa)^{\frac{\log \lambda}{\log d}}\right)$$

And so $\|c'_\kappa\| = \mathcal{O}\left(\|c_\kappa\| \, \lambda^{n(\kappa)}\right) = \mathcal{O}\left(\|c_\kappa\| \, \Theta\left(p(\kappa)^{\frac{\log_2 \lambda}{\log_2 d}}\right)\right) = \mathcal{O}\left(\|c_\kappa\| \, p(\kappa)^e\right)$ which proves the second point, and so the theorem. ☐

D Theorem 2: ERPE to RPR

In this appendix we will prove our main theorem for the expansion: Theorem 2. To do this we first introduce a few other lemmas in order to make the proof more readable.

The following two lemmas are an immediate consequence of Lemma 4.

Lemma 30 (Identity Circuit)**.** *Any correct and leakless n-shares implementation G of the identity circuit for an encoding with strength k is (t, e)-ERPE for any $t \in [0, k] \cap \mathbb{N}$ and for any monotone and continuous $e : [0, 1] \to [0, 1]$.*

Lemma 31 (Swap Circuit)**.** *Any correct and leakless n-shares implementation G of the swap circuit for an encoding with strength k is (t, e)-ERPE for any $t \in [0, k] \cap \mathbb{N}$ and for any monotone and continuous $e : [0, 1] \to [0, 1]$.*

Lemma 32 (Parallel Composition)**.** *Given a compiler (E, CC_b) and two circuits c'_1, c'_2 such that $c_1 := CC_b(c'_1)$ is (t, e)-ERPE of c'_1 and $c_2 := CC_b(c'_2)$ is (t, e)-ERPE of c'_2, then $c_1 \parallel c_2$ is (t, e)-ERPE of $c'_1 \parallel c'_2$.*

Proof. We'll call $c = c_1 \parallel c_2$, $c' = c'_1 \parallel c'_2$, and we choose

$$\begin{aligned}
\breve{I}_c(\breve{O}_1 \parallel \breve{O}_2, \breve{W}_1 \parallel \breve{W}_2) &:= \breve{I}_{c_1}(\breve{O}_1, \breve{W}_1) \parallel \breve{I}_{c_2}(\breve{O}_2, \breve{W}_2)\\
\breve{W}'_c(\breve{O}_1 \parallel \breve{O}_2, \breve{W}_1 \parallel \breve{W}_2) &:= \breve{W}'_{c_1}(\breve{O}_1, \breve{W}_1) \parallel \breve{W}'_{c_2}(\breve{O}_2, \breve{W}_2)\\
\breve{O}_c(\breve{O}_1 \parallel \breve{O}_2, \breve{W}_1 \parallel \breve{W}_2, \breve{I}'_1 \parallel \breve{I}'_2) &:= \breve{O}_{c_1}(\breve{O}_1, \breve{W}_1, \breve{I}'_1) \parallel \breve{O}_{c_2}(\breve{O}_2, \breve{W}_2, \breve{I}'_2)
\end{aligned}$$

Then by using those definitions and the fact that the RPE (t, k)-normalization is compatible with the composition in parallel, the items of the definition of ERPE become the following

1 For all leakage rates $p \in [0, 1]$, we must have that

$$\breve{W}'_{c_1}(\emptyset, \breve{\mathbf{Leak}}_{c_1}(p)) \parallel \breve{W}'_{c_2}(\emptyset, \breve{\mathbf{Leak}}_{c_2}(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'_1}(e(p)) \parallel \breve{\mathbf{Leak}}_{c'_2}(e(p))$$

This has the structure $\mathbf{A} \parallel \mathbf{B} \overset{d}{\leq} \mathbf{C} \parallel \mathbf{D}$, which means we can prove it using Lemma 18 as $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are pairwise independent, and as $\mathbf{A} \overset{d}{\leq} \mathbf{C}$ and $\mathbf{B} \overset{d}{\leq} \mathbf{D}$ by the the item 1 of the ERPE of c_1 and c_2.

Then we need to show that for all possible combinations of output to simulate $\breve{O}_1 \parallel \breve{O}_2 \in [n \cdot (o_1 + o_2)]$, we must have that

$$\breve{W}'_{c_1}(\breve{O}_1, \breve{\mathbf{Leak}}_{c_1}(p)) \parallel \breve{W}'_{c_2}(\breve{O}_2, \breve{\mathbf{Leak}}_{c_2}(p)) \overset{d}{=} \breve{W}'_{c_1}(\emptyset, \breve{\mathbf{Leak}}_{c_1}(p)) \parallel \breve{W}'_{c_2}(\emptyset, \breve{\mathbf{Leak}}_{c_2}(p))$$

This is true as for the item 1 of the ERPE of c_1 and c_2 we obtain that

$$\breve{W}'_{c_1}(\breve{O}_1, \breve{\mathbf{Leak}}_{c_1}(p)) \overset{d}{=} \breve{W}'_{c_1}(\emptyset, \breve{\mathbf{Leak}}_{c_1}(p))$$

$$\breve{W}'_{c_2}(\breve{O}_2, \breve{\mathbf{Leak}}_{c_2}(p)) \overset{d}{=} \breve{W}'_{c_2}(\emptyset, \breve{\mathbf{Leak}}_{c_2}(p))$$

2 We need to show that for all subsets of the required output wires $\breve{O}_1 \parallel \breve{O}_2 \subseteq [n \cdot (o_1 + o_2)]$, for every subset of the internal wires $\breve{W}_1 \parallel \breve{W}_2 \subseteq [w_1 + w_2]$, for all the inputs actually provided $\breve{I}_1$ an RPE (t, k)-normalization of $\breve{I}_{c_1}(\breve{O}_1, \breve{W}_1)$ and $\breve{I}_2$ an RPE (t, k)-normalization of $\breve{I}_{c_2}(\breve{O}_2, \breve{W}_2)$, we need that $\breve{O}_{c_1}(\breve{O}_1, \breve{W}_1, \breve{I}_1)$ an RPE (t, k)-normalization of $\breve{O}_1$ and $\breve{O}_{c_2}(\breve{O}_2, \breve{W}_2, \breve{I}_2)$ an RPE (t, k)-normalization of $\breve{O}_2$. This is true by the item 2 of the ERPE of c_1, c_2.

