Protecting Circuits from Computationally Bounded and Noisy Leakage [*]

Sebastian Faust[†] EPFL

Tal Rabin IBM Research

Leonid Reyzin[‡] Boston University

Eran Tromer[§] Tel-Aviv University

Vinod Vaikuntanathan[¶] MIT

June 30, 2014

Abstract

Physical computational devices leak side-channel information that may, and often does, reveal secret internal states. We present a *general* transformation that compiles any circuit into a circuit with the same functionality but resilience against well-defined classes of leakage. Our construction requires a *small*, *stateless* and *computation-independent* leak-proof component that draws random elements from a fixed distribution. In essence, we reduce the problem of shielding arbitrarily complex circuits to the problem of shielding a single, simple component.

Our approach is based on modeling the adversary as a powerful observer that inspects the device via a limited measurement apparatus. We allow the apparatus to access all the bits of the computation (except those inside the leak-proof component), and the amount of leaked information to grow unbounded over time. However, we assume that the apparatus is limited in the amount of output bits per iteration and the ability to decode certain linear encodings. While our results apply in general to such leakage classes, in particular, we obtain security against:

- *Constant-depth circuits leakage*, where the leakage function is computed by an AC^0 circuit (composed of NOT gates and unbounded fan-in AND and OR gates).
- *Noisy leakage*, where the leakage function reveals all the bits of the internal state of the circuit, perturbed by independent binomial noise. Namely, for some number $p \in (0, 1/2]$, each bit of the computation is flipped with probability p, and remains unchanged with probability $1 - p$.

[*]A preliminary version of this work appears in [15]. This full version will appear in SIAM Journal on Computing.

[†]This work was partially done while Sebastian Faust was at KU Leuven and Aarhus University. Sebastian Faust received funding from the Marie Curie IEF/FP7 project GAPS, grant number: 626467.

[‡]Supported in part by NSF grants 0546614, 0831281, 1012910, and 1012798.

[§]This work was done while Eran Tromer was at MIT and Microsoft Research. Eran Tromer was supported, in part, by NSF CyberTrust grant CNS-0808907 and AFRL grant FA8750-08-1-0088. Views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFRL, NSF, the U.S. Government or any of its agencies.

[¶]This work was done while Vinod Vaikuntanathan was at IBM Research, Microsoft Research, and University of Toronto, and was supported, in part, by a Josef Raviv Postdoctoral Fellowship.

1 Introduction

The best of cryptographic algorithms are insecure when their implementations inadvertently reveal secrets to an eavesdropping adversary. Even when the software is flawless, practical computational devices leak information via numerous side channels, including electromagnetic radiation (visible and otherwise) [38, 29], timing [7], power consumption [28], acoustic emanations [41], and numerous effects at the system architecture level (e.g., cache attacks [5, 34, 35]). Leaked information is even more easily accessible when the computational device is at the hands of an adversary, as is often the case for many modern devices such as smart-cards, TPM chips and (potentially stolen) mobile phones and laptops. Reducing such information leakage has proven excruciatingly difficult and costly, and its complete elimination is nowhere in sight.

Micali and Reyzin [30] proposed a general model for rigorously analyzing protection against side-channel attacks. They model a side-channel attacker as a two part entity – the first is the *measurement apparatus* that performs measurements on the physical state of the device. This is done on behalf of the second entity which is the *adversarial observer*. The observer is assumed to be computationally powerful (e.g., polynomial-time or even unbounded), and takes as input the measurements of the apparatus. Thus, the power of the adversarial observer is primarily constrained by the quality of the information provided by the measurement apparatus.

It is interesting to note that even though computational devices leak abundantly, many side channel attacks are hard to carry out and some devices remain unbroken. This is due to the fact that *useful* measurements can often be difficult to realize in practice. Physical measurement apparatuses often produce a "computationally limited" and "noisy" measurement of the state of the object: they usually do not carry out sophisticated computation. In-depth analysis typically happens in the form of post-processing by the observer (rather than in the measurement apparatus).

In this work, we follow the paradigm of Ishai, Sahai, and Wagner [24], who construct a *general transformation* from any cryptographic algorithm into one that is functionally equivalent, but also leakage-resilient. The particular class of leakage functions they consider is the class of spatially local measurement functions, namely functions that read and output at most k bits of information. In particular, the leakage functions are completely oblivious to a large portion of the circuit's state.

In contrast, we are interested in security against global measurements, which are often easier to carry out than localized measurements that require a focus on specific wires or memory cells; in many side-channel attacks, the main practical difficulty for the attacker lies precisely in obtaining high spatial resolution and accuracy. Furthermore, global measurements are typically also more informative than local measurements. The question that motivates our work is whether, analogously to [24], we can construct a general circuit transformation that tolerates *global* side-channel measurements.

1.1 The Model

Inspired by Ishai et al. [24], we model arbitrary cryptographic computation by circuits computing over a finite field $\mathcal{K}$ (a special case are Boolean circuits that operate over $\mathsf{GF}(2)$). Since cryptographic devices are often implemented as digital circuits, our model of computation directly corresponds to devices in the physical world. Importantly, the circuit has a secret state (such as a cryptographic key for a block cipher), which we want to protect from the adversary.

Our adversarial model considers an adversary who attacks this circuit by adaptively running it on inputs of her choice and learning the result of its computation. The result is computed from the input and the state (which may be updated by the computation). With each query to the circuit, the adversary may choose a leakage function f from some set $\mathcal{L}$ of tolerated leakage functions (this choice, as well as the choice of input to the circuit, may be made adaptively, depending on all the information obtained by the adversary up to this point). We allow f to depend on the secret state and all intermediate results that may occur during computation. We model this by giving as input to f the values that are carried on *all* the wires connecting the circuit's gates. Since we do not make any assumption on the spatial locality of the leakage, we achieve resilience to *global* leakage functions. However, the set $\mathcal{L}$ of leakage functions we can handle is restricted in other ways, as we discuss below.

On computationally weak and noisy leakages. We consider two classes of leakage functions in this work.

- *Computationally-bounded leakage functions:* We assume that $\mathcal{L}$ contains only simple aggregated leakage functions. That is, functions in $\mathcal{L}$ are too weak to perform certain computations (e.g., parity of many bits) *and* their range is bounded by some parameter λ. As a concrete example, we consider functions in the class AC^0—that is, functions that can be computed by circuits of small depth[1]. Some restriction on the computational power of the leakage function is necessary, because of the impossibility results on general circuit obfuscation [3] (see [21, Sections 1 and 1.3] for a discussion of the connection between leakage-resilience and obfuscation). The second restriction—bounding the amount of leakage—is similarly necessary, as otherwise even very simple leakage functions can output the complete secret state (e.g., the identity function). Notice, however, that while the amount of information leaked each time is bounded, the total amount of information leaked over multiple runs of the circuit is not.

- *Noisy leakage functions:* Alternatively, instead of limiting the computational power and output size of f, we assume that $\mathcal{L}$ consists only of noisy functions. That is, functions in $\mathcal{L}$ do not compute exact results, but rather their output is perturbed with some noise. As a concrete example, we assume that the leakage reveals all the bits of the circuit's state, perturbed by independent binomial noise. Notice that in this case the amount of leakage for each run of the circuit is not bounded.

The restriction of computationally bounded and noisy leakages are motivated by the observation that physical measurement apparatuses typically produce a computationally simple or noisy measurement of the device's state. For instance, the power consumption of a cryptographic devices is often described as a simple aggregated functions such as the Hamming weight or the Hamming distance of the processed data. Also, physical measurements are inherently noisy. Such noise may result from the environment (non-algorithmic noise) or from the device itself (algorithmic noise). Thus we model the measurement apparatus as a simple (possibly noisy) aggregated function of the device's state. We stress that once the leakage is in the hands of the adversary, she can carry out arbitrary computation (it need not even be efficient).

Of course, the concrete leakage classes for which we present secure circuit transformations are not particularly strong. For instance, in the computationally bounded setting we instantiate $\mathcal{L}$ with AC^0. AC^0 is weak since it even cannot compute parity and hence many practical side-channel

[1]An earlier version of this paper contained an erroneous claim that our results extend to $\mathsf{ACC}^0[p]$, which is AC^0 augmented with $\mathtt{MOD}_p$ gates. They do not, because average-case hardness results are not known for such circuits.

attacks do not fall into this class (e.g., the Hamming weight of all wires). Nevertheless it is strong enough to allow for measuring approximate Hamming weight [1], or the Hamming weight of a subset of the values on the wires: something routinely measured by side-channel attacks in practice.

For the noisy setting we make the assumption that the value on each wire is perturbed by *independent* noise. This assumption may not hold in practice since, e.g., algorithmic noise is correlated with the processed data. Constructing circuits that remain secure against correlated noisy leakages is an important research question.

Inter-gate wires vs. all of the circuit Formally, we model global leakage functions by giving the values on all wires that connect gates as input to a single leakage function. One may object that this input to the leakage function does not capture all of the physical information present in the circuit, because we do not include the internals of the gates. For instance, we do not give as input the number of conducting transistors inside the gate, while in practice such low-level effects may significantly contribute to the physical leakage [42]. In the case of computationally weak leakage functions, this is not a real limitation, because, although our leakage functions are computationally weak, they suffice to evaluate constant-size operations (e.g., count the number of conducting transistors in a Boolean gate). This allows our leakage functions to simulate the internal behavior of a gate just by knowing its inputs, as long as this behavior is not too complex.

On leak-free components Most gates in our protected circuits are, indeed, very simple constant-size objects. However, we introduce a special component, a so-called *opaque gate*. We assume such gates are *leak-free*. That is, they leak on their inputs and outputs, but the leakage function cannot observe their internals. Naturally, in order for this assumption to be meaningful, an opaque gate must be complex enough so that its internals are not simulatable by a leakage function that gets information only about the wires connected to this gate. In particular, our opaque gates are not constant-size.

However, even though we assume non-constant size leak-free components, this assumption is mitigated by the fact that, in our constructions, these components are *simple, stateless and computation-independent*. In particular, the complexity of implementing our leak-free component is independent of the complexity of the computed circuit, and they neither hold secrets nor maintains state.

We have two kinds of opaque gates, denoted $\mathcal{O}$ (which is used for both constructions) and $\mathcal{Q}$ (which is used only for the noisy leakage construction). They have no inputs. They output an element sampled according to a fixed distribution, which is independent of the computation being carried out. The $\mathcal{O}$ gate, described in more detailed in Section 3.2, samples k uniformly random field elements subject to the condition that they add up to zero (in the case of the field being $\mathsf{GF}(2)$, it simply means sampling a bit string of even parity). The $\mathcal{Q}$ gate, described in more detail in Section 3.3, is more complex: it samples many such k-tuples of field elements and then computes a degree-two polynomial on them.

Although the requirement of a leak-free component is a strong one, the leak-free components we require are minimal in some respects:

1. It is a fixed standardized functionality which can be designed and validated once and added to one's standard cell library — which is far better than having to devise separate protection mechanisms for every circuit of interest.

2. It has no secret keys, no inputs and no internal state, i.e., it is independent of the computation in the circuit and merely samples from a distribution. While the assumption of a shielded physical device that samples perfect randomness is a strong one, we can relax it. For the case of AC^0 leakage we can relax this assumption and require only randomness that is $\mathsf{polylog}(k)$ independent [6]. Also, alternatively, we can de-randomize our construction by using Nisan's unconditional pseudo-random generator against AC^0 [33]. Notice that such a PRG still has to be implemented by a leak-free component, and hence, in this case, our opaque gates become *stateful*, which requires building leak-free memory.

3. Alternatively, because we only need samples from a distribution, we can have the opaque "gate" simply read its output one by one from a pre-computed list. Thus, it suffices to have leak-proof one-time storage (a consumable "tape roll") instead of leak-proof computation. This is an option if the computation is performed only a bounded number of times.

Of course, leak-free components, however simple, are a limitation. The work of Ishai et al. [24], which protects against leakage of some wire values, does not utilize these components. Furthermore, subsequent work (see Section 1.3) eliminated the need for these components even in cases of more complex leakage. On the other hand, many variations of the leak-free component assumption have been made in the literature. We highlight some of these works below; which ones are more or less realistic is debatable.

The model of Micali and Reyzin [30] and many subsequent works (e.g., [13, 36, 14]) assume the presence of leak-free memory. This is captured by the statement that "only computation leaks information" (axiom 1 in [30]), i.e., memory that is not accessed during a computation step does not affect the observable leakage during that step.

The "Oblivious RAM" model of Goldreich and Ostrovsky [17, 19] reverses the roles: while memory is leaky, the computation is assumed to be on a leak-free secure processor. In this model, they show a generic transformation that makes random-access machines resilient to polynomial-time leakage (with polylogarithmic blowup in memory size and running time). We also note that in the recent circuit compilers of Juma and Vahlis [26], Goldwasser and Rothblum [20], and Dziembowski and Faust [11, 12] leak-free components have been used (they have been very recently eliminated in [21]). We discuss these works in more detail in Section 1.3.

1.2 Our Results

We demonstrate *circuit transformations* that compile any circuit into one that is functionally equivalent but resists attacks by leakage functions described above. The circuit transformation TR takes as input the description of a circuit C and compiles it into a *transformed circuit* $\widehat{C}$ that uses the same gates as C (plus the leak-free component). For example, C may be a standard block-cipher, in which case $\widehat{C}$ would be its secure implementation (note that the entire block cipher does not have to be computed with a single combinatorial circuit—because we allow the circuit to maintain state, C can be a clocked circuit with registers).

We define resilience of $\widehat{C}$ against leakage class $\mathcal{L}$ by a simulation-based security notion, which is essentially the same as the one in [24]. A circuit transformation TR is said to be resilient to leakages of $\mathcal{L}$ if observing the computation of $\widehat{C}$ with leakages from $\mathcal{L}$ does not offer any advantage over black-box access to C without any leakage. Formally, we show that for any adversary $\mathcal{A}$ that gets to interact with $\widehat{C}$ by giving it inputs, observing some physical leakage $f \in \mathcal{L}$ from the computation on those inputs, and viewing its outputs, there exists a simulator $\mathcal{S}$ with only black-box access to C such that $\mathcal{A}$ and $\mathcal{S}$ have indistinguishable output distributions. Let us now discuss our constructions in more detail.

Leakage resilience from linear secret sharing. Our constructions, at a high level, are similar to the construction of Ishai et al. [24]: we perform a gate-by-gate transformation. Every wire of C is encoded into a bundle of wires in $\widehat{C}$ using a linear secret sharing scheme Π; each wire of the bundle carries a share of the original wire in C. Each gate in C is replaced by a *gadget* in $\widehat{C}$ which operates on encoded bundles. The gadgets are carefully constructed to use Π internally in a way that looks "essentially random" to functions in the leakage class $\mathcal{L}$ (notice that the internals of these gadgets may leak), and we show that this implies that the whole content of the transformed circuit remains "essentially random" to functions in $\mathcal{L}$. Hence, the adversary gets no advantage from her observation of the leakage. We provide two constructions: one resilient against computationally weak leakage, and the other resilient against noisy leakage.

Resilience to computationally weak leakage. The security of our first construction relies on an assumption on the computational power of $\mathcal{L}$: namely, that it cannot decode Π (even when the output of a function in $\mathcal{L}$ is then processed by a computationally unbounded adversary; see Definition 2 in Section 3.2). This is general for any Π and $\mathcal{L}$ that cannot decode Π. If k is the size of the leakage-resilient encoding Π of a single bit, then our transformation increases the circuit size by a factor of $O(k^2)$. The following is an informal statement of the result (see Theorem 1 for the formal version).

Informal Theorem 1. *Let $\mathcal{K}$ be a finite field and let k be a security parameter. Let $\mathcal{L}$ be some class of leakage functions and let $\mathcal{L}_\Pi$ be the class of functions obtained by adding, to every function in $\mathcal{L}$, any $O(k^2)$ field operations in $\mathcal{K}$, arranged at most 3 layers deep. Suppose the encoding Π is such that the adversary, using leakage functions from $\mathcal{L}_\Pi$ twice, cannot distinguish what field element is encoded. There is a compiler that transforms any circuit over $\mathcal{K}$ into one that is secure against any adversary that executes the transformed circuit a polynomial number of times, using a leakage function from $\mathcal{L}$ at each execution. The compiled circuit uses $O(sk)$ copies of a circuit-independent, input-independent, stateless, randomized leak-free gate. The compiler increases the circuit size by a factor of $O(k^2)$.*

As a special case, to get secure circuit transformations for concrete leakage classes, we can invoke circuit lower bounds [16, 22]. Thus, for the case where the scheme Π is instantiated with the *parity encoding* (i.e., a bit b is shared into random bits whose parity is b), and the leakage class $\mathcal{L} = \mathsf{AC}^0$, the lower bound of Hastad [22] implies that to functions in $\mathcal{L}$ the encoded bit b looks essentially random. The following is an informal statement of the result (see Corollary 3 and discussion thereafter for formal version).

Informal Theorem 2. *Let k be a security parameter. Let $0 < \delta < 1$ and $4 < d < 1/\delta - 1$ be some constants. Let $\mathcal{L}$ be the class of AC0 circuits of depth $d - 4$, consisting of at most $\exp(O(k^{(1-\delta)/d}))$ unlimited-fan-in and-or-not gates, and at most $\lfloor k^\delta/2 \rfloor$ output values. There is a compiler that transforms any boolean circuit of size s into one that is secure against any adversary that executes the transformed circuit at most q times, using a leakage function from $\mathcal{L}$ at each execution. The resulting security is information theoretic, with security level $qs\exp(-\Omega(k^{(1-\delta)/d}))$. The compiled* *circuit uses $O(sk)$ copies of a circuit-independent, input-independent, stateless, randomized leak-free gate. The compiler increases the circuit size by a factor of $O(k^2)$.*

In particular, to be resilient to λ bits of leakage by AC0 leakage functions of depth d_λ and size $\exp(O(\lambda))$*, we can use $k = (2\lambda)^{d_\lambda+6}$ and achieve security level qs* $\exp(-\Omega(\lambda))$.

Resilience to noisy leakage. The security of our second construction relies on the assumption that the functions in $\mathcal{L}$ output wires of $\widehat{C}$ perturbed by independent binomial noise. In this case, we can consider $\mathcal{L}$ that consists of just a single leakage function. Namely, if w_i is the value on wire i, the leakage function $\mathcal{N}_p$ outputs $w_i \oplus \eta_i$, where η_i is 1 with some probability $p > 0$. For such leakages the parity encoding works again. By the XOR Lemma (see Lemma 3 in Section 4.1), $\mathcal{N}_p(\vec{a})$ and $\mathcal{N}_p(\vec{b})$ are statistically close when $\vec{a}$ and $\vec{b}$ are random bit strings of different parity. Similar to the construction for computationally weak leakages, we show that computing with such encodings yields secure computation against $\mathcal{N}_p$ leakages. This transformation increases the circuit size by a factor that is polynomial in $1/p$. The following is an informal statement of the result (see Theorem 2 for the formal version).

Informal Theorem 3. *Let k be a security parameter. There is a compiler that transforms any boolean circuit of size s into one that is secure against any adversary that executes the transformed circuit q times using the leakage function $\mathcal{N}_p$ (which gives each wire value correctly with probability $1 - p$, as defined above) at each execution. The resulting security is information-theoretic, with security level qs* $\exp(-\Omega(kp^6))$*. The compiled circuit uses two kinds of circuit-independent, input-independent, stateless, randomized leak-free gates ($O(sk)$ copies of one and $O(s)$ copies of the other). The compiler increases the circuit size by a factor of $O(k^2)$.*

Our opaque gates. As already outlined above, our constructions make an extra requirement: $\widehat{C}$ uses a small leak-free component $\mathcal{O}$ (used in all constructions) and/or $\mathcal{Q}$ (used in the case of noisy leakages). $\mathcal{O}$ merely outputs samples from a fixed distribution, namely the encoding of 0 under Π (if Π is the parity encoding then $\mathcal{O}$ just outputs random bit strings with parity 0). $\mathcal{Q}$ is more complicated and will be described in Section 3.3. Thus, our results can be interpreted as reducing the physical security of *arbitrary* circuits to the security of a *single simple* component, which can be manufactured in bulk, independently of the larger circuits that need protection. This approach to physical security follows the approach of Micali and Reyzin [30] of reductions between physical devices.

Security proof via general composition. Let us call circuit transformations that replace wires by wire bundles carrying encoded values and gates by gadgets that operate on these bundles *encoding-based.* We show a general technique for proving leakage resilience of encoding-based circuit transformations. Namely, we capture a strong notion of leakage-resilience for transformed circuits (or their gadgets), by saying that they are *reconstructible* if there exist certain simulators (so called *reconstructors*) for the internal wires that fool the leakage class. More precisely, a reconstructor is given only the encoded inputs and outputs of a transformed circuit (or gadget) and has to come up with values for the internal wires that look consistent for leakages in $\mathcal{L}$. We then show a *composition result*: if all parts of a circuit are reconstructible then so is the whole circuit. This will imply security of the transformation. An important contribution of this work are our proof techniques that are particularly useful for computationally weak leakage functions, and have recently been used in [12]. We refer the reader to Section 4.2 for an outline of the proof techniques.

1.3 Related Work

Leakage-resilient cryptography is a very active research area; in particular, there is a considerable amount of recent work that proposes techniques to protect specific cryptographic tasks (such as signing, decrypting, or zero-knowledge proofs) against leakage. We do not survey this work here and focus, instead, on general transformations.

Chari et al. [8] proved the security of linear secret sharing (specifically, XOR-masking) for protection of values against noisy leakage. They did not, however, specify how to compute on the shares without reconstructing the original values.

Ishai, Sahai, and Wagner [24] were the first to propose a general circuit transformation resilient against adversaries who see up to k wires in each invocation of the circuit. While we use similar techniques for our construction, our security analysis differs significantly from the one in [24], because we need to handle a broader class of adversaries and cannot hide any part of the circuit's state from the leakage function. Notice that our transformation that resists AC^0 leakages is trivially secure against k-probing. Hence, our leakage model can be interpreted as a generalization of [24], using the ideas of leakage functions and reductions between physical components introduced in the work of Micali and Reyzin [30]. It is important to note that our generalization of the model of [24] comes at a cost: our transformation is less efficient and requires leak-free components.

Following our work, Rothblum [40] proposed a different circuit transformation that also resists AC^0 leakage but does not require leak-free components; it requires a computational hardness assumption, however. Miles and Viola [32, 31] proposed a circuit transformation that resists more powerful classes of leakage, such as AC^0 augmented with $\log^2 k$ gates that compute any symmetric function (including parity), and, under certain computational assumptions, TC^0 and even NC^1 (their transformation, like ours, assumes a simple leak-free component).

A different generalization of the work of [24] was proposed by Ajtai [2], who showed how to compile RAM programs onto ones that are resilient to wire probing; his transformation can also be viewed in the circuit model, allowing more probed wires than the transformation of [24].

In a somewhat different vein, following our work, other compilers have been proposed that protect arbitrary computation against any *polynomial-time local* leakage function. In particular, Juma and Vahlis [26] and Goldwasser and Rothblum [20] present techniques that allow computation in the presence of arbitrary bounded-range polynomial-time leakage functions. The advantage these schemes have over our work is that they tolerate much stronger leakage functions; the work of Goldwasser and Rothblum [20] is particularly strong in this regard, as it allows for an exponentially higher leakage rate (as a function of the transformed circuit size) than the work of Juma and Vahlis [26]. On the other hand, these works rely on the additional assumption of "only computation leaks information" that we do not use: in other words, they assume that intermediate results can be placed into non-leaking memory, or equivalently, that different parts of the computation leak independently. (In case of [20] this assumption is very strong: the transformed circuit is assumed to be split into very many pieces that leak independently; in case of [26], only two pieces are required).

Similar to our work, Juma and Vahlis [26] and Goldwasser and Rothblum [20] make use of leak-free components. The gate in [26] has to do key refreshing of a secret key for a fully homomorphic encryption scheme, while in [20] for a given public key the component either outputs a random encryption or an encryption of 0. An advantage that the work of Juma and Vahlis enjoys is that its transformation requires only a single copy of the leak-free component for the whole transformed circuit.

Dziembowski and Faust [12] proposed a compiler in the same local-leakage model as [26, 20] but without relying on computational assumptions. Their compiler relies on homomorphic properties of the inner product extractor and simplifies both the compiler and the leak-free gates that were used in earlier works. Goldwasser and Rothblum [21] eliminated both the computational assumptions and the need for leak-free components, but retained the assumption that multiple (and numerous) parts of the circuit leak independently. The already-mentioned transformation of Miles an Viola [32] is also secure in this model of bounded-size polynomial-time "local" leakage, where "local" means a constant number of independently leaking parts (which are not, however, local in the sense of circuit layout or time); their result for this model does not require computational assumptions, but still relies on a leak-free component.

Duc et al. [10], following the work of Prouff and Rivan [37], proposed a compiler in a leakage model that is very similar to our noisy leakage model, with each wire leaking a noisy version of its value. Their work allows for more general noise distributions, does not require any leak-free gates, but requires more overall noise to achieve security.

Reader's Guide. In the next two sections we introduce the security notion of a circuit transformation. In Section 3.2 we give our circuit transformation for computationally weak leakages, and in Section 3.3 for noisy leakages. Security of our construction is proved in Section 4. We first define the notion of *reconstructibility*, then in Section 4.3 and 4.4 we prove that our simple gadgets satisfy these notions. This is followed by our composition lemma in Section 4.5, and finally leads in Section 4.6 of stateful circuit transformations. In Section 5 we show how our construction for computationally weak leakage applies to the specific case when the leakage function is in AC^0.

2 Notation.

In this paper vectors, denoted $\vec{v} = (v_1, \ldots, v_n)$, are column vectors. Matrices are typically denoted by capital letters with an arrow on top. If a matrix $\vec{M}$ has n rows and m columns, then we say that it is a $n \times m$ matrix. $M_{i,j}$ denotes the $(i,j)^{th}$ element, i.e., the entry in row i and column j. Matrix addition and vector multiplication are defined in the standard way.

We denote function composition by $f \circ g : x \mapsto f(g(x))$. If $\mathcal{L}_1$ and $\mathcal{L}_2$ are two sets of functions, then $\mathcal{L}_2 \circ \mathcal{L}_1$ is a set of functions $\{f \circ g \mid f \in \mathcal{L}_2, g \in \mathcal{L}_1\}$. For some set of functions $\mathcal{L}$ we write $n \times \mathcal{L}$ to denote n "parallel" computations of functions in $\mathcal{L}$. That is, $n \times \mathcal{L} = \{(f_1, \ldots, f_n) | f_1, \ldots, f_n \in \mathcal{L}\}$. Notice that we overload notation and sometimes say that functions $f \in n \times \mathcal{L}$ take n copies of the same input (i.e., f computes $(f_1(x), \ldots, f_n(x))$), while sometimes f takes n different inputs $x_1, \ldots, x_n$ (i.e., f computes $(f_1(x_1), \ldots, f_n(x_n))$).

Let $\mathcal{C}(d, s, \lambda)$ denote the class of AND-OR-NOT unlimited fan-in boolean circuits with depth d (not counting NOT gates), size s and λ bits of output. We will sometimes care about arithmetic circuits over a finite field $\mathcal{K}$; in the case, let the function class $\mathsf{SHALLOW}(d, s)$ (for $d, s \in \mathbb{N}$) denote the class of functions that can be computed by deterministic circuits that have at most s field addition, subtraction, and multiplication gates that are arranged at most d deep (i.e., the longest path in the circuit has at most d such gates on it). We will allow an arbitrary number of gates supplying constants from $\mathcal{K}$; such gates will not be counted as part of s and d. Note that even $\mathcal{K} = \mathsf{GF}(2)$, these two classes are different: one deals with unlimited-fan-in boolean operations, while the other deals with algebraic operations of fan-in 2.

The *statistical distance* of two random variables D_1, D_2 is $\frac{1}{2} \sum_x |\Pr[D_1 = x] - \Pr[D_2 = x]|$. They are ϵ-close (denoted by $D_1 \approx_\epsilon D_2$) if their statistical distance is at most ϵ. They are (τ, ϵ)-

computationally close (denoted by $D_1 \approx_{\tau,\epsilon} D_2$) if for any distinguisher D that runs in time τ and outputs 0 or 1, $D(D_1) \approx_\epsilon D(D_2)$. The notation $d \leftarrow D$ denotes sampling a value d according to a random variable D (or uniformly, if D is simply a set).

