Zero-Knowledge PCPs from Leakage-Resilient Circuits, Revisited

Mor Weiss*

May 24, 2019

Abstract

This work studies several enhancements and extensions of known leakage-resilient circuits, motivated by their possible applications to zero-knowledge proof systems.

Leakage-Resilient (LR) circuits are circuits that resist certain "side-channel" attacks, in the sense that such attacks reveal nothing about the (properly encoded) input, other than the output. Probabilistically Checkable Proofs (PCPs) allow a probabilistic verifier, given oracle access to a purported proof, to verify claims of the form "$x \in L$" for an NP-language L, while probing only few proof bits. Zero-Knowledge (ZK) PCPs have the added feature that any verifier querying few proof bits learns nothing about the underlying NP-witness.

Recently, Ishai, Weiss and Yang (TCC 2016-B) showed a surprising connection between these seemingly unrelated notions: they used LR circuits to construct PCPs with ZK guarantees and a *non-adaptive honest verifier* (previous ZK-PCP constructions required adaptive verification). Their technique required *sound* LR circuits which, roughly, are LR circuits that detect and reject ill-formed encoded inputs. They construct sound LR circuits based on the LR circuits of Faust, Rabin, Reyzin, Tromer and Vaikuntanathan (Eurocrypt 2010). Though improving certain properties of the PCP, their construction only obtained *Witness-Indistinguishability* (WI), i.e., simulation with an *inefficient* simulator, and not full-fledged zero-knowledge, which they show to be inherent to a construction based on the LR circuits of Faust et al., unless $\mathsf{NP} = \mathsf{BPP}$.

The initial goal of this work was to use the technique of Ishai et al. to construct ZK-PCPs with efficient simulation, which necessitates replacing the LR circuits of Faust et al. in the transformation. Towards that end, we generalize and improve several other LR circuits, though ultimately failing to construct LR circuits that provably satisfy all needed properties. First, we present a candidate construction based on the probing-resilient circuits of Ishai, Sahai and Wagner (CRYPTO 2003) which we conjecture resists the type of leakage that is useful in constructing ZK-PCPs. Second, we describe a variant of the probing-resilient circuits of Andrychowicz, Dziembowski and Faust (Eurocrypto 2016) which guarantees soundness. Third, we consider the LR circuits of Goyal, Ishai, Maji, Sahai and Sherstov (FOCS 2016) which resists so-called "only computation leaks" leakage, and show inherent limitations towards extending it to resist the type of leakage that is useful towards constructing ZK-PCPs. Finally, we consider the LR circuits of Miles and Viola (STOC 2013) which resist NC^1-leakage, and describe a first step towards making them sound. These variants and insights may be of interest independently of their applications towards constructing ZK-PCPs.

We also consider generalizations of the WI-PCP of Ishai, Weiss and Yang to *PCPs of proximity* (PCPPs), which are PCPs that allow the verifier to check statements of the form "x is *close* to L" while querying only *few bits of* x (and the proof). Specifically, we reduce the problem of constructing WI-PCPs of proximity to a natural coding question.

*Efi Arazi School of Computer Science, IDC Herzliya, Herzliya, Israel. mor.weiss01@post.idc.ac.il.
Work done in part while the author was a PhD student at the Technion and a postdoctoral researcher at Northeastern University.

Contents

1 Introduction

Probabilistically Checkable Proofs. Probabilistically Checkable Proofs (PCPs) [ALM+92, AS92] are proofs systems that allow a Probabilistic Polynomial Time (PPT) verifier, with oracle access to a purported proof, to check statements of the form $x \in L$ for some NP-language L by reading few proof bits. We think of the proof π as generated by a PPT prover that is given the corresponding NP witness w. If $x \in L$ then the verifier accepts an honestly-generated proof π with probability 1, whereas if $x \notin L$ then *any* purported proof π^* is accepted with low probability (the probability that the verifier accepts $x \notin L$ is called the soundness error). The celebrated PCP theorem [AS92, ALM+92, Din06] asserts that any NP language has a PCP system with soundness error 1/2 and a verifier that *non-adaptively* reads a constant number of proofs bits. (A verifier is non-adaptive if his queries are determined solely by his randomness, regardless of the oracle answers to previous queries.)

Zero-Knowledge PCPs. A seemingly unrelated notion is that of Zero-Knowledge (ZK) Proofs [GMR85] which are proofs that carry no extra knowledge other than being convincing. The first to combine the notions of PCP and ZK were Kilian, Petrank and Tardos [KPT97] who defined and constructed Zero-Knowledge PCPs (ZK-PCPs). ZK-PCPs are PCPs with the additional guarantee that the view of any (possibly malicious) verifier that queries a bounded number of proof bits can be efficiently simulated, where the real and simulated views are statistically close. We note that in a ZK-PCP the proof is randomized. Kilian et al. [KPT97] transform standard PCPs for NP into PCPs with statistical ZK against malicious verifiers that are limited to querying only $p(|x|)$ proof bits, for some a-priori fixed polynomial p which is much smaller than the proof length, but can be much larger than the number of queries the honest verifier makes (which is usually $\text{poly}\log(|x|)$).

Adding the ZK guarantee to PCPs did not come without a price. Specifically, the transformation of [KPT97] caused both a polynomial blowup in the proof length, and required adaptive verification, namely even the *honest* verifier is adaptive. Both of these limitations are inherent to the construction of [KPT97], as we explain in Section 1.4 below. This raises the following natural questions:

> *What is the cost of ZK in PCP systems? Do there exist ZK-PCPs of quasilinear length that can be verified non-adaptively?*

Two concurrent and independent works tried to answer these questions by employing different techniques than [KPT97]. Ben-Sasson et al. [BCGV16] relied on algebraic properties of specific PCPs (e.g., the PCPs of [BFLS91, ALM+92, BS08]) to modify the PCP, introducing minor modifications of the prover and verifier algorithms, such that the resultant proof also guarantees ZK. They achieve perfect ZK, i.e., the real and simulated views are identically distributed, with proofs of quasilinear length. However, technically their construction is *not* a PCP since the prover is adaptive, in the sense that it sends the PCP to the verifier in two rounds: after receiving the first-round PCP, the verifier sends a message to the prover, which the prover uses to construct the second-round PCP. This notion of a PCP with adaptive prover was later generalized to Interactive Oracle Proofs (IOPs) [BCS16, RRR16] (also known as Probabilistically Checkable Interactive Proofs) which have seen applications to ZK proofs in different models, including implementations (see, e.g., [BCS16, BCF+17, BBC+17, BBHR18b, BBHR18a] and the references therein).

Ishai et al. [IWY16] focused on obtaining non-adaptive verification in the standard PCP model. They use Leakage-Resilient Circuit Compilers (LRCCs) [ISW03] to construct a PCP with a non-adaptive honest verifier which guarantees ZK with an *unbounded simulator* (namely, they achieve

witness indistinguishability), and proofs of polynomial length. They also obtain standard ZK (with an efficient simulator) in the Common-Random String (CRS) model.

Though introducing new techniques towards the design of ZK-PCPs, the works of [BCGV16, IWY16] fall short of answering the above questions since they do not simultaneously achieve ZK with a non-adaptive verifier in the standard PCP setting. In this work, we consider several directions towards extending and improving the result of [IWY16]. These directions focus on improving the underlying LRCCs which, as explained in Section 1.2 below, are the cause for inefficient simulation. For each direction we either explain why it fails, or point to related open problems which remain to be proven. We first turn our attention to leakage-resilient circuits.

Leakage-Resilient Circuit Compilers (LRCCs). Informally, an LRCC compiles a circuit into a new circuit that takes encoded inputs, and resists side-channel attacks in the sense that these reveal nothing about the (properly encoded) input, other than the output. Since one cannot protect against arbitrary polynomial-time computable leakage [BGI+01], works on LRCCs consider restricted classes of leakage functions. One line of work [ISW03, FRR+10, Rot12, MV13, Mil14, ADF16] restricts the complexity class to which leakage functions belong. Another approach, called "Only Computation Leaks" (OCL) (see, e.g., [MR04, GR10, JV10, GR12, DF12, GIM+16]) assumes leakage functions are "local" in the sense that they operate on disjoint sets of wires of the circuit.

More formally, an LRCC is associated with a class $\mathcal{L}$ of leakage functions, and a randomized encoding scheme E which is used to encode the inputs. The LRCC compiles a given circuit C into a circuit $\widehat{C}$ that emulates C, but operates on encodings generated by E. $\widehat{C}$ is leakage resilient in the sense that for every input x for C, and any leakage function $\ell \in \mathcal{L}$ which operates on the wire values of $\widehat{C}$, the output of ℓ when $\widehat{C}$ is evaluated on a random encoding $\mathsf{E}(x)$ of x can be simulated given only the output $C(x)$.

The construction of ZK-PCPs from LRCCs requires that the underlying LRCC have an additional soundness guarantee which [IWY16] introduce and term *SAT-respecting*. Roughly, an LRCC is SAT-respecting if for any circuit C, if the leakage-resilient circuit $\widehat{C}$ is satisfiable then so is the original circuit C.

1.1 Our Contributions

We explore several avenues towards improving existing LRCCs and the WI-PCP construction of [IWY16]:

- In terms of improving the leakage resilience guarantees of LRCCs, we provide a candidate construction of a SAT-respecting LRCC, based on the probing-secure LRCC of [ISW03], which we conjecture to resists $\mathsf{AC}^0[\oplus]$ leakage, namely leakage computable by constant-depth boolean circuits with AND, OR, NOT, and XOR gates of unbounded fan-in and fan-out. We note that $\mathsf{AC}^0[\oplus]$-leakage is sufficiently strong to allow us to employ the LRCC-to-PCP transformation of [IWY16]. We also point to obstacles in extending the leakage-resilience guarantee of the (SAT-respecting) LRCC of [GIM+16] to $\mathsf{AC}^0[\oplus]$ leakage.

- In terms of adding a soundness guarantee to LRCCs, we describe a SAT-respecting variant of the probing-secure LRCC of [ADF16], as well as a first step towards making the LRCC of [MV13], that resists leakage against a large class of leakage functions (including $\mathsf{AC}^0[\oplus]$), SAT-respecting.

- In terms of improving the transformation of [IWY16], we extend it to apply to PCPs of proximity, and reduce the security of the resultant WI-PCP of proximity to a natural coding question.

As noted above, and explained in detail below, improving the leakage-resilience and SAT-respecting guarantees of existing LRCCs could also potentially improve the transformation of [IWY16].

In the following sections we provide more details about these contributions.