Additionally, we need that the function that takes as parameter the inputs $\vec{x_1} \parallel \vec{x_2} \in E.\vec{\mathbf{Enc}}[\mathbb{S}_{\text{in}}^{i_1+i_2}]$ and the original randoms $\vec{r'_1} \parallel \vec{r'_2} \in \text{supp}[\vec{\mathbf{Rnds}}_{c'_1}() \parallel \vec{\mathbf{Rnds}}_{c'_2}()]$ and calculates

$$\vec{c_1}_{\text{all}}(\vec{x_1}, R.\vec{\mathbf{Enc}}(\vec{r'_1}))_{\breve{W}_1 \parallel \breve{O}_{c_1}(\breve{O}_1, \breve{W}_1, \breve{I}_1)} \parallel \vec{c_2}_{\text{all}}(\vec{x_2}, R.\vec{\mathbf{Enc}}(\vec{r'_2}))_{\breve{W}_2 \parallel \breve{O}_{c_2}(\breve{O}_2, \breve{W}_2, \breve{I}_2)}$$

can be simulated using $(\vec{x_1})_{\breve{I}_1} \parallel (\vec{x_2})_{\breve{I}_2}$, and

$$\vec{c'_1}_{\text{wires}}(E.\vec{\text{Dec}}(\vec{x_1}), \vec{r'_1})_{\breve{W}'_{c_1}(\breve{O}_1,\breve{W}_1)} \parallel \vec{c'_2}_{\text{wires}}(E.\vec{\text{Dec}}(\vec{x_2}), \vec{r'_2})_{\breve{W}'_{c_2}(\breve{O}_2,\breve{W}_2)}$$

This is guaranteed by the item 2 of the ERPE of c_1 and c_2 and by the composition of the simulatability.

□

Lemma 33 (Series Composition). *Given a compiler (E, CC_b) and two circuits c'_1, c'_2 such that $c_1 := CC_b(c'_1)$ is (t,e)-ERPE of c'_1 and $c_2 := CC_b(c'_2)$ is (t,e)-ERPE of c'_2, then $c_1 \circ c_2$ is (t,e)-ERPE of $c'_1 \circ c'_2$.*

Proof. We'll call $c := c_o \circ c_i$, $c' := c'_o \circ c'_i$, and we choose

$$\begin{aligned}
&\breve{I}_c(\breve{O}, \breve{W}_i \parallel \breve{W}_o) := \breve{I}_{c_i}(\breve{I}_{c_o}(\breve{O}, \breve{W}_o), \breve{W}_i)\\
&\breve{W}'_c(\breve{O}, \breve{W}_i \parallel \breve{W}_o) := \breve{W}'_{c_i}(\breve{I}_{c_o}(\breve{O}, \breve{W}_o), \breve{W}_i) \parallel \breve{W}'_{c_o}(\breve{O}, \breve{W}_o)\\
&\breve{O}_c(\breve{O}, \breve{W}_i \parallel \breve{W}_o, \breve{I}) := \breve{O}_{c_o}(\breve{O}, \breve{W}_o, \breve{O}_{c_i}(\breve{I}_{c_o}(\breve{O}, \breve{W}_o), \breve{W}_i, \breve{I}))
\end{aligned}$$

1 We need to prove that for all combinations of the output wires $\breve{O} \in [\mu]^o$ the probability distribution of $\breve{W}'_c(\breve{O}, \breve{\mathbf{Leak}}_{c'}(p)) \overset{d}{=} \breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_{c'}(p))$.

We'll prove this by showing that

$$\breve{W}'_c(\breve{O}, \breve{\mathbf{Leak}}_c(p)) \overset{d}{=} \breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) \parallel \breve{W}'_{c_o}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p))$$

Proof:

$$\breve{W}'_c(\breve{O}, \breve{\mathbf{Leak}}_c(p))$$

By definition, with $\breve{\mathbf{w}}_{\mathbf{o}} := \breve{\mathbf{Leak}}_{c_o}(p)$

$$\begin{aligned}
=&\Pr\Big[\breve{W}'_{c_i}(\breve{I}_{c'_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}), \breve{\mathbf{Leak}}_{c_i}(p)) \parallel \breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}) = \breve{w}'_i \parallel \breve{w}'_o\Big]\\
=&\Pr\Big[\breve{W}'_{c_i}(\breve{I}_{c_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}), \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i \wedge \breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}) = \breve{w}'_o\Big]\\
=&\sum_{w_o} \Pr\Big[\breve{W}'_{c_i}(\breve{I}_{c_o}(\breve{O}, \breve{w}_o), \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i \wedge \breve{W}'_{c_o}(\breve{O}, \breve{w}_o) = \breve{w}'_o \wedge \breve{\mathbf{w}}_{\mathbf{o}} = w_o\Big]\\
=&\sum_{w_o} \Pr\Big[\breve{W}'_{c_i}(\breve{I}_{c_o}(\breve{O}, \breve{w}_o), \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i\Big] \Pr\Big[\breve{W}'_{c_o}(\breve{O}, \breve{w}_o) = \breve{w}'_o \wedge \breve{\mathbf{w}}_{\mathbf{o}} = w_o\Big]
\end{aligned}$$

For the item 1 of the ERPE of c_i, $\breve{W}'_{c_i}$ is identically distributed for every value of its first parameter

$$\begin{aligned}
=&\sum_{w_o} \Pr\Big[\breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i\Big] \Pr\Big[\breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}) = \breve{w}'_o \wedge \breve{\mathbf{w}}_{\mathbf{o}} = w_o\Big]\\
=&\Pr\Big[\breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i\Big] \sum_{w_o} \Pr\Big[\breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}) = \breve{w}'_o \wedge \breve{\mathbf{w}}_{\mathbf{o}} = w_o\Big]\\
=&\Pr\Big[\breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i\Big] \Pr\Big[\breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{w}}_{\mathbf{o}}) = \breve{w}'_o\Big]\\
=&\Pr\Big[\breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) = \breve{w}'_i\Big] \Pr\Big[\breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{Leak}}_{c'_o}(p)) = \breve{w}'_o\Big]
\end{aligned}$$