3 Circuit Transformations

A circuit transformation TR takes as input a security parameter k, a circuit C, and an initial state m_0 and produces a new circuit $\widehat{C}$ and new initial state $\widehat{m}_0$.[2] Let us discuss how C and $\widehat{C}$ look and what properties the transformation TR needs to satisfy. The discussion below adapts the definitions of [24] to general classes of leakage functions, as suggested in [30].

The original circuit C. We assume that the original circuit C carries values from an (arbitrary) finite field $\mathcal{K}$ on their wires and is composed of the following gates (in addition to the memory gates which will be discussed later): $\oplus, \ominus$, and $\odot$ (which compute, respectively, the sum, difference, and product in $\mathcal{K}$, of their two inputs), the "coin flip" gate \$ (which has no inputs and produces a random independently chosen element of $\mathcal{K}$), and for every $\alpha \in \mathcal{K}$, the constant gate $\mathtt{const}_\alpha$ (which has no inputs and simply outputs α).

Fanout in C is handled by a special `copy` gate that takes as input a single value and outputs two copies. If we use one output of a gate ℓ times, then it is passed through a subcircuit of $\ell - 1$ `copy` gates arranged in a tree (the structure of the tree may be chosen arbitrarily). Notice that `copy` gates are just the identity (pass-through wires) and are present mainly for notational convenience.

Stateful circuits. As in the work of Ishai et al. [24], we define the notion of a *stateful* circuit. A stateful circuit additionally contains memory gates, each of which has a single incoming and a single outgoing edge. Memory gates maintain state: at any clock cycle, a memory gate sends its current state down its outgoing edge and updates it according to the value of its incoming edge. Any cycle in the circuit must contain at least one memory gate. The state of all memory gates at clock cycle i is denoted by m_i, with m_0 denoting the initial state. Inputs to and outputs from clock cycle i are denoted, respectively, by x_i and y_i.[3] When a circuit is run in state m_{i-1} on input x_i, the computation will result in a *wire assignment* W_i (a wire assignment to C is a string in $\mathcal{K}^t$, $t \in \mathbb{N}$, where each element represents a value on a wire in C); the circuit will output y_i and the memory gates will be in a new state m_i. We will denote this by $(y_i, m_i, W_i) \Leftarrow C[m_{i-1}](x_i)$.

The transformed circuit $\widehat{C}$. $\widehat{C}$ will make use of the same atomic gates as C and also of leak-free ("opaque") gates described in Sections 3.2 and 3.3. We require from the transformed circuit $\widehat{C}$ with state $\widehat{m}_0$ that it "behaves identically" to C with initial state m_0. We formalize this by the *soundness* property of a circuit transformation TR. That is, for all C and m_0, for any number of clock cycles q and any set of inputs $x_1, x_2, \ldots, x_q$ (one for each clock cycle) the distribution of the outputs $y_1, y_2, \ldots, y_q$ is the same for C starting at state m_0 and $\widehat{C}$ starting at state $\widehat{m}_0$. (We note that [24] defined a more relaxed notion of soundness, requiring only indistinguishable, rather than perfectly identical behavior, but we do not need it in this work.)

[2]Throughout this work, we use the hat notation $\widehat{\square}$ (reminiscent of the proverbial "tinfoil hat") to designate circuits or components that are transformed for leakage-resilience.

[3]m_i, x_i and y_i will be vectors with length n, n_{I} and n_{O} (resp.).

```
**Experiment** $\mathsf{Exp}^{\mathrm{real}}_{\mathsf{TR}}(\mathcal{A}, \mathcal{L}, q, C, m_0, k)$
$(\widehat{C}, \widehat{m}_0) \leftarrow \mathsf{TR}(C, m_0, 1^k)$
$(x_1, f_1) \leftarrow \mathcal{A}(\widehat{C}, 1^k)$, with $f_1 \in \mathcal{L}$
For $i = 1$ to $q - 1$
    $(y_i, \widehat{m}_i, W_i) \Lleftarrow \widehat{C}[\widehat{m}_{i-1}](x_i)$;
    $(x_{i+1}, f_{i+1}) \leftarrow \mathcal{A}(y_i, f_i(W_i))$
$(y_q, \widehat{m}_q, W_q) \Lleftarrow \widehat{C}[\widehat{m}_{q-1}](x_q)$;
Return output of $\mathcal{A}(y_q, f_q(W_q))$.
```

```
**Experiment** $\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}, \mathcal{A}, \mathcal{L}, q, C, m_0, k)$
$(\widehat{C}, \widehat{m}_0) \leftarrow \mathsf{TR}(C, m_0, 1^k)$
$(x_1, f_1) \leftarrow \mathcal{A}(\widehat{C}, 1^k)$, with $f_1 \in \mathcal{L}$
For $i = 1$ to $q - 1$
    $(y_i, m_i) \leftarrow C[m_{i-1}](x_i)$
    $\Lambda_i \leftarrow \mathcal{S}(x_i, y_i, f_i, 1^k)$, with $\Lambda_i$ being the leakage
    $(x_{i+1}, f_{i+1}) \leftarrow \mathcal{A}(y_i, \Lambda_i)$
$(y_q, m_q) \leftarrow C[m_{q-1}](x_q)$; $\Lambda_q \leftarrow \mathcal{S}(x_q, y_q, f_q, 1^k)$
Return output of $\mathcal{A}(y_q, \Lambda_q)$.
```

Figure 1: The real world with the adversary $\mathcal{A}$ observing the computation of the transformed circuit $\widehat{C}[\widehat{m}_i]$ is shown on the left hand side. On the right hand side we describe the simulation.

Security of circuit transformation TR. We want to make sure that the transformed circuit leaks no useful information to an adversary. We use the term $(\mathcal{L}, \tau)$-*adversary* to denote an adversary $\mathcal{A}$ with physical observations limited to functions in the class $\mathcal{L}$ and running time (not including the computation by the leakage function itself) limited to τ. If the adversary $\mathcal{A}$ gets to query the circuit q times, each time choosing a fresh function from $\mathcal{L}$, which computes its output based on the entire wire assignment, we call it a *q-adaptive* $(\mathcal{L}, \tau)$-adversary. To formalize that such an adversary learns nothing useful, we show the existence of a simulator $\mathcal{S}$, and prove that anything the adversary learns can also be learned by $\mathcal{S}$ which does not get any leakage.

Consider the experiments in Figure 1 that start with some circuit C in state m_0, and allow it to run for q iterations. In both experiments, we assume that $\mathcal{A}$ and $\mathcal{S}$ are stateful, i.e., remember their state from one invocation to the next.

The security definition below says that the transformed circuit is secure if the outputs of the two experiments are indistinguishable. We are now ready to state our security notion precisely. The definition is for both computational and statistical indistinguishability (the latter is obtained by setting the parameter τ_D to ∞).

Definition 1 (Security of Circuit Transformation). *Recall that k is the security parameter. A circuit transformation TR is $(\mathcal{L}, \tau_{\mathcal{A}}, \tau_{\mathcal{S}}, \tau_D, q, \epsilon)$-secure if for every q-adaptive $(\mathcal{L}, \tau_{\mathcal{A}})$-adversary $\mathcal{A}$ there is a simulator $\mathcal{S}$ running in time $\tau_{\mathcal{S}}$ such that for all (stateful) circuits C with initial states m_0*

$$\mathsf{Exp}^{\mathrm{real}}_{\mathsf{TR}}(\mathcal{A}, \mathcal{L}, q, C, m_0, k) \approx_{\tau_D, \epsilon} \mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}, \mathcal{A}, q, C, m_0, k) \,,$$

where the random variables are outputs of the experiments. Sometimes, we abbreviate parameters and refer to a circuit transformation being $\mathcal{L}$-secure, which means it is $(\mathcal{L}, \tau_{\mathcal{A}}(k), \tau_{\mathcal{S}}(k), \tau_D(k), q(k), \epsilon(k))$-secure for some polynomials $\tau_{\mathcal{A}}$, $\tau_{\mathcal{S}}$, τ_D, q, and some negligible function ϵ.

Note that a stronger result is obtained when $\mathcal{L}$, $\tau_{\mathcal{A}}$, τ_D and q are larger (as it allows for more leakage functions and stronger adversaries), and when $\tau_{\mathcal{S}}$ and the distinguishing advantage ϵ are smaller (because it indicates tighter simulation).

In the following sections, we discuss three circuit transformations, which follow a similar approach that we call *encoding based circuit transformations*. We then show three examples of encoding based circuit transformations. In Section 3.2, we propose encoding based transformations for computationally weak and in Section 3.3 for noisy leakages.

3.1 Encoding Based Circuit Transformations

At a high-level, in an encoding based circuit transformation, each wire w in the original circuit C is represented by a *wire bundle* in $\widehat{C}$, consisting of k wires $\vec{w} = (w_1, \ldots, w_k)$, that carry an *encoding* of w. The gates in C are replaced gate-by-gate with so called *gadgets*, computing on encoded values. Our construction will ensure that each wire bundle between two gadgets carries an encoding of its value that is chosen *uniformly and independently of all the wires in the transformed circuit.* The main difficulty in the concrete constructions of Section 3.2–3.3 will be to construct gadgets that remain "secure" even if their internals may leak. Before devising the concrete gadgets, let us explain the main ingredients of encoding based circuit transformations.

Encoding scheme Π. The main ingredient of the transformations presented below is an *encoding scheme* $\Pi = (\mathsf{Enc}, \mathsf{Dec})$, which maps a single element of a finite field $\mathcal{K}$ to a vector in $\mathcal{K}^k$ and back. More precisely, for $k \in \mathbb{N}_{>0}$, $\mathsf{Dec} : \mathcal{K}^k \to \mathcal{K}$ is a deterministic function, and Enc is a (probabilistic) algorithm that, on input $x \in \mathcal{K}$, chooses uniformly at random an element of $\mathsf{Dec}^{-1}(x)$ (where $\mathsf{Dec}^{-1}(x)$ is the set $\{y \in \mathcal{K}^k : \mathsf{Dec}(y) = x\}$).

Linear encoding schemes. A special type of an encoding scheme is a *linear encoding scheme*, which requires that Dec is an affine function. For $k \in \mathbb{N}_{>0}$, Dec is defined by a *decoding vector* $\vec{r} = (r_1, \ldots, r_k) \in \mathcal{K}^k$ as $\mathsf{Dec} : (y_1, \ldots, y_k) \mapsto \sum_i y_i r_i = \vec{r}^{\mathsf{T}} \vec{y}$. In the simplest case of $\mathcal{K} = \mathsf{GF}(2)$, a linear encoding of a bit x is a random string of k bits whose exclusive-or is x. The case when $\mathcal{K} = \mathsf{GF}(2)$ will be called *parity encoding*, denoted with Π_{parity}. Further examples of linear encoding schemes are any threshold or non-threshold linear secret sharing scheme [4].

We abuse notation and use $\mathsf{Enc}(x)$ to denote the distribution of encodings of x, $\vec{x}$ to denote a particular encoding from this distribution and $\mathsf{Enc}(x_1, \ldots, x_n) = (\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_n)) = (\vec{x}_1, \ldots, \vec{x}_n)$ to denote the encoding of a set of elements $x_1, \ldots, x_n \in \mathcal{K}$. Furthermore, denote by $\mathsf{Enc}(\cdot)$ the uniform distribution over all encodings.

Encoding inputs – decoding outputs. Because the transformed gadgets in $\widehat{C}$ operate on encodings, $\widehat{C}$ needs to have a subcircuit at the beginning that encodes the inputs and another subcircuit at the end that decodes the outputs. However, in our proofs, we want to be able to also reason about transformed circuits without encoding and decoding. Thus, we do not require that every transformed circuit $\widehat{C}$ should have such encoding and decoding. Instead, we introduce artificial input and output gates that can be part of C for syntactic purposes. If such gates are present (as they would be on any "complete" circuit that one would actually wish to transform), then $\widehat{C}$ will include input encoding and output decoding. If they are not, then $\widehat{C}$ will operate on already encoded inputs and produce encoded outputs.

More precisely, if we wish for $\widehat{C}$ to include input encoding and output decoding, then the circuit C given to TR must have a special `encoder` gate on every input wire. In C the `encoder` gate is simply the identity, since no encoding is needed. Also, on every output wire there must be a special `decoder` gate, which is also the identity. These special gates must not appear anywhere else in C and do not count for the size of the circuit. In $\widehat{C}$ each `encoder` gate is replaced by an $\widehat{\texttt{encoder}}$

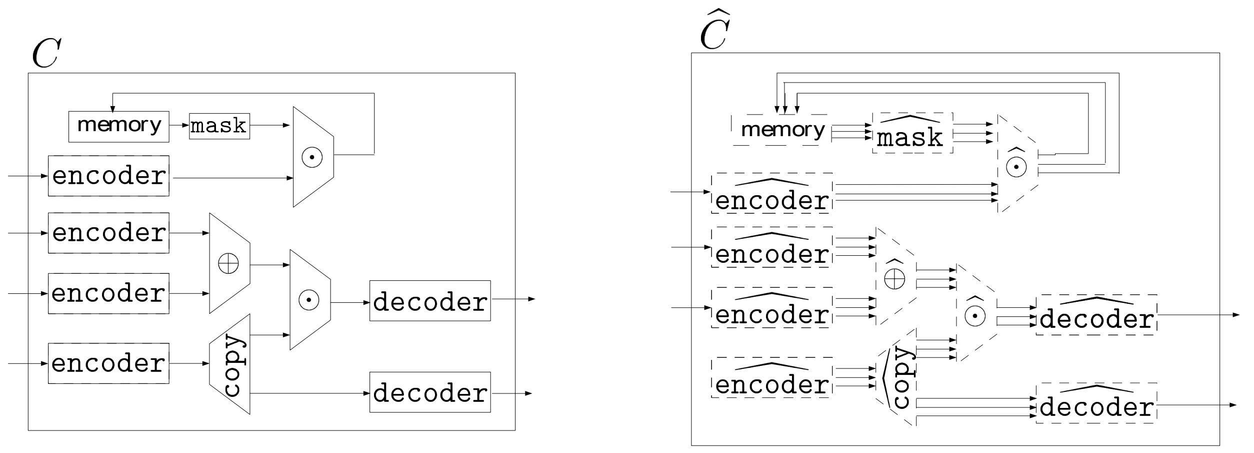

Figure 2: An example of a circuit C and its transformation $\widehat{C}$.

gadget which performs encoding (see below) and each `decoder` gate is replaced by a $\widehat{\texttt{decoder}}$ gadget that performs decoding (see below).

The $\widehat{\texttt{encoder}}$ gadget takes an input $a \in \mathcal{K}$ and outputs an encoding (i.e., a wire bundle) $\vec{a} \in \mathcal{K}^k$ of a. The encoding can be chosen arbitrarily from the support of $\mathsf{Enc}(a)$, e.g., $\vec{a} = (r_1^{-1}a, 0, \ldots, 0)$. The $\widehat{\texttt{decoder}}$ gadget takes an encoding (i.e., a wire bundle) $\vec{a} \in \mathcal{K}^k$ of a and outputs $a \leftarrow \mathsf{Dec}(\vec{a})$. For our concrete transformations below, $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ can be implemented with just $\texttt{const}_\alpha$, $\oplus$, and $\odot$ gates.

Transformation of the state. So far we have considered the transformation of stateless circuits. A stateful circuit C additionally has an initial state m_0 that is stored in memory cells. For syntactical reasons we assume that each such memory cell in C is followed by a `mask` gate, which is implemented in C by the identity function.

To augment the circuit transformation to handle stateful circuits, we have to explain how to transform the initial state m_0 and what to do with each memory gate. The initial state is replaced by a randomly chosen encoding $\mathsf{Enc}(m_0)$. Each memory gate is replaced by a gadget that consists of k memory gates to store the encoding, followed by a $\widehat{\texttt{mask}}$ gadget that represents the `mask` gate in C. The implementation of the $\widehat{\texttt{mask}}$ gadget is transformation specific and will be described for our concrete instantiations below. Notice that in contrast to $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$, the $\widehat{\texttt{mask}}$ gadget is necessary to achieve security; otherwise we cannot safely tolerate leakage that exceeds the length of a single encoding.[4] The high-level structure of the circuit transformation is given in Figure 2.

3.2 Circuit Transformations Resilient to Computationally-Bounded Leakage

The transformation of [24] protects against leakage that reveals the values of up to $k-1$ wires, and hence, is oblivious to huge parts of the computation. In this section we now present our circuit transformation TR_C that protects circuits against global, but computationally bounded leakage. More precisely, we will show that from any linear encoding scheme Π over $\mathcal{K}$, we can construct a

[4]The purpose of $\widehat{\texttt{mask}}$ is to guarantee re-randomization of the memory and destroy partial information that an adversary may have learnt about the secret state.

circuit transformation TR_C that protects circuits against leakages from class $\mathcal{L}$, where $\mathcal{L}$ contains all functions that cannot decode Π.

To capture formally that leakage functions in $\mathcal{L}$ cannot decode Π, we will need the notion of *leakage-indistinguishability.* Roughly speaking, this notion formalizes what it means for an encoding of two values to be indistinguishable in the presence of leakage from $\mathcal{L}$. But let us first introduce a more general definition that speaks about leakage-indistinguishability of two distributions:

Definition 2 (Leakage-Indistinguishability of Distributions and Encodings). *Two distributions X, X' are said to be p-adaptive $(\mathcal{L}, \tau, \epsilon)$*-leakage-indistinguishable*, if for any adversary $\mathcal{A}$, running in time τ and making at most p queries to its oracle,*

$$|\Pr[x \leftarrow X : \mathcal{A}^{\mathsf{Eval}(x,\cdot)}(1^k) = 1] - \Pr[x \leftarrow X' : \mathcal{A}^{\mathsf{Eval}(x,\cdot)}(1^k) = 1| \leq \epsilon, \tag{1}$$

where $\mathsf{Eval}(x, \cdot)$ takes as input a leakage function $f \in \mathcal{L}$ and outputs $f(x)$.

We say that an encoding scheme Π *is p-adaptive $(\mathcal{L}, \tau, \epsilon)$*-leakage-indistinguishable *if for any $a, a' \in \mathcal{K}$ the two distributions $\mathsf{Enc}(a)$ and $\mathsf{Enc}(a')$ are p-adaptive $(\mathcal{L}, \tau, \epsilon)$-leakage-indistinguishable.*

Recall that the function class $\mathsf{SHALLOW}(d, s)$ (for $d, s \in \mathbb{N}$) is the class of functions that can be computed by deterministic circuits (i.e., ones without \$ gates) that have at most s $\oplus, \ominus$, and $\odot$ gates that are arranged at most d deep (i.e., the longest path in the circuit has at most d such gates on it).[5]

The theorem below shows the existence of a circuit transformation TR_C based on arbitrary leakage-indistinguishable linear encoding schemes Π.

Theorem 1 (Security against computationally bounded leakage). *Recall that k is the security parameter. Furthermore, let $\mathcal{L}_\Pi$ be some class of leakage functions and let $q, \epsilon_\Pi, \tau_\Pi \geq 0$.[6] If there exists a linear encoding scheme Π over $\mathcal{K}$ with output outputs of size k that is 2-adaptive $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable, then there exists a circuit transformation TR_C that is $(\mathcal{L}, \tau_\mathcal{A}, \tau_\mathcal{S}, \tau_D, q, \epsilon)$-secure for*

- *any $\tau_\mathcal{A}$ and τ_D satisfying $\tau_\mathcal{A} + \tau_D \leq \tau_\Pi - qsO(k^2)$, where s is the number of gates in C,*
- *some $\tau_\mathcal{S} \leq \tau_\mathcal{A} + qsO(k^2)$,*
- *some $\epsilon \leq \epsilon_\Pi(q + 1)(s(k + 2) + n)$, where n is the number of memory gates in C,*
- *any $\mathcal{L}$ that satisfies $\mathcal{L} \circ \mathsf{SHALLOW}(3, O(k^2)) \subseteq \mathcal{L}_\Pi$ (for $\mathcal{K} = \mathsf{GF}(2)$, $\mathcal{L} \circ \mathsf{SHALLOW}(2, O(k^2)) \subseteq \mathcal{L}_\Pi$).*

In particular, if the encoding scheme is 2-adaptive $\mathcal{L}_\Pi$-leakage indistinguishable, then the transformation is $\mathcal{L}$-secure.

The transformation increases the size of each multiplication gate by a factor of $O(k^2)$ and the size of the rest of the circuit by a factor of $O(k)$.

Notice that we define $\mathcal{L}$ implicitly by $\mathcal{L}_\Pi$ and the function class $\mathsf{SHALLOW}(3, O(k^2))$. Loosely speaking, if we want that our circuits are resilient to functions in $\mathcal{L}$, then we need an encoding scheme Π that is resilient to functions at least from some set $\mathcal{L} \circ \mathsf{SHALLOW}(3, O(k^2))$. Our result

[5]Note that `copy` and $\mathtt{const}_\alpha$ gates are allowed in the circuit and do not count towards d or s.

[6]These values are all parameterized by k, and thus, for ease of notation, we omit k.

is better (or tighter) if the difference between $\mathcal{L}$ and $\mathcal{L}_\Pi$ can be described by functions that are as simple as possible. In our case the difference between $\mathcal{L}$ and $\mathcal{L}_\Pi$ (in terms of computational power) is described by the class $\mathsf{SHALLOW}(3, O(k^2))$. We will present a concrete instantiation of the above theorem in Section 5, where we set Π to Π_{parity} and $\mathcal{L}$ to AC^0, in which case we obtain statistical, rather than computational, indistinguishability.

The leak-free gate $\mathcal{O}$. Ideally, one would hope that our transformation TR_C requires only standard gates of constant size to transform arbitrary circuits. Unfortunately, it is not clear how to prove security when we restrict TR_C to only use such gates. To show security, we allow TR_C to additionally use the leak-free gate $\mathcal{O}$. The $\mathcal{O}$ gate has size linear (in k), but as discussed in the introduction is very simple otherwise. In particular, it has *no inputs*, and merely outputs an encoding sampled from the distribution $\mathsf{Enc}(0)$. Crucially, while the wires coming out of this gate can be observed by the leakage function, we assume that the gate itself does not leak information. For the case of $\mathcal{K} = \mathsf{GF}(2)$ our leak-free component can be implemented by a leak-free subcircuit that works as follows: generate k random bits $b_0, \ldots, b_{k-1}$ and output $b_i \oplus b_{i+1 \bmod k}$ for $0 \leq i \leq k-1$. Note that this subcircuit is of constant depth.

Our transformation follows the outline given in Section 3.1 using the gadgets given in Figure 3. A visual description of its main component, the $\widehat{\odot}$ gadget, is presented in Figure 4. At a high-level the $\widehat{\odot}$ gadget first computes a (not uniformly random) encoding of $a \odot b$ represented by the $k \times k$ matrix $\vec{B}$. Next, $\vec{B}$ is "randomized" with the matrix $\vec{S}$, resulting in $\vec{U}$. Since eventually $\widehat{\odot}$ has to output a uniform encoding of $a \odot b$ with length k, we compress the "rows" of $\vec{U}$ by decoding to obtain $\vec{q}$. The transformation increases the size of each multiplication gate by a factor of $O(k^2)$ and the rest of the circuit by a factor of $O(k)$, where the constants hidden in $O(\cdot)$ are small.

Incidentally, observe in Figure 3 that because every gadget other than $\widehat{\texttt{encoder}}$ or $\widehat{\texttt{decoder}}$ ends with a masking by an output of $\mathcal{O}$, [7] and wire bundles do not fan-out (instead, they go through the $\widehat{\texttt{copy}}$ gadget), each wire bundle between two gadgets carries an encoding of its value that is chosen uniformly and independently of all the wires in the transformed circuit. This fact, together with the construction of the gadgets, is what enables to show security.

Transformation of stateful circuits. To augment the transformation to handle stateful circuits we proceed as outlined in Section 3.1 – namely, we encode the initial memory m_0 by $\widehat{m}_0 \leftarrow \mathsf{Enc}(m_0)$ and use $|\widehat{m}_0|$ memory cells to store the result. Each such bundle of k memory cells is followed by the $\widehat{\texttt{mask}}$ gadget to re-randomize the state. This masking is necessary to guarantee security as will be discussed later.

In the following Lemma we show that our transformation outputs circuits that are functionally equivalent to C. The security proof is given in Section 4 within our general framework to analyze security of encoding based circuit transformations.

Lemma 1 (Soundness of TR_C). *The circuit transformation TR_C is sound.*

Proof. Since we encode the input, do a gate-by-gate transformation, and then decode the output, it suffices to prove that our gate gadgets work correctly on encoded values (recall that $\vec{r}$ is the decoding vectors):

[7]One can instead define the basic gadgets as not including this masking with $\mathcal{O}$, and instead place a `mask` gate on every wire. The resulting transformation is similar. However, this does not cleanly generalize to the case of transformations not necessarily based on linear encodings.

```
**Transformation** $c = a \odot b \ \Rightarrow\ \vec{c} \leftarrow \vec{a}\widehat{\odot}\vec{b}$:
    Compute the $k \times k$ matrix
        $\vec{B} = \vec{a}\vec{b}^{\mathsf{T}} = (a_i \odot b_j)_{1 \le i,j \le k}$ using $k^2$ $\odot$ gates
    Compute the $k \times k$ matrix $\vec{S}$
        where each column of $\vec{S}$ is output by $\mathcal{O}$
    $\vec{U} = \vec{B} \oplus \vec{S}$ (using $k^2$ $\oplus$ gates)
    Decode each row of $\vec{U}$ using $k-1$ $\oplus$ gates,
        $k$ $\odot$ gates, and $k$ $\texttt{const}_\alpha$ gates
        to obtain $\vec{q} = \vec{U}\vec{r}$,
        where $\vec{r}$ is the decoding vector
        (it does not matter how this decoding is
        performed as long as there are $O(k)$ wires
        in the decoding subcircuit and each one
        carries some linear combination of the
        wires being decoded, plus possibly a
        constant)
    $\vec{o} \leftarrow \mathcal{O}$
    $\vec{c} = \vec{q} \oplus \vec{o}$ (using $k$ $\oplus$ gates)
```

```
**Transformation** $c \leftarrow \$ \ \Rightarrow\ \vec{c} \leftarrow \widehat{\$}$:
    $c_i \leftarrow \$$    for $i \in [1, k]$
```

```
**Transformation** $c = a \oplus b \ \Rightarrow\ \vec{c} \leftarrow \vec{a}\widehat{\oplus}\vec{b}$
            (or $c = a \ominus b \ \Rightarrow\ \vec{c} \leftarrow \vec{a}\widehat{\ominus}\vec{b}$):
    $\vec{q} = \vec{a} \oplus \vec{b}$ (or $\vec{q} = \vec{a} \ominus \vec{b}$)
        using $k$ $\oplus$ (or $\ominus$) gates
    $\vec{o} \leftarrow \mathcal{O}$
    $\vec{c} = \vec{q} \oplus \vec{o}$ (using $k$ $\oplus$ gates)
```

```
**Transformation** $b = \texttt{mask}(a) \ \Rightarrow\ \vec{b} \leftarrow \widehat{\texttt{mask}}(\vec{a})$
    $\vec{o} \leftarrow \mathcal{O}$
    $\vec{b} = \vec{a} \oplus \vec{o}$ (using $k$ $\oplus$ gates)
```

```
**Transformation** $a = \texttt{const}_\alpha \ \Rightarrow\ \vec{a} \leftarrow \widehat{\texttt{const}}_\alpha$,
        for any $\alpha \in \mathcal{K}$
    Let $\vec{\alpha}$ be a fixed arbitrary encoding of $\alpha$.
    $\vec{o} \leftarrow \mathcal{O}$
    $\vec{a} = \vec{\alpha} \oplus \vec{o}$ (using $k$ $\oplus$ gates)
```

```
**Gadget** $(\vec{b}, \vec{c}) \leftarrow \widehat{\texttt{copy}}(\vec{a})$
    $\vec{o_b} \leftarrow \mathcal{O}$, $\vec{o_c} \leftarrow \mathcal{O}$
    $\vec{b} = \vec{a} \oplus \vec{o_b}$ (using $k$ $\oplus$ gates)
    $\vec{c} = \vec{a} \oplus \vec{o_c}$ (using $k$ $\oplus$ gates)
```

Figure 3: Gadgets used in the stateless circuit transformation TR_C.