1.2 ZK-PCPs from Leakage-Resilience

Before discussing several (attempts to obtain) improvements to the Witness-Indistinguishable (WI) PCP of [IWY16], we first explain in detail their approach towards constructing ZK-PCPs from LRCCs.

The construction of [IWY16]. The main observation of [IWY16] is that one can think of the PCP bits queried by a (possibly malicious) verifier V^* as *leakage* on the NP-witness. Indeed, the PCP is obtained by applying the honest prover algorithm P to the NP-witness w, and the queried PCP bits are the restriction of the prover's output to few bits. Namely, any subset I of queried bits defines a leakage function ℓ_I which on input w applies the prover algorithm P to w to obtain a proof π, and outputs π_I. Ishai et al. [IWY16] observe that if the original PCP is resilient against such leakage functions, then the PCP system is ZK.

In general, PCPs are not resilient against such leakage, but they can be made resilient using an LRCC as follows. Consider the verification circuit C of an NP relation $\mathcal{R}_L$, and assume without loss of generality that the NP-witness w is the entire wire values of C when evaluated on x and a witness y for the membership of x in L. To prove that $x \in L$, the prover P should convince the verifier V that C_x (i.e., C with x hard-wired into it) is satisfiable. To do so in zero-knowledge, both P, V replace C_x with its leakage-resilient version $\widehat{C_x}$, and P proves to V that $\widehat{C_x}$ is satisfiable by constructing a PCP π from the wire values of $\widehat{C_x}$.

This transformation crucially relies on the fact that $\widehat{C_x}$ emulates C_x (e.g., if $\widehat{C_x}$ always outputs 1 then the resultant PCP system is not sound). Standard LRCCs (e.g., [FRR+10, DF12]) only guarantee this when the encoded input to $\widehat{C_x}$ is honestly generated, so a malicious prover P^* can cheat by generating the proof from the wire values of $\widehat{C_x}$ on *ill-formed* inputs. More specifically, the leakage-resilient circuits in these constructions expect to obtain, as part of an input encoding, structured randomness (so-called "masks" or "opaque gates") which is then used to mask intermediate computations to achieve leakage resilience. The leakage-resilience simulator uses ill-formed randomness to simulate the wire values. The main technical contribution of [IWY16] is the construction of an LRCC with an additional soundness guarantee which they call *SAT-respecting*. Recall that an LRCC is SAT-respecting if for any circuit C, if the leakage-resilient circuit $\widehat{C}$ is satisfiable then so is the original circuit C. Their construction, which is based on the LRCC of Faust et al. [FRR+10] is leakage-resilient with an *inefficient* simulator, which is the reason their resultant PCP only guarantees WI and not ZK.

Ishai et al. [IWY16] observe that inefficient simulation is in certain cases inherent to the leakage-resilient based technique, in the sense that efficient simulation is possible only for languages in BPP. More specifically, the underlying LRCC of [FRR+10] has a strong leakage-resilience guarantee: it has a universal simulator which, roughly, simulates the entire wire values of the circuit *without knowing the leakage function*, and these simulated wires are *simultaneously "good"* for all leakage functions (i.e., leakage functions cannot distinguish between the simulated and actual wire values). They use this universal simulation property, together with the SAT-respecting guarantee of the leakage-resilient circuit, to decide the language.

1.2.1 Towards ZK-PCPs with efficient simulation: Going beyond [IWY16]

Given the aforementioned observation of [IWY16], it is clear that constructing ZK-PCPs with efficient simulation from LRCCs requires using an LRCC in which the simulator is *not universal*. We focus on three such general compilers: the compiler of Ishai, Sahai and Wagner [ISW03], the compiler of Andrychowicz, Dziembowski, and Faust [ADF16], and the compiler of Goyal et al. [GIM+16]. All these compilers follow the same paradigm of replacing wires in the circuit with *bundles* of wires that carry encodings of the wire values of the original circuit, and replacing gates with *gadgets* which are sub-circuits that emulate the gate operation over bundles (see Section 2.2 for a more detailed description of gadget-based LRCCs). We discuss each of these in turn.

In the following, we say a class F of functions "contains a PCP" if the PCP prover algorithm, when its output is restricted to any "small" subset of bits, computes a function in the class F. We rely on the following observation from [IWY16]: the class $\mathsf{AC}^0[\oplus]$ of constant-depth boolean circuits with AND, OR, NOT, and XOR gates of unbounded fan-in and fan-out contains the PCP of Arora and Safra [AS92]. In fact, the AS-PCP only uses a small (poly-logarithmic in the input length) number of XOR gates, so in the following we use $\mathsf{AC}^0[\oplus]$ to denote circuits as described above with *few* XOR gates.

The ISW LRCC [ISW03]. The ISW LRCC transforms boolean circuits into leakage-resilient counterparts, by encoding each bit b with a bit-string whose parity is b, and using gadgets that emulate AND and NOT gates. An important property of the compiled circuit is that it does not use any structured randomness, and is consequently SAT-respecting. (In more detail, any input encoding $\widehat{x}$ for the leakage-resilient circuit $\widehat{C}$ corresponds to *some* input x for the original circuit C, and in particular if $\widehat{x}$ satisfies $\widehat{C}$ then x satisfies C.) However, the ISW-LRCC only resist probing attacks that output a small subset of the wire values, and this function class does not contain a PCP. (Indeed, the local verification property of the PCP requires the prover to perform some computation over the NP-witness.) The ISW-LRCC is conjectured to resist AC^0 leakage, i.e., leakage computable by constant-depth boolean circuits with AND, OR, and NOT gates of unbounded fan-in and fan-out. However, proving this seems to be a hard problem (where later works that protect against wider leakage classes, e.g. [FRR+10, Rot12], provide different constructions instead of extending the security proof of the ISW-LRCC), and currently there isn't even a candidate simulator for proving this conjecture.

Recall that when constructing a ZK-PCP from an LRCC, the LRCC should resist leakage from a function class that contains the PCP. The PCP of [AS92] is *not* contained in AC^0 - the use of XOR gates is inherent to the construction.[1]. It is easy to see that the ISW-LRCC is *insecure* against such leakage, because its intermediate computations are encoded using parity encoding which $\mathsf{AC}^0[\oplus]$ circuits can decode. For example, consider a circuit $C(x_1, x_2) = x_1 \wedge x_2$ consisting of a single AND gate. Then $\mathsf{AC}^0[\oplus]$-leakage-resilience requires the existence of an efficient simulator that can simulate the leakage given only the output of the circuit, and the simulated leakage should be indistinguishable from the actual leakage. Since $C(0,0) = C(0,1) = 0$, this implies that applying an $\mathsf{AC}^0[\oplus]$-leakage function ℓ to the wire values in a computation on inputs $(0,0)$ and $(0,1)$ should yield indistinguishable distributions. However, the leakage function ℓ that decodes the second input gives efficiently distinguishable distributions (whose distance is 1).

A possible approach towards obtaining $\mathsf{AC}^0[\oplus]$-leakage-resilience is to modify the ISW-LRCC to operate over $\mathbb{F}_3$ instead of $\mathbb{F}_2$. The high-level idea is conceptually simple: replace the parity encoding with mod-3 encoding over $\mathbb{F}_3$ which encodes every element $z \in \mathbb{F}_3$ by a vector over

[1] In fact, this is the case in most PCPs in the literature, which use a sum-check protocol that requires performing additions over a field.

$\mathbb{F}_3$ that sums to z, and generalize the AND and NOT gadgets to operate over mod-3 encodings. However, formalizing this intuition requires some work since the ISW-LRCC was designed to work specifically over boolean circuits, whereas here we are using arithmetic circuits to emulate boolean computations. In particular, it is not even clear what "AND gates" and "NOT gates" mean when operating over $\mathbb{F}_3$. However, since we are only interested in emulating boolean computations, we can meaningfully define these operations to emulate the desired functionality, such that when given inputs in $\{0,1\}$, the operation is consistent with standard AND and NOT. Additionally, for the SAT-respecting property we need to ensure that the inputs to the leakage-resilient circuit encode bits. This can be achieved by checking, in a leakage-resilient manner, that each input symbol is a zero of the polynomial $x \cdot (x-1)$, and if not then force the output to be 0. (A similar idea was used in [IWY16].) We describe this construction in Section 3, and show that is resists probing attacks (as the original ISW-LRCC). We note that similar to the original ISW-LRCC, we do not currently have any candidate simulator for proving $\mathsf{AC}^0[\oplus]$-security of this modified construction.

The ADF-LRCC [ADF16]. Similar to [ISW03], the ADF-LRCC protects against probing attacks. The construction is more general than [ISW03], where the compiler can operate over any field, and using any "multiplication-friendly" encoding[2] such as Shamir secret sharing [Sha79] or Algebraic-Geometry (AG) codes [CC06]. Though both [ISW03, ADF16] only protect against probing attacks, a class which doesn't contain a PCP, the ADF-LRCC has the advantage of generality: since it can be instantiated using various different encodings over different fields, it may be more easily modified to achieve stronger leakage-resilience guarantees.[3] In particular, an implementation over extension fields of $\mathbb{F}_3$ with Shamir secret sharing might resist $\mathsf{AC}^0[\oplus]$ leakage, though we currently do not have a candidate simulator towards proving this.

One disadvantage of the ADF-LRCC is that it uses structured randomness, where correctness of the computation relies on this randomness having the "right" structure. As noted above, most LRCCs (e.g. [FRR+10, DF12]) rely on structure randomness, but whereas the amount of structured randomness used in these construction scales with the computation size, the amount of structured randomness used in the ADF-LRCC scales only with the input size, and is independent of the computation size. This gives the hope of being able to verify, inside the compiled circuit, that the randomness has the desired structure, without harming leakage-resilience. In section 4, we describe a variant of the ADF-LRCC that includes this additional check, which gives a probing-attach secure SAT-respecting LRCC over general fields (with improved leakage rate).

The GIMSS-LRCC [GIM+16]. The LRCC of [GIM+16] uses MPC protocols to resist leakage from a broad function class that includes so-called "Only Computation Leaks" (OCL) leakage, namely leakage which is "local" in the sense that leakage functions operate on disjoint sets of wires of the circuit. More specifically, [GIM+16] protect against an extended version of the OCL model of Micali and Reyzin [MR04], also known as "OCL+" [BCG+11], which [GIM+16] call *Bounded Communication Leakage* (BCL). Informally, in this context, the wires of the leakage-resilient circuit $\widehat{C}$ are partitioned into a "left component" $\widehat{C_L}$ and a "right component" $\widehat{C_R}$. Leakage functions correspond to bounded-communication 2-party protocols between $\widehat{C_L}, \widehat{C_R}$, where the output of the

[2] Roughly, an encoding scheme is "multiplication friendly" if the product of two encodings can be computed as a linear combination of the component-wise products. That is, given two encodings $\widehat{a} = (a_1, \ldots, a_n)$ and $\widehat{b} = (b_1, \ldots, b_n)$ of values a, b (respectively), the product $c = ab$ can be obtained as a linear combination of the products $a_1 b_1, \ldots, a_n b_n$.