For the item 1 of the ERPE of c_o, $\breve{W}'_{c_o}$ is identically distributed for every value of its first parameter

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

Additionally, We need to prove that for all leakage rates $p \in [0,1]$ the distribution of the set of leaking virtual wires $\breve{W}'_c$ must be upper bounded by the leakage of the virtual circuit:

$$\breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_c(e(p))$$

We can prove this by showing that

$$\breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p)) \overset{d}{=} \breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) \parallel \breve{W}'_{c_o}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'}(e(p))$$

We already proved the left part in the proof, and to prove the right part we can use the item 1 of the ERPE of c_o and c_i and obtain that:

$$\breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'_i}(e(p)) \wedge \breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{Leak}}_{c_i}(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'_o}(e(p))$$

As the four expressions are all independent, we can use Lemma 18 and obtain that

$$\breve{W}'_{c_i}(\emptyset, \breve{\mathbf{Leak}}_{c_i}(p)) \parallel \breve{W}'_{c_o}(\breve{O}, \breve{\mathbf{Leak}}_{c_i}(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'_i}(e(p)) \parallel \breve{\mathbf{Leak}}_{c'_o}(e(p))$$

2 For all combinations of the output wires $\breve{O} \in [n \cdot o]$ for all the internal wires $\breve{W}_i \parallel \breve{W}_o \subseteq [w]$, we define $\breve{M} := \breve{I}_{c_o}(\breve{O}, \breve{W}_o)$, and $\breve{I} := \breve{I}_{c_i}(\breve{M}, \breve{W}_i)$. For all the possible provided inputs $\breve{I}'$ an RPE (t,k)-normalization of $\breve{I}$ we define $\breve{M}' := \breve{O}_{c_i}(\breve{M}, \breve{W}_i, \breve{I}')$, and $\breve{O}' := \breve{O}_{c_o}(\breve{O}, \breve{W}_o, \breve{M}')$.

From item 2 of the ERPE of c_i, $\breve{M}'$ an RPE (t,k)-normalization of $\breve{M}$; and from item 2 of ERPE of c_o, $\breve{O}'$ an RPE (t,k)-normalization of $\breve{O}$, which proves the first part of this point by definition.

Then for all $\vec{x}, \vec{r'_i}, \vec{r'_o}$ we need to prove that

$$\vec{x}_{\breve{I}'} \parallel \vec{c'}_{\text{wires}}(E.\vec{\text{Dec}}(\vec{x}), \vec{r'_i} \parallel \vec{r'_o})_{\breve{W}'_c(\breve{O},\breve{W})} \xrightarrow{\text{sim}} \vec{c}_{\text{all}}(\vec{x}, R_{c_i}.\vec{\mathbf{Enc}}(\vec{r'_i}) \parallel R_{c_o}.\vec{\mathbf{Enc}}(\vec{r'_i}))_{\breve{W}_o \parallel \breve{W}_i \parallel \breve{O}'}$$

We first define $(\mathbf{w_i}, \vec{\mathbf{v}}) := \vec{c_i}_{\text{all}}(\vec{x}, R_{c_i}.\vec{\mathbf{Enc}}(\vec{r'_i}))$ and $(\mathbf{w_o}, \vec{\mathbf{y}}) := \vec{c_o}_{\text{all}}(\vec{\mathbf{v}}, R_{c_o}.\vec{\mathbf{Enc}}(\vec{r'_o}))$. We also define the corresponding values in the virtual circuit $\vec{x'} := E.\vec{\text{Dec}}(\vec{x})$, $(\vec{w'_i}, \vec{v'}) := \vec{c'_i}_{\text{all}}(\vec{x'}, \vec{r'_i})$, and $(\vec{w'_o}, \vec{y'}) := \vec{c'_o}_{\text{all}}(\vec{v'}, \vec{r'_o})$.

Thanks to our definition of correctness,

$$
\begin{aligned}
\vec{v'} &= (\vec{c'_i})_{\text{outs}}(\vec{x'}, \vec{r'_i})\\
&= (\vec{c'_i})_{\text{outs}}(E.\vec{\text{Dec}}(\vec{x}), R_{c_i}.\vec{\text{Dec}}(R_{c_i}.\vec{\mathbf{Enc}}(\vec{r'_i})))\\
&= E.\vec{\text{Dec}}((\vec{c_i})_{\text{outs}}(\vec{x}, R_{c_i}.\vec{\mathbf{Enc}}(\vec{r'_i})))\\
&= E.\vec{\text{Dec}}(\vec{\mathbf{v}})
\end{aligned}
$$

Then what we need to prove is that

$$\vec{x}_{\breve{I}'} \parallel (\vec{w'_i})_{\breve{W}'_{c_i}(\breve{M},\breve{W}_i)} \parallel (\vec{w'_o})_{\breve{W}'_{c_o}(\breve{O},\breve{W}_o)} \xrightarrow{\text{sim}} \vec{\mathbf{y}}_{\breve{O}'} \parallel (\vec{\mathbf{w_i}})_{\breve{W}_i} \parallel (\vec{\mathbf{w_o}})_{\breve{W}_o}$$

By the item 2 of the ERPE of c_i (as $\breve{M}'$ an RPE (t,k)-normalization of $\breve{M}$, $\breve{I}$ an RPE (t,k)-normalization of $\breve{I}$; $\vec{w}'_i$ was calculated with input $\vec{x}'$ and $\vec{x}' = E.\overrightarrow{\mathrm{Dec}}(\vec{x})$ by definition) that

$$\vec{x}_{\breve{I}'} \parallel (\vec{w}'_i)_{\breve{W}'_{c_i}(\breve{M},\breve{W}_i)} \xrightarrow{\text{sim}} \vec{\mathbf{v}}|_{\breve{M}'} \parallel (\vec{\mathbf{w}}_{\mathbf{i}})_{W_i}$$