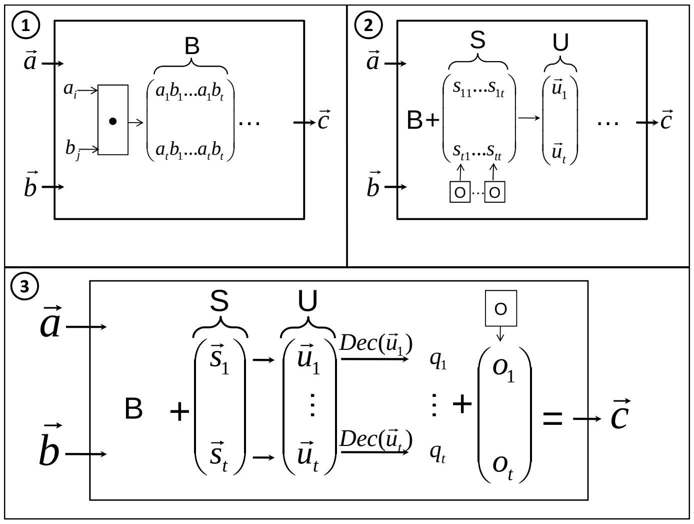

Figure 4: A step-by-step illustration of the $\widehat{\odot}$ gadget. Steps (1-3) are all part of the transformed gadget $\widehat{\odot}$.

$\widehat{\oplus}$**:** For $\vec{c} = \vec{a} \oplus \vec{b} \oplus \vec{o}$, with $\vec{o}$ being an encoding of 0, we get by linearity that $\mathsf{Dec}(\vec{c}) = a \oplus b$.

$\widehat{\odot}$: $\mathsf{Dec}(\vec{c}) = \vec{r}^{\mathsf{T}}(\vec{q} \oplus \vec{o}) = \vec{r}^{\mathsf{T}}((\vec{B} \oplus \vec{S})\vec{r} \oplus \vec{o}) = \vec{r}^{\mathsf{T}}((\vec{a}\vec{b}^{\mathsf{T}} \oplus \vec{S})\vec{r} \oplus \vec{o}) = (\vec{r}^{\mathsf{T}}\vec{a})(\vec{b}^{\mathsf{T}}\vec{r}) \oplus (\vec{r}^{\mathsf{T}}\vec{S})\vec{r} \oplus \vec{r}^{\mathsf{T}}\vec{o} = ab \oplus \vec{0}^{\mathsf{T}}\vec{r} \oplus 0 = ab$

$\widehat{\ominus}, \widehat{\texttt{copy}}, \widehat{\texttt{const}}_\alpha, \widehat{\texttt{mask}}$ **and** $\widehat{\$}$**:** Similar to $\widehat{\oplus}$, by linearity.

□

3.3 Circuit Transformations Resilient to Noisy Leakage

So far, Ishai et al. [24] discussed leakage classes that consider probing attacks, and we discussed those that are constrained in terms of their computational power and output length (Section 3.2). In this section, we consider the noisy leakage model, and present a transformation TR_N that makes arbitrary Boolean circuits resilient to leakage that consists of the values of *all the wires* in the circuit, except that each bit is flipped with some probability $p \in (0, 1/2]$. [8]

Noisy leakage. For some noise parameter $p \in (0, 1/2]$ the single tolerated leakage function is $\mathcal{L} = \{\mathcal{N}_p\}$, where $\mathcal{N}_p$ is probabilistic, and is defined as follows: Let B_p be the binomial distribution with parameter p which outputs 1 with probability p and 0 otherwise. Then, for some input $\vec{x} \in \{0,1\}^*$ we have $\mathcal{N}_p(\vec{x}) = \vec{x} \oplus \vec{b}$, where each bit b_i is drawn independently from the distribution B_p.

Ideally, we would hope that the circuit transformation from the previous section provides security against noisy leakage as well. Indeed, since Theorem 1 is very general, we could just plug-in $\mathcal{N}_p$ for $\mathcal{L}$, and get "some" $\mathcal{N}_p$ resilience for our circuit transformation TR_C. The security, however, would rely on the additional assumption that Π_{parity} is $\mathcal{N}_p \circ \mathsf{SHALLOW}(3, O(k^2))$-leakage indistinguishable, and it is not clear what such an assumption would mean in a setting where $\mathcal{N}_p$ reveals all its inputs perturbed by independent binomial noise (recall that $\mathsf{SHALLOW}(3, O(k^2))$ describes the difference between the class of leakages that is tolerated by Π_{parity} and the transformed circuit).

Furthermore, by inspection of our construction it turns out that TR_C is not particularly tailored to protect against noisy leakages. Indeed, TR_C remains "secure" for very large noise p, but this is not what we want. In practice, we are typically interested in what happens for low noise and, unfortunately, we can show that in such a case there is an explicit attack against the transformation TR_C (as well as against the transformation of [24]).

An attack with noisy leakages. Specifically, the attack is against the construction of the multiplication gadget $\widehat{\odot}$ in Figure 3. The gadget takes as input two encodings $\vec{a}$ and $\vec{b}$ and first computes the k^2 bits $\{a_i \odot b_j : i, j \in [k]\}$. Consider the first k bits $(a_1 \odot b_1, \ldots, a_1 \odot b_k)$. If $a_1 = 0$, then all these bits are 0, whereas if $a_1 = 1$, then roughly half of them are 1. Given such disparity, using error correction the adversary can determine whether a_1 is 0 or 1, even if he is given a noisy version of these k bits. Proceeding in a similar way, he can reconstruct all the bits a_i, and thus the input bit a itself. The fundamental reason why this attack works is that the construction of the $\widehat{\odot}$ gadget in Figure 3 does not use its input in a *local* way, namely it accesses the input bits a large number of times.

To avoid this attack we propose a new circuit transformation TR_N. TR_N uses as underlying encoding scheme the parity encoding and proceeds in the same way as the transformation TR_C from Section 3.2 (cf. Figure 3), except for the construction of the multiplication gadget $\widehat{\odot}$. This new construction of the multiplication gadget avoids the attack outlined above, and is constructed using a new opaque gate that we call $\mathcal{Q}$ (in addition to the opaque gate $\mathcal{O}$). We stress that the opaque gate $\mathcal{Q}$ inherits the main characteristics of the opaque gate $\mathcal{O}$ in that it is *stateless*, and *independent of the computation*. In other words, $\mathcal{Q}$ simply produces samples from a fixed distribution.

[8]Notice that similar to Section 3.2 we can generalize the class of circuits that we can transform (i.e., to circuits that do computation over an arbitrary field $\mathcal{K}$) and the class of noisy leakage that we can tolerate. For ease of description we omit the details here.

Before we give the specification of the opaque gate $\mathcal{Q}$ and the construction of the new $\widehat{\odot}$ gadget, let us state our main theorem that deals with noisy leakages. Note that, in contrast to Theorem 1, this theorem deals with a much more restricted leakage class, but, as a result, obtains statistical, rather than computational, indistinguishability.

Theorem 2. *Recall that k is the security parameter. Let $p \in (0, 1/2]$, $q > 0$ and the leakage function $\mathcal{N}_p$ be defined as above. There exists a circuit transformation TR_N that is $(\mathcal{N}_p, \tau_\mathcal{A}, \tau_\mathcal{S}, \tau_D = \infty, q, \epsilon)$-secure for any $\tau_\mathcal{A}$, for some $\tau_\mathcal{S} = \tau_\mathcal{A} + qsO(k^2)$, and*

$$\epsilon \leq (q+1)(n + (2k+3)s)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512)),$$

where s is the number of gates in C and n is the size of the state of C. The transformation increases the size of each multiplication gate by a factor of $O(k^2)$ and the rest of the circuit by a factor of $O(k)$. Note that for fixed ϵ, $k = O(p^{-6} \log 1/p)$.

We prove this theorem in a similar way as Theorem 1. Namely, we base the security of TR_N on the indistinguishability of the underlying parity encoding scheme Π_{parity}. As it turns out (which will be discussed in Section 4.1), the parity encoding scheme is information theoretic $\mathcal{N}_p$ leakage indistinguishable (note that this is the reason why in the above theorem we can eliminate to explicitly condition on the leakage indistinguishability of Π_{parity}). For the details of the proof, we defer to the next section.

Similar to TR_C, the transformation TR_N increases the size of each multiplication gate by a factor of $O(k^2)$ and the rest of the circuit by a factor of $O(k)$. Notice that in contrast to the circuit transformation TR_C, our circuit transformation for noisy leakages requires leak-free gates of size $O(k^2)$. All constants hidden in $O(\cdot)$ notation are small.

The new opaque gate $\mathcal{Q}$. The opaque gate $\mathcal{Q}$ is probabilistic, takes no inputs and outputs $2k^2+1$ bits. It operates in the following way: Sample $2k$ uniformly random 0-encodings $\vec{r}^{(1)}, \ldots, \vec{r}^{(k)}$ and $\vec{s}^{(1)}, \ldots, \vec{s}^{(k)}$. Let $\vec{R}$ and $\vec{S}$ be the following two $k \times k$ matrices:

$$\vec{R} = \begin{pmatrix} \vec{r}^{(1)} \\ \vdots \\ \bigoplus_{j\in[1,i]} \vec{r}^{(j)} \\ \vdots \\ \bigoplus_{j\in[1,k]} \vec{r}^{(j)} \end{pmatrix} \text{ and } \vec{S} = \begin{pmatrix} \vec{s}^{(1)} \\ \vdots \\ \bigoplus_{j\in[1,i]} \vec{s}^{(j)} \\ \vdots \\ \bigoplus_{j\in[1,k]} \vec{s}^{(j)} \end{pmatrix}$$

Let us denote by $\vec{R} \bigcirc \vec{S}$ the sum of the componentwise product, i.e., $\bigoplus_{i,j\in[1,k]} R_{i,j} \odot S_{i,j}$. Then the output of the opaque gate $\mathcal{Q}$ is the tuple $(\vec{r}^{(1)}, \ldots, \vec{r}^{(k)}, \vec{s}^{(1)}, \ldots, \vec{s}^{(k)}, \vec{R} \bigcirc \vec{S}^{\mathsf{T}})$. Unlike the case for $\mathcal{O}$ described in Section 3.2 , we do not know how to sample the distribution produced by $\mathcal{Q}$ in constant depth.

The new multiplication gadget $\widehat{\odot}$. The operation of the multiplication gadget $\widehat{\odot}$ proceeds in two stages.

- The first stage uses a sub-gadget $\widehat{\texttt{mult}}$ that takes as input two encodings $\vec{a} = (a_1, \ldots, a_k)$ and $\vec{b} = (b_1, \ldots, b_k)$ of a and b (resp.), and outputs a k^2 long encoding $\vec{q} = (q_1, \ldots, q_{k^2})$ of $a \odot b$.

```
Subgadget $\vec{q} \leftarrow \widehat{\texttt{mult}}(\vec{a}, \vec{b})$:
    Compute $2k^2 + 1$ bits with $\mathcal{Q}$:
        $(\vec{r}^{(1)}, \ldots, \vec{r}^{(k)}, \vec{s}^{(1)}, \ldots, \vec{s}^{(k)}, u) \leftarrow \mathcal{Q}$.
    Let $\vec{a}^{(0)} = \vec{a}$ and $\vec{b}^{(0)} = \vec{b}$. Compute for $i \in [k]$:
        $\vec{a}^{(i)} = \vec{a}^{(i-1)} \oplus \vec{r}^{(i)}$ and $\vec{b}^{(i)} = \vec{b}^{(i-1)} \oplus \vec{s}^{(i)}$
            (using $O(k^2)$ $\oplus$ gates).
        Let $\vec{A} = (\vec{a}^i)_{i\in[1,k]}$ and $\vec{B} = (\vec{b}^i)_{i\in[1,k]}$.
    Compute $z_1^{(1)} = A_{1,1} \odot B_{1,1} \oplus u$ and
        for $(i,j) \neq (1,1)$: $z_j^{(i)} = A_{i,j} \odot B_{j,i}$
        (using $O(k^2)$ $\odot$ gates).
    For $i \in [k]$: $\vec{w}^{(i)} \leftarrow \mathcal{O}$ (using $O(k)$ opaque gates $\mathcal{O}$).
    Compute $\vec{q}^{(i)} = \vec{z}^{(i)} \oplus \vec{w}^{(i)}$; let $\vec{q} = (\vec{q}^{(1)}, \ldots, \vec{q}^{(k)})$
        (using $O(k^2)$ $\oplus$ gates).
```

```
Subgadget $\vec{c} \leftarrow \widehat{\texttt{compress}}(\vec{q})$:
    Let $\vec{q} = (\vec{q}^{(1)}, \ldots, \vec{q}^{(k)})$, with $\vec{q}^{(i)} \in \{0,1\}^k$.
    $\vec{c} = \vec{q}^{(1)} \widehat{\oplus} \ldots \widehat{\oplus} \vec{q}^{(k)}$ (using $O(k)$ $\widehat{\oplus}$ gadgets).
```

```
Transformation $c = a \odot b \;\Rightarrow\; \vec{c} \leftarrow \vec{a} \widehat{\odot} \vec{b}$
    $\vec{q} \leftarrow \widehat{\texttt{mult}}(\vec{a}, \vec{b})$
        (using $O(k^2)$ standard gates and $O(k)$
        $\mathcal{O}$ gates and one $\mathcal{Q}$ gate).
    $\vec{c} \leftarrow \widehat{\texttt{compress}}(\vec{q})$
        (using $O(k^2)$ $\oplus$ gates and $O(k)$ $\mathcal{O}$ gates).
```

Figure 5: Gadget $\widehat{\odot}$ in the circuit transformation TR_N and its sub-gadgets $\widehat{\texttt{mult}}$ and $\widehat{\texttt{compress}}$. The constants hidden in the O-Notation are small.

- The second stage "compresses" this longer encoding into an encoding $\vec{c} = (c_1, \ldots, c_k)$, using a gadget $\widehat{\texttt{compress}}$.

We describe these two stages and their costs in more detail in Figure 5. Notice that $\widehat{\odot}$ is carefully constructed to prevent the attack outlined above. Crucially, this requires that the inputs, $\vec{a}$ and $\vec{b}$, are not used too often. We achieve this by generating k copies of $\vec{a}$ and $\vec{b}$ as $\vec{a}^{(i)} = \vec{a}^{(i-1)} \oplus \vec{r}^{(i)}$ and $\vec{b}^{(i)} = \vec{b}^{(i-1)} \oplus \vec{s}^{(i)}$, with $\vec{r}^{(i)}, \vec{s}^{(i)} \leftarrow \mathsf{Enc}(0)$. Next, $\vec{A} = (\vec{a}^{(i)})_i$ and $\vec{B} = (\vec{b}^{(i)})_i$ are used to compute the k^2 long encoding $\vec{q} = (q_1, \ldots, q_{k^2})$ of $a \odot b$. For this to be correct we need the value u that is output by $\mathcal{Q}$. Unfortunately, it is not clear how to compute u in clear as, e.g., $\vec{r}^{(1)}$ and $\vec{s}^{(1)}$ are used k times in the computation of u.

The transformation of the other gates, i.e., of $\oplus$, \$, `const`, `mask` and `copy`, is done as in the transformation TR_C and is omitted in Figure 5.

In the following lemma we prove the correctness of our circuit transformation TR_N. The security proof follows our general approach and will we presented in the next section.

Lemma 2 (Soundness of TR_N). *The circuit transformation TR_N is sound.*

Proof. Since the only difference between TR_N and TR_C is the transformation of the $\odot$ gate, we focus on correctness of $\widehat{\odot}$. Let $\vec{R}$ and $\vec{S}$ be the matrices defined above, and let $\hat{\vec{A}}$ be a matrix whose rows are k copies of the vector $\vec{a}$, and $\hat{\vec{B}}$ be a matrix whose rows are k copies of the vector $\vec{b}$. First, a simple calculation shows that

$$\hat{\vec{A}} \bigcirc \vec{S}^{\mathsf{T}} = \bigoplus_{i\in[k]} \left[a_i \odot \left(\bigoplus_{j\in[i]} \overbrace{\vec{s}_1^{(j)} \oplus \ldots \oplus \vec{s}_k^{(j)}}^{=0} \right) \right] = 0 \tag{2}$$

$$\vec{R} \bigcirc \hat{\vec{B}}^{\mathsf{T}} = 0 \tag{3}$$

and

$$\hat{\vec{A}} \bigcirc \hat{\vec{B}}^{\mathsf{T}} = \bigoplus_{i,j} a_j^{(i)} \odot b_i^{(j)} = \left(\bigoplus_i a_i \right) \odot \left(\bigoplus_j b_j \right) = a \odot b \tag{4}$$

We now establish the correctness of the $\widehat{\odot}$ gadget by the following computation:

$$\begin{aligned}
\bigoplus_{i,j\in[1,k]} q_j^{(i)} &= \bigoplus_{i,j\in[1,k]} \left[w_j^{(i)}\right] \oplus u \oplus (\hat{\vec{A}} \oplus \vec{R}) \bigcirc (\hat{\vec{B}} \oplus \vec{S})^{\mathsf{T}} \\
&= u \oplus \hat{\vec{A}} \bigcirc \hat{\vec{B}}^{\mathsf{T}} \oplus \hat{\vec{A}} \bigcirc \vec{S}^{\mathsf{T}} \oplus \vec{R} \bigcirc \vec{S}^{\mathsf{T}} \oplus \vec{R} \bigcirc \hat{\vec{B}}^{\mathsf{T}} \\
&= u \oplus \hat{\vec{A}} \bigcirc \hat{\vec{B}}^{\mathsf{T}} \oplus \vec{R} \bigcirc \vec{S}^{\mathsf{T}} \quad \text{(by Equation 2 and 3)} \\
&= \hat{\vec{A}} \bigcirc \hat{\vec{B}}^{\mathsf{T}} \quad (\text{since } \vec{R} \bigcirc \vec{S}^{\mathsf{T}} = u, \text{ by definition}) \\
&= a \odot b \quad \text{(by Equation 4)}
\end{aligned}$$

□

4 Proof of Security

Before we outline the security proof and introduce our technical tools, we first discuss useful properties of the parity encoding Π_{parity}. This will allow us to prove security for concrete function classes, namely, for AC^0 and $\mathcal{N}_p$ noisy leakages.

4.1 Leakage-Indistinguishability of the Parity Encoding Π_{parity}

We show leakage indistinguishability of the Π_{parity} encoding against multi-bit range AC^0 functions and noisy leakages $\mathcal{N}_p$ (for $p \in (0, 1/2]$).

AC^0 leakage-indistinguishability of Π_{parity}. The decoding function of the Π_{parity} encoding is exactly the parity function, which is hard for AC^0. This observation enables us to prove AC^0 leakage-indistinguishability of Π_{parity}.

Recall that $\mathcal{C}(d, s, \lambda)$ denotes the class of AND-OR-NOT unlimited fan-in circuits with depth d, size s and λ bits of output. If we translate the classical result of Hastad [22] (as cited in [27, Corollary 1]) to our definition of leakage indistinguishability, we get that the parity encoding is $(\mathcal{C}(d, 2^{k^{1/d}}, 1), \infty, 2^{-k^{1/d+1}})$-leakage-indistinguishable, for any constant d. In other words, this protects against AC^0 circuits that output 1 bit. Using the result of Dubrov and Ishai [9, Theorem 3.4], we get the following corollary that later will be applied to show security of TR_C against multi-bit AC^0 leakages (cf. Section 5).

Proposition 1 **(AC^0 leakage indistinguishability of Π_{parity}).** *For any $0 < \delta < 1$, and $d \in \mathbb{N}_{>1}$ the parity encoding Π_{parity} is $(\mathcal{C}(d, \exp(O(k^{(1-\delta)/d})), k^\delta), \infty, \exp(-\Omega(k^{(1-\delta)/d})))$-leakage-indistinguishable.*[9]

$\mathcal{N}_p$ leakage-indistinguishability of Π_{parity}. In this paragraph we prove some useful properties of the parity encoding Π_{parity} for $\mathcal{N}_p$ leakages, which yields into Proposition 2 showing that Π_{parity} is $\mathcal{N}_p$ leakage indistinguishability (for $p \in (0, 1/2]$).

We first present a simple version of the information-theoretic XOR Lemma.

[9] An even better result is obtained if one restricts d to $d = 1$: in that case, the ϵ parameter gets reduced to $\exp(-\Omega(k - k^\delta \log k))$

Lemma 3 (XOR Lemma [18, 43]). *Let X_0 and X_1 be two distributions. For any $k \in \mathbb{N}_{>0}$ and $b \in \{0,1\}$ we define the distributions*

$$\overrightarrow{X_b} = (X_{b_1}, \ldots, X_{b_k}) \text{ with } b_1 \oplus \ldots \oplus b_k = b.$$

If $\Delta(X_0; X_1) \leq \epsilon$, then $\Delta(\overrightarrow{X_0}, \overrightarrow{X_1}) \leq \epsilon^k$.

The above lemma can be used to show that $\mathsf{Enc}(0)$ is indistinguishable from $\mathsf{Enc}(1)$ by noisy leakage. However, if the same share of an encoding appears on multiple wires, and each wire leaks with independent noise, the noise can cancel out, making it easier for the adversary to distinguish. Thus, we now show a technical lemma that bounds the statistical distance of ℓ copies of $\mathcal{N}_p(\mathsf{Enc}(0))$ from ℓ copies of $\mathcal{N}_p(\mathsf{Enc}(1))$.

Lemma 4. *For any constant ℓ, any vectors $\vec{c}_1, \ldots, \vec{c}_\ell \in \{0,1\}^k$ and any $b \in \{0,1\}$, we have $\Delta(D_0, D_1) \leq (1-(2p)^\ell)^k$ with*

$$D_b := D_b(\vec{c}_1, \ldots, \vec{c}_\ell) = \Big((\mathcal{N}_p(\vec{e} \oplus \vec{c}_1), \ldots, \mathcal{N}_p(\vec{e} \oplus \vec{c}_\ell)) \Big)_{\vec{e} \leftarrow \mathsf{Enc}(b)}.$$

Proof. Since the vectors $\vec{c}_i$ are known, it suffices to show that

$$\Delta(D_0(\mathbf{0}, \ldots, \mathbf{0}), D_1(\mathbf{0}, \ldots, \mathbf{0}) \leq (1-(2p)^\ell)^k$$

That is, given ℓ copies of an encoding $\vec{e}$ perturbed by independent binomial noise drawn from $\mathcal{N}_p$, it is (information-theoretically) hard to distinguish whether $\vec{e}$ is an encoding of 0 or an encoding of 1.

Towards showing this, first consider the ith coordinate of the encoding $\vec{e}$ perturbed by ℓ independent noise terms from $\mathcal{N}_p$, i.e., consider the distribution

$$D'_{e_i} = (e_i + \eta_{i,1}, \ldots, e_i + \eta_{i,\ell})$$

where each $\eta_{i,j}$ is drawn independently from the binomial distribution B_p. The statistical distance between D'_0 and D'_1 is at most $1-(2p)^\ell$ by an elementary calculation. By applying the XOR Lemma (cf. Lemma 3) we get

$$\Delta(D_0, D_1) \leq \Delta(D'_0, D'_1)^k \leq (1-(2p)^\ell)^k,$$

which concludes the proof. □

Of course, the situation in the actual transformed circuit won't be as simple as in Lemma 4: we will not have just multiple copies of the wires, but wires that depend in predictable ways on other wires. Intuitively, noise cancellation will not be a problem if a single wire doesn't influence too many other wires, and hence its value is not leaked too many times with independent noise. To formalize this constraint, we introduce the function class $\mathsf{LOCAL}(\ell)$. For some $\ell, m, n \in \mathbb{N}$, a function $f : \{0,1\}^{mk} \to \{0,1\}^n$ with inputs $\vec{x}_1, \ldots, \vec{x}_m \in \{0,1\}^k$ is said to be in $\mathsf{LOCAL}(\ell)$ if the following holds for each $i \in [1,m]$:

> For any fixed $m-1$ inputs $\vec{x}_1, \ldots, \vec{x}_{i-1}, \vec{x}_{i+1}, \ldots, \vec{x}_m$, all but at most $k\ell$ output bits of the function $f(\vec{x}_1, \ldots, \vec{x}_m)$ (as a function of x_i) are constant (i.e., do not depend on x_i); the remaining output bits are computed as $\vec{x}_i \oplus \vec{d}$, for some constant $\vec{d}$.

The identity function, for instance, is in $\mathsf{LOCAL}(1)$, while a function that outputs ℓ copies of its inputs is in $\mathsf{LOCAL}(\ell)$.

Informally, the proposition below says that an adversary that picks q times functions $f \in \mathcal{N}_p \circ \mathsf{LOCAL}(\ell)$ obtains nothing more than $q\ell$ noisy copies of the target encoding (essentially $\mathcal{N}_p \circ \mathsf{LOCAL}(\ell)$ takes as input an encoding and outputs ℓ noisy copies). To sum it up, we get

Proposition 2 (Noisy leakage indistinguishability of Π_{parity}). *For any $p \in (0, 1/2]$ and any constant $\ell, q \in \mathbb{N}_{\geq 1}$ the parity encoding Π_{parity} is q-adaptive $(\mathcal{N}_p \circ \mathsf{LOCAL}(\ell), \infty, (1-(2p)^{q\ell})^k)$-leakage indistinguishable.*

In particular, Π_{parity} is $(\mathcal{N}_p, \infty, (1-2p)^k)$-leakage indistinguishable (since $\mathcal{N}_p \circ \mathsf{LOCAL}(1) = \mathcal{N}_p$ outputs a single noisy copy of its input).

The proof follows immediately from definition of locality and Lemma 4 and is omitted.

4.2 Outline of the proof techniques

Notation. By $\mathcal{W}_C(x)$ we denote a distribution of wire assignments (recall that a wire assignment of a circuit C is a string in $\mathcal{K}^t$, where each element represents a value on a wire of C) that is induced when a circuit C is being evaluated on an input x (in particular, if C is deterministic, then $\mathcal{W}_C(x)$ has only one element in its support). We use $\mathcal{W}_C(x|y)$ to denote the same distribution conditioned on the fact that the output of $C(x)$ was y.

In order to show security of our transformation according to Definition 1 (see Section 3), we need to build a simulator that comes up with an indistinguishable environment for arbitrary adversaries $\mathcal{A}$. This simulation must provide answers to $\mathcal{A}$'s leakage queries $f \in \mathcal{L}$, which will be done as follows. For a leakage query $f \in \mathcal{L}$, the simulator comes up with an assignment of all the internal wires of $\widehat{C}$ that is consistent with the inputs and outputs of the circuit. This assignment is fed into f and the simulator returns the result to $\mathcal{A}$.

The computation of the wire assignment is quite simple: wire bundles that are between two gadgets will be assigned random values, and the internals of the gadgets will be simulated to be consistent with those random values (note that this will imply that the simulated outputs of $\mathcal{O}$ (and $\mathcal{Q}$) used within gadgets will no longer be encodings of 0). The wires that are used to encode the inputs of $\widehat{C}$ (in the $\widehat{\texttt{encoder}}$ gadget) and decode the outputs (in the $\widehat{\texttt{decoder}}$ gadget) will be simulated honestly, because the simulator knows its inputs and its outputs. The difficult part is showing that $\mathcal{A}$ cannot distinguish the real wire distribution from the simulated one when its access to the wire values is limited by functions available in the class $\mathcal{L}$.

A mental experiment – Security proof with leak-free gadgets. Conceptually, proving this indistinguishability proceeds in two steps. First, consider a mental experiment where each gadget in the transformed circuit $\widehat{C}$ is *perfectly opaque*. Namely, the only wires that the adversary $\mathcal{A}$ can "see" (via the leakage function f) are the *external wires* of the gadgets that connect the output of a gadget to the input of another gadget (these are exactly the wires that carry encodings of the values in the circuit C). The internals of the gadgets are off-limits to $\mathcal{A}$. Once in this (imaginary) world, we use the first key property of our gadget transformations presented in Figure 3 and 5, which is captured by the following definition.