[3] We note that the ADF-LRCC also achieves a better leakage rate (i.e., the blowup in size of the compiled circuit is smaller). In fact, this was the goal of [ADF16], however it is of lesser importance for us, though we note that a better leakage rate might help in reducing the proof length in the resultant ZK-PCP.

leakage function is the transcript of the protocol when the views of $\widehat{C_L}, \widehat{C_R}$ consist of the internal values of the wires of these two "components".

BCL leakage is quite powerful. In particular, it is broad enough to capture several realistic leakage attacks such as the sum of all circuit wires over the integers, as well as linear functions over the wires of the circuit. Unfortunately, Abboud, Rubinstein, and Williams [ARW17] recently proved that there are no PCPs (with non-trivial query complexity) in this class. In fact, we show that in terms of low-depth computation, the GIMSS-LRCC is *as insecure* as the ISW-LRCC (which it uses as a building block), in the sense that any successful $\mathsf{AC}^0[\oplus]$ leakage attack on the ISW-LRCC, such as the one described above, gives rise to a related $\mathsf{AC}^0[\oplus]$ leakage attack on the GIMSS-LRCC. A direct (informal) consequence is that generalizing the LRCC-GIMSS to resist $\mathsf{AC}^0[\oplus]$ leakage is as hard as generalizing the ISW-LRCC to resist such leakage. We discuss this in more detail in Section 5. We note that though OCL and $\mathsf{AC}^0[\oplus]$ are incomparable leakage classes, they both contain leakage functions that compute sums of wires of the circuit.

1.2.2 Towards Going Beyond $\mathsf{AC}^0[\oplus]$-Leakage

Even if one settles for inefficient simulation, there are still possible directions towards improving and extending the SAT-respecting LRCC and the WI-PCP of [IWY16]. One such direction is to strengthen the IWY-LRCC to resist leakage from wider classes than $\mathsf{AC}^0[\oplus]$, which is motivated both from a leakage-resilience prospective of protecting against wider leakage classes, as well as from the ZK-PCP prospective. Indeed, in the context of constructing ZK-PCPs, resisting a wider class of leakage functions translates to the ability to apply the transformation to a wider class of PCPs. In particular, we might be able to apply the transformation to more efficient PCPs, and thus improve the proof length of the resultant ZK-PCP.

Towards that end, we consider the LRCC of Miles and Viola [MV13]. The MV-LRCC follows the standard paradigm of replacing wires and gates with bundles and gadgets, but diverges from previous construction in using circuits that operate over *non-commutative groups* (instead of arithmetic circuits over a field). This allows them to protect against large leakage classes. Specifically, their LRCC withstands leakage from any function class against which average-case lower bounds are known, including $\mathsf{AC}^0[\oplus]$ circuits (when they have few XOR gates). Moreover, assuming $\mathsf{L} \neq \mathsf{NC}^1$, their LRCC resists NC^1-leakage [Mil14]. We note that similar to [FRR+10], the MV-LRCC has a universal simulator, so using it in the transformation of [IWY16] will only give WI-PCPs (and not ZK-PCPs).

The MV-LRCC is not SAT-respecting, due to two reasons. First, it uses computations over a group to emulate boolean computations, so one must ensure that the inputs to the leakage-resilient circuit encode bits. Second, it relies on structured randomness to achieve leakage-resilience. To address the first issue, one needs to design some leakage-resilient test to check that the inputs correspond to bits. As noted above, in a field this can be done by checking that the inputs are zeros of the polynomial $x \cdot (x-1)$. Therefore, we need to extend this check to a non-commutative group. In Section 6 we design a gadget that performs this test while preserving leakage-resilience.

Addressing the second issue is more involved, and we currently do not know how to solve it. We review the obstacles in trying to extend the ideas of [IWY16] to non-commutative rings. Recall that the FRRTV-LRCC [FRR+10] is not SAT-respecting because it employs structured randomness to mask intermediate computations. Ishai et al. [IWY16] describe a variant of the FRRTV-LRCC that *is* SAT-respecting, in which the leakage-resilient circuit $\widehat{C}$ includes an additional component checking that the randomness has the "right" structure. To preserve leakage-resilience (with an inefficient simulator), the final construction is more involved. For a circuit C, let $\widehat{C}^F$ denote the leakage-resilient circuit obtained from C by applying the FRRTV-LRCC. The leakage-resilient

circuit $\widehat{C}$ of [IWY16] contains two copies of $\widehat{C}^F$, as well as a component C^R verifying that *at least one* of the copies used "well-formed" randomness (i.e., randomness with the correct structure). An important property of C^R which is *crucial* for the leakage-resilience guarantee is that it hides *which* of the copies used well-formed randomness.[4] To obtain these properties, Ishai et al. [IWY16] crucially rely on the fact that the FRRTV-LRCC operates over a field. In particular, a field is both commutative and has a 0. Thus, the outcome 0 of a product $a \times b = 0$ erases all information about which of a, b was 0. Both properties are not satisfied by non-commutative rings, and we therefore do not know how to design a similar test for the MV-LRCC.

1.3 Towards WI-PCPs of Proximity from LRCCs

We generalize the LRCC-to-PCP construction of [IWY16] to apply also to PCPs of Proximity (PCPPs), and reduce the security of the construction to the task of designing a code with "good" distance that resists $\mathsf{AC}^0[\oplus]$ leakage.

Informally, PCPPs generalize PCPs by allowing the verifier, which has oracle access to his input x and a purported proof, to check that $x \in L$ for an NP-language L by querying only few bits *of the input* x (in addition to querying few proof bits). If x is accepted then the verifier is guaranteed that with high probability x is *close* (in relative hamming distance) to some $x' \in L$. ZK-PCPPs have the added feature that the view of any verifier querying a bounded number of input or proof bits can be efficiently simulated by making the same number of queries *to the input oracle* x *alone*. We say that a PCPP system is WI if this holds with an *inefficient* simulator.

Generalizing the transformation of [IWY16] to work for PCPPs turns out to be non-trivial, and requires overcoming several obstacles, as we now describe. Recall that in the transformation, the prover proves to the verifier that $\widehat{C_x}$ is satisfiable, where $\widehat{C_x}$ is the SAT-respecting and leakage-resilient version of the verification circuit C of the NP-relation, with the input x hard-wired into it.

The first obstacle we face in generalizing this construction to work for PCPPs is that the PCPP verifier does not know x, so we cannot hard-wire x into C. The natural solution would be to apply the LRCC to C (without hard-wiring x), and use x as part of the witness proving that the leakage-resilient $\widehat{C}$ is satisfiable. Clearly, this would not work since the input and witness have significantly different roles: the verifier is only interested in verifying the *existence of some* satisfying witness, whereas the verifier needs to verify its *specific* input oracle satisfies the circuit. Thus, the verifier must check consistency of the encoded input $\widehat{x}$ (used in $\widehat{C}$) with its own input oracle x.

Therefore, we turn our attention to the question of verifying consistency of an encoding $\widehat{x}$ with a given x, for which it suffices to show how to check that $\widehat{b}$ encodes a given bit b. (Indeed, given such a method, longer strings can be checked by checking a random subset of their bits; we note that the encoding used in [IWY16] anyway encodes long messages by encoding every bit separately.) Notice that the statement "$\widehat{b}$ encodes b" is a statement in P (i.e., in deterministic polynomial time), and verifying it is made non-trivial by the fact that it should be done with only few queries to $\widehat{b}$. This is because the query complexity, i.e., the number of bits the verifier reads from the proof, is an important measure of a PCPP system which we would like to minimize. To allow the verifier to check such claims with few queries, we give it an additional PCPP that attests to the fact that

[4] Indeed, given the output y of C, the unbounded leakage-resilience simulator finds an arbitrary input x' such that $C(x') = y$, honestly emulates $\widehat{C}$ on an encoding of x', and uses this to generate the leakage. The proof uses a hybrid argument in which the real-world input of the copies of $\widehat{C}^F$ are replaces with encodings of x' one copy at a time. Replacing the input to a copy of $\widehat{C}^F$ requires using ill-formed randomness in this copy. Indistinguishability of adjacent hybrids relies on the fact that C^R hides which (and how many) of the copies use well-formed randomness. We refer the interested reader to [IWY16] for additional details.

$\widehat{b}$ encodes b. It is important to note that this PCPP *need not provide any ZK guarantee.* Notice that since the verifier doesn't read its entire input $\left(\widehat{b}, b\right)$, the best one can hope for is that if the verifier accepts then $\widehat{b}$ is *close* to an encoding of b. Consequently, the underlying encoding scheme must have good distance. Otherwise, by flipping few bits one can change an encoding of 0 to an encoding of 1, and so the fact that $\widehat{b}$ is close to an encoding of b doesn't rule out the possibility that it actually encodes $1-b$, i.e., that the input used in $\widehat{C}$ is inconsistent with the actual input x of the verifier.

However, the LRCC of [IWY16] cannot use an encoding scheme with good (in fact, any even non-trivial) distance, because the SAT-respecting property crucially relies on the fact that any input to the leakage-resilient circuit is a valid encoding of *some* input for the original circuit. Thus, we modify the LRCC of [IWY16] to take part of its input in the clear, and guarantee an "average-case" leakage-resilience property when this part of the input is encoded using a leakage-resilient encoding. Crucially, this average-case leakage-resilience property *does not* necessitate encoding the input using the same encoding scheme which the LRCC uses to perform the internal computations. We then show that this average-case leakage-resilience property, though weaker than standard leakage-resilience, is nonetheless sufficient to obtain WI-PCPPs.

We emphasize that obtaining soundness and WI simultaneously in our PCPP system requires encoding the inputs to the leakage-resilient circuit using an encoding scheme with "good" distance that resists $\mathsf{AC}^0[\oplus]$ leakage. We do not currently know how to construct such a code, or even whether such a code exists.

We provide a more detailed description, including the full transformation, in Section 7.

1.4 On the Limits of the ZK-PCP of [KPT97]

We now explain why adaptive verification, and a polynomial blowup in the proof length, are inherent to the ZK-PCP construction of [KPT97].