By the item 2 of the ERPE of c_o (as $\breve{O}'$ an RPE (t,k)-normalization of $\breve{O}$, $\breve{M}'$ an RPE (t,k)-normalization of $\breve{M}$; $\vec{w}'_o$ was calculated with input $\vec{v}'$ and $\vec{v}' = E.\overrightarrow{\mathrm{Dec}}(\vec{\mathbf{v}})$)

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

Combining those two we obtain that

$$\begin{aligned}\vec{x}_{\breve{I}'} \parallel (\vec{w}'_i)_{\breve{W}'_{c_i}(\breve{M},\breve{W}_i)} \parallel (\vec{w}'_o)_{\breve{W}'_{c_o}(\breve{O},\breve{W}_o)} &\xrightarrow{\text{sim}} \vec{\mathbf{v}}_{\breve{M}'} \parallel (\vec{\mathbf{w}}_{\mathbf{i}})_{\breve{W}_i} \parallel (\vec{w}'_o)_{\breve{W}'_{c_o}(\breve{O},\breve{W}_o)} \\ &\xrightarrow{\text{sim}} \vec{\mathbf{y}}_{\breve{O}'} \parallel (\vec{\mathbf{w}}_{\mathbf{i}})_{\breve{W}_i} \parallel (\vec{\mathbf{w}}_{\mathbf{o}})_{\breve{W}_o}\end{aligned}$$

□

Lemma 34 (Link Between Dependencies)**.** *Given a circuit c' with n-shares encoding D', w the number of internal wires, and the compiled circuit c (with encoding $D := E \circ D'$, i inputs, o outputs) that is (t,e)-ERPE of c', given $\breve{I}_c$,$\breve{W}'_c$,$\breve{O}_c$ from the definition of ERPE, then for all $\breve{W} \subseteq [w]$*

$$\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\overrightarrow{\mathbf{Enc}}]}(\breve{W}) \subseteq \breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}'_{\mathbf{wires}} \circ D'.\overrightarrow{\mathbf{Enc}}]}(\breve{W}'_c(\emptyset,\breve{W}))$$

Proof. We know from the item 2 of the ERPE with $\breve{O} := \emptyset$, for all $\breve{I}$ an RPE (t,k)-normalization of $\breve{I}_c(\emptyset,\breve{W})$, for all $\vec{x} \in E.\overrightarrow{\mathbf{Enc}}[\mathbb{S}^i_{\text{in}}]$ and for all $\vec{r}' \in \breve{\mathrm{supp}}[\mathbf{Rnds}_{c'}()]$,

$$\vec{x}_{\breve{I}} \parallel \vec{c}'_{\text{wires}}(E.\overrightarrow{\mathrm{Dec}}(\vec{x}),\vec{r}')_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} \vec{c}_{\text{all}}(\vec{x},R.\overrightarrow{\mathbf{Enc}}(\vec{r}'))_{\breve{W} \parallel \breve{O}_c(\breve{\mathrm{I}},\breve{W},\breve{I})}$$

This implies that $\vec{x}_{\breve{I}} \parallel \vec{c}'_{\text{wires}}(E.\overrightarrow{\mathrm{Dec}}(\vec{x}),\vec{r}')_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} \vec{c}_{\text{wires}}(\vec{x},R.\overrightarrow{\mathbf{Enc}}(\vec{r}'))_{\breve{W}}$

We can choose $\vec{\mathbf{r}}' := \mathbf{Rnds}_{c'}()$ (and so $R.\overrightarrow{\mathbf{Enc}}(\vec{\mathbf{r}}') = \mathbf{Rnds}_c()$) and so $\forall \vec{x} \in E.\overrightarrow{\mathbf{Enc}}[\mathbb{S}^i_{\text{in}}]$

$$\vec{x}_{\breve{I}} \parallel \vec{\mathbf{c}}'_{\mathbf{wires}}(E.\overrightarrow{\mathrm{Dec}}(\vec{x}))_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} \vec{\mathbf{c}}_{\mathbf{wires}}(\vec{x})_{\breve{W}}$$

Then $\forall \vec{y} \in \mathbb{S}^i_{\text{in}}$ We can choose $\vec{\mathbf{x}} := E.\overrightarrow{\mathbf{Enc}}(\vec{y})$ (and so $E.\overrightarrow{\mathrm{Dec}}(\vec{\mathbf{x}}) := \vec{y}$) and we obtain

$$E.\overrightarrow{\mathbf{Enc}}(\vec{y})_{\breve{I}} \parallel \vec{\mathbf{c}}'_{\mathbf{wires}}(\vec{y})_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\overrightarrow{\mathbf{Enc}}(\vec{y}))_{\breve{W}}$$

As $\breve{I}$ an RPE (t,k)-normalization of $\breve{I}_c$ we have that $\breve{\mathrm{Dep}}_{[E.\overrightarrow{\mathbf{Enc}}]}(\breve{I}) = \emptyset$ i.e. $[] \xrightarrow{\text{sim}} E.\overrightarrow{\mathbf{Enc}}(\vec{y})_{\breve{I}}$. So, $\forall \vec{y} \in \mathbb{S}^i_{\text{in}}$ $\vec{\mathbf{c}}'_{\mathbf{wires}}(\vec{y})_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\overrightarrow{\mathbf{Enc}}(\vec{y})))_{\breve{W}}$

As the original circuit has encoding D', we can choose $\vec{\mathbf{y}} := D'.\overrightarrow{\mathbf{Enc}}(\vec{z})$, then

$$\vec{\mathbf{c}}'_{\mathbf{wires}}(D'.\overrightarrow{\mathbf{Enc}}(\vec{z}))_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} \vec{\mathbf{c}}_{\mathbf{wires}}(E.\overrightarrow{\mathbf{Enc}}(D'.\overrightarrow{\mathbf{Enc}}(\vec{z})))_{\breve{W}}$$

This is equivalent to $(\vec{\mathbf{c}}'_{\mathbf{wires}} \circ D'.\overrightarrow{\mathbf{Enc}})_{\breve{W}'_c(\emptyset,\breve{W})} \xrightarrow{\text{sim}} (\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\overrightarrow{\mathbf{Enc}})_{\breve{W}}$