Definition 3 (Rerandomizing). *Let C be a stateless circuit with n_{I} inputs, and no* `encoder` *or* `decoder` *gates. Let $\widehat{C}$ be the corresponding transformed circuit. We say that $\widehat{C}$ is* rerandomizing *if, for any fixed input* $(x_1, x_2, \ldots, x_{n_\mathrm{I}})$ *and its encoded input* $X \in \mathsf{Enc}(x_1, x_2, \ldots, x_{n_\mathrm{I}})$*, the encoded output* $\widehat{C}(X)$ *is distributed like* $\mathsf{Enc}(C(x_1, x_2, \ldots, x_{n_\mathrm{I}}))$*, i.e., independently of the particular encoding* X*.*

This definition applies, in particular, to single-gadget circuits. In Section 4.3 (for TR_C) and Section 4.4 (for TR_N), we show that all our gadgets are rerandomizing, i.e., the gadget's encoded output is uniformly distributed and independent from the gadget's encoded input. For a circuit $\widehat{C}$ composed of such gadgets, this implies that wire-bundles external to gadgets are distributed like $(\vec{w_1}, \ldots, \vec{w_m})$ where $\vec{w_i} \leftarrow \mathsf{Enc}(w_i)$ are *random and independent* encodings of the values $w_1, \ldots, w_m$ on the wires in C. In the mental experiment this observation suffices to simulate wire assignments without getting noticed by adversaries equipped with leakages in $\mathcal{L}$. More precisely, the simulator (who does not know the real w_is as they may depend on the secret state) uses an independent random vector $\vec{w_i}'$ for the external wire-bundles. By the leakage indistinguishability of the encoding scheme, such a change in the distribution will not get noticed by the adversary. By a hybrid argument, the same holds for a vector of *independent* encodings of m values as well.

Reduction. Before we declare victory (in this imaginary world), let us look a little more carefully at the hybrid argument. At each hybrid step, we will prove indistinguishability by a reduction to the security of the underlying encoding scheme. In other words, we will show by reduction that if $\mathcal{A}$ equipped with functions from $\mathcal{L}$ can distinguish two hybrid wire distributions, then some adversary $\mathcal{A}_\Pi$, equipped with functions from $\mathcal{L}_\Pi$, can distinguish two encodings. Given a target encoding, our reduction will need to fake the remaining wires of the circuit in a consistent way and give them as input to the function from $\mathcal{L}$ (notice that functions from $\mathcal{L}$ expect as input a full wire assignment for $\widehat{C}$).

Efficient reduction for computationally weak leakages. If $\mathcal{A}$ specifies a leakage function $f \in \mathcal{L}$ for $\widehat{C}$, then $\mathcal{A}_\Pi$ will specify its own leakage function $f_\Pi \in \mathcal{L}_\Pi$ for the target encoding and return its result to $\mathcal{A}$. Since $\mathcal{A}_\Pi$ has only access to its target encoding via f_Π, f_Π has to fake (in a way that will look real to f and $\mathcal{A}$) all the wires of $\widehat{C}$ before it can invoke f. At the same time, f_Π should not be much more complex than f, because our result is more meaningful when the difference between the power of $\mathcal{L}_\Pi$ and the power of $\mathcal{L}$ is smaller (recall that in Theorem 1 the difference between $\mathcal{L}$ and $\mathcal{L}_\Pi$ was described by $\mathsf{SHALLOW}(3, O(k^2))$. The main trick is for $\mathcal{A}_\Pi$ to *hardwire* as much as possible into f_Π, so that when f_Π observes the encoding, it has to do very little work before it can invoke f. In fact, in this imaginary situation, all the remaining wires of the hybrid wire distribution can be hardwired into f_Π because of the rerandomizing property (i.e., the encodings are independent), so f_Π has to simply invoke f on its input wires and hardwired values.

Local reduction for noisy leakages. To show security against noisy leakages, we show a reduction to the security of the underlying encoding scheme, similarly to the computationally weak case. In contrast to the computationally weak leakages, where the reduction needs to be as weak as the leakage function, we have no such constraint in the noisy case. However, the reduction should not get to see any wire too many times, because repeated observations of a noisy wire reduce the amount of uncertainty about the true value of the wire. Thus, from a single target encoding, we will fake the wire assignment of $\widehat{C}$ by using the target encoding as little as possible before feeding it into the noisy leakage function $\mathcal{N}_p$. (We will call such reductions "local"; see Section 4.1.) We ensure this by choosing most of the wires in $\widehat{C}$ *independently* of the target encoding. In fact, in our imaginary world in each hybrid the target encoding is only used once for $\vec{w}_i$ while all the other wires are independent. Since in such a case the adversary only gets a single noisy copy of the target encoding, by the $\mathcal{N}_p$ leakage indistinguishability of the parity encoding (cf. Proposition 2), she will not be able to tell apart two consecutive hybrid distributions.

The real world – Gadgets may leak. The second step in the proof is to move from the mental experiment to the real world, where the internals of the gadgets also leak. Unlike in the mental experiment, where the values of all wire bundles were independent, values of wires inside a gadget are correlated to its input and output wire bundles. Thus, they cannot be hardwired. Nor can they be efficiently (or locally) computed, because the complexity of the gadgets is too high. Instead, they will be simulated efficiently (or locally) in a way that is leakage-indistinguishable.

The simulation will take, as input, a gadget's input X and its output Y. Because our circuits are randomized, the simulation will be randomized, as well. The trick to getting an efficient (or local) simulator is to pick the randomness ahead of time, and then hardwire as many wire values into the simulator as possible without knowing X and Y. Thus, we define a "reconstructor" as a family of functions (one for each value of the randomness); the functions themselves will need to be very efficient (or local). However, the cost of sampling a function from such a family is incurred not inside the simulated leakage function, but rather by the simulated adversary; and thus it does not have to be efficient (or local), but merely polynomial-time.

Definition 4 (Reconstructor). *Let $\widehat{C}$ be a (transformed) stateless circuit. We say that a pair of strings (X, Y) is* plausible *for $\widehat{C}$ if $\widehat{C}$ might output Y on input X, i.e., if* $\Pr[\widehat{C}(X) = Y] > 0$.

Consider a distribution $\mathsf{REC}_{\widehat{C}}$ over the functions whose input is a plausible pair (X, Y), and whose output is an assignment to the wires of $\widehat{C}$. Define $\mathsf{REC}_{\widehat{C}}(X, Y)$ as the distribution obtained by sampling a function $R_{\widehat{C}}$ from $\mathsf{REC}_{\widehat{C}}$ and computing $R_{\widehat{C}}(X, Y)$. Such a distribution is called a $(\mathcal{L}, \tau, \epsilon)$-reconstructor for $\widehat{C}$ *if for any plausible (X, Y), the following two wire assignment distribution are $(\mathcal{L}, \tau, \epsilon)$-leakage-indistinguishable:*

- $\mathcal{W}_{\widehat{C}}(X|Y)$,
- $\mathsf{REC}_{\widehat{C}}(X, Y)$.

If the support of the distribution $\mathsf{REC}_{\widehat{C}}$ is in some set of functions $\mathcal{R}$, we say that $\widehat{C}$ is $(\mathcal{L}, \tau, \epsilon)$-reconstructible by $\mathcal{R}$.

Intuitively, if a circuit $\widehat{C}$ is $\mathcal{L}$-reconstructible in $\mathcal{R}$, then a random function in $\mathcal{R}$, given $\widehat{C}$'s encoded inputs X and outputs Y, can compute a wire assignment for $\widehat{C}$ that is $\mathcal{L}$-leakage indistinguishable from $\widehat{C}$'s real wire distribution (conditioned on the inputs being X and the outputs being Y). Putting it differently, reconstructibility of $\widehat{C}$ allows to simulate $\widehat{C}$'s internals from just knowing its inputs and outputs.

On a high-level, in the simulation we replace each gadget with its reconstructor in addition to replacing connecting wire bundles, i.e., the wires that go between gadgets, with random encodings. The proof that the simulation is indistinguishable requires first doing a hybrid argument over gadgets as they are replaced by reconstructors one-by-one, and then modifying the hybrid argument over the wire-bundles (replacing them by random encodings) as described above. In the hybrid argument over the wire-bundles, we can hardwire values for every wire in the circuit except the gadgets connected to the challenge encoding. Simulating the internals of these gadgets will be done using the reconstructor. We show that all gadgets in $\widehat{C}$ have either very efficient reconstructors (for the gadgets of TR_C refer to Section 4.3), or reconstructors that make very little use of their inputs (for the gadgets of TR_N refer to Section 4.4). We then show that (efficient or local) reconstructibility composes, which allows us to efficiently (or locally) reconstruct the internals of a complete circuit $\widehat{C}$ given only its inputs. Formally such composition is proven in Lemma 13 (cf. Section 4.5).

To conclude the proof we rely on the reconstructibility of $\widehat{C}$ to show security of the transformation according to Definition 1. Informally, the simulator replaces the secret state by random encodings and lets the reconstructors of the gadgets compute the internals of $\widehat{C}$ in a way that is consistent with the random state and the inputs and outputs of $\widehat{C}$. Again, we rely on the efficiency (or locality) of the reconstructors to keep our reductions as tight as possible.

A cautionary remark on our proofs. The proofs that we present in the next sections are technical and require a careful bookkeeping of the involved parameters, in particular, of the relation between the leakage classes $\mathcal{L}_\Pi$ and $\mathcal{L}$. This is necessary, if we want to make meaningful security statements about low-complexity leakages such as circuits of low-depth. In particular, we require that the reduction can be evaluated by low-depth circuits: If we start with some low-depth leakage $\mathcal{L}_\Pi$, and loose too much in the reduction, then $\mathcal{L}$ will become the empty set and no meaningful security statement can be made. Surprisingly, we show that our reductions are very efficient (namely, they can be computed by depth 3 circuits), even though they may have to "fake" computation carried out by deep and complex circuits.

Reader's guide to the security proof. For readers only interested in the high-level concept, we give the following reading guide. In Section 4.3 we present reconstructors for the gadgets of TR_C. We advise to read Lemma 5 as it gives a simple application of efficient reconstructors. The main technical part of Section 4.3 are Lemma 9 and Lemma 10. In Section 4.4 we present reconstructors for TR_N. A simple example of a local reconstructor is given in Lemma 11. The main technical part is Lemma 12, whose proof is moved to the appendix. In Section 4.5 we show that reconstructors compose. The outline from the last pages highlighted the main ideas of such composition. Finally, in Section 4.6 we discuss security of stateful circuits in Lemma 15 whose proof may safely be skipped. The proofs of the main Theorem 1 and Theorem 2 only sum up parameters and do not make use of any interesting new techniques.

4.3 Single Gadgets Reconstructors for TR_C

We show in this section that all single-gate gadgets from Figure 3 of the transformation TR_C have efficient reconstructors and are rerandomizing. The rerandomizing property follows immediately from the fact that every gadget's output is, as the last step of the gadget, masked by the output of $\mathcal{O}$. Therefore, in the following we focus to show existence of efficient reconstructors. Efficiency is described by the circuit class $\mathsf{SHALLOW}(d, s)$. If a reconstructor is in $\mathsf{SHALLOW}(d, s)$, then it can be computed by circuits of depth d and size s. We are particular interested in keeping d small as later we want to talk about AC^0 reductions (i.e., reductions that can be computed by constant depth circuits).

We show first existence of reconstructors for the simple gadgets of TR_C, namely, for $\widehat{\texttt{copy}}, \widehat{\texttt{mask}}$, $\widehat{\texttt{const}}_\alpha, \widehat{\$}, \widehat{\oplus}$, and $\widehat{\ominus}$. Except for the $\widehat{\oplus}$ reconstructor the proofs are moved to Appendix A.

Lemma 5 ($\widehat{\oplus}$ **and** $\widehat{\ominus}$ **gadgets of** TR_C **are reconstructible).** *Recall that k is the security parameter. The $\widehat{\oplus}$ and $\widehat{\ominus}$ gadgets are $(\mathcal{L}, \infty, 0)$-reconstructible by $\mathsf{SHALLOW}(2, O(k))$ for any $\mathcal{L}$.*

Proof. We will do the proof for $\widehat{\oplus}$; the proof for $\widehat{\ominus}$ is similar. The reconstructor $\mathsf{REC}_{\widehat{\oplus}}$ is the distribution whose only support is the following circuit $R_{\widehat{\oplus}}$. On inputs (X, Y) where $X = (\vec{a}, \vec{b})$ (i.e., the desired input of the $\widehat{\oplus}$ gate), and $Y = (\vec{c})$ (i.e., its desired output), $R_{\widehat{\oplus}}$ assigns the wires of $\widehat{\oplus}$ to $\vec{q} = \vec{a} \oplus \vec{b}$ and $\vec{o} = \vec{c} \ominus \vec{q}$.

If X, Y are chosen as in the definition of a reconstructor (i.e., they are plausible), then the resulting output of $R_{\widehat{\oplus}}(X, Y)$ is identically distributed to the wire distribution $\mathcal{W}_{\widehat{\oplus}}(X|Y)$, since in both cases $\vec{o}$ takes the only possible consistent value $\vec{o} = \vec{c} \ominus \vec{q}$. Notice that $R_{\widehat{\oplus}}$ can be computed by a circuit of depth 2 because on inputs X, Y it first will compute $\vec{q} = \vec{a} \oplus \vec{b}$ and based on that $\vec{o} = \vec{c} \ominus \vec{q}$. The $\ominus$ and $\oplus$ gates above operate only on single field elements, so $R_{\widehat{\oplus}}$ requires $O(k)$ size. □

Lemma 6 ($\widehat{\$}$ **of** TR_C **is reconstructible).** *The $\widehat{\$}$ gadget is $(\mathcal{L}, \infty, 0)$-reconstructible by $\mathsf{SHALLOW}(0, O(k))$ for any $\mathcal{L}$.*

Lemma 7 ($\widehat{\texttt{copy}}, \widehat{\texttt{mask}}$**, and** $\widehat{\texttt{const}_\alpha}$ **of** TR_C **are reconstructible).** *The $\widehat{\texttt{copy}}$ gadget, the $\widehat{\texttt{mask}}$ gadget, and, for every $\alpha \in \mathcal{K}$, the $\widehat{\texttt{const}_\alpha}$ gadget are $(\mathcal{L}, \infty, 0)$-reconstructible by $\mathsf{SHALLOW}(1, O(k))$, for any $\mathcal{L}$.*

We are now going to prove the reconstructibility of the $\widehat{\odot}$ gadget. For our result to be more meaningful it is of vital importance that our simulation is efficient. Presenting such an efficient simulator for the $\widehat{\odot}$ gadget, is the main technical difficulty of this section, since $\widehat{\odot}$ is a deep, complex circuit. But before we present our shallow simulator in Lemma 9, we first prove a simple technical lemma which relates two leakage-indistinguishability statements using a "shallow" wire simulator f_S.

Lemma 8. *Let $\mathcal{W}_0, \mathcal{W}_0'$ be distributions over $\mathcal{K}^n$ for some $n > 0$.*[10] *Let F be a distribution over n-input functions in some class $\mathcal{L}$. For $f_S \leftarrow F$ define the following distributions:*

$$\mathcal{W}_1 \equiv f_S(\mathcal{W}_0) \quad \textit{and} \quad \mathcal{W}_1' \equiv f_S(\mathcal{W}_0'). \tag{5}$$

Let $\mathcal{L}_0$ be a class of leakage functions and let $\epsilon_0 > 0$, $\tau_0 > 0$. If $\mathcal{W}_0$ and $\mathcal{W}_0'$ are $(\mathcal{L}_0, \tau_0, \epsilon_0)$-leakage-indistinguishable, then $\mathcal{W}_1$ and $\mathcal{W}_1'$ are $(\mathcal{L}_1, \tau_1, \epsilon_1)$-leakage-indistinguishable. Here, for any $\mathcal{L}_1$ that satisfies $\mathcal{L}_1 \circ \mathcal{L} \subseteq \mathcal{L}_0$, $\epsilon_0 = \epsilon_1$, and $\tau_0 - \tau_1$ is the time needed to sample from F.

Proof. We show by contradiction that for all adversaries $\mathcal{A}_1$ running in time at most τ_1

$$|\Pr[\mathcal{A}_1^{\mathsf{Eval}_1(\mathcal{W}_1, \cdot)} = 1] - \Pr[\mathcal{A}_1^{\mathsf{Eval}_1(\mathcal{W}_1', \cdot)} = 1| \leq \epsilon_1, \tag{6}$$

where Eval_1 can be queried once by $\mathcal{A}_1$ with a leakage function $f_1 \in \mathcal{L}_1$, where $\mathcal{L}_1$ satisfies $\mathcal{L}_1 \circ \mathcal{L} \subseteq \mathcal{L}_0$.

Suppose for contradiction that (6) is violated for some $(\mathcal{L}_1, \tau_1)$-adversary $\mathcal{A}_1$, then we construct an $(\mathcal{L}_0, \tau_0)$-adversary $\mathcal{A}_0$ that breaks the leakage-indistinguishability of the distributions $\mathcal{W}_0$ and $\mathcal{W}_0'$. The adversary $\mathcal{A}_0$ will invoke $\mathcal{A}_1$ as a subroutine, answering $\mathcal{A}_1$'s leakage query and eventually

[10]In our case, these will be wire assignments to a circuit with n wires. Notice that this can also just be a single encoding.

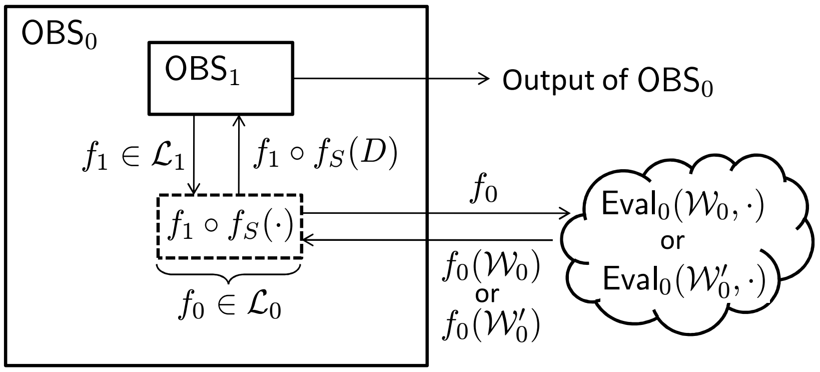
OBS_0
OBS_1
Output of OBS_0
$f_1 \in \mathcal{L}_1$
$f_1 \circ f_S(D)$
$f_1 \circ f_S(\cdot)$
$f_0 \in \mathcal{L}_0$
f_0
$f_0(\mathcal{W}_0)$ or $f_0(\mathcal{W}_0')$
$\mathsf{Eval}_0(\mathcal{W}_0,\cdot)$ or $\mathsf{Eval}_0(\mathcal{W}_0',\cdot)$

Figure 6: Outline of the reduction in Lemma 8

outputting whatever $\mathcal{A}_1$ outputs (see Figure 6). To answer the leakage query $f_1 \in \mathcal{L}_1$, the adversary $\mathcal{A}_0$ will use its own oracle Eval_0. The difficulty is that Eval_0 evaluates a leakage function $f_0 \in \mathcal{L}_0$ on a sample either from $\mathcal{W}_0$ or $\mathcal{W}_0'$, whereas $\mathcal{A}_1$ produces a query f_1 to be evaluated on a (possibly much larger) wire assignment sampled from $\mathcal{W}_1$ or $\mathcal{W}_1'$.

We address this by using a function f_S, drawn from the distribution F, that takes as input a single "challenge" that is either sampled from $\mathcal{W}_0$ or $\mathcal{W}_0'$ and outputs a full wire assignment from either $\mathcal{W}_1$ or $\mathcal{W}_1'$, respectively. To recap, $\mathcal{A}_0$ lets $\mathcal{A}_1$ choose $f_1 \in \mathcal{L}_1$, and draws a function f_S from F. It then queries Eval_0 on $f_0 = f_1 \circ f_S$ and forwards the answer back to $\mathcal{A}_1$. Finally, if $\mathcal{A}_1$ returns a bit b, then $\mathcal{A}_0$ outputs b as its own guess.

To analyze the distinguishing advantage of $\mathcal{A}_0$, consider the following two cases:

$$\begin{aligned}
\Pr[\mathcal{A}_0^{\mathsf{Eval}_0(\mathcal{W}_0,\cdot)} = 1] &= \Pr[\mathcal{A}_1^{\mathsf{Eval}_1(f_S(\mathcal{W}_0),\cdot)} = 1] \overset{(5)}{=} \Pr[\mathcal{A}_1^{\mathsf{Eval}_1(\mathcal{W}_1,\cdot)} = 1] \\
\Pr[\mathcal{A}_0^{\mathsf{Eval}_0(\mathcal{W}_0',\cdot)} = 1] &= Pr[\mathcal{A}_1^{\mathsf{Eval}_1(f_S(\mathcal{W}_0'),\cdot)} = 1] \overset{(5)}{=} \Pr[\mathcal{A}_1^{\mathsf{Eval}_1(\mathcal{W}_1',\cdot)} = 1]
\end{aligned}$$

By taking the difference and with Eqn. (6) we get

$$|\Pr[\mathcal{A}_0^{\mathsf{Eval}_0(\mathcal{W}_0,\cdot)} = 1] - \Pr[\mathcal{A}_0^{\mathsf{Eval}_0(\mathcal{W}_0',\cdot)} = 1| \le \epsilon_1,$$

which yields that $\epsilon_0 = \epsilon_1$. Observe also that $f_0 \in \mathcal{L}_0$ (i.e., the reduction does not lose much in the leakage function's power): since $f_S \in \mathcal{L}$ indeed we have that $f_0 = f_1 \circ f_S \in \mathcal{L}_1 \circ \mathcal{L} \subseteq \mathcal{L}_0$. Finally, note that the only extra time $\mathcal{A}_0$ spends (i.e., $\tau_0 - \tau_1$) is the time required to sample from the distribution F. □

Let us now show that the $\widehat{\odot}$ gadget is reconstructible by shallow circuits. The lemma below describes the reconstructor for $\widehat{\odot}$ and gives the high-level idea of the proof; the technical details are moved to Lemma 10.

Lemma 9 (**$\widehat{\odot}$ of TR_C is reconstructible).** *Let $\mathcal{L}_\Pi$ be a class of leakage functions and let $\tau > 0, \epsilon > 0$. If Π is $(\mathcal{L}_\Pi, \tau, \epsilon)$-leakage-indistinguishable, then the $\widehat{\odot}$ gadget is $(\mathcal{L}, \tau - O(k^2), k\epsilon)$-reconstructible by $\mathsf{SHALLOW}(2, O(k^2))$, for any $\mathcal{L}$ that satisfies $\mathcal{L} \circ \mathsf{SHALLOW}(3, O(k^2)) \subseteq \mathcal{L}_\Pi$ (and if $\mathcal{K} = \mathsf{GF}(2)$, then $\mathcal{L} \circ \mathsf{SHALLOW}(2, O(k^2)) \subseteq \mathcal{L}_\Pi$).*

Proof of Lemma 9. We first describe the reconstructor $\mathsf{REC}_{\widehat{\odot}}$ for $\widehat{\odot}$ gadgets, and then prove that it is indistinguishable from a wire assignment of a real evaluation of $\widehat{\odot}$ conditioned on plausible inputs X and outputs Y.

$R_{\widehat{\odot}}$ sampled from the reconstructor $\mathsf{REC}_{\widehat{\odot}}$ takes as inputs plausible values (X, Y), where $X = (\vec{a}, \vec{b})$ (i.e., the desired input of the $\widehat{\odot}$ gate), and $Y = (\vec{c})$ (i.e., its desired output) and is defined as follows:

1. Sample $\vec{U}$ uniformly from $\mathcal{K}^{k\times k}$ and compute the values on the wires in the subsequent decoding sub-circuits for the computation of $\vec{q}$. (Note that this procedure need not be in low depth, because it is done as part of sampling $R_{\widehat{\odot}}$, rather than by $R_{\widehat{\odot}}$ itself.) Hardwire the results as $R_{\widehat{\odot}}$'s outputs.

2. On input X, $R_{\widehat{\odot}}$ computes the matrix $\vec{B} = (a_i \odot b_j)_{i,j\in[1,k]}$ and outputs it as part of the wire assignment.

3. $R_{\widehat{\odot}}$ computes online $\vec{S} = \vec{B} \ominus \vec{U}$ and $\vec{o} = \vec{c} \ominus \vec{q}$ (i.e., once using $\vec{B}$ that depends on input X and once using the input $Y = \vec{c}$).

Circuits sampled from $\mathsf{REC}_{\widehat{\odot}}$ have size $O(k^2)$ (because they need to compute matrices $\vec{B}$ and $\vec{S}$) and depth 2, because $\vec{S}$ is computed from $\vec{B}$, that in turn has been computed from the inputs.

It remains to show that if X, Y are chosen as in the definition of reconstructors, then $R_{\widehat{\odot}}(X, Y)$ and $\mathcal{W}_{\widehat{\odot}}(X|Y)$ are $(\mathcal{L}, \tau - O(k^2), k\epsilon)$-leakage-indistinguishable. The reconstructor from above differs from the real wires assignment in that $\vec{U}$ is a random matrix (instead of being $\vec{B} \oplus \vec{S}$, where $\vec{S}$ is a matrix whose columns decode to 0). In the following we will show that one can replace the matrix $\vec{S}$ by a matrix sampled uniformly at random from $\mathcal{K}^{k\times k}$. Since $\vec{U}$ is computed as $\vec{B} \oplus \vec{S}$ (i.e., for random $\vec{S}$ the matrix $\vec{U}$ is random as required by the reconstructor distribution) this will give us the desired property of reconstructibility and concludes the proof.

We prove that $\vec{S}$ can be replaced by a random matrix using a hybrid argument. We define hybrid distributions $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$ $(\ell \in [0, k])$ as $\mathcal{W}_{\widehat{\odot}}(X|Y)$, except that for the first ℓ columns of $\vec{S}$ the elements are drawn uniformly from $\mathcal{K}$. It is easy to see that the 0th hybrid uses matrix $\vec{S}$ as defined in our construction and the kth hybrid distributions uses a matrix $\vec{S}$ drawn uniformly at random. We show the leakage-indistinguishability between two consecutive hybrids by a reduction to the encoding leakage-indistinguishability. More precisely, we will show that for all $\ell \in [1, k]$ and all plausible X, Y, $\mathcal{W}^{\ell-1}_{\widehat{\odot}}(X|Y)$ and $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$ are $(\mathcal{L}, \tau - O(k^2), \epsilon)$-leakage-indistinguishable.

To this end in the next technical lemma, we show for any $\ell \in [1, k]$ and any X the existence of a distribution F^{ℓ} of functions in $\mathsf{SHALLOW}(3, O(k^2))$ samplable in time $O(k^2)$ that take as input a single encoding and map it either to $\mathcal{W}^{\ell-1}_{\widehat{\odot}}(X|Y)$ or $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$, depending on whether the given encoding was an encoding of 0 or of a random value. By applying Lemma 8 to Lemma 10 below (setting $\mathcal{W}_0 = \mathsf{Enc}(0), \mathcal{W}'_0 = \mathsf{Enc}(\cdot)$) we get that $\mathcal{W}^{\ell-1}_{\widehat{\odot}}(X|Y)$ and $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$ are $(\mathcal{L}, \tau - O(k^2), \epsilon)$-leakage-indistinguishable, where $\mathcal{L} \circ \mathsf{SHALLOW}(3, O(k^2)) \subseteq \mathcal{L}_{\Pi}$. Using triangle inequality we get together with the k hybrids that $\mathcal{W}_{\widehat{\odot}}(X|Y)$ and $R_{\widehat{\odot}}(X, Y)$ are $(\mathcal{L}, \tau - O(k^2), k\epsilon)$-leakage-indistinguishable, if Π is $(\mathcal{L}_{\Pi}, \tau, \epsilon)$-leakage-indistinguishable. This concludes the proof. $\square$

The following technical lemma proves the existence of the distribution F^{ℓ} used above in Lemma 9 and may be skipped by the reader.