At a very high level, [KPT97] provide a general transformation from a PCP with ZK against the *honest* verifier in which the honest verifier queries a small (polylogarithmic) number of proof bits, to a PCP with full-fledged ZK. The transformation employs an unconditionally secure oracle-based commitment primitive called a "locking scheme". The fact that the underlying PCP has small query complexity (polynomial in the security parameter) is crucial for the construction. As any such PCP necessarily has a large soundness error, they amplify soundness through repetition. However, since the original proof is only secure against a verifier making few queries, each iteration in the amplification should use a different proof, so the proof has to be duplicated, causing a polynomial blowup.

Moreover, to move from ZK against only the honest verifier to full-fledged ZK, the idea is to "mix" the proof symbols, where the honest verifier's queries are "translated" to a different (random) set of queries of the honest verifier, and the locations of these new queries are then revealed to the verifier. Thus, the verifier must adaptively make queries to the proof: the first set of queries establishes the locations of the queried symbols in the "mixed" proof, and the second set of queries reads those bits. (The adaptivity of the verifier in the construction is actually larger.) Finally, we note that the use of locking schemes in of itself inherently causes a polynomial blowup and adaptive verification, due to the information-theoretic properties of the locking scheme. We note that though quasilinear-length PCPs were not known at the time, it is even now unclear how to leverage the constructions we have today [BGH+04, Din06, BS08, Mie08] to improve the proof length of ZK-PCPs.

2 Preliminaries

Parts of this section were taken almost verbatim from [IWY16, GIW17].

Let $\mathbb{F}$ be a finite field, and Σ be a finite alphabet (i.e., a set of symbols). In the following, function composition is denoted as $f \circ g$, where $(f \circ g)(x) := f(g(x))$. If F, G are families of functions then $F \circ G := \{f \circ g : f \in F, g \in G\}$. For a vector $\vec{x} \in \Sigma^n$, and a subset $I \subseteq [n]$, we use $\vec{x}_I$ to denote the restriction of $\vec{x}$ to the indices in I, i.e., $\vec{x}_I = (x_i)_{i \in I}$. For a randomized algorithm A taking input x, we sometimes make the random choices of A explicit by writing $A(x; r)$ to denote that A has input x and uses random string r. If $\mathcal{D}$ is a distribution then $X \leftarrow \mathcal{D}$, or $X \in_R \mathcal{D}$, denotes sampling X according to the distribution $\mathcal{D}$. Given two distributions X, Y, $\mathsf{SD}(X, Y)$ denotes the statistical distance between X and Y. For a natural n, $\mathsf{negl}(n)$ denotes a function that is negligible in n, and $\mathrm{poly}(n)$ denotes a polynomial in n. For a function family $\mathcal{L}$, we sometimes use the term "leakage family $\mathcal{L}$", or "leakage class $\mathcal{L}$". In the following, n, m denote the input length, t denotes a security parameter, and d, s denote depth and size, respectively (e.g., of circuits, as defined below).

Circuits. We consider boolean circuits $\mathcal{C}$ over the set $X = \{x_1, \cdots, x_n\}$ of variables. $\mathcal{C}$ is a directed acyclic graph whose vertices are called *gates* and whose edges are called *wires*. The wires of C are labeled with functions over X. Every gate in $\mathcal{C}$ of in-degree 0 has out-degree 1 and is either labeled by a variable from X and referred to as an *input gate*; or is labeled by a constant $\alpha \in \{0, 1\}$ and referred to as a const_α *gate*. Following [FRR+10], all other gates are labeled by one of the operations $\wedge, \vee, \neg$, or copy, where $\wedge, \vee$ vertices have fan-in 2 and fan-out 1; copy has fan-in 1 and fan-out 2, where the labels of the outgoing edges carry the same function as the incoming edge; and $\neg$ has fan-in and fan-out 1. We will also consider boolean circuits that additionally have $\oplus$ gates of unbounded fan-in and fan-out. We write $\mathcal{C} : \{0,1\}^n \to \{0,1\}^k$ to indicate that $\mathcal{C}$ is a boolean circuit with n inputs and k outputs. The *size* of a circuit $\mathcal{C}$, denoted $|\mathcal{C}|$, is the number of wires in $\mathcal{C}$, together with input and output gates.

We also consider arithmetic circuits $\mathcal{C}$ over a finite field $\mathbb{F}$ and the set X. Similarly to the boolean case, $\mathcal{C}$ has input and constant gates, and following [FRR+10], all other gates are labeled by one of the following functions $+, -, \times$ or copy, where $+, -, \times$ are the addition, subtraction, and multiplication operations of the field (i.e., the outgoing wire is labeled with the addition, subtraction, or product (respectively) of the labels of the incoming wires), and these vertices have fan-in 2 and fan-out 1; and copy vertices have fan-in 1 and fan-out 2, with the same operation as in the boolean case. We write $\mathcal{C} : \mathbb{F}^n \to \mathbb{F}^k$ to indicate that $\mathcal{C}$ is an arithmetic circuit over $\mathbb{F}$ with n inputs and k outputs. We use $\mathsf{AC}^0[\oplus]$ to denote the class of all boolean circuits of constant depth and polynomial size (in the input length) with $\wedge, \vee$ gates of unbounded fan-in and fan-out, $\neg$ gates of unbounded fan-out, and a *polylogarithmic* number of $\oplus$ gates of unbounded fan-in and fan-out. Somewhat abusing notation, we sometimes use $\mathsf{AC}^0[\oplus]$ to denote the class of *functions* computable by such circuits.

Encoding Schemes. An encoding scheme E over alphabet Σ is a pair $(\mathsf{Enc}, \mathsf{Dec})$ of algorithms, where the *encoding algorithm* Enc is a PPT algorithm that given a message $x \in \Sigma^n$ outputs an encoding $\widehat{x} \in \Sigma^{\widehat{n}}$ for some $\widehat{n} = \widehat{n}(n)$; and the *decoding algorithm* Dec is a deterministic algorithm, that given an $\widehat{x}$ of length $\widehat{n}$ in the image of Enc, outputs an $x \in \Sigma^n$. Moreover, $\Pr[\mathsf{Dec}(\mathsf{Enc}(x)) = x] = 1$ for every $x \in \Sigma^n$. It would sometimes be convenient to explicitly describe the randomness used by Enc, in which case we think of Enc as a deterministic function $\mathsf{Enc}(x; r)$ of its input x, and random input r. By default, the encoding schemes considered in this work encode each input symbol separately, i.e., for $x = (x_1, \ldots, x_n)$, $\mathsf{Enc}(x) = (\mathsf{Enc}(x_1), \ldots, (x_n))$.

Parameterized encoding schemes. We consider encoding schemes in which the encoding and decoding algorithms are given an additional input 1^t, which is used as a security parameter. Concretely, the encoding length depends also on t (and not only on n), i.e., $\widehat{n} = \widehat{n}(n,t)$, and for every t the resultant scheme is an encoding scheme (in particular, for every $x \in \Sigma^n$ *and every* $t \in \mathbb{N}$, $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(x,1^t\right),1^t\right)=x\right]=1$). We call such schemes *parameterized*. Furthermore, we sometimes consider encoding schemes that take *a pair* of security parameters $1^t, 1^{t_{\text{in}}}$. (t_{in} is used in cases when the encoding scheme employs an "internal" encoding scheme, and is used in the internal scheme.) In such cases, the encoding length depends on n, t, t_{in}, and the resultant scheme should be an encoding scheme for every $t, t_{\text{in}} \in \mathbb{N}$. We will usually omit the term "parameterized", and use "encoding scheme" to describe both *parameterized and non-parameterized* encoding schemes.

2.1 Leakage-Resilient Circuit Compilers

In this section we define leakage resilient circuit compilers. We first define the notion of a circuit compiler.

Definition 2.1 (Circuit Compiler)**.** A circuit compiler over $\mathbb{F}$ is a pair $(\mathsf{Comp},\mathsf{E})$ of algorithms with the following syntax.

- $\mathsf{E}=(\mathsf{Enc},\mathsf{Dec})$ is an encoding scheme, where Enc on input $x \in \mathbb{F}^n$, and $1^t, 1^{t_{\text{in}}}$, outputs a vector $\widehat{x}$ of length $\widehat{n}$ for some $\widehat{n}=\widehat{n}(n,t,t_{\text{in}})$.
- Comp is a polynomial-time algorithm that given an arithmetic circuit C over $\mathbb{F}$, and 1^t, outputs a (randomized) arithmetic circuit $\widehat{C}$.

We require that $(\mathsf{Comp},\mathsf{E})$ satisfy the following *correctness* requirement. There exists a negligible function $\epsilon(t)=\mathsf{negl}(t)$ such that for any arithmetic circuit C, and any input x for C, we have $\Pr\left[\widehat{C}(\widehat{x})=C(x)\right]=1$, where $\widehat{x} \leftarrow \mathsf{Enc}\left(x,1^t,1^{|C|}\right)$.

A boolean circuit compiler is defined similarly, except that it operates on boolean circuits C.

We consider circuit compilers that are also "sound", meaning that satisfying inputs for the compiled circuit exist only if the original circuit is satisfiable. As discussed in Section 1.2.1 this is non-trivial because one might possibly satisfy the compiled circuits using inputs that do not encode any inputs to the original circuit.[5]

Definition 2.2 (SAT-respecting circuit compiler)**.** A circuit compiler $(\mathsf{Comp},\mathsf{E})$ is *SAT-respecting* if it satisfies the following *soundness* requirement for every circuit $C:\mathbb{F}^n \to \mathbb{F}$. If $\widehat{C}=\mathsf{Comp}(\mathsf{C})$ is satisfiable then C is satisfiable, i.e., if $\widehat{C}(\widehat{x}^*)=0$ for some $\widehat{x}^* \in \mathbb{F}^{\widehat{n}}$, then there exists an $x \in \mathbb{F}^n$ such that $C(x)=0$. (For $\mathbb{F}=\mathbb{F}_2$, we require that if $\widehat{C}$ outputs 1 on some input, then so does C.)

Next, we define leakage-resilience of a circuit compiler. The following notation will be useful.

Notation 2.3. For a Circuit C, a leakage function $\ell:\mathbb{F}^{|C|} \to \mathbb{F}^k$ for some natural k, and an input x for C, $[C,x]$ denotes the wire values of C when evaluated on x, and $\ell[C,x]$ denotes the output of ℓ on $[C,x]$.

[5] We note that LRCCs were defined for general circuits, whose output may be long. In this work, we focus on circuits that output a single bit or field element, because the SAT-respecting property is only meaningful for such circuits.