As by definition the function $(\vec{\mathbf{c}}'_{\mathbf{wires}} \circ D'.\overrightarrow{\mathbf{Enc}})_{\breve{W}'}$ can be simulated using the inputs $\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}'_{\mathbf{wires}} \circ D'.\overrightarrow{\mathbf{Enc}}]}(\breve{W}')$, we obtain that the function $(\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\overrightarrow{\mathbf{Enc}})|_{\breve{W}}$ can be simulated using the inputs $\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}'_{\mathbf{wires}} \circ D'.\overrightarrow{\mathbf{Enc}}]}(\breve{W}'_g(\emptyset,\breve{W}))$. By definition, $\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\overrightarrow{\mathbf{Enc}}]}$ is minimal, and so $\breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\overrightarrow{\mathbf{Enc}}]}(\breve{W}) \subseteq \breve{\mathrm{Dep}}_{[\vec{\mathbf{c}}'_{\mathbf{wires}} \circ D'.\overrightarrow{\mathbf{Enc}}]}(\breve{W}'_g(\emptyset,\breve{W}))$ □

Lemma 35 (All outputs ERPE to RPR)**.** *Given a compiler (E, CC) such that for all circuits $c' \in C_{\text{in}}$ the compiled circuit $c := CC(c')$ is (t,e)-ERPE of c', then c is e-RPR.*

Proof. Let's call D' the encoding of c' and $D := E \circ D'$ the encoding of c, and $\breve{W}'_c$ from the definition of ERPE

To say that it's e-RPR is to say that given a circuit c, if c' is $(e(p), \varepsilon)$-SRPS, then c is (p, ε)-SRPS.

From the definition of the SRPS of c', $\Pr\left[\breve{\text{Dep}}_{[\vec{\mathbf{c}'}_{\mathbf{wires}} \circ D'.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leak}}_{c'}(e(p))) \neq \emptyset\right] \leq \varepsilon$

The item 1 of the ERPE says that $\breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p)) \overset{d}{\leq} \breve{\mathbf{Leak}}_{c'}(e(p))$ and as $\breve{W} \mapsto \breve{\text{Dep}}_{[\vec{\mathbf{c}'}_{\mathbf{wires}} \circ D'.\vec{\mathbf{Enc}}]}(\breve{W}) \neq \emptyset$ is monotone, we can apply the definition of $\overset{d}{\leq}$ and the hypothesis, and obtain that $\Pr\left[\breve{\text{Dep}}_{[\vec{\mathbf{c}'}_{\mathbf{wires}} \circ D'.\vec{\mathbf{Enc}}]}(\breve{W}'_c(\emptyset, \breve{\mathbf{Leak}}_c(p))) \neq \emptyset\right] \leq \varepsilon$

We know from Lemma 34 that for all $\breve{W} \subseteq [w]$

$$\breve{\text{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\vec{\mathbf{Enc}}]}(\breve{W}) \subseteq \breve{\text{Dep}}_{[\vec{\mathbf{c}'}_{\mathbf{wires}} \circ D'.\vec{\mathbf{Enc}}]}(\breve{W}'_c(\emptyset, \breve{W}))$$

and so we obtain that $\Pr\left[\breve{\text{Dep}}_{[\vec{\mathbf{c}}_{\mathbf{wires}} \circ D.\vec{\mathbf{Enc}}]}(\breve{\mathbf{Leak}}_c(p)) \neq \emptyset\right] \leq \varepsilon$. I.e. c is (p, ε)-SRPS. ☐

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,e)-ERPE of g, then the compiler $(E, CC_{\vec{G}})$ is e-RPR.

Proof. We first show that for all circuits $c \in C_{\text{in}}$, the compiled circuit $CC_{\vec{G}}(c)$ is (t,e)-ERPE of c. We can do this by induction. We know that for all gates g, $\vec{G}_g$ is (t,e)-ERPE of g. Also, if c is the identity circuit $CC_{\vec{G}}(c)$ is (t,e)-ERPE for Lemma 30, and same for the swap circuit and Lemma 31. The two induction steps are the composition in parallel and in series, and are proven respectively by Lemma 32 and Lemma 33. This means we can use Lemma 35 to show that the compiler $(E, CC_{\vec{G}})$ is e-RPR. ☐

E Proofs For the Main Compiler

We'll present in this appendix the proofs necessary to analyze our main compiler.

E.1 Security from the RPE

Proposition 1: The (t,e)-RPE implies the (t,e)-wRPE which implies the (t,e)-ERPE.

Proof. The first implication is true as the RPE satisfies the conditions of the wRPE, while we can prove the second implication by constructing the wRPE from the ERPE by only limiting it. We write the definition of Definition 30 and we call G the implementation, we limit it to deterministic fully-leakable gate g, so $\vec{r'} := []$, $R.\vec{\mathbf{Enc}}(\vec{r'}) := \vec{\mathbf{Rnds}}_G()$, and $\vec{\mathbf{g}}_{\mathbf{wires}}$ is the identity function. We then limit and rename $\breve{O}_G(\breve{O}, \breve{W}, \cdot) := \breve{O}'(\breve{O}, \breve{W})$. We define $\breve{W}'_G(\breve{O}, \breve{W}) := \breve{F}(\breve{O}, \breve{W})$ with $\breve{F}$ defined like in the RPE:

$$j \in \breve{F}(\breve{O}, \breve{W}) \iff \left|\breve{I}_G(\breve{O}, \breve{W}) \cap \breve{s}(j)\right| > t$$

Then we have that if $\breve{I}_G(\breve{O}, \breve{W}) \cap \breve{s}(j) \neq \breve{I} \cap \breve{s}(j)$ then $j \in \breve{F}(\breve{O}, \breve{W})$ and $|\breve{I} \cap \breve{s}(j)| = k$. Thanks to the completion property implicit in the definition of 'E has strength k' we obtain the following simulation

$$\vec{x}_{\breve{I}} \parallel E.\vec{\text{Dec}}(\vec{x})_{\breve{F}(\breve{O}, \breve{W})} \xrightarrow{\text{sim}} \vec{x}_{\breve{I}_G(\breve{O}, \breve{W})}$$

So we can use the right side as the basis for the simulation of item 2 instead of the left side, and so we can avoid the quantification over $\breve{I}$. The result is the definition of (t, e)-wRPE, which makes it a sufficient condition for the (t, e)-ERPE. □

Proposition 2: For the n-shares additive encoding, given a $t \in [0, n-1] \cap \mathbb{N}$, and any continuous monotone $e : [0, 1] \to [0, 1]$, we consider the gadget obtained by a parallel of at least $t+1$ random gates and the remaining to reach n are constant-0 gates. This gadget is (t, e)-ERPE for the random gate.