Lemma 10. *For any $\ell \in [1,k]$ and any plausible $X = (\vec{a}, \vec{b}), Y = (\vec{c})$, there exists a distribution F^ℓ over functions in $\mathsf{SHALLOW}(3, O(k^2))$ (if $\mathcal{K} = \mathsf{GF}(2)$ then $\mathsf{SHALLOW}(2, O(k^2))$) that take as input a single encoding and output a wire assignment for $\widehat{\odot}$, such that for $f_S \leftarrow F^\ell$:*

$$\mathcal{W}_{\widehat{\odot}}^{\ell-1}(X|Y) \equiv f_S(\mathsf{Enc}(0)), \tag{7}$$

$$\mathcal{W}_{\widehat{\odot}}^{\ell}(X|Y) \equiv f_S(\mathsf{Enc}(\cdot)). \tag{8}$$

Proof. $f_S \leftarrow F^\ell$ on input an encoding $\vec{e}$ shall output a full wire assignment of $\widehat{\odot}$, with $\vec{e}$ embedded into the ℓth column of $\vec{S}$, and with the correct distribution on the remaining wire values. This guarantees that if the target encoding $\vec{e}$ is drawn uniformly-and-independently from $\mathsf{Enc}(0)$ then $f_S(\vec{e})$ is distributed identically to the hybrid wire distribution $\mathcal{W}_{\widehat{\odot}}^{\ell-1}(X|Y)$. On the other hand, if $\vec{e}$ is drawn uniformly-and-independently from $\mathsf{Enc}(\cdot)$, then $f_S(\vec{e})$ is distributed identically to $\mathcal{W}_{\widehat{\odot}}^{\ell}(X|Y)$.

The difficulty is that f_S must have small (constant) depth, but needs to output a wire assignment for the deep circuit $\widehat{\odot}$. We solve this problem by hard-wiring most of the resulting wire assignment directly into f_S. The only parts of the wire assignment that cannot be hardwired are those that depend on the input $\vec{e}$, but fortunately, they can be easily computed (indeed, this was the main goal in designing the $\widehat{\odot}$ gadget).

Concretely, the distribution F^ℓ is defined by drawing f_S as follows:

1. From given $X = (\vec{a}, \vec{b})$ compute consistently the matrix $\vec{B} = (a_i b_j)_{i,j \in [1,k]}$ and hardwire $\vec{a}, \vec{b}, \vec{B}$ into f_S.

2. Most columns of $\vec{S}$ are hardwired into f_S: left of the ℓth column they are drawn at random, and right of the ℓth column they are drawn from $\mathsf{Enc}(0)$. Only the ℓth column depends on the input and is filled with the challenge encoding $\vec{e}$.

3. Using $\vec{B}$ and $\vec{S}$ hardwire all elements of $\vec{U} = \vec{B} \oplus \vec{S}$ into f_S except for the ℓth column. For the ℓth column, f_S computes on input $\vec{e}$, for each $i \in [1,k]$, the value $U_{i,\ell} = B_{i,\ell} \oplus e_i$.

4. Consider, for $i \in [1,k]$, the decoding subcircuits in $\widehat{\odot}$ that compute $\vec{q}$ with values from $\vec{U}$. In each subcircuit the wires carry the linear combination of $\{U_{i,j}\}_j$, plus possibly a constant. If this linear combination does not depend on $U_{i,\ell}$ (i.e., the input to f_S), then pre-compute this wire and hardwire the result into f_S. On the other hand, if it does depend on $U_{i,\ell} = B_{i,\ell} \oplus e_i$, then pre-compute the partial linear combination except the term that depends on e_i and hardwire the result into the description of f_S. On input $\vec{e}$, f_S computes the missing outputs by $\oplus$-ing the partial linear combination with the missing term (which is e_i times a constant).

5. With fixed Y and $\vec{q}$ from the previous step compute $\vec{o} = Y \ominus \vec{q}$ and output it.

Let us first consider the outputs of f_S that are independent of $\vec{e}$. In $\mathcal{W}_{\widehat{\odot}}^{\ell-1}(X|Y)$ and $\mathcal{W}_{\widehat{\odot}}^{\ell}(X|Y)$ the first $\ell - 1$ columns in $\vec{S}$ are independently-and-uniformly drawn from $\mathsf{Enc}(\cdot)$, whereas the last $k - \ell - 1$ columns are sampled from $\mathsf{Enc}(0)$. The other hardwired outputs that do not depend on $\vec{e}$, are computed honestly from X, Y and $\vec{S}$, thus with respect to only these values, $\mathcal{W}_{\widehat{\odot}}^{\ell-1}(X|Y)$, $\mathcal{W}_{\widehat{\odot}}^{\ell}(X|Y)$ and the outputs of f_S are identically distributed. If on the other hand an output of f_S depends on $\vec{e}$ we distinguish two cases:

1. $\vec{e} \leftarrow \mathsf{Enc}(0)$: This means the ℓth column of $\vec{S}$ is assigned an encoding drawn from $\mathsf{Enc}(0)$. Together with the observation that all remaining wires are computed honestly using $\vec{S}$ and $\vec{B}$, we get that $f_S(\mathsf{Enc}(0))$ and $\mathcal{W}^{\ell-1}_{\widehat{\odot}}(X|Y)$ are distributed identically.

2. $\vec{e} \leftarrow \mathsf{Enc}(\cdot)$: Here, the ℓth column of $\vec{S}$ is assigned a random value in $\mathcal{K}^k$. With the same observation as above we get that $f_S(\mathsf{Enc}(\cdot))$ and $\mathcal{W}^{\ell}_{\widehat{\odot}}(X|Y)$ are distributed identically.

It is clear that functions from F^ℓ can be sampled in time $O(k^2)$. It remains to show that they can indeed be computed by shallow circuits. The input to f_S is used to adjust the ℓth column of $\vec{U}$, which requires a circuit of depth 1 and size k. Additionally, adjusting the values in the subcircuits for the computation of $\vec{q}$ requires computation of depth 2 (for the computation of e_i times a constant and $\oplus$-ing it) and $O(k)$ size. Finally, once $\vec{q}$ is evaluated, f_S needs to compute $\vec{o}$ which increases the depth by 1. Overall, we get circuits of size $O(k^2)$ and depth 3. In the case of $\mathsf{GF}(2)$, there is no need to multiply e_i by a constant, so the depth is only 2. $\square$

4.4 Single Gadgets Reconstructors for TR_N

In this section, we prove that the gadgets of TR_N from Figure 5 are rerandomizing and reconstructibile. We only present statements and proofs for reconstructibility of the $\widehat{\oplus}$ and $\widehat{\odot}$ gadgets; for the simpler gadgets, the reconstructors are the same as in the previous section; it is easy to see that those are all within $\mathsf{LOCAL}(3)$.

Gadgets of TR_N are rerandomizing. By inspection of our gadgets, it is easy to see that they satisfy the rerandomizing property: similarly to the gadgets of TR_C, the reason is that the output of each gadget is masked with the output of the opaque gate $\mathcal{O}$.

Gadgets of TR_N are reconstructible. On a high-level, the reconstructors for the gadgets of TR_N follow the reconstructor constructions from Section 4.3. The main difference is that in the noisy case, we are not concerned about the *computational efficiency* of the reconstructor, but rather the *number of times the reconstructor's output depends on its input bits.* That is, given the encoded inputs and outputs of a gadget, the reconstructor has to simulate the internals in a way that looks indistinguishable (from the real wire distribution) to noisy leakage $\mathcal{N}_p$. Since the gadget's internal wires depend on its encoded inputs and outputs, the reconstructor will need to use them to compute the remaining wires of the gadget. We say that a reconstructor operates *locally* if it does not have to use its inputs (i.e., the encoded inputs and outputs of the gadgets) too often, which essentially means that for a large gadget most of its internal wires are independent of the encoded inputs and outputs, and, hence, can be hardwired into the reconstructor. Because local functions computed on $\mathsf{Enc}(0)$ and $\mathsf{Enc}(1)$ are indistinguishable through $\mathcal{N}_p$, our reconstructor will work.

We show that the gadgets of TR_N exhibit such a locality property. In particular, we prove that to reconstruct the $\widehat{\oplus}$ gadget, the reconstructor needs to use its inputs only 3 times, and can hardwire the remaining wires. We show similar locality for the large $\widehat{\odot}$ gadget.

Importance of locality. One may ask why locality of reconstruction is such an important property in the presence of noisy leakage $\mathcal{N}_p$. To explain this, let us go back to the outline of the security proof from Section 4.2. To show security of TR_N according to Definition 1, we need to build a simulator $\mathcal{S}$ that answers the adversary's leakage queries $\mathcal{N}_p$. Since $\mathcal{S}$ does not know the circuit's secret state, it will use random encodings instead. We show by reduction to the leakage indistinguishability of Π_{parity} that such a change in the wire distribution (namely, replacing the real secret state with random encodings) will not get noticed by the adversary. To this end, we put the target encoding in the secret state and let the reconstructor for $\widehat{C}$ (that we will construct in the next section by composition from reconstructors of single gadgets) simulate all of $\widehat{C}$'s internals. If the reconstructor for $\widehat{C}$ works in a local way, i.e., most of the wires in $\widehat{C}$ are independent of the target encoding, then by Proposition 2 the statistical distance between the wire distribution using the correct state and the simulated distribution using random encodings is small. To formally describe such locality, we use the function class $\mathsf{LOCAL}(\ell)$ formally described in Section 4.1. We remind the reader that functions in $\mathsf{LOCAL}(\ell)$ allow each input bit to affect at most ℓ output bits.

Formal Statements. Let us first describe reconstructibility of the $\widehat{\oplus}$ gadget.

Lemma 11 ($\widehat{\oplus}$ gadgets of TR_N are reconstructible). *The $\widehat{\oplus}$ gadget is $(\mathcal{L}, \infty, 0)$-reconstructible by $\mathsf{LOCAL}(3)$ for any $\mathcal{L}$.*

Proof. The reconstructor $\mathsf{REC}_{\widehat{\oplus}}$ is the distribution whose only support is the following circuit $R_{\widehat{\oplus}}$. On inputs (X, Y) where $X = (\vec{a}, \vec{b})$, and $Y = (\vec{c})$, $R_{\widehat{\oplus}}$ assigns the wires of $\widehat{\oplus}$ to $\vec{q} = \vec{a} \oplus \vec{b}$ and $\vec{o} = \vec{c} \oplus \vec{q}$.

If X, Y are chosen as in the definition of a reconstructor (i.e., they are plausible inputs), then the resulting output of $R_{\widehat{\oplus}}(X, Y)$ is distributed as the real wire distribution $\mathcal{W}_{\widehat{\oplus}}(X|Y)$, since in both cases $\vec{o}$ takes the only possible consistent value $\vec{o} = \vec{c} \oplus \vec{q}$.

It remains to show that $R_{\widehat{\oplus}}$ is in $\mathsf{LOCAL}(3)$. We must show that for each input $\vec{a}, \vec{b}, \vec{c}$ (when the remaining inputs are fixed) the output of the reconstructor is either a fixed constant or can be written as the free input plus a constant vector. For input $\vec{a}$ the inputs $\vec{b}$ and $\vec{c}$ are fixed constants and hence, the output of $R_{\widehat{\oplus}}$ is constant except $(\vec{a}, \vec{q} := \vec{a} \oplus \vec{b}, \vec{o} := \vec{a} \oplus (\vec{b} \oplus \vec{c}))$. The same analysis works for $\vec{b}$. For $\vec{c}$ observe that $\vec{a}$ and $\vec{b}$ are fixed and all outputs are constant except $(\vec{c}, \vec{c} \oplus \vec{q})$. Hence, we get that $R_{\widehat{\oplus}} \in \mathsf{LOCAL}(3)$. □

The above lemma works for arbitrary leakage classes $\mathcal{L}$. Later, we set $\mathcal{L} = \mathcal{N}_p$ to get reconstruction for $\widehat{\oplus}$ that is resilient to noisy leakages.

The reconstructibility of $\widehat{\odot}$ is significantly more complicated. Indeed, we will only fully establish it in the next section, when we show composability of reconstructors. $\widehat{\odot}$ is built from two sub-gadgets: $\widehat{\texttt{mult}}$ and $\widehat{\texttt{compress}}$, where $\widehat{\texttt{compress}}$ itself is composed of $\widehat{\oplus}$ gadgets. Hence, to apply composability and show existence of a reconstructor for $\widehat{\odot}$ (cf. Lemma 14 in Section 4.5) it remains to show reconstructibility of $\widehat{\texttt{mult}}$. Indeed, showing this is the main technical part of this section and is given in the lemma below, whose proof is moved to Appendix B.

Lemma 12 ($\widehat{\texttt{mult}}$ gadgets of TR_N are reconstructible). *For every $p \in (0, \frac{1}{2}]$, the $\widehat{\texttt{mult}}$ gadget is $(\mathcal{N}_p, \infty, \epsilon(k))$-reconstructible by $\mathsf{LOCAL}(2)$, where $\epsilon(k) \leq (2k+1)(\exp(-15kp^5) + \exp(-k/512))$.*

4.5 Multi-Gadget Circuit Reconstructors

In the previous two sections we showed reconstructors for the gadgets used by the transformations TR_C and TR_N. In this section, we are interested in composing such reconstructors to obtain a multi-gadget circuit reconstructor, i.e., we show the existence of reconstructors for arbitrary complex circuits. To keep our composition lemma below as general as possible, we do not focus on our transformations TR_C or TR_N but rather prove reconstructor composition for arbitrary encoding-based circuit transformations.

Recall that in an encoding-based circuit transformation each wire w in the original circuit C is represented by a wire bundle in $\widehat{C}$, consisting of k wires $\vec{w} = (w_1, \ldots, w_k)$, that carry an encoding of w. The gates in C are replaced gate-by-gate with so called gadgets, computing on encoded values. For the detailed description of encoding based circuit transformations we refer to Section 3.1.

In this section, we consider transformed circuits $\widehat{C}$ without $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ gadgets, i.e., we assume that $\widehat{C}$'s inputs are already given in encoded form, and the outputs are not explicitly decoded. The reason for this restriction is that the $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ gadgets are by definition not reconstructible, since reconstructors are only defined for gadgets that take encoded values as inputs and output encoded values.

Lemma 13 (Reconstructor composition for encoding-based circuits). *Let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be any (not necessarily linear) encoding scheme that is $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable for some $\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi$. Let* TR *be an encoding based circuit transformation and suppose that each corresponding gadget, $\widehat{g}$, is rerandomizing and $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-reconstructible by $\mathcal{R}_{\widehat{g}}$, for some $\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}}$. Then for any* stateless *circuit C of size s with n_I inputs, n_O outputs, and m wires, $\widehat{C} \leftarrow \mathsf{TR}(C)$ is rerandomizing and $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{C}}, \epsilon_{\widehat{C}})$-reconstructible by $\mathcal{R}_{\widehat{C}}$, for*

- *any $\mathcal{L}_{\widehat{C}}$ that satisfies $(\mathcal{L}_{\widehat{C}} \circ (2 \times \mathcal{R}_{\widehat{g}})) \subseteq \mathcal{L}_\Pi$ (where $(2 \times \mathcal{R}_{\widehat{g}})$ denotes two parallel executions of $\mathcal{R}_{\widehat{g}}$) and $\mathcal{L}_{\widehat{C}} \subseteq \mathcal{L}_{\widehat{g}}$,*
- *any $\tau_{\widehat{C}} \le \min(\tau_\Pi, \tau_{\widehat{g}}) - s\tau_{\mathsf{samp}}$, where τ_{samp} is the maximum time to sample $R_{\widehat{g}} \leftarrow \mathsf{REC}_{\widehat{g}}$ for all gadgets $\widehat{g}$.*[11]
- *some $\epsilon_{\widehat{C}} \le m\epsilon_\Pi + s\epsilon_{\widehat{g}}$,*
- *$\mathcal{R}_{\widehat{C}} \subseteq (n_\mathrm{I} + n_\mathrm{O}) \times \mathcal{R}_{\widehat{g}}$; moreover, if $\mathcal{R}_{\widehat{g}} \subseteq \mathsf{LOCAL}(\ell)$, then also $\mathcal{R}_{\widehat{C}} \subseteq \mathsf{LOCAL}(\ell)$*

Before we give the proof, let us discuss an interpretation of the important parameters. To apply the lemma we require

- a $\mathcal{L}_\Pi$-leakage-indistinguishable encoding scheme, and
- that all gadgets in $\widehat{C}$ are $\mathcal{L}_{\widehat{g}}$-reconstructible by $\mathcal{R}_{\widehat{g}}$.

If that is given, then by the composition lemma it is guaranteed that $\widehat{C} \leftarrow \mathsf{TR}(C)$ is $\mathcal{L}_{\widehat{C}}$-reconstructible for any class $\mathcal{L}_{\widehat{C}}$ that satisfies

- $(\mathcal{L}_{\widehat{C}} \circ (2 \times \mathcal{R}_{\widehat{g}})) \subseteq \mathcal{L}_\Pi$,
- $\mathcal{L}_{\widehat{C}} \subseteq \mathcal{L}_{\widehat{g}}$.

Or put otherwise: if we want that $\widehat{C}$ is $\mathcal{L}_{\widehat{C}}$-reconstructible, for some $\mathcal{L}_{\widehat{C}}$, then we need an encoding scheme that "tolerates" at least functions from $(\mathcal{L}_{\widehat{C}} \circ (2 \times \mathcal{R}_{\widehat{g}}))$ and all the gadgets have to be at least $\mathcal{L}_{\widehat{C}}$-reconstructible.

[11]For simplicity we assume that τ_{samp} is larger than the maximal time to compute gadgets $\widehat{g}$.

Proof of Lemma 13. Let $\widehat{C}$ be the transformed circuit, with inputs denoted $X = (\vec{x}_1, \ldots, \vec{x}_{n_{\mathrm{I}}})$ and outputs denoted $Y = (\vec{y}_1, \ldots, \vec{y}_{n_{\mathrm{O}}})$.Let *first gadgets* denote the set of topologically-first gadgets in $\widehat{C}$, and let *last gadgets* denote the set of topologically-last gadgets in $\widehat{C}$. The wires that go between gadgets (i.e., not directly connected to X or Y, and not part of the innards of some gadget) are called *connecting wires*. (We will assume, in order to simplify the proof, that for every two-input gadget, either both inputs are connected to X, or both inputs are connected to internal wires. Moreover, we will assume that no gadget is connected to both X and Y. These assumptions ensure that a first gadget has no connecting wires going into it and that a gadget cannot be simultaneously first and last. They are without loss of generality, because they can be easily satisfied by adding copy gates to the inputs as needed.)

The fact that $\widehat{C}$ is rerandomizing follows immediately from the fact that the last gadgets are rerandomizing, and the randomness used in each gadget is independent.

The reconstructor $\mathsf{REC}_{\widehat{C}}$ is a distribution over circuits $\mathcal{R}_{\widehat{C}}$ with inputs (X, Y). We define $\mathsf{REC}_{\widehat{C}}$, with $R_{\widehat{C}} \leftarrow \mathsf{REC}_{\widehat{C}}$, for input (X, Y) that is plausible for $\widehat{C}$, as follows:

1. For each $\widehat{g}$ gadget in $\widehat{C}$, sample $R_{\widehat{g}} \leftarrow \mathsf{REC}_{\widehat{g}}$.

2. For each connecting wire bundle, sample a random encoding of a random field element, i.e., $\vec{v} \leftarrow \mathsf{Enc}(v)$ with $v \leftarrow \mathcal{K}$.

3. For each gadget $\widehat{g}$ in $\widehat{C}$ except for the first gadgets and last gadgets, pre-compute $R_{\widehat{g}}(U, V)$ and hardwire the result into $R_{\widehat{C}}$. Here, U (resp., V) are the encodings assigned above to the wire bundles that are the inputs (resp., outputs) of $\widehat{g}$.

4. On input (X, Y) the reconstructor $R_{\widehat{C}}$ computes the reconstructors of all the first and last gadgets. For the first gadgets, the input wire bundles are given in X and the outputs have been hardwired above. Similarly, for the last gadgets, the inputs have been hardwired and the outputs are given in Y.

We now analyze the class of the reconstructor $\mathsf{REC}_{\widehat{C}}$. For a circuit C with n_{I} inputs and n_{O} outputs, $R_{\widehat{C}} \leftarrow \mathsf{REC}_{\widehat{C}}$ on inputs (X, Y) only needs to compute $n_{\mathrm{I}} + n_{\mathrm{O}}$ reconstructors (for the first gadgets and last gadgets). Hence, $\mathsf{REC}_{\widehat{C}}$ lies in $(n_{\mathrm{I}} + n_{\mathrm{O}}) \times \mathcal{R}_{\widehat{g}}$ as claimed in the statement. Moreover, each input of $R_{\widehat{C}}$ is used only in a single gadget reconstructor, and thus locality provided by gadget reconstructors is preserved by $R_{\widehat{C}}$.

It remains to show that for any plausible input/output pair (X, Y), $\mathsf{REC}_{\widehat{C}}(X, Y)$ is $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{C}}, \epsilon_{\widehat{C}})$-leakage-indistinguishable from $\mathcal{W}_{\widehat{C}}(X|Y)$. The proof is by a hybrid argument, outlined as follows. First, we replace all gadgets in $\widehat{C}$ by their corresponding reconstructors. Then, we replace all connecting wire bundles with random encodings of random values, keeping the innards of gadgets consistent with these random encodings.

We first prove that we can replace each gadget in $\widehat{C}$ with an appropriate gadget reconstructor keeping the connecting wires consistent. We will use the following notation. Let $\{\widehat{g}_i\}$ for $i \in [1, s]$ denote the gadgets in $\widehat{C}$. Drawing a wire assignment from the distribution $\mathcal{W}_{\widehat{C}}(X|Y)$ of the real circuit, we denote its elements as follows. For the ith gadget $\widehat{g}_i$ in $\widehat{C}$, U_i are its inputs and V_i are its outputs (these are identified with elements of X or Y if $\widehat{g}_i$ is a first gadget or a last gadget). Note that (U_i, V_i) is always plausible for $\widehat{g}_i$, by definition. Let us define the following hybrid wire assignment distributions:

$\mathcal{W}^0_{\widehat{C}}$: $\mathcal{W}_{\widehat{C}}(X|Y)$.

$\mathcal{W}^i_{\widehat{C}}$ ($i \in [1,s]$): Same as $\mathcal{W}^{i-1}_{\widehat{C}}$ except that the assignment to the wires inside $\widehat{g}_i$ is replaced by $R_{\widehat{g}_i}(U_i, V_i)$ with $R_{\widehat{g}_i} \leftarrow \mathsf{REC}_{\widehat{g}_i}$.

The following claim shows that $\mathcal{W}^{i-1}_{\widehat{C}}$ and $\mathcal{W}^i_{\widehat{C}}$ are $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-leakage-indistinguishable for all $i \in [1,s]$. More precisely,

Claim 1. *For any $i \in [1,s]$, if $\widehat{g}_i$ is $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-reconstructible, then the distributions $\mathcal{W}^{i-1}_{\widehat{C}}$ and $\mathcal{W}^i_{\widehat{C}}$ are $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}} - s\tau_{\mathsf{samp}}, \epsilon_{\widehat{g}})$-leakage-indistinguishable.*

Proof. For any $i \in [1,s]$ we use Lemma 8 with the following mapping: $\mathcal{W}_1 = \mathcal{W}^{i-1}_{\widehat{C}}, \mathcal{W}'_1 = \mathcal{W}^i_{\widehat{C}}$ and $\mathcal{W}_0 = \mathcal{W}_{\widehat{g}_i}(U_i|V_i), \mathcal{W}'_0 = \mathsf{REC}_{\widehat{g}_i}(U_i, V_i)$. To apply Lemma 8, we need to define the distribution F, where $f_S \leftarrow F$:

1. For all $j \geq i+1$ sample from $\mathcal{W}_{\widehat{g}_j}(U_j|V_j)$ and hardwire the result into the description of f_S.
2. For all $j \leq i-1$ run $\mathsf{REC}_{\widehat{g}_j}(U_j, V_j)$ to obtain a valid wire assignment for that part of the circuit. Hardwire the result into the description of f_S.
3. For the part of the wire assignment that represents $\widehat{g}_i$, f_S just outputs its input.

Note that f_S takes as long to sample as the time required to either compute or reconstruct the $s-1$ gadgets, which, in our case is upper bounded by τ_{samp}. It is easy to see that for its input f_S is the identity function (it just outputs its inputs together with hardwired values). Moreover, if f_S takes as input a sample from $\mathcal{W}_{\widehat{g}_i}(U_i|V_i)$ then its output is distributed as $\mathcal{W}^{i-1}_{\widehat{C}}$. On the other hand if the input is $R_{\widehat{g}_i}(U_i, V_i)$, then f_S's output is identically distributed to $\mathcal{W}^i_{\widehat{C}}$. These facts, combined with Lemma 8 and the fact that $\mathcal{W}_0$ and $\mathcal{W}'_0$ are $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-leakage-indistinguishable, show that $\mathcal{W}_1 = \mathcal{W}^{i-1}_{\widehat{C}}$ and $\mathcal{W}'_1 = \mathcal{W}^i_{\widehat{C}}$ are $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}} - s\tau_{\mathsf{samp}}, \epsilon_{\widehat{g}})$-leakage-indistinguishable. This concludes the claim. $\square$

Next, we show that we can replace the connecting wire bundles in $\widehat{C}$ with random encodings. Associate each bundle of connecting wires with integer $i \in [1,m]$ and denote the encoding carried by this bundle by $\vec{v}_i$. Denote by $\widehat{g}^i_-$ the gadget that has $\vec{v}_i$ as an output wire bundle, and by $\widehat{g}^i_+$ the gadget that has $\vec{v}_i$ as input (see Figure 7). We define iteratively the following hybrid wire assignment distributions:

$\mathcal{W}^i_{\widehat{C}}$ with $i \in [s+1, s+m]$: Same as $\mathcal{W}^{i-1}_{\widehat{C}}$ except that $\vec{v}_i$ is replaced with a random encoding $\vec{v}'_i \leftarrow \mathsf{Enc}(\cdot)$ (and the internal wires in $\widehat{g}^i_-$ and $\widehat{g}^i_+$ are adjusted accordingly, as the wire bundles are given as inputs to the reconstructors of $\widehat{g}^i_-$ and $\widehat{g}^i_+$).

Intuitively: $\mathcal{W}^s_{\widehat{C}}$ is the wire assignment distribution that results from running, for each gadget in $\widehat{C}$, its corresponding reconstructor using honestly-computed connecting wires. Then, in $\mathcal{W}^i_{\widehat{C}}$ for $i = s+1, \ldots, s+m$, we replace step-by-step the honest encodings at the connecting wires with random encodings. These random encodings no longer correctly capture the computation of the circuit, but are still leakage-indistinguishable. The final distribution, $\mathcal{W}^{s+m}_{\widehat{C}}$, is identical to $\mathsf{REC}_{\widehat{C}}(X, Y)$.

We next prove a claim stating that for all $i \in [s+1, s+m]$ the distributions $\mathcal{W}^{i-1}_{\widehat{C}}$ and $\mathcal{W}^i_{\widehat{C}}$ are $(\mathcal{L}_{\mathcal{W}}, \tau_{\mathcal{W}}, \epsilon_{\mathcal{W}})$-leakage-indistinguishable.

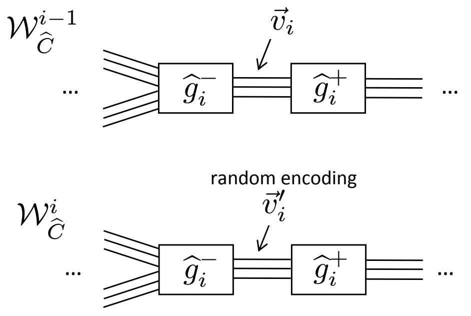

Figure 7: Notation used in Claim 2. $\widehat{g}_-^i$ and $\widehat{g}_+^i$ are reconstructors for gadgets preceding and following the wire bundle in question that is changed between the two successive experiments. In the two successive experiments, $\mathcal{W}_{\widehat{C}}^{i-1}$ and $\mathcal{W}_{\widehat{C}}^{i}$, all the other wires bundles are sampled from the same distribution, whereas $\vec{v}_i$ is drawn from the honest distribution in $\mathcal{W}_{\widehat{C}}^{i-1}$, while $\vec{v}'_i$ is a random encoding of a random element in $\mathcal{W}_{\widehat{C}}^{i}$.