Definition 2.4 (LRCC)**.** Let t be a security parameter, and $\mathbb{F}$ be a finite field. For a function class $\mathcal{L}$, $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$, and a size function $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$, we say that $(\mathsf{Comp}, \mathsf{E})$ is $(\mathcal{L}, \epsilon(t), \mathsf{S}(n))$-*leakage-resilient* if there exists a PPT algorithm Sim such that the following holds. For all sufficiently large t, every arithmetic circuit C over $\mathbb{F}$ of input length n and size at most $\mathsf{S}(n)$, every $\ell \in \mathcal{L}$ of input length $\left|\hat{C}\right|$, and every $x \in \mathbb{F}^n$, we have $\mathsf{SD}\left(\ell\left[\mathsf{Sim}\left(C, C\left(x\right)\right)\right], \ell\left[\widehat{C}, \widehat{x}\right]\right) \leq \epsilon(t)$, where $\widehat{x} \leftarrow \mathsf{E}\left(x, 1^t, 1^{|C|}\right)$.

If the above holds with an inefficient simulator Sim, then we say that $(\mathsf{Comp}, \mathsf{E})$ is $(\mathcal{L}, \epsilon(t), \mathsf{S}(n))$-*relaxed leakage-resilient*.

Probing Leakage. One leakage class which we consider in this work is the class of t-*probing* (or t-*probing attacks*) for a parameter $t \in \mathbb{N}$ with input length n over some field $\mathbb{F}$, which consists of all functions that output t symbols of their input. Formally: $\mathcal{L}_{t,n,\mathbb{F}} := \left\{\ell : \mathbb{F}^n \to \mathbb{F}^t \ : \ \exists I \subseteq [n], |I| = t \text{ s.t. } \forall x \in \mathbb{F}^n, \ell(x) = x_I\right\}$.

2.2 Gadget-Based Leakage-Resilient Circuit Compilers

In this section we describe the paradigm of gadget-based LRCCs. We note that all LRCCs considered in this work are gadget-based. We describe the compilers over a finite field $\mathbb{F}$, which can be naturally adjusted to the boolean case as well. At a high level, given a circuit C, a gadget-based LRCC replaces every wire in C with a bundle of wires, which carry an encoding of the wire value, and every gate with a sub-circuit that emulates the operation of the gate on encoded inputs. We now describe this technique in more details.

Gadgets. A bundle is a string of field elements, encoding a field element according to some encoding scheme E; and a gadget is a circuit which operates on bundles and emulates the operation of the corresponding gate in C. A gadget has both standard inputs, that represent the wires in the original circuit, and masking inputs, that are used to achieve privacy. More formally, a gadget emulates a specific boolean or arithmetic operation on the standard inputs, and outputs a bundle encoding the correct output. Every gadget $\mathcal{G}$ is associated with a set $M_{\mathcal{G}}$ of "well-formed" masking input bundles (e.g., in the LRCC of [FRR+10], $M_{\mathcal{G}}$ consists of sets of 0-encodings). For every standard input x, on input bundles $\widehat{x}$ encoding x, and *any* masking input bundles $\mathsf{m} \in M_{\mathcal{G}}$, the output of the gadget $\mathcal{G}$ should be consistent with the operation on x. For example, if $\mathcal{G}$ computes the $\times$ operation, then for every standard input $x = (x_1, x_2)$, for every bundle encodings $\widehat{x} = (\widehat{x_1}, \widehat{x_2})$ of x according to E, and for every masking input bundles $\mathsf{m} \in M_{\mathcal{G}}$, $\mathcal{G}(\widehat{x}, \mathsf{m})$ is a bundle encoding $x_1 \times x_2$ according to E. The privacy of the internal computations in the gadget will be achieved when the masking input bundles of the gadget are uniformly distributed over $M_{\mathcal{G}}$, *regardless* of the actual values encoded by the masking input bundles.

Gadget-based LRCCs. In our constructions, the compiled circuit $\widehat{C}$ is obtained from a circuit C by replacing every wire with a bundle, every gate with the corresponding gadget, and adding decoding sub-circuits (computing the decoding function of E) following the output gates of C. Recall that the gadgets also have masking inputs. These are provided as part of the encoded input of $\hat{C}$, in the following way. $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ uses an "inner" encoding scheme $\mathsf{E}^{\text{in}} = \left(\mathsf{Enc}^{\text{in}}, \mathsf{Dec}^{\text{in}}\right)$, where Enc uses Enc^{in} to encode the inputs of C, concatenated with κ masking values (whose identity is determined by the definition of the gadget, e.g., in [FRR+10] the masking values are all zero) for a sufficiently large κ; and Dec uses Dec^{in} to decode its input, and discards the last κ symbols.

2.3 Zero-Knowledge and Witness-Indistinguishable Probabilistically Checkable Proofs

We now define PCPs with zero-knowledge guarantees. Informally, these are PCPs with the additional property that the view of any (possibly malicious) verifier that is restricted in the number of queries it can make to the proof, can be simulated given only the input.

Definition 2.5 (ZK- and WI-PCPs, [KPT97])**.** A probabilistic proof system (P, V) is a *Zero-Knowledge Probabilistically Checkable Proof (ZK-PCP)* system for an NP-relation $\mathcal{R}_L = \mathcal{R}_L(x, w)$, if the following holds.

- *Syntax.* The prover P has input $\epsilon, 1^{q^*}, x, w$, and outputs a proof π for (x, w) (i.e., $P\left(\epsilon, 1^{q^*}, x, w\right)$ defines a distribution over proofs for (x, w)). The verifier V has input ϵ, q^*, x, and oracle access to π, and outputs either acc or rej.

  We associate with P, V as above the following efficiency measures. The *alphabet* $\Sigma = \Sigma(\epsilon, q^*, |x|)$ over which π is defined; The *length* $\ell = \ell(\epsilon, q^*, |x|)$ of the proof π; the *query complexity* $q = q(\epsilon, q^*, |x|)$ of V (i.e., the number of queries V makes to his oracle); and the *randomness complexity* $r = r(\epsilon, q^*, |x|)$ of V (namely, the number of random bits it uses).

- *Semantics.* (P, V) should have the following properties.

  - *Completeness.* For every $(x, w) \in \mathcal{R}$ and every proof $\pi \in P\left(\epsilon, 1^{q^*}, x, w\right)$, $\Pr\left[V^{\pi}(\epsilon, q^*, x) = \mathsf{acc}\right] = 1$, where the probability is over the randomness of V.
  - *Soundness.* For every $x \notin L_{\mathcal{R}}$ and every π^*, $\Pr\left[V^{\pi^*}(\epsilon, q^*, x) = \mathsf{acc}\right] \leq \epsilon$.
  - (ϵ, q^*)*-Zero-Knowledge (ZK).* For every (possibly malicious) verifier V^* that reads at most q^* proof symbols there exists a PPT simulator Sim such that for every $(x, w) \in \mathcal{R}_L$, $\mathsf{SD}\left(\mathsf{V}_{V^*,P}(\epsilon, q^*, x, w), \mathsf{Sim}\left(\epsilon, 1^{q^*}, x\right)\right) \leq \epsilon$ where $\mathsf{V}_{V^*,P}(\epsilon, q^*, x, w)$ denotes the view of V^* given input x and oracle access to a proof π generated by $P\left(\epsilon, 1^{q^*}, x, w\right)$.

    If the above holds with an *inefficient* simulator, we say the PCP system is (ϵ, q^*)*-witness indistinguishable*.

In Section 7.1 we defined ZK- and WI-PCPs *of Proximity*.

3 Extensions of the ISW-LRCC [ISW03]

In this section we describe a variant of the ISW-LRCC over $\mathbb{F}_3$, which we conjecture to be secure against $\mathsf{AC}^0[\oplus]$-leakage. We first briefly recall the original ISW-LRCC. The ISW-LRCC operates over boolean circuits with AND and NOT gates. It encodes the inputs using the parity encoding (which replaces a bit b with a random bit-string whose parity is b), and provides gadgets that emulate NOT and AND gates. The NOT gadget simply applies a NOT gate on the first element of the input encoding. The AND gadget uses the observation that given two parity encodings $\widehat{a} = (a_1, \ldots, a_m), \widehat{b} = (b_1, \ldots, b_m)$ of values a, b, $\sum_{i,j} a_i b_j = ab$. Using this observation, the AND gadget computes, for every $i \neq j$, an additive secret sharing of $a_i b_j + a_j b_i$ into 2 shares, and the i'th output of the gadget is the sum of all secret shares with index i, together with $a_i b_i$. (See Construction 3.3 below for more details.)

Extending the construction to work over $\mathbb{F}_3$ requires generalizing the gadgets to work over $\mathbb{F}_3$. As we show in Construction 3.3 below, this can be achieved with minor alterations to the original construction. However, using arithmetic circuits over $\mathbb{F}_3$ to emulate boolean circuit introduces the additional complication that the input bundles are no longer guaranteed to encode bits. Thus,

we design a simple circuit $C_{\mathsf{inp}} : \{0,1\} \to \{0,1\}$ to check that the inputs correspond to bits.[6] Given a circuit $C : \{0,1\}^n \to \{0,1\}$, we define $C' : \{0,1\}^n \to \{0,1\}$ as follows: $C'(x_1, \ldots, x_n) = C(x_1, \ldots, x_n) \wedge C_{\mathsf{inp}}(x_1) \wedge \ldots \wedge C_{\mathsf{inp}}(x_n)$. We then compile C' using the modified ISW-LRCC (with encodings and gadgets that operate over $\mathbb{F}_3$). As we show in Lemma 3.6 below, $\widehat{C'}(\widehat{x_1}, \ldots, \widehat{x_n}) = 1$ if and only if for every $1 \leq i \leq n$, $\widehat{x_i}$ encodes some $x_i \in \{0,1\}$, and additionally $C(x_1, \ldots, x_n) = 1$.

Notice that we add the input-checker circuit C_{inp} to the circuit C *before* it is compiled into its leakage-resilient counterpart. This will be useful when arguing leakage resilience. Specifically, the compiled circuit C' now includes the input checker, and we can argue leakage-resilience of C' directly, without having to explicitly consider the input checker.

We now formalize this intuition, starting with the input checker.

Construction 3.1 (Input checker C_{inp})**.** $C_{\mathsf{inp}} : \{0,1\} \to \{0,1\}$ is defined as $C_{\mathsf{inp}}(x) = \neg(x \wedge \neg(x))$.

Next, we describe the encoding scheme used in our construction. Following [ISW03], we use t to denote the security parameter of the encoding scheme, namely the scheme is secure against probing of t wires. For simplicity, we define encoding for a single element in $\mathbb{F}_3$. This can be extended to encoding vectors of elements over $\mathbb{F}_3$ by encoding each element separately.