Proof. First of all, let's call G the gadget, and ℓ the number of randoms in G. G is correct for the random gate by using the ℓ-shares additive encoding for the randoms. Then we rewrite Lemma 4 with $i = 0$, $\breve{I} := \emptyset$, $\breve{I}' := \emptyset$, $\vec{x} := []$, with $o = 1$, $e(\cdot) = 0$ and then apply the definition of RPE $(t, n-1)$-normalization, and we obtain that G is $(t, 0)$-ERPE of the random gate iff

- for all $\breve{O} \subseteq [n]$ with $|\breve{O}| \leq t$, for all virtual randoms $\vec{r'} \in \operatorname{s\breve{u}pp}[\vec{\mathbf{Rnds}}_g()]$, we have that $[] \overset{\text{sim}}{\longrightarrow} \vec{G}_{\text{wires}}([], R.\vec{\mathbf{Enc}}(\vec{r'}))_{\breve{O}}$
- and there is a $j \in [n]$, such that for all virtual randoms $\vec{r'} \in \operatorname{s\breve{u}pp}[\vec{\mathbf{Rnds}}_g()]$, we have that $[] \overset{\text{sim}}{\longrightarrow} \vec{G}_{\text{wires}}([], R.\vec{\mathbf{Enc}}(\vec{r'}))_{[o]\setminus\{j\}}$.

In the first case we have $|\breve{O}| \leq t$, so the adversary obtains at most t randoms. As the encoding of the randoms has strength $\ell - 1 \geq (t+1) - 1 \geq t$, we have that $\vec{G}_{\text{wires}}([], R.\vec{\mathbf{Enc}}(\vec{r'}))|_{\breve{O}}$ can be simulated with no virtual random.

For the second case, we can chose $j := 1$ and one of the randoms is not selected, and so due to the additive encoding having strength $\ell - 1$, $\vec{G}_{\text{wires}}([], R.\vec{\mathbf{Enc}}(\vec{r'}))|_{[o]\setminus\{1\}}$ can be simulated with no virtual random, as required. □

E.2 Composition of Compiler Sequences

Lemma 5: Given a compiler sequence C and given two compilers I, O such that we can define $C'_m := O \circ C_m \circ I$ then C' has all the size amplification order of C.

Proof. By definition of circuit complexity matrix,

$$M_{C'_m} = M_O \cdot M_{C_m} \cdot M_I$$

And by the properties of the matrix norm induced by the vector p-norms,

$$\left\|M_{C'_m}\right\|_1 \leq \|M_O\|_1 \cdot \|M_{C_m}\|_1 \cdot \|M_I\|_1$$

Which means that given a generic λ, size amplification order of C

$$\left\|M_{C'_m}\right\|_1 \leq \|M_O\|_1 \cdot \|M_{C_m}\|_1 \cdot \|M_I\|_1 = \Theta\left(\|M_{C_m}\|_1\right) = \mathcal{O}\left(\lambda^m\right)$$

which means that λ is also a size amplification order for C' □

Lemma 6: Given a compiler sequence C and given a compiler I such that we can define $C'_n := C_n \circ I$ and such that I is $x^{\Omega(1)}$-RPR then C' has all the tolerated leakage, security amplification order, size amplification order and expansion exponent of C.

Proof. First of all, if I is $x^{\Omega(1)}$-RPR then there is an f such that I is f-RPR with $f(x) = x^{\Omega(1)}$ as $x \to 0$. This means that $\log_2 f(x) = \Omega(1) \log_2 x = \Omega(\log_2 x)$. Let's consider a generic P tolerated leakage of C and relative security amplification order d. Then there is a sequence of functions e_n such that C_n is e_n-RPR and $\log_2 e_n(P) = \Omega(d^n)$, and so $\log_2 f(e_n(P)) = \Omega\left(\log_2 e_n(P)\right) = \Omega\left(\Omega\left(d^n\right)\right) = \Omega\left(d^n\right)$. As C'_i is $(f \circ e_i)$-RPR by Lemma 2, then P is a tolerated leakage of C', and d is a relative security amplification order. By Lemma 5 also the size amplification order is preserved, and so is the expansion exponent. □

Lemma 7: Given a compiler sequence I and a compiler sequence O (with respectively a tolerated leakage of P_i, P_o; and security amplification order of d_i, d_o for the aforementioned tolerated leakage) then there is a k such that $C_n := O_k \circ I_n$ has a security amplification order of d_i relative to a tolerated leakage of P_o.

Proof. Let's define e^i the e from the definition of d_i and of P_i, then I_n is e^i_n-RPR and $\log_2 e^i_n(P_i) = \Omega\left(d_i^n\right)$. Let's also define e^o the e from the definition of d_o and of P_i, then O_n is e^o_n-RPR, and $\log_2 e^o_n(P_o) = \Omega\left(d_o^n\right)$. This means that there is a k such that $e^o_k(P_o) \leq P_i$, and for that k (as e^i_n is monotone) we have that $\log_2 e^i_n(e^o_k(P_o)) \leq \log_2 e^i_n(P_i) = \Omega\left(d_i^n\right)$, which proves the thesis as $O_k \circ I_n$ is $(e^i_n \circ e^o_k)$-RPR per Lemma 2. □

E.3 Proofs For the Classic Expansion

Lemma 8: 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.

Proof. We will consider the circuit complexity matrix M_X, which is a square matrix with elements in $\Re$. If it's diagonalizable in $\mathbb{C}$, then there are the matrices P, J such that $M_X = P^{-1} \cdot J \cdot P$, where J is a diagonal matrix with the values λ_i which are the eigenvalues of M_X, and λ_1 is the highest eigenvalue in module.