Claim 2. *Let $\mathcal{L}_\Pi$ be some class of leakage functions and let $\tau_\Pi > 0, \epsilon_\Pi > 0$. If Π is $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable, then for all $i \in [s+1, s+m]$ the distributions $\mathcal{W}_{\widehat{C}}^{i-1}$ and $\mathcal{W}_{\widehat{C}}^{i}$ are $(\mathcal{L}_\mathcal{W}, \tau_\mathcal{W}, \epsilon_\mathcal{W})$-leakage-indistinguishable with $\epsilon_\mathcal{W} = \epsilon_\Pi$, $\tau_\mathcal{W} = \tau_\Pi - s\tau_{\mathsf{samp}}$, and any $\mathcal{L}_\mathcal{W}$ that satisfies $(\mathcal{L}_\mathcal{W} \circ (2 \times \mathcal{R}_{\widehat{g}})) \subseteq \mathcal{L}_\Pi$.*

Proof. To prove this statement for any $i \in [s+1, s+m]$, we apply Lemma 8 with the following assignment for the distributions: $\mathcal{W}_1 = \mathcal{W}_{\widehat{C}}^{i-1}, \mathcal{W}'_1 = \mathcal{W}_{\widehat{C}}^{i}$ and $\mathcal{W}_0 = \mathsf{Enc}(v_i), \mathcal{W}'_0 = \mathsf{Enc}(v'_i)$, with $v'_i \leftarrow \mathcal{K}$. Furthermore, we define the distribution F, with $f_S \leftarrow F$ that takes as input a single encoding $\vec{e}$:

1. Sample the values for all the connecting wire bundles except $\vec{v}_i$ according to $\mathcal{W}_{\widehat{C}}^{i}$ (which is the same as $\mathcal{W}_{\widehat{C}}^{i-1}$ for those wire bundles).

2. For each gadget $\widehat{g}$ in $\widehat{C}$ except $\widehat{g}_-^i$ and $\widehat{g}_+^i$, pick a reconstructor from the appropriate reconstructor distribution $R_{\widehat{g}} \leftarrow \mathsf{REC}_{\widehat{g}}$, and run $R_{\widehat{g}}(U, V)$, where (U, V) are the sampled values for the input and output wire bundles of $\widehat{g}$. The resulting wire assignments for each gadget are hardwired into f_S.

3. Pick and hardwire reconstructors $R_{\widehat{g}_-^i} \leftarrow \mathsf{REC}_{\widehat{g}_-^i}$ and $R_{\widehat{g}_+^i} \leftarrow \mathsf{REC}_{\widehat{g}_+^i}$ and wire their descriptions into f_S. On input $\vec{e}$, run on-line the reconstructors $R_{\widehat{g}_-^i}$ and $R_{\widehat{g}_+^i}$, using as their inputs and outputs the wire bundles already sampled and $\vec{v}_i$ set to $\vec{e}$. Output their resulting wire assignments together with the hardwired wire assignments for all the other gadget reconstructors from the previous steps.

We claim that

$$\mathcal{W}_{\widehat{C}}^{i-1} \equiv f_S(\vec{e}), \text{ if } \vec{e} \leftarrow \mathsf{Enc}(v_i),$$
$$\mathcal{W}_{\widehat{C}}^{i} \equiv f_S(\vec{e}), \text{ if } \vec{e} \leftarrow \mathsf{Enc}(v'_i).$$

Indeed, in either case, all the wires internal to gadgets are computed according to reconstructors, and the connecting wire bundles except $\vec{v}_i$ are sampled identically in the two distributions. If $\vec{e} \leftarrow \mathsf{Enc}(v_i)$ then, because all the gadgets are rerandomizing, the joint distribution of $\vec{e}$ together with all the other wires is indeed $\mathcal{W}_{\widehat{C}}^{i-1}$ (note that this is the only place where we use the fact that the gadgets are rerandomizing, but the use of this fact here is crucial: if $\mathsf{Enc}(v_i)$ was correlated with some other connecting wire bundle, we could not hardwire that bundle into f_S, because it would not be known until $\vec{e}$ was given).

Sampling $f_S \leftarrow F$ takes $s\tau_{\mathsf{samp}}$ time, because that is the time required to sample the reconstructors. Let us now analyze the complexity of f_S. Since most of the wire assignments are hardwired in advance into f_S, on input $\vec{e}$ f_S only needs to run $\widehat{g}_-^i$ and $\widehat{g}_+^i$. Thus, we get that functions $f_S \leftarrow F$ can be computed in $2 \times \mathcal{R}_{\widehat{g}}$. If we now apply Lemma 8 with the fact that $\mathcal{W}_0$ and $\mathcal{W}_0'$ are $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable, we get that $\mathcal{W}_1 = \mathcal{W}_{\widehat{C}}^{i-1}$ and $\mathcal{W}_1' = \mathcal{W}_{\widehat{C}}^{i}$ are $(\mathcal{L}_\mathcal{W}, \tau_\mathcal{W}, \epsilon_\mathcal{W})$-leakage-indistinguishable for

- $\tau_\mathcal{W} = \tau_\Pi - s\tau_{\mathsf{samp}}$,
- for any class of functions $\mathcal{L}_\mathcal{W}$ that satisfies $\mathcal{L}_\Pi \supseteq (\mathcal{L}_\mathcal{W} \circ (2 \times \mathcal{R}_{\widehat{g}}))$ and
- $\epsilon_\mathcal{W} = \epsilon_\Pi$.

This concludes the proof of the claim. □

Putting now the results from Claim 1 and Claim 2 together and setting $\mathcal{L}_{\widehat{g}} = \mathcal{L}_{\widehat{C}}$ and $\mathcal{L}_\mathcal{W} = \mathcal{L}_{\widehat{C}}$, we get that $\mathcal{W}_{\widehat{C}}^0 = \mathcal{W}_{\widehat{C}}(X|Y)$ and $\mathcal{W}_{\widehat{C}}^{s+m} = \mathsf{REC}_{\widehat{C}}(X, Y)$ are $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{C}}, \epsilon_{\widehat{C}})$-leakage-indistinguishable. Here, $\tau_{\widehat{C}} = \min(\tau_\Pi, \tau_{\widehat{g}}) - s\tau_{\mathsf{samp}}$ and

$$\epsilon_{\widehat{C}} = m\epsilon_\Pi + s\epsilon_{\widehat{g}}. \tag{9}$$

This concludes the proof of Lemma 13. □

Below we give applications of our general composition lemma by showing that circuits transformed by TR_C and TR_N are reconstructible.

Reconstructibility of $\widehat{C} \leftarrow \mathsf{TR}_C(C)$. We establish composition of the single gadget reconstructors presented in Section 4.3 in the corollary below. Since its proof merely adds up parameters, we move its formal version to Appendix C and give here only a simple sketch.

Corollary 1 (Reconstructor for $\widehat{C} \leftarrow \mathsf{TR}_C(C)$). *Let $\mathcal{L}_\Pi$ be some set of leakage functions and $\epsilon_\Pi > 0, \tau_\Pi > 0$. Let Π be the underlying encoding scheme of TR_C with Π being $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable. Let C be a stateless circuit of size s, with n_I inputs and n_O outputs. Then the transformed circuit $\widehat{C} \leftarrow \mathsf{TR}_C(C)$ is rerandomizing and $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{C}}, \epsilon_{\widehat{C}})$-reconstructible by $\mathsf{SHALLOW}(2, (n_\mathrm{I} + n_\mathrm{O})O(k^2))$. Here, we have $\epsilon_{\widehat{C}} = \epsilon_\Pi s(k+2)$, $\tau_{\widehat{C}} = \tau_\Pi - O(sk^2)$, and $\mathcal{L}_{\widehat{C}}$ satisfies $\mathcal{L}_\Pi \subseteq \mathcal{L}_{\widehat{C}} \circ \mathsf{SHALLOW}(3, O(k^2))$ (for $\mathcal{K} = \mathsf{GF}(2)$, $\mathcal{L}_\Pi = \mathcal{L}_{\widehat{C}} \circ \mathsf{SHALLOW}(2, O(k^2))$).*

Proof (sketch). At a high-level the proof is simple. Since TR_C is an encoding based circuit transformation, and all gadgets used by TR_C are rerandomizing and reconstructible, Lemma 13 from above establishes the corollary. In the formal proof in Appendix C we rigorously analyze the parameters which establish the above corollary. □

Reconstructibility of $\widehat{C} \leftarrow \mathsf{TR}_N(C)$. Before we show reconstructibility of composed circuits $\widehat{C} \leftarrow \mathsf{TR}_N(C)$ we present a reconstructor for $\widehat{\odot}$ of TR_N. We do this by viewing $\widehat{\odot}$ as a circuit composed of $\widehat{\oplus}$ and $\widehat{\texttt{mult}}$ gadgets, for which we have shown reconstructibility in the last section. Then we apply the composition lemma, which gives us a reconstructor for $\widehat{\odot}$. The proof of the lemma below and of Corollary 2 can be found in Appendix C.

Lemma 14. *For every $p \in (0, \frac{1}{2}]$, the $\widehat{\odot}$ gadget is $(\mathcal{N}_p, \infty, \epsilon_{\widehat{\odot}})$-reconstructible by $\mathsf{LOCAL}(3)$, where $\epsilon_{\widehat{\odot}}(k) \leq (2k+1)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512))$.*

We establish reconstructibility of transformed circuits $\widehat{C} \leftarrow \mathsf{TR}_N(C)$.

Corollary 2 (Reconstructor for $\widehat{C} \leftarrow \mathsf{TR}_N(C)$). *Let $\mathcal{L}_{\widehat{C}} = \mathcal{N}_p$ for some $p \in (0, 1/2]$. Let C be a stateless circuit of size s, with n_{I} inputs and n_{O} outputs. Then the transformed circuit $\widehat{C} \leftarrow \mathsf{TR}_N(C)$ is rerandomizing and $(\mathcal{N}_p, \infty, \epsilon_{\widehat{C}}(k))$-reconstructible by $\mathsf{LOCAL}(3)$ with $\epsilon_{\widehat{C}}(k) \leq s(2k + 3)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512))$*

In Lemma 13 and the two corollaries above we considered only the stateless case, and proved that it is simulatable by a special simulator called "reconstructor." In the next section, we will use this special simulator to provide a simulator that proves security of stateful circuits directly according to Definition 1.

4.6 Security of Full Circuit Transformation

To prove security of the full transformation according to Definition 1 we need to consider stateful circuits. Stateful circuits have memory and are run and observed multiple times. In contrast, so far we have only considered stateless circuits that have no persistent state and are observed once. The main challenge in the stateful setting is the following. Note that memory has input wires (used at the end of round $i-1$ to store the state $\widehat{m}_i$) and output wires (used at the beginning of round i to retrieve $\widehat{m}_i$), and therefore the adversary can obtain leakage related to memory twice. This problem is exacerbated by adaptivity, which allows the adversary to choose a leakage function for round i that depends on the results of round $i-1$. Thus, the ultimate information obtained by the adversary about $\widehat{m}_i$ is considerably more complex than can be computed by any two fixed functions from $\mathcal{L}$.

We address these difficulties in the proof of the following lemma.

Lemma 15. *Let $\tau_\Pi > 0, \epsilon_\Pi > 0$, let $\mathcal{L}_\Pi$ be some class of leakage functions, and let TR be an encoding based circuit transformation with underlying encoding scheme Π. Let C be an arbitrary (stateful) circuit with n memory gates, s other gates (not counting dummy `encoder` and `decoder` gates), and m wires connecting those s gates. Let $\widehat{C} \leftarrow \mathsf{TR}(C)$ be its transformation. Suppose all gadgets in $\widehat{C}$, except $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$, are rerandomizing and $(\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-reconstructible by $\mathcal{R}_{\widehat{g}}$, for some $\mathcal{L}_{\widehat{g}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}}$. Suppose also that Π is $(\mathcal{L}_\Pi, \tau_\Pi, \epsilon_\Pi)$-leakage-indistinguishable and 2-adaptive $(\mathcal{L}_{2\Pi}, \tau_{2\Pi}, \epsilon_{2\Pi})$-leakage-indistinguishable. Then TR is $(\mathcal{L}, \tau_\mathcal{A}, \tau_\mathcal{S}, \tau_D, q, \epsilon)$-secure for*

- *any $\tau_\mathcal{A}$ and τ_D satisfying $\tau_\mathcal{A} + \tau_D \leq \min(\tau_\Pi, \tau_{2\Pi}, \tau_{\widehat{g}}) - qs\tau_{\mathsf{samp}}$, where τ_{samp} is the time to sample $R_{\widehat{g}} \leftarrow \mathsf{REC}_{\widehat{g}}$,*[12]
- *some $\tau_\mathcal{S} \leq \tau_\mathcal{A} + qs\tau_{\mathsf{samp}}$,*

[12]Like in Lemma 13, we assume that τ_{samp} is larger than the time required to compute gadgets $\widehat{g}$.

- *some* $\epsilon \leq qs\epsilon_{\widehat{g}} + qm\epsilon_{\Pi} + (q+1)n\epsilon_{2\Pi}$
- *any* $\mathcal{L}$ *that satisfies* $(\mathcal{L} \circ (2 \times \mathcal{R}_{\widehat{g}})) \subseteq \mathcal{L}_{\Pi}$, $(\mathcal{L} \circ \mathcal{R}_{\widehat{g}}) \subseteq \mathcal{L}_{2\Pi}$, *and* $\mathcal{L} \subseteq \mathcal{L}_{\widehat{g}}$.

Before starting with the proof, we discuss an interpretation of the important parameters. To apply the lemma we require

- that all gadgets in $\widehat{C}$ (except $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$) are $\mathcal{L}_{\widehat{g}}$-reconstructible, and
- the underlying encoding scheme Π is $\mathcal{L}_{\Pi}$-leakage-indistinguishable and 2-adaptive $\mathcal{L}_{2\Pi}$-leakage-indistinguishable.

If that is given, then by the above lemma, the encoding-based circuit transformation TR is $\mathcal{L}$-secure for any class $\mathcal{L}$ that satisfies

- $(\mathcal{L} \circ (2 \times \mathcal{R}_{\widehat{g}})) \subseteq \mathcal{L}_{\Pi}$, where $(2 \times \mathcal{R}_{\widehat{g}})$ denotes parallel execution of $\mathcal{R}_{\widehat{g}}$ (which is the function class in which the gadget reconstructors lie),
- $(\mathcal{L} \circ \mathcal{R}_{\widehat{g}}) \subseteq \mathcal{L}_{2\Pi}$
- $\mathcal{L} \subseteq \mathcal{L}_{\widehat{g}}$.

Stated differently, if we want TR to be $\mathcal{L}$-secure for some $\mathcal{L}$, then we need an encoding scheme that is secure against functions from $(\mathcal{L} \circ (2 \times \mathcal{R}_{\widehat{g}}))$ for a single query, functions from $(\mathcal{L} \circ \mathcal{R}_{\widehat{g}})$ (which is a smaller class) for two queries; we also need that all gadgets $\widehat{g}$ in $\widehat{C}$ (except $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$) are at least $\mathcal{L}$-reconstructible. Before we instantiate the above lemma to show security of our transformations TR_C and TR_N, we give a high-level outline of the proof (the details of the proof are in Appendix D).

Outline of the proof. Define a circuit $\widehat{C}^*$ as the circuit that computes a single clock cycle of $\widehat{C}$ (recall the notion of clock cycle from the definition of stateful circuits on p. 10), without encoding and decoding and without state. Specifically, $\widehat{C}^*$ is obtained from $\widehat{C}$ by removing $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ gadgets, removing memory, taking the wire bundles that exit the memory and making them part of the input, and taking the wire bundles that enter the memory and making them part of the output. We can then write one clock cycle of the computation (without input encoding and output decoding) as $(\widehat{m}_i, \mathsf{Enc}(y_i)) \leftarrow \widehat{C}^*(\widehat{m}_{i-1}, \mathsf{Enc}(x_i))$. We know from Lemma 13 that $\widehat{C}^*$ has a reconstructor.

We need to show that for every q-adaptive $(\mathcal{L}, \tau_{\mathcal{A}})$-adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ with only black-box access to $C[m_i]$ such that for every stateful circuit $C[m_0]$, the output distribution of $\mathcal{A}$ and $\mathcal{S}$ are computationally close.

The idea of the proof is simple: $\mathcal{S}$ runs $\mathcal{A}$ as a subroutine, simulates its environment and outputs $\mathcal{A}$'s result. $\mathcal{S}$ needs to simulate the environment without knowledge of the initial secret state m_0; thus, it has to answer $\mathcal{A}$'s leakage queries without knowing the secret state. Each such leakage query gets as input the wire assignment of a single clock cycle of $\widehat{C}$. To simulate the wire assignment, we can use the reconstructor for $\widehat{C}^*$, giving it the circuit's true public inputs and outputs and random values instead of the memory contents. This approach proves security of a single observation. However, extending this argument to many rounds requires some additional care.

Main difficulty of the proof. During computation of $(\widehat{m}_i, \mathsf{Enc}(y_i)) \leftarrow \widehat{C}^*(\widehat{m}_{i-1}, \mathsf{Enc}(x_i))$, the adversary can pick a leakage function f_i and obtain some knowledge about the secret state $\widehat{m}_i$. Adaptively, based on that knowledge (i.e., on the output y_i and the leakage that may depend on $\widehat{m}_i$), the adversary may pick a leakage function f_{i+1} and get leakage from the execution of $\widehat{C}^*(\widehat{m}_i, \mathsf{Enc}(x_{i+1}))$. The difficulty is that the leakage from both observations, number i and number $(i+1)$, may depend on the secret state $\widehat{m}_i$, which is the reason why we require the underlying encoding scheme to be 2-adaptive leakage indistinguishable. (This difficulty does not occur for leakage from prior or subsequent states, or from other wire bundles, essentially because of rerandomization.)

We now give a high-level description of the simulator $\mathcal{S}$. The formal specification is given in Figure 8 in the appendix.

Simulation. $\mathcal{S}$ needs to answer the adversary's query (f_i, x_i) for each clock cycle $1 \leq i \leq q$. $\mathcal{S}$ generates a simulated wire assignment and then applies f_i to it. Generation of the simulated wire assignment is done as follows. Since $\mathcal{S}$ does not know the content of the secret memory (but the wire assignment may depend on it), $\mathcal{S}$ uses random encodings instead. $\mathcal{S}$ also uses the public input x_i and output y_i. Note that, y_i may no longer be the correct output on x_i if the contents of the memory are changed to random values. So the "correct" wire assignment may not exist. However, a simulated assignment may still be computed: $\mathcal{S}$ does so by using the reconstructors of the gadgets inside $\widehat{C}$ (except $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ gadgets, whose wire assignments are computed honestly from x_i and y_i).

We need to show that such a simulation is indistinguishable from the adversary's observations in the real experiment. Like in Lemma 13, this is done by a hybrid argument. We first show that instead of the real wire assignment, we can replace all wire bundles with random encodings, and use the gadget reconstructors for the internal wires of each gadget (except $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$). Notice that these steps are essentially the same as in the proof of Lemma 13. We could just directly apply Lemma 13 and use the reconstructor for the entire stateless circuit $\widehat{C}^*$ rather than for each gadget, but that would give us worse parameters: specifically, the reduction in leakage class would be larger, because the reduction in leakage class depends on the size of the reconstructor.

Next, we consider $n(q+1)$ different hybrids, i.e., we make a hybrid argument over the number of observations q and the size of the secret state n. In each hybrid step, we replace the content of a single encoded memory cell with some random encoding. By the leakage-indistinguishability of the underlying encoding scheme Π, two consecutive hybrids will be indistinguishable. Notice that this is the place where we require Π to be secure even against 2-adaptive adversaries, since the observation of two consecutive clock cycles will depend on the target encoding.

In Appendix D we give the technical description of the ideas outlined above.

4.7 Proofs of our Main Theorems: Theorem 1 and Theorem 2

Proof of Theorem 1. The proof of Theorem 1 merely puts together the parameters from Lemma 9 and Lemma 15, and may be skipped by the reader.

In this theorem, we are taking assumed parameters from 2-adaptive leakage indistinguishability and also using them for 1-adaptive leakage indistinguishability (since it is no worse), in order to simply the theorem statement. So, when applying Lemma 15, we use $(\mathcal{L}_{2\Pi}, \tau_{2\Pi}, \epsilon_{2\Pi}) = (\mathcal{L}_{\Pi}, \tau_{\Pi}, \epsilon_{\Pi})$. By definition, TR_C is an encoding based circuit transformation for arbitrary circuits C with size s and n memory cells. Since all gates have fan-in at most 2, it has $m \leq 2s$ wires. Recall from the theorem statement that $\epsilon_{\Pi} > 0, \tau_{\Pi} > 0$ and $\mathcal{L}_{\Pi}$, $\mathcal{L}$ are some leakage classes that satisfy

$\mathcal{L} \circ \mathsf{SHALLOW}(3, O(k^2)) \subseteq \mathcal{L}_\Pi$ (replace $\mathsf{SHALLOW}(3, O(k^2)$ with $\mathsf{SHALLOW}(2, O(k^2)$ in the case of $\mathsf{GF}(2)$). Notice, further that as proven in Section 4.3 all gadgets in $\widehat{C}$ are rerandomizing and $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-reconstructible by $\mathsf{SHALLOW}(3, O(k^2))$ (respectively, $\mathsf{SHALLOW}(2, O(k^2))$ for the case of $\mathsf{GF}(2)$) for some parameters $\tau_{\widehat{g}}, \epsilon_{\widehat{g}}$. Since our transformation has to work for *any* circuit we can assume[13] that C is made solely of $\odot$ gates. By Lemma 9, we get then $\tau_{\widehat{g}} = \tau_\Pi - O(k^2)$ and $\epsilon_{\widehat{g}} = k\epsilon_\Pi$.

We are now ready to apply Lemma 15 and get $\tau_\mathcal{A} + \tau_D \leq \min(\tau_\Pi, \tau_{\widehat{g}}) - qs\tau_{\mathsf{samp}}$. With $\tau_{\mathsf{samp}} = O(k^2)$ this yields $\tau_\mathcal{A} + \tau_D \leq \tau_\Pi - qsO(k^2)$. Similarly, we get $\tau_\mathcal{S} \leq \tau_c A + qsO(k^2)$. Next, we compute the computational distance between the real experiment and the simulated experiment:

$$\epsilon = qs\epsilon_{\widehat{g}} + qm\epsilon_\Pi + (q+1)n\epsilon_{2\Pi} \leq \epsilon_\Pi(q+1)(s(k+2)+n).$$

This concludes the proof. $\square$

Theorem 1 relies on the assumption that Π is 2-adaptive leakage-indistinguishable. We will eliminate this additional assumption in the next section using circuit lower bounds, and show an unconditional results for AC^0 leakages.

Proof of Theorem 2. The proof of the main theorem for noisy leakages (Theorem 2) follows the same line as the proof of Theorem 1 above. We use Proposition 2 to note that the Π_{parity} encoding is $(\mathcal{N}_p \circ \mathsf{LOCAL}(6), \infty, (1-(2p)^6)^k)$-leakage indistinguishable and 2-adaptive $(\mathcal{N}_p \circ \mathsf{LOCAL}(3), \infty, (1-(2p)^6)^k)$-leakage indistinguishable (thus, in this case we can apply Lemma 15 with $\epsilon_\Pi = \epsilon_{2\Pi} < \exp(-64kp^6)$, using $1 - x \leq \exp(-x)$). Recall from Lemma 14 that the gadgets are rerandomizing and $(\mathcal{N}_p, \infty, \epsilon_{\widehat{C}}(k))$-reconstructible by $\mathsf{LOCAL}(3)$ with $\epsilon_{\widehat{g}} \leq (2k+1)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512))$. Also note that the number m of internal wires is at most $2s$, because each gate has fan-in at most two.

Notice that Theorem 2, unlike Theorem 1, doesn't need to make computational assumptions about hardness of decoding Π. $\square$

5 Instantiation of TR_C Against AC^0 leakage

As mentioned in the last section Theorem 2 doesn't require computational hardness assumptions, while Theorem 1 relies on the assumption that decoding is "hard" for functions in $\mathcal{L}$. Lower bounds on computational tasks are notoriously difficult to prove, and therefore, given our current state of knowledge, applying our results for computationally bounded leakages will, in most cases, require computational assumptions about hardness of decoding for a given class of leakage functions. In this section, however, we highlight a case in which Theorem 1 can be applied for an explicit leakage class.

Recall that $\mathcal{C}(d, s, \lambda)$ denotes the class of AND-OR-NOT unlimited fan-in circuits with depth d, size s and λ bits of output. In Proposition 1 (cf. Section 4.1) we showed that the parity encoding Π_{parity} is $(\mathcal{C}(d, \exp(O(k^{(1-\delta)}/d)), k^\delta), \infty, \exp(-\Omega(k^{(1-\delta)}/d)))$ leakage-indistinguishable, for any constants $d \in \mathbb{N}_{>0}$ and $0 < \delta < 1$.

If we instantiate TR_C with Π_{parity} then by Theorem 1 we almost instantly obtain security against leakages modeled as constant depth circuits. However, there is one caveat. In Theorem 1 we require that Π_{parity} is 2-*adaptive* leakage-indistinguishable, while Proposition 1 only talks about a single observation.

[13]since the parameters are worst in this case

In the following lemma we show generically that a leakage-indistinguishable encoding scheme Π is also secure against 2-adaptive adversaries. We would like to emphasize that the bounds in the lemma are rather bad, since the leakage circuit size and the adversarial running time lose exponentially in λ (i.e., the amount of leakage that we tolerate per observation). However, as it turns out, this loss will not matter much in the application to Corollary 3, because the adversary there is information-theoretic (and thus has arbitrary running time), and the circuit size loss will be absorbed into the exponent.

The lemma is given specifically for leakage functions modeled by circuits with unlimited fan-in AND and OR gates, such as it is the case for AC^0. The proof is moved to Appendix E.

Lemma 16. *Let D, E be two distributions and $d, s, \tau, \epsilon \geq 0$ and $\mathcal{L} = \mathcal{C}(d, s, \lambda)$. If D and E are $(\mathcal{L}, \tau, \epsilon)$-leakage-indistinguishable, then D and E are 2-adaptive $(\mathcal{L}', 2^{-\lambda}\tau, \epsilon)$-leakage-indistinguishable, where $\mathcal{L}' = \mathcal{C}(d-2, O(s2^{-\lambda}), \lfloor \lambda/2 \rfloor)$.*

We would like to note that we can generalize this lemma in two ways: first, by a similar argument we can prove security against p-adaptive adversaries. This, however, increases the function's size exponentially in p *and* λ. Second, observe that we state this lemma for the special case that the leakage functions are circuits with particular gates. This can be generalized in a straightforward way to other function classes.

We obtain the following corollary by instantiating Theorem 1 with the parity encoding, the tolerable leakage functions $\mathcal{L}$ with AC^0 circuits that output up to k^δ bits, and using the above lemma about adaptivity.

Corollary 3. *Recall that k is the security parameter. Let $0 < \delta < 1$, $4 < d < 1/\delta - 1$, and q be some constants. There exists a circuit transformation for circuits over $\mathcal{K} = \mathsf{GF}(2)$ that is $(\mathcal{L}, \tau_{\mathcal{A}}, \tau_{\mathcal{S}}, \tau_D = \infty, q, \epsilon)$-secure for*

- *any $\tau_{\mathcal{A}}$,*
- *some $\tau_{\mathcal{S}} \leq \tau_{\mathcal{A}} + qsO(k^2)$, where s is the number of gates in C*
- *some $\epsilon \leq (q+1)(s(k+2)+n)\exp(-\Omega(k^{(1-\delta)/d}))$, where n is the number of memory gates in C,*
- *$\mathcal{L} = \mathcal{C}(d-4, \exp(O(k^{(1-\delta)/d})), \lfloor k^\delta/2 \rfloor)$. Notice that this is in AC^0 since $d-4$ is constant.*

The transformation increases the size of each multiplication gate by a factor of $O(k^2)$ and the size of the rest of the circuit by a factor of $O(k)$.

For example, fixing the relationship $\delta = 1/(d+2)$, we now figure out what we need to do in order to be resilient to λ bits of leakage computed by an AC^0 circuit of any depth d_λ (which, using the notation from the Corollary statement, is $d-4$). Since $\lambda = \lfloor k^\delta/2 \rfloor$, the security parameter k needs to be at least $(2\lambda)^{d_\lambda+6}$. The size of the leakage circuit can go up to $\exp(c_1 k^{1/(d_\lambda+6)}) > \exp(c_1\lambda)$; and security (i.e., distinguishing advantage ϵ) will be at most $qs\exp(-c_2 k^{1/(d_\lambda+6)}) < qs\exp(-c_2\lambda)$, for some constants c_1 and c_2 (which follow from O and Ω used in Proposition 1; they depend on d and δ, but the exact dependence is not worked out in [9]). Increasing k further would allow increasing the size of the leakage circuit, decrease the distinguishing advantage, and allow for more leakage. Unfortunately, because the circuit size would grow in proportion to k^2, the amount of leakage as a fraction of the circuit size would decrease.