Definition 3.2 (Mod-3 encoding scheme)**.** Let $t \in \mathbb{N}$ be a security parameter, and $m := 2t + 1$. The *mod-3 encoding scheme* $\mathsf{E}_3 = (\mathsf{Enc}_3, \mathsf{Dec}_3)$ over $\mathbb{F}_3$ is a parameterized encoding scheme defined as follows:

- For every input $x \in \mathbb{F}_3$, and security parameter $t \in \mathbb{N}$, $\mathsf{Enc}_3(x, 1^t) = (x_1, \ldots, x_m)$, where $x_1, \ldots, x_m$ are random in $\mathbb{F}_3$ subject to the constraint that $\sum_{i=1}^m x_i = x$.
- For every $t \in \mathbb{N}$, and every $(x_1, \ldots, x_m) \in \mathbb{F}^m$, $\mathsf{Dec}_3(x_1, \ldots, x_m) = \sum_{i=1}^m x_i \pmod 3$.

We now describe the gadgets used in our construction.

Construction 3.3 (Gadgets for ISW-LRCC over $\mathbb{F}_3$)**.** Let $\mathsf{E}_3 = (\mathsf{Enc}_3, \mathsf{Dec}_3)$ be the mod-3 encoding scheme of Definition 3.2. The gadgets are arithmetic circuits over $\mathbb{F}_3$ that emulate the boolean operations NOT and AND.

The NOT gadget $\mathcal{G}_\neg$: takes as input $\widehat{a} = (a_1, \ldots, a_m) \in \mathsf{Enc}_3(a, 1^t)$. It outputs $(2a_1 + 1, 2a_2, \ldots, 2a_m)$.[7]

The AND gadget $\mathcal{G}_\wedge$: takes as input $\widehat{a} = (a_1, \ldots, a_m) \in \mathsf{Enc}_3(a, 1^t)$ and $\widehat{b} = (b_1, \ldots, b_m) \in \mathsf{Enc}_3(b, 1^t)$, as well as masking inputs $\{z_{ij}\}_{1 \leq i < j \leq m}$ which are uniformly random in $\mathbb{F}_3$, and operates as follows.

1. For every $1 \leq i < j \leq m$, compute $z_{ji} = (2 \cdot z_{ij} + a_i \cdot b_j) + a_j \cdot b_i$.[8]
2. For every $1 \leq i \leq m$, compute $c_i = a_i \cdot b_i + \sum_{j \neq i} z_{ij}$.

It is easy to see that if the inputs to $\mathcal{G}_\neg, \mathcal{G}_\wedge$ encode bits then the outputs are encodings of the correct values (i.e., the NOT and AND of the inputs, respectively).

[6] Since we have defined C_{inp} to operate over bits, this check seems useless. However, in the final construction C_{inp} will be implemented using an arithmetic circuit over $\mathbb{F}_3$, in which case the test will be meaningful.

[7] In [ISW03], the NOT gadget simply negates the first coordinate.

[8] In [ISW03], $z_{ji} = (z_{ij} \oplus a_i \cdot b_j) \oplus a_j \cdot b_i$. Intuitively, z_{ij}, z_{ji} is an additive secret sharing of $a_i b_j + a_j b_i$. Notice that in $\mathbb{F}_2$, $-z_{ij} = z_{ij}$, whereas in $\mathbb{F}_3$ for $z_{ij} \in \{0,1\}$, $-z_{ij} = 2z_{ij}$, which is why we multiply by 2.

Construction 3.4 (ISW-LRCC variant over $\mathbb{F}_3$)**.** Let $t, t_{\text{in}} \in \mathbb{N}$ be security parameters, and $n \in \mathbb{N}$ be an input length parameter. Let $m = 2t + 1$, and let $\mathsf{E}_3 = (\mathsf{Enc}_3, \mathsf{Dec}_3)$ be the encoding scheme of Definition 3.2. The LRCC is defined as follows.

- The encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ operates as follows:
  - for every $x \in \mathbb{F}^n$, $\mathsf{Enc}\left(x, 1^t, 1^{t_{\text{in}}}\right) = \left(\mathsf{Enc}_3\left(x, 1^t\right), r\right)$ for $r \in_R \mathbb{F}_3^{(t_{\text{in}}+n)\cdot M}$, where $M = \frac{m(m-1)}{2}$ is the number of random values used in the gadget $\mathcal{G}_\wedge$. (Intuitively, r provides the randomness needed for all the AND gadgets in the compiled circuit.)
  - $\mathsf{Dec}\left(\left(\widehat{x}, r\right), 1^t, 1^{t_{\text{in}}}\right) = \mathsf{Dec}_3\left(x, 1^t\right)$.
- The compiler, given a circuit $C : \{0,1\}^n \to \{0,1\}$ with s AND gates, outputs the circuit $\widehat{C} : \mathbb{F}_3^{n+(s+n)M} \to \mathbb{F}_3$ constructed as follows:[9]
  - Let $C' : \{0,1\}^n \to \{0,1\}$ such that $C'\left(x_1, \ldots, x_n\right) = C\left(x_1, \ldots, x_n\right) \wedge \left(\wedge_{i=1}^n C_{\mathsf{inp}}\left(x_i\right)\right)$, where C_{inp} is the input checker from Construction 3.1.
  - Let $\widehat{C'}$ denote the circuit obtained from C' when all $\neg, \wedge$ gates are replaced with $\mathcal{G}_\neg, \mathcal{G}_\wedge$ gadgets, respectively.
  - $\widehat{C}$ is obtained from $\widehat{C'}$ by concatenating the decoding circuit Dec to $\widehat{C'}$'s output.

We make the following observations regarding the gadgets of Construction 3.3:

Observation 3.5. *The gadgets of Construction 3.3 satisfy the following:*

- *For any $a \in \{0,1\}$, and any $(a_1, \ldots, a_m) \in \mathsf{Enc}_3\left(a, 1^t\right)$, $\mathcal{G}_\neg\left(a_1, \ldots, a_m\right) = \left(c_1, \ldots, c_m\right)$ such that $\sum_{i=1}^m c_i = \neg\left(a\right)$ (namely, the output encodes $\neg\left(a\right)$).*
- *For any $\left(a_1, \ldots, a_m\right) \in \mathsf{Enc}_3\left(2, 1^t\right)$, $\mathcal{G}_\neg\left(a_1, \ldots, a_m\right) \in \mathsf{Enc}_3\left(2, 1^t\right)$ (namely, the output encodes 2).*
- *For any $a, b \in \mathbb{F}_3$, any $\left(a_1, \ldots, a_m\right) \in \mathsf{Enc}_3\left(a, 1^t\right)$, and $\left(b_1, \ldots, b_m\right) \in \mathsf{Enc}_3\left(b, 1^t\right)$, $\mathcal{G}_\wedge\left(\left(a_1, \ldots, a_m\right), \left(b_1, \ldots, b_m\right)\right) = \left(c_1, \ldots, c_m\right)$ such that $\sum_{i=1}^m c_i = a \cdot b$ (namely, the output encodes $a \cdot b$).*

We now show that Construction 3.4 is SAT-respecting. Recall that for arithmetic circuits, SAT-respecting means that if $\widehat{C}\left(\widehat{x}\right) = 0$ for some $\widehat{x}$ (which isn't necessarily a valid encoding), then there exists an x such that $C\left(x\right) = 0$. However, since we are interested in compiling *boolean* circuits, and we implement the field operations using any correct boolean sub-circuits, the property we actually want is that if $\widehat{C}\left(\widehat{x}\right) = 1$ for some $\widehat{x}$, then there exists an x such that $C\left(x\right) = 1$, which we now prove.

Lemma 3.6. *Let $C : \{0,1\}^n \to \{0,1\}$, and let $\widehat{C}$ be the circuit obtained from C in Construction 3.4. If $\widehat{C}\left(\widehat{x}\right) = 1$ for some $\widehat{x}$ then there exits an $x \in \{0,1\}^n$ such that $C\left(x\right) = 1$.*

[9] Syntactically, the compiler should return a boolean circuit (i.e., a circuit operating over the same field as the original circuit). To achieve this we implement the field operations used in $\widehat{C}$ using boolean sub-circuits. However, to simplify the description, we chose to describe $\widehat{C}$ as an arithmetic circuit. We note that this doesn't affect $\mathsf{AC}^0\left[\oplus\right]$-leakage-resilience.

Proof. We know from Observation 3.5 that if the inputs of the gadgets of Construction 3.3 encode bits then the output encodes the correct value (which, in particular, is also a bit). Therefore, if the input encodings $\widehat{x_1}, \ldots, \widehat{x_n}$ of $\widehat{C}$ are valid encodings of bits $x_1, \ldots, x_n$ then $\widehat{C}\left(\widehat{x_1}, \ldots, \widehat{x_n}\right) = C\left(x_1, \ldots, x_n\right)$, and in particular $\widehat{C}\left(\widehat{x_1}, \ldots, \widehat{x_n}\right) = 1$ if and only if $C\left(x_1, \ldots, x_n\right) = 1$.

Therefore, to prove the construction is SAT-respecting, it suffices to prove that if not all $x_1, \ldots, x_n$ are bits, then $\widehat{C}\left(\widehat{x_1}, \ldots, \widehat{x_n}\right) = 0$. (We note that $x_1, \ldots, x_n$ are well defined because any vector $\widehat{x} \in \mathbb{F}_3^{n \cdot m}$ encodes *some* value.) Assume without loss of generality that $x_1 \notin \{0, 1\}$. We claim in this case $\widehat{C_{\mathsf{inp}}}\left(x_1\right) = 0$, where $\widehat{C_{\mathsf{inp}}}$ is the circuit obtained from C_{inp} when all gates are replaced with the corresponding gadgets.

To see why this holds, notice that if $x_1 \notin \{0, 1\}$ then $x_1 = 2$. Let $\left(x_{1,1}, \ldots, x_{1,m}\right) \in \mathsf{Enc}\left(x_1, 1^t\right)$ be some arbitrary encoding of x_1. Then by Observation 3.5, $\mathcal{G}_{\neg}\left(x_{1,1}, \ldots, x_{1,m}\right) = \left(z_1, \ldots, z_m\right)$ such that $\sum_{i=1}^{m} z_i = 2$. Again using Observation 3.5, $\mathcal{G}_{\wedge}\left(\left(x_{1,1}, \ldots, x_{1,m}\right), \left(z_1, \ldots, z_m\right)\right) = \left(z'_1, \ldots, z'_m\right)$ such that $\sum_{i=1}^{m} z'_i = 1$ (indeed, both inputs to $\mathcal{G}_{\wedge}$ encode 2 so the output encodes $2 \cdot 2 (\mathrm{mod} 3) = 1$). Finally, using Observation 3.5 again, $\mathcal{G}_{\neg}\left(z'_1, \ldots, z'_m\right) = \left(z''_1, \ldots, z''_m\right)$ such that $\sum_{i=1}^{m} z''_i = 0$, i.e., $\widehat{C_{\mathsf{inp}}}\left(x_{1,1}, \ldots, x_{1,m}\right)$ encodes 0.