Additionally, from [HJ85] we know that

$$\|M_{C_n}\|_1 = \|M_X^n\|_1 = \left\|P^{-1} J^n P\right\|_1 \leq \left\|P^{-1}\right\|_1 \|J\|_1^n \|P\|_1$$

And that $\|J\|_1 = \max_{j \in [n]} \sum_{i=1}^{n} |J_{i,j}| = \max_{j \in [n]} |\lambda_j| = |\lambda_1|$

Which means that $\|M_{C_n}\|_1 = \mathcal{O}\left(|\lambda_1|^n\right)$, and so $|\lambda_1|$ is a size amplification order. □

Lemma 9: 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.

Proof. From Theorem 3 we know that X is e-RPR with $e(p) = \max_i e_i(p)$, where e_i is taken from the definition of t-RPE-tolerated leakage, and is such that for all $p \in (0, P_i]$, $e_i(p) < p$.

This also means that for all $p \in (0, P]$, $e(p) < p$.

As by definition of RPR e is continuous, then for every $\epsilon \in (0, P)$, we can apply the extreme value theorem to $p \mapsto e(p) - p$ with $p \in [\epsilon, P]$ and because for all $p \in (0, P]$ we have that $e(p) - p < 0$ by hypothesis, then we obtain that $M := \max_{p \in [\epsilon, P]} e(p) - p < 0$. This means that for all $p \in [\epsilon, P]$ we have that $e(p) \leq M + p$.

By applying this iteratively we obtain that $e^m(p) \leq M \cdot m + p$ as long as $e^{m-1}(p) \geq \epsilon$. If instead $e^{m-1}(p) < \epsilon$ then $e^m(p) < e^{m-1}(p) < \epsilon$. This means that for every $\epsilon \in (0, P)$ eventually as $m \to \infty$ we have that $e^m(P) < \epsilon$.

This by definition means that $e^m(P) \to 0$ as $m \to \infty$. Additionally from Corollary 1 we have that the compiler C_m is e^m-RPR, this means by definition that P is a tolerated leakage. □

Lemma 10: Given a compiler sequence $C_n := X^n$ if d is a security amplification order for some tolerated leakage, then it's a security amplification order for any tolerated leakage.

Proof. This is an immediate consequence of Lemma 7: given a security amplification order d of C and a tolerated leakage P of C there is a k such that eventually as $n \to \infty$, $C_n = C_k \circ C_{n-k}$ and such that C has that security amplification order d relative to the tolerated leakage P. □

Lemma 11: 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 tolerated leakage.

Proof. Thanks to Theorem 3, X is e-RPR with $e(p) := \max e_i(p)$, the e_i taken from the definition of t-RPE-amplification order. Then as $e_i(p) = \Theta\left(p^{d_i}\right) = \mathcal{O}\left(p^d\right)$ we have that $e(p) = \mathcal{O}\left(p^d\right)$.

By definition of $e(x) = \mathcal{O}\left(x^d\right)$ we know that there is a $x' < 1$ such that for all $0 < x < x'$ we have that $e(x)/x^d$ is upper bounded by some constant c'. Then for $c := \max\{c'^{1/d}, 1/x'\}$ we have that $f(x) := min(1, (cx)^d)$ is such that $e(x) \leq f(x)$. This is proven for $x = 0$ as $e(x) = 0 = f(x)$ as e is continuous, below x' as $e(x) \leq c'x^d \leq (cx)^d$ and above x' as $e(x) \leq 1 \leq (cx')^d \leq (cx)^d$.

As $e(x) \leq f(x)$, we can use Lemma 25 and obtain that X is f-RPR. Thanks to Corollary 1 the compiler C_n is f^n-RPR.

We can split f and write it as $f(x) = min(1, f'(x))$ and $f'(x) := (cx)^d$. As f' maps a value ≥ 1 to one ≥ 1, we have that $f^n(x) = min(1, f'^n(x))$.

We then have by induction that $f'^n(x) = x^{d^n} \prod_{i=1}^{n} c^{d^i}$. This because $f'^0(x) := x$ and

$$f'(f'^n(x)) = (cx^{d^n} \prod_{i=1}^{n} c^{d^i})^d = x^{d^n d} c^d \prod_{i=1}^{n} c^{d^i d} = x^{d^{n+1}} \prod_{i=1}^{n+1} c^{d^i}$$

As $\sum_{i=1}^{n} d^{i-n} = \sum_{i=0}^{n-1} d^{-i} \leq \sum_{i=0}^{\infty} d^{-i} = 1/(1 - 1/d) = d/(d-1)$, then

$$f'^n(x) = x^{d^n} \prod_{i=1}^{n} c^{d^i} = (c^{\sum_{i=1}^{n} d^{i-n}} x)^{d^n} \leq (c^{d/(d-1)} x)^{d^n}$$

This means C_n is e_n-RPR with $e_n(x) := min(1, c^{d/(d-1)}x)^{d^n}$ by Lemma 25. This means that for any $P < c^{-d/(d-1)}$ we have that $\log_2 e_n(P) = d^n \log_2(c^{d/(d-1)}P) \leq 0$ which means that $\log_2 e_n(P) = \Theta\left(d^n\right)$ which means P is a tolerated leakage and d is a security amplification order for P. We can apply Lemma 10 and we obtain that d is a security amplification order for any tolerated leakage. □

E.4 Gadgets Without Strength for Fully-leakable Deterministic Gates

Lemma 12: 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 $(0, e)$-ERPE, with $e(x) := \min\{1, (w \cdot x)^{1/i}\}$.

Proof. To prove that we can prove that it's $(0, e)$-wRPE and use Proposition 1. As the encoding has strength 0, in the definition of wRPE we have that $t, k := 0$, and so

$\breve{O}'(\cdot,\cdot) := \emptyset$ as $\emptyset$ is the only RPE $(0,0)$-normalization of anything. This makes the second point automatically true.

We can then chose $\breve{I}(\breve{W})$ to $\emptyset$ if $\breve{W} = \emptyset$ and $[i \cdot n]$ otherwise. Those obviously satisfy the first point as the internal wires $\breve{W} = \emptyset$ can be simulated with no input and no randoms, while all the others internal wires can always be simulated using all the inputs and randoms.