We could set δ to be smaller, which would lead a higher k to support the same amount of leakage, which would lead to higher allowable leakage circuit size and lower ϵ, at the expense of bigger increase in the transformed circuit size. The tradeoff is not easy to quantify, because the constants c_1 and c_2 would also change.

Proof. By Proposition 1 the parity encoding is $(\mathcal{C}(d, \exp(O(k^{(1-\delta)/d})), k^\delta), \infty, \exp(-\Omega(k^{(1-\delta)/d})))$-leakage-indistinguishable. Lemma 16 then shows that the encoding scheme is 2-adaptive $(\mathcal{L}_{2\Pi}, \infty, \epsilon_{2\Pi})$-leakage-indistinguishable, where

- $\mathcal{L}_{2\Pi}$ is $\mathcal{C}(d-2, \exp(O(k^{(1-\delta)/d}) - k^\delta), \lfloor k^\delta/2 \rfloor)$. Since by assumption $d < 1/\delta - 1$ we get that $(1-\delta)/d > \delta$ and therefore $\exp(O(k^{(1-\delta)/d}) - k^\delta) = \exp(O(k^{(1-\delta)/d}))$. Thus, we can simplify $\mathcal{L}_{2\Pi}$ to $\mathcal{C}(d-2, \exp(O(k^{(1-\delta)/d})), \lfloor k^\delta/2 \rfloor)$, and
- $\epsilon_{2\Pi} = \exp(-\Omega(k^{(1-\delta)/d}))$.

We now apply Theorem 1 with $\mathcal{K} = \mathsf{GF}(2)$. To see that the depth of the leakage class goes from $d-2$ to $d-4$, we only need to observe that $\mathsf{SHALLOW}(2, O(k^2))$ can be implemented in by depth-two boolean circuits (where we don't count NOT gates and allow sufficient fan-in) by expressing arithmetic gates of fan-in-2 and depth-2 as a constant-size CNF or DNF. $\square$

Improving the security loss. The bounds from Corollary 3 imply that asymptotically parity encoding and our transformed circuits can tolerate similar leakage functions as long as $d < 1/\delta - 1$. This restriction can be eliminated by relaxing the security definition. More precisely, if in Definition 1 we restrict the adversary to choose the leakage function f_i, $i \geq 2$, adaptively *only* depending on the output of the leakage functions $f_1, \ldots f_{i-2}$[14], then in Theorem 1 we will *not* require that the underlying encoding scheme Π is 2-adaptive leakage-indistinguishable. The parameters of the our reduction will then be significantly improved[15] for this relaxed security definition.

Acknowledgments

We thank Yuval Ishai for discussion on [24] and pointing out the result in [9]. Thanks also to Ran Raz and Debajyoti Bera for discussions on circuit lower bounds, and Ronen Shaltiel for discussions on alternative constructions. We are grateful to Eric Miles and Emanuele Viola for pointing out that a claim made in a previous version of our paper—namely, that our construction can achieve security against $\mathsf{ACC}^0[p]$ circuits—required stronger circuit lower bounds than are currently known. We appreciate the detailed and helpful comments made by anonymous referees.

References

[1] Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In Jin-Yi Cai, editor, *Advances in Computational Complexity Theory*, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, pages 1–20. American Mathematical Society, 1993.

[14] Notice that the choice of f_i and the input x_i may still depend on the circuit's outputs $y_1, \ldots, y_{i-1}$.

[15] Specifically, the exponential security loss in λ that stems from Lemma 16 will not be needed, because 2-adaptivity will not be needed.

[2] Miklós Ajtai. Secure computation with information leaking to an adversary. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 715–724. ACM, 2011.

[3] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[4] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, The Technion—Israel Institute of Technology, June 1996.

[5] Daniel J. Bernstein. Cache-timing attacks on AES. `http://cr.yp.to/papers.html#cachetiming`, 2005.

[6] Mark Braverman. Poly-logarithmic independence fools AC0 circuits. Technical Report TR09-011, ECCC, 2009.

[7] David Brumley and Dan Boneh. Remote timing attacks are practical. *Comput. Netw.*, 48(5):701–716, 2005.

[8] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[9] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 711–720, New York, NY, USA, 2006. ACM.

[10] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[11] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 702–721. Springer, 2011.

[12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 230–247. Springer, 2012.

[13] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. *Annual IEEE Symposium on Foundations of Computer Science*, pages 293–302, 2008.

[14] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 343–360. Springer, 2010.

[15] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, 2010.

[16] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[17] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *STOC*, pages 182–194, 1987.

[18] Oded Goldreich. Three xor-lemmas—An exposition. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(56), 1995.

[19] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[20] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In Rabin [39], pages 59–79.

[21] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *FOCS*, pages 31–40. IEEE Computer Society, 2012.

[22] Johan Hastad. Almost optimal lower bounds for small depth circuits. In *Symposium on the Theory Of Computing*, 1986.

[23] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[24] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO'03*, pages 463–481, 2003. revised and abbreviated version of [25].

[25] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. unpublished manuscript ([24] is a revised and abbreviated version), 2003.

[26] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In Rabin [39], pages 41–58.

[27] Adam Klivans. On the derandomization of constant depth circuits. In *APPROX '01/RANDOM '01*, pages 249–260. Springer-Verlag, 2001.

[28] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[29] Markus G. Kuhn. *Compromising emanations: eavesdropping risks of computer displays*. PhD thesis, University of Cambridge, 2003. Technical Report UCAM-CL-TR-577.

[30] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC'04*, pages 278–296, 2004.

[31] Eric Miles. Iterated group products and leakage resilience against nc1. In Moni Naor, editor, *ITCS*, pages 261–268. ACM, 2014.

[32] Eric Miles and Emanuele Viola. Shielding circuits with groups. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 251–260. ACM, 2013.

[33] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[34] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA*, pages 1–20, 2006.

[35] Colin Percival. Cache missing for fun and profit. presented at BSDCan 2005, Ottawa, 2005; see `http://www.daemonology.net/hyperthreading-considered-harmful`, 2005.

[36] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.

[37] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[38] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.

[39] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[40] Guy N. Rothblum. How to compute under AC0 leakage without secure hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, 2012.

[41] Adi Shamir and Eran Tromer. Acoustic cryptanalysis: on nosy people and noisy machines. presented at the Eurocrypt 2004 rump session; see `http://tromer.org/acoustic`, 2004.

[42] Kris Tiri and Ingrid Verbauwhede. A VLSI design flow for secure side-channel attack resistant ICs. In *DATE*, pages 58–63. IEEE Computer Society, 2005.

[43] Salil Vadhan. *A Study of Statistical Zero-Knowledge Proofs.* PhD thesis, Massachusetts Institute of Technology, 1999.

A Proofs Omitted from Section 4.3

Proof of Lemma 6. The reconstructor $\mathsf{REC}_{\widehat{\$}}$ is the distribution whose only support is the following circuit $R_{\widehat{\$}}$. Given an empty X (i.e., the desired input of $\widehat{\$}$) and a $Y = (\vec{y})$ (i.e., the desired output of $\widehat{\$}$), $R_{\widehat{\$}}(X, Y)$ outputs a wire assignment that simply lets the output of $\widehat{\$}$ carry the only consistent value, namely Y. This is distributed identically to the honest case. □

Proof of Lemma 7. We will do the proof for the $\widehat{\mathsf{copy}}$ gadget; the other two are similar. The reconstructor $\mathsf{REC}_{\widehat{\mathsf{copy}}}$ is the distribution whose only support is a circuit $R_{\widehat{\mathsf{copy}}}$ that on inputs (X, Y) where $X = (\vec{a})$ (i.e., the desired input of the $\widehat{\mathsf{copy}}$ gate), and $Y = (\vec{b}, \vec{c})$ (i.e., its desired output), assigns the wires of $\widehat{\mathsf{copy}}$ in the only consistent way: $\vec{o_b} = \vec{b} \ominus \vec{a}$ and $\vec{o_c} = \vec{c} \ominus \vec{a}$.

If $\vec{a}, \vec{b}, \vec{c}$ are chosen as in the definition of a reconstructor (i.e., they are plausible inputs), then the resulting output of $R_{\widehat{\mathsf{copy}}}(X, Y)$ is identically distributed to the wire distribution $\mathcal{W}_{\widehat{\mathsf{copy}}}(X|Y)$, since in both cases $\vec{o_b}$ and $\vec{o_c}$ take the only possible consistent value $\vec{o_b} = \vec{b} \ominus \vec{a}$ and $\vec{o_c} = \vec{c} \ominus \vec{a}$. Notice that $R_{\widehat{\mathtt{copy}}}$ can be computed by a circuit of depth 1 because on inputs $\vec{a}, \vec{b}, \vec{c}$ it needs only to compute $\vec{o_b}, \vec{o_c}$, both requiring a $\ominus$ operation. The size of $\mathsf{REC}_{\widehat{\mathtt{copy}}}$ is $O(k)$ for computing the $2k$ $\ominus$ operations. $\square$

B Proofs Omitted from Section 4.4

Our goal is to prove Lemma 12, i.e., to show a local reconstructor for $\widehat{\mathtt{mult}}$ gadgets. First, we present a technical lemma (Lemma 17), which says that an encoding of 0 is cannot be distinguished from an encoding of 1 even given access to noisy leakage of several copies of that encoding (offset by fixed vectors), as well as the inner product of that encoding and a another vector whose Hamming weight is not too high. This lemma will then be used in a hybrid argument of Lemma 12 to show that the reconstructor we build for the $\widehat{\mathtt{mult}}$ gadget is leakage indistinguishable.

Specifically, Lemma 17 talks about statistical closeness of the distributions D_0 and D_1 defined as follows.

For any constant ℓ, any constant vectors $\vec{c}_1, \ldots, \vec{c}_\ell, \vec{c} \in \{0,1\}^k$ and any $b \in \{0,1\}$, let:

$$D_b := D_b(p, \ell, \vec{c}_1, \ldots, \vec{c}_\ell, c) = \Big((\mathcal{N}_p(\vec{e} \oplus \vec{c}_1), \ldots, \mathcal{N}_p(\vec{e} \oplus \vec{c}_\ell)), \langle \vec{e}, \vec{c} \rangle \Big)_{\vec{e} \leftarrow \mathsf{Enc}(b)}$$

Here, $\langle \vec{e}, \vec{c} \rangle$ denotes the inner product of $\vec{e}$ and $\vec{c}$.

Lemma 17. *Let $p \in (0, 1/2]$, and $\ell \in \mathbb{N}$ be constants, and let $\vec{c}_1, \ldots, \vec{c}_\ell \in \{0,1\}^k$ and $\vec{c} \in \{0,1\}^k$ be k-bit vectors such that $\vec{c}$ has Hamming weight at most t (and all the other $\vec{c}_i$ are unrestricted). Then, $\Delta(D_0; D_1) \leq (1 - (2p)^\ell)^{k-t}$.*

Proof. We prove this statement based on two claims. Let I denote the positions where $\vec{c}$ is 0, and let $\vec{e}_I$ denote the restriction of $\vec{e}$ to the positions in I. For every $\varsigma \in \{0,1\}^t$ and $b \in \{0,1\}$, define the distribution $D_{b,\varsigma}$ as follows. Construct a vector $\vec{e}$ such that $\vec{e}_{\bar{I}} = \varsigma$, and $\vec{e}_I$ is a random sharing of the bit $b' = b \oplus \langle \varsigma, 111\ldots11 \rangle$. Generate the distribution D_b using this vector $\vec{e}$ and the vectors $\vec{c}, \vec{c}_1, \ldots, \vec{c}_\ell$.

Claim 3. *For every $\varsigma \in \{0,1\}^t$ the statistical distance $\Delta(D_{0,\varsigma}; D_{1,\varsigma}) \leq (1 - (2p)^\ell)^{k-t}$.*

Proof. For a fixed ς the last bit of both distributions $D_{0,\varsigma}$ and $D_{1,\varsigma}$ is fixed to $\langle \vec{e}, \vec{c} \rangle = \langle \varsigma, 111\ldots11 \rangle$. Additionally, in both distributions it fixes t bits of each $\vec{e} \oplus \vec{c}_i$ to the same string. Hence, what remains are $k-t$ coordinates in each $\vec{e} \oplus \vec{c}_i$. By definition of the distribution we have that $\bigoplus_{j \in I} e_j = b'$, with b' defined as $b \oplus \langle \varsigma, 111\ldots11 \rangle$. Notice that this guarantees that $\bigoplus_j e_j = b$. Since the only difference between these two distributions are that in each $\vec{e} \oplus \vec{c}_i$ those $k-t$ coordinates are either a random encoding of $0 \oplus \langle \varsigma, 111\ldots11 \rangle$ or $1 \oplus \langle \varsigma, 111\ldots11 \rangle$ we get $\Delta(D_{0,\varsigma}; D_{1,\varsigma}) \leq (1 - (2p)^\ell)^{k-t}$ by applying Lemma 4. This concludes the proof. $\square$

Claim 4. *If $\Delta(D_{0,\varsigma}; D_{1,\varsigma}) \leq \epsilon$ for every $\varsigma \in \{0,1\}^t$, then $\Delta(D_0; D_1) \leq \epsilon$.*

Proof. Obviously the distribution D_b can alternatively be obtained by first sampling a random ς, and then producing a sample from $D_{b,\varsigma}$. Such a sampling gives us by an elementary calculation

$$\Delta(D_0; D_1) \leq \left(\frac{1}{2}\right)^t 2^t \epsilon = \epsilon.$$

□

Putting the two claims together we get that $\Delta(D_0; D_1) \leq (1-(2p)^\ell)^{k-t}$, which concludes the proof. □

Proof of Lemma 12. The internals of the $\widehat{\texttt{mult}}$ gadget consist of the inputs $(\vec{a}, \vec{b})$, the output $\vec{q}$ and the intermediate values, namely

$$(\vec{r}^{(1)}, \ldots, \vec{r}^{(k)}, \vec{s}^{(1)}, \ldots, \vec{s}^{(k)}, \vec{a}^{(1)}, \ldots, \vec{a}^{(k)}, \vec{b}^{(1)}, \ldots, \vec{b}^{(k)}, u, \vec{w}, \vec{z}).$$

The reconstructor $\mathsf{REC}_{\widehat{\texttt{mult}}}$ for the $\widehat{\texttt{mult}}$ gadget is a distribution over functions $R_{\widehat{\texttt{mult}}}$, which for plausible inputs $(X = (\vec{a}, \vec{b}), Y = (\vec{q}))$ proceeds as follows:

1. Set $\vec{a}^{(0)} = \vec{a}$ and $\vec{b}^{(0)} = \vec{b}$ and choose uniformly random vectors $\vec{a}^{(1)}, \ldots, \vec{a}^{(k)}, \vec{b}^{(1)}, \ldots, \vec{b}^{(k)}$. $\vec{A} = \{\vec{a}^{(i)}\}_{i \in [1,k]}$ and $\vec{B} = \{\vec{b}^{(i)}\}_{i \in [1,k]}$ can be hard-wired into the output of $R_{\widehat{\texttt{mult}}}$.
2. From $\vec{a}^{(0)}, \ldots, \vec{a}^{(k)}, \vec{b}^{(0)}, \ldots, \vec{b}^{(k)}$ compute for $i \in [1,k]$: $\vec{r}^{(i)} = \vec{a}^{(i-1)} \oplus \vec{a}^{(i)}$ and $\vec{s}^{(i)} = \vec{b}^{(i-1)} \oplus \vec{b}^{(i)}$. All $\{\vec{r}^{(i)}\}_{i \in [2,k]}$ and $\{\vec{s}^{(i)}\}_{i \in [2,k]}$ can be hard-wired into the reconstructor. $\vec{r}^{(1)}$ is computed from the reconstructor's input $\vec{a}^{(0)}$ and $\vec{s}^{(1)}$ from $\vec{b}^{(0)}$, respectively.
3. Choose u uniformly at random and compute from $\vec{A} = (\vec{a}^{(i)})_{i \in [1,k]}$, $(\vec{b}^{(i)})_{i \in [1,k]}$ and u the vector $\vec{z}$. These values can be hardwired into the reconstructor.
4. Compute $\vec{w} = \vec{z} \oplus \vec{q}$ from the reconstructor's input $\vec{q}$.
5. Output $(\vec{a}, \vec{b}, \{\vec{r}^{(i)}, \vec{s}^{(i)}, \vec{a}^{(i)}, \vec{b}^{(i)}\}_{i=1}^k, u, \vec{q}, \vec{z}, \vec{w})$.

We first discuss the underlying function class of $\mathsf{REC}_{\widehat{\texttt{mult}}}$.

Claim 5. *The support of* $\mathsf{REC}_{\widehat{\texttt{mult}}}$ *is in* $\mathsf{LOCAL}(2)$.

Proof. We must show that for each input $\vec{a}, \vec{b}, \vec{q}$ the output of the reconstructor is either a fixed constant or can be written as one of the inputs plus a constant vector. For input $\vec{a}$ the other inputs $\vec{b}$ and $\vec{q}$ are fixed constants and, hence, the output of $R_{\widehat{\texttt{mult}}}$ is constant except $(\vec{a}^{(0)} = \vec{a}, \vec{r}^{(1)} = \vec{a} \oplus \vec{a}^{(1)})$, where $\vec{a}^{(1)}$ is a constant vector. The same analysis works for $\vec{b}$ and $\vec{q}$, which yields that $R_{\widehat{\texttt{mult}}} \in \mathsf{LOCAL}(2)$. □

It is easy to see that reconstructed wire assignment differs from the wire assignment produced by a real execution of $\widehat{\texttt{mult}}$. Nevertheless, we show that the output of the reconstructor (on plausible inputs $\vec{a}, \vec{b}, \vec{q}$) is $\mathcal{N}_p$ leakage-indistinguishable from the distribution produced during the operation of the $\widehat{\texttt{mult}}$ gadget on inputs $\vec{a}$ and $\vec{b}$, conditioned on the output being $\vec{q}$. There are two main differences between the reconstructor distribution and the real wire distribution. First,the vectors $\vec{a}^{(i)}$ and $\vec{b}^{(i)}$ are uniformly random whereas in the real world, they are random encodings of a (resp. b). Secondly, in the reconstructor distribution, the bit u is uniformly random whereas in the real world, $u = \vec{R} \bigcirc \vec{S}^{\mathsf{T}}$ (where the matrices $\vec{R}$ and $\vec{S}$ are as in Section 3.3).

The indistinguishability is shown by a hybrid argument – consider the following $2k+1$ hybrids.

- *Hybrid H_0*: This is the real distribution, conditioned on plausible values $\vec{a}, \vec{b}, \vec{q}$. We will take an alternative view of this distribution, by first sampling the vectors $\vec{a}^{(i)}$ and $\vec{b}^{(i)}$ as random encodings of a and b respectively, and then defining the vectors $\vec{r}^{(i)}$ and $\vec{s}^{(i)}$ as

  $$\vec{r}^{(i)} = \vec{a}^{(i-1)} \oplus \vec{a}^{(i)} \text{ and } \vec{s}^{(i)} = \vec{b}^{(i-1)} \oplus \vec{b}^{(i)}$$

  exactly as in the real wire distribution. It is easy to see that this is exactly distributed as the real wire assignment (conditioned on the inputs being $\vec{a}, \vec{b}$ and the output being $\vec{q}$).

- *Hybrid H_ℓ, for $1 \le \ell \le k$*: H_ℓ is the same as $H_{\ell-1}$, except that the vector $\vec{a}^{(\ell)}$ – which is a random encoding of a in $H_{\ell-1}$ – is replaced with a uniformly random vector. The vector $\vec{r}^{(\ell)}$ is computed as $\vec{a}^{(\ell-1)} \oplus \vec{a}^{(\ell)}$ and the vector $\vec{r}^{(\ell+1)}$ is $\vec{a}^{(\ell+1)} \oplus \vec{a}^{(\ell)}$. The rest of the values are computed exactly as in the previous hybrids.

- *Hybrid $H_{k+\ell}$ for $1 \le \ell \le k$*: $H_{k+\ell}$ is the same as $H_{k+\ell-1}$, except that the vector $\vec{b}^{(\ell)}$ – which is a random encoding of b in $H_{k+\ell-1}$ – is replaced with a uniformly random vector. The vector $\vec{s}^{(\ell)}$ is computed as $\vec{b}^{(\ell-1)} \oplus \vec{b}^{(\ell)}$ and $\vec{s}^{(\ell+1)}$ is $\vec{b}^{(\ell+1)} \oplus \vec{b}^{(\ell)}$. The rest of the values are computed exactly as in the previous hybrids.

- *Hybrid H_{2k+1}*: H_{2k+1} is the same as H_{2k} except that the bit $u = \vec{R} \bigcirc \vec{S}^\intercal$ is replaced with a uniformly random bit.

We show that the hybrid H_0 (the real distribution) is indistinguishable from the hybrid H_{2k+1} (the output of the reconstructor) in the following three claims:

Claim 6. *For every $p \in (0, \frac{1}{2}]$ and $1 \le \ell \le k$, the hybrids H_ℓ and $H_{\ell-1}$ are $(\mathcal{N}_p, \infty, \epsilon(k))$-leakage-indistinguishable, where $\epsilon(k) \le \exp(-15kp^5) + \exp(-k/512)$.*

Proof. The only difference between hybrids H_ℓ and $H_{\ell-1}$ is in the vector $\vec{a}^{(\ell)}$, and also, some of the other quantities that are computed using $\vec{a}^{(\ell)}$. Similar to Lemma 10 we define a distribution F^ℓ ($\ell \in [1, k]$) over functions f_S that take as input an encoding $\vec{e}$ and embeds it at $\vec{a}^{(\ell)}$. If $\vec{e}$ is an encoding of a then f_S outputs a wire assignment that is distributed as $H_{\ell-1}$. If on the other hand f_S takes as input a random encoding then it produces the distribution H_ℓ. In contrast to Lemma 10 where we were mainly concerned that f_S is shallow, in this claim we must guarantee that f_S uses its input only a limited number of times. We solve this by hard-wiring most of f_S outputs directly into the function.

For ease of notation below we will denote the vectors $\tilde{\vec{r}}^{(i)}$ as the row vectors of the matrix $\vec{R}$ (notice that these vectors are not visible to the leakage function), i.e.,

$$\vec{R} = \begin{pmatrix} \tilde{\vec{r}}^{(1)} \\ \vdots \\ \tilde{\vec{r}}^{(k)} \end{pmatrix} = \begin{pmatrix} \vec{r}^{(1)} \\ \vdots \\ \bigoplus_{j \in [1,i]} \vec{r}^{(j)} \\ \vdots \\ \bigoplus_{j \in [1,k]} \vec{r}^{(j)} \end{pmatrix}$$

We define the distribution F^ℓ by drawing f_S as follows:

1. Set $\vec{b}^{(0)} = \vec{b}$ and sample uniformly at random $\{\vec{s}_i\}_{i\in[1,k]}$ which are encodings of 0. Compute $\vec{b}_i = \vec{b}_{i-1} \oplus \vec{s}_i$ and hard-wire the results as fixed outputs into f_S. Notice also that $\{\vec{s}_i\}_{i\in[1,k]}$ allows us to compute (without making use of the inputs) the matrix $\vec{S}$.

2. Hard-wire the vectors $\{\vec{a}^{(i)}\}_{i\neq\ell}$ into f_S: for $i < \ell$ $\vec{a}^{(i)}$ is a uniformly random k bit string. For $i > \ell$ $\vec{a}^{(i)}$ is a uniformly random encoding of a. Further, set $\vec{a}^{(0)} = \vec{a}$ and compute for all $i \notin \{\ell, \ell+1\}$ the vectors $\vec{r}^{(i)} = \vec{a}^{(i-1)} \oplus \vec{a}^{(i)}$. Hard-wire the result into f_S.

3. For all $i \neq \ell$ compute $\tilde{\vec{r}}^{(i)}$ as $\vec{a}^{(0)} \oplus \vec{a}^{(i)}$. Hard-wire these results into the description of f_S as intermediate values that will later be used to compute u.

4. On input $\vec{e}$ f_S sets $\vec{a}^{(\ell)} = \vec{e}$ and computes $\vec{r}^{(\ell)} = \vec{a}^{(\ell-1)} \oplus \vec{a}^{(\ell)}$ and $\vec{r}^{(\ell+1)} = \vec{a}^{(\ell)} \oplus \vec{a}^{(\ell+1)}$. Further compute $\tilde{\vec{r}}^{\ell}$ as $\vec{a}^{(0)} \oplus \vec{a}^{(\ell)}$. The vectors $\vec{r}^{(\ell)}$ and $\vec{r}^{(\ell+1)}$ will be part of f_S's output. $\tilde{\vec{r}}^{\ell}$ will be used in the next step to compute the bit u.

5. Notice that the whole matrix $\vec{S}$ and all rows except $\tilde{\vec{r}}^{(\ell)}$ of $\vec{R}$ are hard-wired into f_S. To compute u the function f_S computes $u = \vec{R} \bigcirc \vec{S}^{\mathsf{T}}$.

6. From $u, \vec{A}, \vec{B}$ f_S computes $\vec{z}$ and together with $\vec{q}$ (which is hard-wired into f_S) the vector $\vec{w} = \vec{z} \oplus \vec{q}$.

By inspection of the above it is easy to see that if $\vec{e}$ is an encoding of a then $\vec{a}^{(\ell)}$ is an encoding of a. Since all other values are computed honestly from $\vec{a}^{(\ell)}$, and hard-wired values (that have the correct distribution), we get $f_S(\vec{e}) \equiv H_{i-1}$. Similarly if $\vec{e}$ is a random vector, then $f_S(\vec{e}) \equiv H_i$. Let us next bound $\Delta(H_{i-1}; H_i)$.

Most of the outputs of f_S are hard-wired into the function. There are some exceptions. The values computed in Step 4, $\vec{a}^{(\ell)}$, $\vec{r}^{(\ell)}$ and $\vec{r}^{(\ell+1)}$, have the form of $\vec{e}$ plus some constant. In Step 5 we compute u, which indirectly depends on $\vec{e}$ (via $\tilde{\vec{r}}^{(\ell)}$). Essentially, for all $i \neq \ell$ we can compute the inner product of the ith row of $\vec{R}$ and the ith column of $\vec{S}$ "off-line" and hardwire the result into f_S as an intermediate value. Additionally, $\tilde{\vec{r}}^{(\ell)}$ can be written as $\vec{a}^{(0)} \oplus \vec{a}^{(\ell)}$, where $\vec{a}^{(0)}$ is a fixed constant. Hence, the inner product of $\vec{a}^{(0)}$ and the ℓth column of $\vec{S}$ is a fixed constant as well. We denote the sum of all these fixed inner products as d. What remains is the inner product of $\vec{a}^{(\ell)} = \vec{e}$ and a fixed constant vector $\vec{c}$ which represents the ℓth column of $\vec{S}$. To sum it up we have $u = \langle \vec{e}, \vec{c} \rangle \oplus d$, where $\vec{c}$ is a uniformly random fixed vector and d some fixed constant.

Finally, in Step 6 we compute vectors $\vec{z}$ and $\vec{w}$ of length k^2. $k^2 - k$ bits of these vectors can be fixed and are independent of $\vec{e}$. For $\vec{z}$, these elements have the form $e_i \odot c_i$ for some constant c_i. For $\vec{w}$, they are equal to $e_i \odot c_i \oplus q_i$, where q_i are fixed constants. If the bits c_i are all 1 then we have to vectors of the form $\vec{e}$ plus a constant.