By Observation 3.5, the gadget $\mathcal{G}_{\wedge}$ is correct for all inputs in $\mathbb{F}_3$. Therefore, the fact that $\widehat{C_{\mathsf{inp}}}\left(x_{1,1}, \ldots, x_{1,m}\right)$ encodes 0 implies that $\widehat{C'}$ outputs 0.

□

As discussed in Section 1.2.1, we do not currently know how to prove that Construction 3.4 resists $\mathsf{AC}^0\left[\oplus\right]$-leakage. However, as a sanity check, we show that the modified construction at least is as secure as the original ISW-LRCC, by proving that it resists probing attacks. (For simplicity, we consider the circuit $\widehat{C}$ as described over $\mathbb{F}_3$, so each probed wire carries a field element in $\mathbb{F}_3$.)

Lemma 3.7. *Construction 3.4 is secure against t-probing leakage.*

The proof follows the security proof of [ISW03, Theorem 1], with slight modifications that are needed due to the slightly modified construction. The main difference in the proof is due to the slightly different security definition we use: in [ISW03], both the inputs and outputs of the circuit were considered private, and any probing of the wires of the circuit *excluding the input encoder and output decoder* should be simulated "from scratch". We consider the output to be public (indeed, this is needed when using LRCCs to construct ZK-PCPs), and require that any probing of the wires of the circuit *and the output decoder* can be simulated given the output of the circuit. In the proof, we account for this difference in definitions, and show how to adapt the proof for the case of public output. We stress that we are able to protect against probes on the decoder because we do not try to hide the output.

Proof of Lemma 3.7. Similar to [ISW03], we first construct a simulator that can perfectly simulate t-probing of a single gadget $\mathcal{G}$ in Construction 3.4, given at most $m-1$ shares of its inputs. Since the inputs are shared using an additive secret sharing with m shares, any $m-1$ shares are uniformly distributed in $\mathbb{F}_3^{m-1}$, and can therefore be simulated as random values.

Simulating an AND gate. Let $w_1, \ldots, w_t$ denote the probed wires of $\mathcal{G}$, and denote its inputs by $\widehat{a} = \left(a_1, \ldots, a_m\right), \widehat{b} = \left(b_1, \ldots, b_m\right)$. We define a set $I \subseteq [m]$ of size $|I| \leq m - 1$ such that $w_1, \ldots, w_t$ can be perfectly simulated given the restriction of the inputs of $\mathcal{G}$ to I. This is done as follows:

1. Initialize $I = \emptyset$ and $w_1, \ldots, w_t$ to be unassigned.
2. For every $1 \leq k \leq t$:

(a) If w_k is of the form $a_i, b_i, a_i b_i, z_{ij}$ or $2 \cdot z_{ij}$ for some $i \neq j$, or a sum of values of this form, then add i to I.

(b) Otherwise, w_k is of the form $a_i b_j$ or $2 \cdot z_{ij} + a_i b_j$ for some $i \neq j$, in which case add both i and j to I.

Notice that each w_k adds at most two indices to I, so $|I| \leq 2t = m - 1$. This completes the description of I. The following steps describe how to simulate $w_1, \ldots, w_t$ given only $\widehat{a}_I, \widehat{b}_I$.

3. Assign values to the z_{ij} as follows:

   (a) If $i \in I$ and $j \notin I$ then assign to z_{ij} a random value in $\mathbb{F}_3$. Notice that if $i < j$ then this is exactly how z_{ij} is chosen in the construction. If $j < i$ then from the definition of I, z_{ji} does not appear in the computation of any w_k. Since (z_{ij}, z_{ji}) is an additive secret sharing of $a_i b_j + a_j b_i$, when z_{ji} remains secret then z_{ij} is uniformly distributed, so the simulated z_{ij} is distributed identically to the real z_{ij} (conditioned on the event that x_{ji} remains hidden).

   (b) If $i, j \in I$ then the simulator knows a_i, b_i, a_j, b_j and can compute z_{ij} identically to the real world, namely if $i < j$ then z_{ij} is assigned a random value in $\mathbb{F}_3$ and $z_{ji} = 2z_{ij} + a_i b_j + a_j b_i$, otherwise z_{ji} is assigned a random value, and $z_{ij} = 2z_{ji} + a_i b_j + a_j b_i$.

   (c) If $i \notin I$ then leave z_{ij} unassigned. This will not affect the simulation since by definition of I, z_{ij} is never used in the computation of any w_k.

4. Simulate the values of any wire $w_k, 1 \leq k \leq t$ as follows:

   (a) If w_k is of the form $a_i, b_i, a_i b_i$, z_{ij} for some $i \neq j$, or any sum of such values (including c_i as a special case) then since $i \in I$ we know a_i, b_i and all z_{ij} have been perfectly simulated in Step 3, so w_k can be perfectly simulated.

   (b) Otherwise, w_k is of the form $a_i b_j$ or $2z_{ij} + a_i b_j$. By the definition of I, it holds that $i, j \in I$ so the simulator knows a_i, b_j. Additionally, z_{ij} has already been perfectly simulated in Step 3b, so w_k can be perfectly simulated.

Simulating a NOT gate. Let $w_1, \ldots, w_t$ denote the probed wires of $\mathcal{G}$, and denote its input by $\widehat{a} = (a_1, \ldots, a_m)$. We define a set $I \subseteq [m]$ of size $|I| \leq m - 1$ such that $w_1, \ldots, w_t$ can be perfectly simulated given the restriction $\widehat{a}_I$ of $\widehat{a}$ to I. This is done as follows:

1. Initialize $I = \emptyset$ and $w_1, \ldots, w_t$ to be unassigned.

2. For every $1 \leq k \leq t$, w_k is of the form $a_i, 2a_i$ or $2a_i + 1$. Add i to I.

3. For every $1 \leq k \leq t$, notice that w_k depends on a single a_i, and $i \in I$ so the simulator knows a_i. Therefore, the simulator can perfectly simulate w_k.

In particular, notice that all wires $c_i, i \in I$ in the outputs of both NOT and AND gadgets can be perfectly simulated in the simulation described above. This will be useful when constructing the simulator for the entire circuit.

Finally, we use the gadget simulators to simulate the entire circuit. The simulator is given the output $y \in \mathbb{F}_3$ of the circuit, and operates as follows. For each gadget $\mathcal{G}$, it compute the set $I_{\mathcal{G}}$ as described above. Notice that since at most t wires are probed *throughout* the circuit, $I := \cup_{\mathcal{G}} I_{\mathcal{G}}$ has size $|I| \leq m - 1$. Next, it simulates the probed wires in the gadgets, working from the inputs to the outputs. For this strategy to work, we need to show that when simulating a gadget $\mathcal{G}$, the

simulator already knows the restriction of $\mathcal{G}$'s inputs to $I_{\mathcal{G}}$. This holds due to the observation made above, that the wires $c_i, i \in I_{\mathcal{G}'}$ can be perfectly simulated for every gadget $\mathcal{G}'$. Thus, if the inputs of $\mathcal{G}$ are the inputs of the circuit, then these are perfectly simulated by the simulator as uniformly random values in $\mathbb{F}_3$. Otherwise, these are the outputs of a previous gadget $\mathcal{G}'$, and therefore were already perfectly simulated.

Finally, the simulator needs to simulate probes to the output decoder. Let $\widehat{a} = (a_1, \ldots, a_m)$ denote the input to the output decoder. Some subset of these values have already been assigned during the simulation, however since at most $m-1$ of these were already assigned values, there exists some $i^* \in [m]$ such that a_{i^*} is unassigned. The simulator assigns random values to all unassigned $a_i, i \neq i^*$, and sets $a_{i^*} = \Sigma_{i \neq i^*} a_i - y$. It then honestly applies the decoder to $\widehat{a}$ and uses the resultant wire values to answer the probes to the wires of the decoder.

It remains to prove that the simulation of the decoder wires is perfect. For that, it suffices to prove that the simulated inputs to the decoder are perfectly simulated (because the simulator honestly evaluates the decoder). To see why this holds, notice that any $m-1$ shares of $\widehat{a}$ are uniformly distributed (due to the z_{ij}'s), and the output determines the final share. That is exactly how the shares are computed in the simulation, so the simulation is perfect.

□

4 Extensions of the ADF-LRCC [ADF16]

In this section we describe a variant of the ADF-LRCC that is SAT-respecting. Recall from Section 1.2.1 that the ADF-LRCC uses a linear "multiplication friendly" encoding scheme in which, roughly given a pair $\widehat{a} = (a_1, \ldots, a_n)$ and $\widehat{b} = (b_1, \ldots, b_n)$ of encodings of a, b (respectively), the product $c = ab$ can be computed as a linear combination of the products $a_1 b_1, \ldots, a_n b_n$. Using this property, Andrychowicz et al. design a multiplication gadget that is much more efficient than the gadget of [ISW03]. They then instantiate the construction with Shamir's secret sharing [Sha79] or with AG codes [CC06], which are both error-correcting codes. We focus here on the Shamir-based construction over fields $\mathbb{F} = \mathbb{F}_q$, where q is a prime.

When designing SAT-respecting LRCCs, using an error-correcting code as the underlying encoding has the disadvantage that not all vectors of field elements of the "right" length correspond to valid encodings. That is, if Enc outputs elements in $\mathbb{F}^m$ for some $m \in \mathbb{N}$, then there exist $\widehat{v} \in \mathbb{F}^m$ which are *not* valid encodings. Consequently, the leakage-resilient version $\widehat{C}$ of a circuit C might be satisfiable using invalid encodings (that do not encode any value), even if C is *not* satisfiable, meaning the LRCC would not be SAT-respecting. Consequently, $\widehat{C}$ must verify its inputs are valid encodings. (Once the inputs are valid encodings, all intermediate encodings computed in $\widehat{C}$ are guaranteed to be valid encoding from the correctness of the compiler.)

Therefore, a SAT-respecting variant of the ADF-LRCC must verify the input encodings are indeed valid encodings. Luckily, this check can be carried out using (almost only) the gadgets of the ADF-LRCC construction (described in Construction 4.2), as we now explain.