As per the third point, the previous assignment make $\breve{F}(\breve{O}, \breve{W})$ equal to $\emptyset$ if $\breve{W} = \emptyset$ and $[i]$ otherwise, which means that we also satisfy the requirement that the distribution of $\breve{F}(\breve{O}, \mathbf{Le\breve{a}k}_G(p))$ is the same for all $\breve{O}$.

What we need to prove is that for every leakage probability vector $p \in [0,1]$, the failure events must have the probability distribution $\breve{F}(\emptyset, \mathbf{Le\breve{a}k}_G(p)) \overset{d}{\leq} \mathbf{Le\breve{a}k}_g(e(p))$.

Let's call $\breve{\mathbf{v}}(p)$ the probabilistic function that returns a random variable that is $\emptyset$ with probability $(1-p)^w$ and is $[i]$ with probability $1-(1-p)^w$. Then $\breve{F}(\emptyset, \mathbf{Le\breve{a}k}_G(p)) \overset{d}{=} \breve{\mathbf{v}}(p)$, so we need that for all monotone P and for all $p \in [0,1]$

$$\Pr[P(\breve{\mathbf{v}}(p))] \leq \Pr\Big[P(\mathbf{Le\breve{a}k}_g(e(p)))\Big]$$

If we exclude the constant P (which obviously satisfy the inequality), then all monotone non-constant P have $P(\emptyset) = \mathit{false}$, $P([i]) = \mathit{true}$. We can then define the monotone non-constant predicate $P'(\breve{U}) := (\breve{U} = [i])$, and we obtain that for $\breve{U} \neq [i]$ we have that $P'(\breve{U}) = \mathit{false} \leq P(\breve{U})$. This means that

$$\Pr\Big[P'(\mathbf{Le\breve{a}k}_g(e(p)))\Big] \leq \Pr\Big[P(\mathbf{Le\breve{a}k}_g(e(p)))\Big]$$

And as $\breve{\mathbf{v}}(p)$ only has the values $\emptyset$ and $[i]$, $\Pr[P(\breve{\mathbf{v}}(p))] = \Pr[P'(\breve{\mathbf{v}}(p))]$. Those two together mean that we can prove the lemma by showing that for every $p \in [0,1]$

$$\Pr[P'(\breve{\mathbf{v}}(p))] \leq \Pr\Big[P'(\mathbf{Le\breve{a}k}_g(e(p)))\Big]$$

I.e. $1-(1-p)^w \leq e(p)^i$. If G has no wires ($w = 0$) then is $0 \leq e(p)^i = 0$. Otherwise $w \geq 1$, $p \in [0,1]$ so we can use Bernulli's inequality, which that states that $pw \geq 1-(1-p)^w$, which implies that $1-(1-p)^w \leq \min\{1, pw\}$. The thesis is proven as

$$\min\{1, wp\} = \min\{1^i, ((wp)^{1/i})^i\} = \min\{1, (wp)^{1/i}\}^i = e(p)^i$$

$\square$

References

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with o(1/ log(n)) leakage rate. Cryptology ePrint Archive, Paper 2016/173, 2016. https://eprint.iacr.org/2016/173.

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. Cryptology ePrint Archive, Paper 2018/566, 2018. https://eprint.iacr.org/2018/566.

[Ajt11] Miklós Ajtai. Secure computation with information leaking to an adversary. *Electron. Colloquium Comput. Complex.*, TR11, 2011.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. *IACR Cryptol. ePrint Arch.*, 2016:211, 2016.

[BCP$^+$20] Sonia Belaïd, Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Abdul Rahman Taleb. Random probing security: Verification, composition, expansion and new constructions. Cryptology ePrint Archive, Paper 2020/786, 2020. https://eprint.iacr.org/2020/786.

[BRT21] Sonia Belaïd, Matthieu Rivain, and Abdul Rahman Taleb. On the power of expansion: More efficient constructions in the random probing model. Cryptology ePrint Archive, Paper 2021/434, 2021. https://eprint.iacr.org/2021/434.

[BRTV21] Sonia Belaïd, Matthieu Rivain, Abdul Rahman Taleb, and Damien Vergnaud. Dynamic random probing expansion with quasi linear asymptotic complexity. Cryptology ePrint Archive, Paper 2021/1455, 2021. https://eprint.iacr.org/2021/1455.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, 1999.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: from probing attacks to noisy leakage. Cryptology ePrint Archive, Paper 2014/079, 2014. https://eprint.iacr.org/2014/079.

[GJR18] Dahmun Goudarzi, Antoine Joux, and Matthieu Rivain. How to securely compute with noisy leakage in quasilinear complexity. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 547–574, Cham, 2018. Springer International Publishing.

[GPRV22] Dahmun Goudarzi, Thomas Prest, Matthieu Rivain, and Damien Vergnaud. Probing security through input-output separation and revisited quasilinear masking. Cryptology ePrint Archive, Paper 2022/045, 2022. https://eprint.iacr.org/2022/045.

[HJ85] Roger A. Horn and Charles R. Johnson. *Norms for vectors and matrices*, page 257–342. Cambridge University Press, 1985.

[HU05] Dennis Hofheinz and Dominique Unruh. On the notion of statistical security in simulatability definitions. Cryptology ePrint Archive, Paper 2005/032, 2005. https://eprint.iacr.org/2005/032.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual International Cryptology Conference*, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, 1996.

[MT10] Ueli Maurer and Stefano Tessaro. A hardcore lemma for computational indistinguishability: Security amplification for arbitrarily weak prgs with optimal stretch. In *Theory of Cryptography Conference*, 2010.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a rényi day. Cryptology ePrint Archive, Paper 2019/138, 2019. https://eprint.iacr.org/2019/138.

[PR13a] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *International Conference on the Theory and Application of Cryptographic Techniques*, 2013.

[PR13b] 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*, pages 142–159, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Research in Smart Cards*, 2001.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. Cryptology ePrint Archive, Paper 2010/441, 2010. https://eprint.iacr.org/2010/441.