To conclude, in total we have at most 5 outputs of the form $\vec{e}$ plus a fixed constant vector, and the bit $\langle \vec{e}, \vec{c} \rangle \oplus d$. If we apply a noisy function $\mathcal{N}_p$ (for some $p \in (0, 1/2]$) to the outputs of f_S, then conditioned that $\vec{c}$ having Hamming weight at most $17k/32$, we get with Lemma 17 that the statistical distance between H_{i-1} and H_i is upper bounded by

$$(1 - (2p)^5)^{15k/32} \leq \exp(-15kp^5)$$

(here we are using $(1 - x) \leq \exp(-x)$). Since $\vec{c}$ is a random k bit vector, the probability that its Hamming is greater than $17k/32$ is less than $\exp(-k/512)$ by Hoeffding's Inequality [23]. Thus, we get

$$\begin{aligned}\Delta(H_{i-1}, H_i) &\le \Delta(H_{i-1}, H_i|E)\Pr[E] + \Delta(H_{i-1}, H_i|\bar{E})\Pr[\bar{E}] \\ &< \exp(-15kp^5)(1-\exp(-k/512)) + 1\cdot\exp(-k/512) \\ &< \exp(-15kp^5) + \exp(-k/512)\,.\end{aligned}$$

□

Claim 7. *For every $p \in (0, \frac{1}{2}]$ and $1 \le \ell \le k$, the hybrids $H_{k+\ell}$ and $H_{k+\ell-1}$ are $(\mathcal{N}_p, \infty, \epsilon(k))$-leakage-indistinguishable, where $\epsilon(k) \le \exp(-15kp^5) + \exp(-k/512)$.*

Proof. The proof follows along the lines of the proof of Claim 6. □

Claim 8. *For every $p \in (0, \frac{1}{2}]$, the hybrids H_{2k+1} and H_{2k} are $(\mathcal{N}_p, \infty, \epsilon(k))$-leakage-indistinguishable, where $\epsilon(k) \le \exp(-15kp/16) + \exp(-k/512)$.*

Proof. The difference between the two hybrids is that in the former, the bit u is computed as $\vec{R}\bigcirc\vec{S}^{\mathsf{T}}$, whereas in the latter, it is uniformly random. To show that they are leakage-indistinguishable, observe that for every setting of the matrix $\vec{S}$ and all the rows of $\vec{R}$ except the first, $u = d \oplus \langle \vec{r}^{(1)}, \vec{c}\rangle$, where d is a fixed bit that depends on $\vec{S}$ and the remaining rows of $\vec{R}$, and $\vec{c}$ is the first column of $\vec{S}$. Assuming $\vec{c}$ has Hamming weight at least $15k/32$ (which it does with probability at least $1 - \exp(-k/512)$ by Hoeffding's Inequality [23]), distinguishing u from random is equivalent to distinguishing the XOR of at least $15k/32$ bits of $\vec{r}^{(1)}$ from random given $\mathcal{N}_p(\vec{r}^{(1)})$. By Lemma 4, it cannot be distingushed from random with advantage better than $(1-2p)^{15k/32} \le \exp(-15kp/16)$ (using $(1-x) \le \exp(-x)$). □

Putting the three claims together we get that the distribution output by the reconstructor and the real wire distribution are $(\mathcal{N}_p, \infty, \epsilon(k))$-leakage-indistinguishable where

$$\begin{aligned}\epsilon(k) &\le k\cdot(\exp(-15kp^5) + \exp(-k/512)) + \\ &\quad\; k\cdot(\exp(-15kp^5) + \exp(-k/512)) + \exp(-15kp/16) + \exp(-k/512) \\ &\le (2k+1)(\exp(-15kp^5) + \exp(-k/512))\end{aligned}$$

for any constant $p \in (0, 1/2]$ (because $15kp/16 \ge 15kp^5$ for $p \le 1/2$). This concludes the proof. □

C Proofs Omitted from Section 4.5

Proof of Corollary 1. Recall from the corollary statement that $\epsilon_\Pi > 0, \tau_\Pi > 0$ and $\mathcal{L}_\Pi$, $\mathcal{L}_{\widehat{C}}$ are some leakage classes that satisfy $\mathcal{L}_{\widehat{C}} \circ \mathsf{SHALLOW}(3, O(k^2))) \subseteq \mathcal{L}_\Pi$. By definition, TR_C is an encoding based circuit transformation, where by Section 4.3 all gadgets in $\widehat{C}$ are rerandomizing and $(\mathcal{L}_{\widehat{C}}, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-reconstructible by $\mathsf{SHALLOW}(2, O(k^2))$ for some parameters $\tau_{\widehat{g}}, \epsilon_{\widehat{g}}$. Since our transformation has to work for *any* circuit we can assume wlog[16] that C is made solely of $\odot$ gates. By Lemma 9, we get then $\tau_{\widehat{g}} = \tau_\Pi - O(k^2)$ and $\epsilon_{\widehat{g}} = k\epsilon_\Pi$.

[16]since the parameters are worst in this case

We are now ready to apply Lemma 13 and get:

$$\begin{aligned}\mathcal{R}_{\widehat{C}} &= (n_{\rm I}+n_{\rm O}) \times \mathcal{R}_{\widehat{g}} = (n_{\rm I}+n_{\rm O}) \times \mathsf{SHALLOW}(2, O(k^2)) \\ &= \mathsf{SHALLOW}(2, (n_{\rm I}+n_{\rm O})O(k^2)).\end{aligned}$$

Notice that $(n_{\rm I}+n_{\rm O}) \times \mathcal{R}_{\widehat{g}}$ denotes parallel execution of the reconstructors with different inputs. Further, we have

- $\tau_{\widehat{C}} \leq \min(\tau_\Pi, \tau_{\widehat{g}}) - s\tau_{\mathsf{samp}} = \tau_\Pi - O(sk^2)$, since all our gadgets have size $O(k^2)$,
- $\epsilon_{\widehat{C}} \leq s(m\epsilon_\Pi + \epsilon_{\widehat{g}}) = s\epsilon_\Pi(k+2)$ (because every gate has fan-in at most two, the number of wires m is at most $2s$).

This concludes the proof. □

Proof of Lemma 14. We can view $\widehat{\odot}$ as a circuit composed of k $\widehat{\oplus}$ gadgets (from $\widehat{\texttt{compress}}$) and a single $\widehat{\texttt{mult}}$ gadget. Hence, the circuit consists of $s = k+1$ gadgets, takes $n_{\rm I} = 2$ inputs (in encoded form), outputs $n_{\rm O} = 1$ encodings, and has $m = 2k-2$ wire bundles that connect gadgets. In Section 4.4 we showed that $\widehat{\oplus}$ is $(\mathcal{L}, \infty, 0)$-reconstructible by $\mathsf{LOCAL}(3)$, for any $\mathcal{L}$, and $\widehat{\texttt{mult}}$ is $(\mathcal{N}_p, \infty, \epsilon_{\widehat{\texttt{mult}}})$-reconstructible by $\mathsf{LOCAL}(2)$ with $\epsilon_{\widehat{\texttt{mult}}} \leq (2k+1)(\exp(-15kp^5) + \exp(-k/512))$.

Further, by Proposition 2 the Π_{parity} encoding is $(\mathcal{N}_p \circ \mathsf{LOCAL}(6), \infty, (1-(2p)^6)^k)$-leakage indistinguishable. Note that $1-(2p)^6 \leq \exp(-(2p)^6)$.

We can put these things together by Lemma 13. In fact, because $\epsilon_{\widehat{\oplus}} = 0$ for all the $\widehat{\oplus}$ gadgets, we can improve the analysis of Lemma 13 (which assumed, for simplicity, that every gadget has the same ϵ as the worst gadget): since Claim 1 is applied to $\widehat{\texttt{mult}}$ only once, we can replace $s\epsilon_{\widehat{g}}$ by just $\epsilon_{\widehat{\texttt{mult}}}$ in the statement of Lemma 13. We therefore get that $\widehat{\odot}$ is $(\mathcal{N}_p, \infty, \epsilon_{\widehat{\odot}})$-reconstructible by $\mathsf{LOCAL}(3)$ with

$$\begin{aligned}\epsilon_{\widehat{\odot}} &\leq m\epsilon_\Pi + \epsilon_{\widehat{\texttt{mult}}} = (2k-2)\cdot(1-(2p)^6)^k + (2k+1)(\exp(-15kp^5) + \exp(-k/512)) \\ &\leq (2k+1)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512))\,.\end{aligned}$$

Notice that the conditions on the leakage classes required by Lemma 13 are satisfied. The reason for this is that by our choice of the parameters the Π_{parity} encoding tolerates leakages from $2 \times \mathsf{LOCAL}(3) = \mathsf{LOCAL}(6)$, where $2 \times \mathsf{LOCAL}(3)$ is parallel execution on the same inputs. □

Proof sketch of Corollary 2. The proof is similar to the proof of Corollary 1 and we only provide a sketch here. By definition, TR_N is an encoding based circuit transformation, where by Section 4.4 all gadgets in $\widehat{C}$ are rerandomizing and $(\mathcal{N}_p, \tau_{\widehat{g}}, \epsilon_{\widehat{g}})$-reconstructible by $\mathsf{LOCAL}(3)$ for some parameters $\tau_{\widehat{g}}, \epsilon_{\widehat{g}}$. Since our transformation has to work for *any* circuit we can assume without loss of generality[17] that C is made solely of $\odot$ gates. By Lemma 14, we get then $\tau_{\widehat{g}} = \infty$, $\mathcal{L}_{\widehat{g}} = \mathcal{N}_p$ and $\epsilon_{\widehat{g}} \leq (2k+1)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512))$.

Further, by Proposition 2 the Π_{parity} encoding is $(\mathcal{L}_\Pi, \infty, (1-(2p)^6)^k)$-leakage indistinguishable, for $\mathcal{L}_\Pi = \mathcal{N}_p \circ \mathsf{LOCAL}(6)$.

We are now ready to apply Lemma 13 and get $\mathcal{R}_{\widehat{C}} \subseteq \mathsf{LOCAL}(3)$. Further, we have

- $\tau_{\widehat{C}} \leq \infty$

[17] since the parameters are worst in this case

```
Simulator $\mathcal{S}(\mathcal{A}, q, C)$
1. Sample uniformly at random encodings $(Z_0, \ldots, Z_q)$, where each $Z_i$ consists of
      $n$ encodings of random elements of $\mathcal{K}$
2. Run $\mathcal{A}(q, C)$
3. For each query $(f_i, x_i)$ of $\mathcal{A}$:
4.     Query $C[m_{i-1}]$ on input $x_i$ to obtain $y_i$ and sample $Y_i \leftarrow \mathsf{Enc}(y_i)$
5.     Compute wire assignment $W_E$ for the $\widehat{\texttt{encoder}}$ with input $x_i$ and its output $X_i$
6.     Compute wire assignment $W_D$ for $\widehat{\texttt{decoder}}$ gadget with input $Y_i$ and output $y_i$
7.     Sample for each connecting wire bundle in $\widehat{C}^*$ a random encoding $\vec{v} \leftarrow \mathsf{Enc}(v)$ with $v \leftarrow \mathcal{K}$
8.     For each gadget $\widehat{g}$ in $\widehat{C}^*$ run the corresponding reconstructor $\mathcal{W}_{\widehat{g}} \leftarrow \mathsf{REC}_{\widehat{g}}(U, V)$,
          where $U$ are the encoded inputs and $V$ are the encoded outputs of the gadgets $\widehat{g}$.
          Notice that $U$ and $V$ are part of the connecting wire bundles sampled above.
9.     Let $\mathcal{W}_{\widehat{C}^*}$ denote the wire assignment composed from $\mathcal{W}_{\widehat{g}}$ in Step (7) and (8).
          Return $(f_i(W_E, \mathcal{W}_{\widehat{C}^*}, W_D), y_i)$ to $\mathcal{A}$
10. Return the output of $\mathcal{A}$.
```

Figure 8: Description of the simulator $\mathcal{S}$ that runs in the experiment $\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}$.

- $\epsilon_{\widehat{C}} \leq s(2(1-(2p)^6)^k + (2k+1)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512)) \leq s(2k+3)(\exp(-64kp^6) + \exp(-15kp^5) + \exp(-k/512))$ (here we use the fact that every gate has fan-in at most two, the number of wires m is at most $2s$).

This concludes the proof. □

D Proofs Omitted from Section 4.6

Proof of Lemma 15. Without loss of generality assume that $\widehat{C}$ contains at least one rerandomizing and reconstructible gadget (i.e., $\widehat{C}$ is not the empty circuit). The simulator $\mathcal{S}$ is formally defined in Figure 8. $\mathcal{S}$ does not know the real initial secret state m_0 but instead uses random encodings $(Z_0, \ldots, Z_q)$. Furthermore, it computes the internal wires of $\widehat{C}^*$ for each round $i \in [1, q]$ with the reconstructor $\mathsf{REC}_{\widehat{g}}$ of the corresponding gadgets (cf. line 8 in Figure 8). We show in this lemma that $\mathcal{S}$ running in experiment $\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}$ produces an output that is indistinguishable from $\mathcal{A}$'s output in $\mathsf{Exp}^{\mathrm{real}}_{\mathsf{TR}}$ (cf. Definition 1 in Section 3 for the description of these experiments). The proof relies on techniques developed in Lemma 13 and uses a hybrid argument with a series of hybrid simulators.

The first hybrid simulators will replace the real wire assignment of $\widehat{C}^*$ with wire assignments produced by reconstructors of the gadgets. Then, we replace step-by-step the elements of the secret state $\widehat{m}_i$ and the connecting wires (between the gadgets) with random encodings. Once we have done this replacement, the simulator is as in Figure 8.

More formally, we consider the following series of hybrid simulators:

- Simulators $\mathcal{S}_1^0, \ldots, \mathcal{S}_1^{q \cdot s}$: for each $i \in [1, qs]$, let $j = i \mod q$. We define the simulator $\mathcal{S}_1^i$ as $\mathcal{S}_1^{i-1}$ except that in the $(\lfloor i/q \rfloor + 1)$th execution, we replace the $(j+1)$th gadget $\widehat{g}$ with its reconstructor $R_{\widehat{g}} \leftarrow \mathsf{REC}_{\widehat{g}}$. Notice that $\mathcal{S}_1^0$ is essentially the real execution of the original circuit. Notice further that as in Lemma 13 the inputs and outputs of the reconstructor are as in the real execution with $\widehat{g}$.

- Simulators $\mathcal{S}_2^0, \dots, \mathcal{S}_2^{q \cdot m}$:
  - $\mathcal{S}_2^0$ is defined as $\mathcal{S}_1^{q \cdot s}$
  - for each $i \in [1, q \cdot m]$ we define the simulator $\mathcal{S}_2^i$ as $\mathcal{S}_2^{i-1}$ except that we replace the ith connecting wire bundle by a random encoding sampled from $\mathsf{Enc}(\cdot)$. Notice that in each of the q execution of $\widehat{C}^*$ there are at most m connecting wires.
- Simulators $\mathcal{S}^{0,0}, \mathcal{S}^{0,1}, \dots, \mathcal{S}^{0,n}, \mathcal{S}^{1,1}, \dots, \mathcal{S}^{q,n-1}, \mathcal{S}^{q,n}$:
  - $\mathcal{S}^{0,0}$ is defined as $\mathcal{S}_2^{q \cdot m}$
  - $\mathcal{S}^{i,j}$ for $i \in [0, q], j \in [1, n]$: This is as the previous simulator, but where the jth element of the ith state is replaced with a random encoding.

Notice that in the simulation given by $\mathcal{S}_2^{q \cdot m}$ the simulator essentially replaces the wire assignment of $\widehat{C}^*$ with the wire assignment produced by the reconstructor of the stateless circuit $\widehat{C}^*$. Hence, the indistinguishability of the simulations given by $\mathcal{S}_1^0$ and $\mathcal{S}_2^{q \cdot m}$ follows essentially from Lemma 13. We repeat the important parts here to obtain the final parameters for our result.

Before we show indistinguishability of the hybrid simulators, we notice that for ease of notation we omit to explicitly specify the wire assignment for the $\widehat{\texttt{encoder}}$ and $\widehat{\texttt{decoder}}$ gadgets (i.e., W_E and W_D in Figure 8). Indeed, we can easily incorporate them into the simulation, since their inputs and outputs are known to the simulator.

The indistinguishability of the hybrid simulations $\mathcal{S}_1^{i-1}$ and $\mathcal{S}_1^i$, follows directly from Claim 1 (cf. Lemma 13 in Section 4.5) where it was shown that for transformed stateless circuits we can replace the real wire assignments of gadgets with the wire assignments of reconstructors. This yields that the computational distance between the simulation of $\mathcal{S}_1^{i-1}$ and $\mathcal{S}_1^i$ is upper bounded by $\epsilon_{\widehat{g}}$ for any $(\mathcal{L}, \tau_{\mathcal{A}})$-adversary and distinguisher running in time τ_D, where $\mathcal{L} \subseteq \mathcal{L}_{\widehat{g}}$ and $\tau_{\mathcal{A}} + \tau_D \leq \tau_{\widehat{g}} - qs\tau_{\mathsf{samp}}$. By applying this result repeatedly over the q rounds and s gates that are evaluated in each round, we get for any $(\mathcal{L}, \tau_{\mathcal{A}})$-adversary $\mathcal{A}$:

$$\mathsf{Exp}_{\mathsf{TR}}^{\mathrm{real}}(\mathcal{A}, \mathcal{L}, q, C, m_0, k) \approx_{\tau_D, q \cdot s \cdot \epsilon_{\widehat{g}}} \mathsf{Exp}_{\mathsf{TR}}^{\mathrm{sim}}(\mathcal{S}_1^{qs}, \mathcal{A}, q, C, m_0, k)\,. \tag{10}$$

We next need to show that for each $i \in [1, q \cdot m]$ the simulations given by $\mathcal{S}_2^{i-1}$ and $\mathcal{S}_2^i$ are computationally close. That is, we can show that for each connecting wire we can replace the real encoding with a random encoding without getting noticed. This was shown in Claim 2 (cf. Lemma 13 in Section 4.5). More precisely, the computational distance between the simulation of $\mathcal{S}_2^{i-1}$ and $\mathcal{S}_2^i$ is upper bounded by ϵ_Π for any $(\mathcal{L}', \tau_{\mathcal{A}})$-adversary and distinguisher running in time τ_D, where $\mathcal{L}' \circ (2 \times \mathcal{R}_{\widehat{g}}) \subseteq \mathcal{L}_\Pi$ and $\tau_{\mathcal{A}} + \tau_D \leq \tau_\Pi - qs\tau_{\mathsf{samp}}$. Applying this result repeatedly over the q rounds and m wires for each round, we get for any $(\mathcal{L}', \tau_{\mathcal{A}})$-adversary $\mathcal{A}$:

$$\mathsf{Exp}_{\mathsf{TR}}^{\mathrm{sim}}(\mathcal{S}_2^0, \mathcal{A}, q, C, m_0, k) \approx_{\tau_D, q \cdot m \cdot \epsilon_\Pi} \mathsf{Exp}_{\mathsf{TR}}^{\mathrm{sim}}(\mathcal{S}_2^{qm}, \mathcal{A}, q, C, m_0, k)\,. \tag{11}$$

Notice that in the simulation of $\mathcal{S}^{0,0}$ we replaced the gadgets in all of the q rounds by reconstructors and the encodings on the connecting wires between the gadgets by random encodings. To obtain our final simulation, we show that we can replace step-by-step the memory by random encodings. We prove this along the lines of Claim 2 in Lemma 13. For ease of notation, we identify in the following $\mathcal{S}^{i,0}$ with $\mathcal{S}^{i-1,n}$, for $i > 0$.

Claim 9. *Suppose Π is 2-adaptive $(\mathcal{L}_{2\Pi}, \tau_{2\Pi}, \epsilon_{2\Pi})$-leakage indistinguishable, $\mathcal{A}$ is a q-adaptive $(\mathcal{L}_{\mathcal{W}}, \tau_{\mathcal{W}})$-adversary such that $(\mathcal{L}_{\mathcal{W}} \circ \mathcal{R}_{\widehat{g}}) \subseteq \mathcal{L}_{2\Pi}$, where $\mathcal{R}_{\widehat{g}}$ is the class of reconstructors for gadgets $\widehat{g}$, and τ_D is the distinguisher running time satisfying $\tau_{\mathcal{W}} + \tau_D \leq \tau_{2\Pi} - qs\tau_{\mathsf{samp}}$. Then for any initial state m_0 and any $i \in [0, q], j \in [1, n]$*

$$\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}^{i,j-1}, \mathcal{A}, q, C, m_0, k) \approx_{\tau_D, \epsilon_{2\Pi}} \mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}^{i,j}, \mathcal{A}, q, C, m_0, k)\,. \tag{12}$$

Proof. We prove this claim by contradiction. Suppose there exists an adversary $\mathcal{A}$, a distinguisher D, a state m_0, and values $i \in [0, q], j \in [1, n]$ such that (12) does not hold, then we build a 2-adaptive $(\mathcal{L}_{2\Pi}, \tau_{2\Pi})$-adversary $\mathcal{A}_\Pi$ that distinguishes an encoding of the jth element of $\widehat{m}_i$ from a random encoding. Such $\mathcal{A}_\Pi$ will simulate the environment for $\mathcal{A}$, placing its target encoding as the jth encoding in the ith state. Notice that $\mathcal{A}_\Pi$ can observe its target encoding twice. This enables the simulator to answer to all of $\mathcal{A}$'s queries (x_ℓ, f_ℓ) in a consistent way.

We distinguish three cases to answer the query (f_ℓ, x_ℓ), $\ell \in [1, q]$, depending on the value of i:

1. *The ℓth observation does not contain the ith state:* For such a query $\mathcal{A}_\Pi$ knows the secret inputs and outputs (which either is the ith real state or a random encoding) and can compute the answer correctly with the appropriate reconstructor. This simulation is identical to the simulation of $\mathcal{S}^{i,j-1}$ (which is identical to $\mathcal{S}^{i,j}$ for such queries).

2. *The $\ell = (i+1)$th observation accesses the ith state as part of the input memory:* $\mathcal{A}_\Pi$ puts its target encoding at the jth position of the ith state and uses the reconstructor (together with other hard-wired inputs) to compute a wire assignment for $\widehat{C}^*$. If the target encoding encodes the element of the real state then the simulation is identical to $\mathcal{S}^{i,j-1}$. On the other hand, if it is an encoding of a random value, then the simulation is identical to $\mathcal{S}^{i,j}$. The difficulty is that $\mathcal{A}_\Pi$ has to come up with a wire assignment for $\widehat{C}^*$ that is consistent with the target encoding. Since the target encoding is only known to the leakage function, this has to be done inside the leakage function. Hence, as part of the leakage function, we run the appropriate reconstructor $\mathsf{REC}_{\widehat{g}}$ for the gadget that has the target value as input.

3. *The $\ell = i$th observation accesses the ith state as part of the output:* The analysis is similar to step 2, except that inside the leakage function, we need to use the reconstructor for the gadget that outputs the target encoding.

A crucial point in the above simulation is that a consistent simulation requires $\mathcal{A}_\Pi$ to query its target oracle twice: once when the ith state is an input to an evaluation of $\widehat{C}^*$ (i.e., in the $(i+1)$th round), and a second time when it is part of the output (i.e., in the ith round). This is the reason why we need to rely on a 2-adaptive leakage indistinguishable encoding scheme. For the details we refer the reader to Claim 2 in the proof of Lemma 13. □

Applying Claim 9 repeatedly, we obtain

$$\mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}^{0,0}, \mathcal{A}, q, C, m_0, k) \approx_{\tau_D, (q+1)n\epsilon_{2\Pi}} \mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}^{q,n}, \mathcal{A}, q, C, m_0, k)\,. \tag{13}$$

Note that Equations 10, 11, and 13 hold as long as $\tau_{\mathcal{A}} + \tau_D \leq \min(\tau_\Pi, \tau_{2\Pi}, \tau_{\widehat{g}}) - qs\tau_{\mathsf{samp}}$. Combining them and recalling that the simulation given by $\mathcal{S}_1^{q \cdot s}$ is identical to the simulation given by $\mathcal{S}_2^0$, and the simulation given by $\mathcal{S}_2^{q \cdot m}$ is identical to the simulation of $\mathcal{S}^{0,0}$, we get

$$\mathsf{Exp}^{\mathrm{real}}_{\mathsf{TR}}(\mathcal{A}, \mathcal{L}, q, C, m_0, k) \approx_{\tau_D, qs\epsilon_{\widehat{g}} + qm\epsilon_\Pi + (q+1)n\epsilon_{2\Pi}} \mathsf{Exp}^{\mathrm{sim}}_{\mathsf{TR}}(\mathcal{S}, q, C, m_0, k)\,.$$

□

E Proofs Omitted from Section 5

Proof of Lemma 16. Assume for contradiction that D and E are not 2-adaptive $(\mathcal{L}', 2^{-\lambda}\tau, \epsilon)$-leakage-indistinguishable, then there exists a 2-adaptive $(\mathcal{L}', 2^{-\lambda}\tau)$-adversary $\mathcal{A}'$ that breaks the leakage-indistinguishability of C and D with functions from $\mathcal{L}'$. We will build a $(\mathcal{L}, \tau)$-adversary $\mathcal{A}$ such that

$$|\Pr[\mathcal{A}^{\mathsf{Eval}(D,\cdot)} = 1] - \Pr[\mathcal{A}^{\mathsf{Eval}(E,\cdot)} = 1]| > \epsilon.$$

$\mathcal{A}$ runs $\mathcal{A}'$ as a subroutine and has to adaptively answer its 2 leakage queries f_1, f_2, while having only a single query access to its target oracle Eval (i.e., with the function $f \in \mathcal{L}$). We will resolve this by letting f simulate the adaptivity, and outputting the results of both leakage queries f_1 and f_2. This will increase the size of the function f exponentially in λ.

$\mathcal{A}$ runs in two phases. A learning phase, where it is supposed to learn all possible leakage functions that $\mathcal{A}'$ may pick for the second leakage query. Then, a leakage phase, where it builds a leakage function, obtains valid leakage from Eval with just a single query, and finally returns the reply to $\mathcal{A}'$.

The learning phase is pretty simple: $\mathcal{A}$ runs $\mathcal{A}'$ as a subroutine and gets back f_1. Since $\mathcal{A}$ is only allowed to query Eval once, it cannot query Eval with f_1 directly. Instead, it needs to figure out f_2 that $\mathcal{A}'$ will use as its second query for every possible return value $\Lambda \in \{0,1\}^\lambda$ of f_1. To do so, it rewinds $\mathcal{A}'$ 2^λ times, each time giving a different Λ to $\mathcal{A}'$ to obtain the function f_2^Λ. (Observe that some values of Λ may be invalid for the leakage function f_1. This might give $\mathcal{A}'$ indication that she is run in a simulated environment; in that case, $\mathcal{A}'$ may run forever, but $\mathcal{A}$ will stop her after $2^{-\lambda}\tau$ steps.)

Let us now describe the leakage phase. $\mathcal{A}$ will build its leakage function f as follows: on input S, f computes $\Lambda_1 = f_1(S)$, $f_2^{\Lambda_1}(S)$, and outputs both values.

The rest of the proof is straightforward: $\mathcal{A}$ uses its return from the oracle Eval to answer the two leakage queries f_1, f_2 of $\mathcal{A}'$. Since this is a perfect simulation, we get that if $\mathcal{A}'$ can distinguish with advantage more than ϵ, then so can $\mathcal{A}$. Notice that the running time of $\mathcal{A}$ is $2^{-\lambda}\tau 2^\lambda \approx \tau$.

We need to compute the circuit complexity of f. All 2^λ possible functions of f_2 need to be hardwired into the circuit, but they can be computed in parallel with f_1 (so they increase the size, but not the depth of the circuit). Then, the output of one of these functions needs to be "selected" according to the output of f_1. This selection can be done by increasing the depth by 2 (not counting NOT gates) and size $O(2^\lambda)$ (cf. Figure 9 for the case when $\lambda = 1$). Thus, we get $\mathcal{L}' = \mathcal{C}(d-2, O(s2^{-\lambda}), \lfloor \lambda/2 \rfloor)$ as stated in the lemma.

□

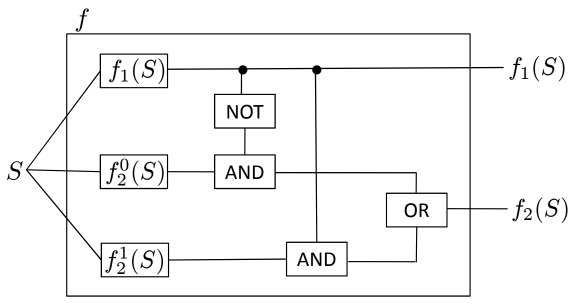

Figure 9: The structure of f when $\mathcal{L} = 1$