The ADF-LRCC compiles arithmetic circuits over a finite field $\mathbb{F}$ into leakage-resilient counterparts over $\mathbb{F}$. It uses Shamir's encoding scheme (Definition 4.1) to encode the field elements, where encodings have length $m = 2t + 2$, and correspond to degree-t polynomials, represented as a list of $t + 2$ coefficients.[10] The ADF-LRCC uses also degree-$2t$ encodings, and we represent

[10] Andrychowicz et al. [ADF16] set $m = 2t+1$ and use lists of $t+1$ coefficients. However, increasing m does not hurt security or efficiency (since we are increasing m only by a constant). We need a larger m, and longer coefficient-list, to verify validity of the input encodings.

those using $2t+2$ coefficients.[11] Notice that a polynomial $p(x)$ represented as a list of $t+2$ coefficients (of the monomials $1, x, \ldots, x^t, x^{t+1}$) is of degree t if and only if the leading coefficient is 0. Thus, given the list of coefficients of p, we can easily verify that it has degree t. The question is how to obtain, from a given encoding, the coefficients of the underlying polynomial *without violating leakage-resilience*. Our main observation is that the multiplication gadget of [ADF16] in fact computes these coefficients.

In a little more detail, the multiplication gadget, given encodings $\widehat{a} = (a_1, \ldots, a_m)$ and $\widehat{b} = (b_1, \ldots, b_m)$, computes $\widehat{c} = (a_1 \cdot b_1, \ldots, a_m \cdot b_m)$, masks $\widehat{c}$ by adding to it a degree-$2t$ Shamir encoding $\widehat{r}$ of a random field element, and decodes $\widehat{c} + \widehat{r}$. (The multiplication gadget contains additional steps which are not important for us, see Construction 4.2 for the full description of the gadget.) If $\widehat{a}, \widehat{b}$ are degree-t encodings, then $\widehat{c}$, and consequently also $\widehat{c} + \widehat{r}$, are degree-$2t$ encodings.[12] The decoding procedure reconstructs all the coefficients of the underlying polynomial, then discards all except the free coefficient.

Since we are representing polynomials with one redundant coefficient (i.e., we include the coefficient of x^{t+1} in a degree-t polynomial, and the coefficient of x^{2t+1} in a degree-$2t$ polynomial), we can use the discarded coefficients to check validity of the encodings. In particular, $\widehat{a}, \widehat{b}$ are both valid (i.e., correspond to degree-t polynomials) if and only if the leading coefficient when decoding $\widehat{c} + \widehat{r}$ is 0. This can be checked without violating leakage-resilience because the coefficients are anyway computed by the leakage-resilient circuit $\widehat{C}$, and any computation performed on the leading coefficient can be simulated by setting it to 0 in the simulation. Indeed, this will perfectly simulate the computation because in an honest execution the leading coefficient is guaranteed to be 0. This intuition is formalized in Construction 4.5 below. We first define Shamir's encoding scheme (this is simply Shamir's secret sharing scheme) and the ADF-LRCC multiplication gadget.

Definition 4.1 (Shamir's encoding)**.** Let $\mathbb{F}$ be a finite field, $t \in \mathbb{N}$ be a security parameter, and $m := t+2$. Let $\alpha_1, \ldots, \alpha_m \in \mathbb{F}$ be m distinct field elements. *Shamir's encoding scheme* $\mathsf{E}_S = (\mathsf{Enc}_S, \mathsf{Dec}_S)$ over $\mathbb{F}$ is a parameterized encoding scheme defined as follows:

- For every input $a \in \mathbb{F}$, and security parameter $t \in \mathbb{N}$, $\mathsf{Enc}_S\left(x, 1^t\right)$ operates as follows:
  - Pick $a_1, \ldots, a_t \in_R \mathbb{F}$.
  - Set $a_{t+1} := 0$.
  - Let $p_x(y) = x + \sum_{i=1}^{t+1} a_i x^i$.
  - Output $(p_x(\alpha_1), \ldots, p_x(\alpha_m))$.
- For every $t \in \mathbb{N}$, and every $\widehat{x} = (x_1, \ldots, x_m) \in \mathbb{F}^m$, $\mathsf{Dec}_S(x_1, \ldots, x_m)$ interpolates the degree-$(t+1)$ polynomial $p(y) = \sum_{i=0}^{t+1} a_i y^i$ such that $p(\alpha_i) = x_i$ for every $1 \le i \le m$,[13] and output x_0.

The multiplication gadget of [ADF16] uses as a subroutine a randomness sampling procedure called $\mathsf{RandSamp}$ which resists t-probing attacks. Informally, $\mathsf{RandSamp}\left(1^t\right)$ samples a random $r \in_R \mathbb{F}$, and outputs a degree-t and a degree-$2t$ encoding of r. Since the actual implementation details of this procedure is of no importance to us, we refer the interested reader to [ADF16] for the exact description of this procedure.

[11] In [ADF16], degree-t and degree-$2t$ polynomials were represented using lists of $t+1$ and $2t+1$ coefficients, respectively.

[12] We note that $\widehat{r}$ is generated inside the leakage-resilient circuit, and is therefore guaranteed to have degree $2t$.

[13] This is done by multiplying $\widehat{x}$ to the right with the inverse of the Vandermonde matrix.

Construction 4.2 (Multiplication gadget of ADF-LRCC [ADF16])**.** Let M denote the number of random field elements used in $\mathsf{RandSamp}\left(1^t\right)$. The multiplication gadget takes as input $\widehat{a} = (a_1, \ldots, a_m) \in \mathsf{Enc}_S\left(a, 1^t\right)$ and $\widehat{b} = (b_1, \ldots, b_m) \in \mathsf{Enc}_S\left(b, 1^t\right)$, as well as masking inputs $\vec{r} \in \mathbb{F}^M$. It outputs $\widehat{c} = (c_1, \ldots, c_m) \in \mathsf{Enc}_S\left(a \cdot b, 1^t\right)$ computed as follows.

1. Sample $(\widehat{r}', \widehat{r}'') = \mathsf{RandSamp}\left(1^t; \vec{r}\right)$. ($\widehat{r}', \widehat{r}''$ are degree-t and degree$-2t$ encodings, respectively, of the same random field element r.)
2. For every $1 \leq i \leq m$, compute $v_i = a_i \cdot b_i$.
3. Let $\widehat{v} = (v_1, \ldots, v_m)$. Compute $\widehat{w} = \widehat{v} + \widehat{r}''$.
4. Decode $\widehat{w}$ using Dec_S (from Definition 4.1), and let w denote the value which Dec_S outputs.
5. Set $\widehat{z} = (z_1, ..., z_n)$ such that $z_i = w$ for every $1 \leq i \leq m$, and output $\widehat{c} = \widehat{z} - \widehat{r}'$.

We now use the multiplication gadget of [ADF16] to design an input-checker circuit, similar to the one constructed in Section 3.

Construction 4.3 (Input checker C_{inp})**.** The input checker circuit $C_{\mathsf{inp}} : \mathbb{F}^m \times \mathbb{F}^{2M_\times} \to \mathbb{F}$, where $M_\times$ denotes the number of random field elements used in the multiplication gadget of Construction 4.2, on input $(\widehat{x}, \vec{r})$ operates as follows:

1. Apply the copy gadget of Construction 4.8 to $\widehat{x}$ to obtain two copies $\widehat{x}', \widehat{x}''$ encoding the same value as $\widehat{x}$, where the random field elements used in the gadget are taken from the first half of $\vec{r}$.
2. Evaluate the multiplication gadget of Construction 4.2 on inputs $(\widehat{x}', \widehat{x}')$, where the random field elements used in the gadget are taken from the second half of $\vec{r}$.
3. Let $(a_0, \ldots, a_{2t+1})$ denote the field elements computed during the decoding of $\widehat{w}$ in Step 4 of Construction 4.2.
4. Output $1 - a_{2t+1}^{q-1}$, where $q := |\mathbb{F}|$.

We make the following observation regarding the input checker, which follows directly from its description and from Fermat's little theorem.

Observation 4.4. *For every $\widehat{x} \in \mathbb{F}^m$, and any $\vec{r} \in \mathbb{F}^{M_\times}$:*

- *If $\widehat{x} \in \mathsf{Enc}_S\left(x, 1^t\right)$ for some $x \in \mathbb{F}$, then $C_{\mathsf{inp}}\left(\widehat{x}, \vec{r}\right) = 1$.*
- *Otherwise, $C_{\mathsf{inp}}\left(\widehat{x}, \vec{r}\right) = 0$.*

Finally, we combine the input checker with the ADF-LRCC to obtain a SAT-respecting variant.

Construction 4.5 (SAT-respecting ADF-LRCC variant)**.** Let $t, t_{\text{in}} \in \mathbb{N}$ be security parameters, and $n \in \mathbb{N}$ be an input length parameter. Let $m = 2t + 2$, and let $\mathsf{E}_S = (\mathsf{Enc}_S, \mathsf{Dec}_S)$ be the encoding scheme of Definition 4.1. The LRCC is defined as follows.

- The encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ operates as follows:
  - for every $x \in \mathbb{F}^n$, $\mathsf{Enc}\left(x, 1^t, 1^{t_{\text{in}}}\right) = \left(\mathsf{Enc}_S\left(x, 1^t\right), r\right)$, where $r \in_R \mathbb{F}^{t_{\text{in}} \cdot M}$, and M is the number of random field elements used in the multiplication gadget of Construction 4.2. (Intuitively, r provides the randomness needed for all the gadgets in the compiled circuit.)

– $\mathsf{Dec}\left((\widehat{x}, r), 1^t, 1^{t_{\text{in}}}\right) = \mathsf{Dec}_S\left(\widehat{x}, 1^t\right)$.

- The compiler, given a circuit $C : \mathbb{F}^n \to \mathbb{F}$ with s gates, outputs the circuit $\widehat{C}$ constructed as follows:[14]
  - Let $\widehat{C'}$ denote the circuit obtained from C when all gates are replaced with the corresponding gadgets of the ADF-LRCC (as described in Constructions 4.2 and 4.8), and the random field elements they use are taken from $\widehat{C}$'s second input.
  - Let $\widehat{C''} : \mathbb{F}^{n\cdot(m+M)} \to \mathbb{F}$ such that $\widehat{C''}\left(\widehat{x_1}, \ldots, \widehat{x_m}, \vec{r_1}, \ldots, \vec{r_m}\right) = \prod_{i=1}^{n} C_{\mathsf{inp}}\left(\widehat{x_i}, \vec{r_i}\right)$. (Notice that by Observation 4.4, $\widehat{C''}$ outputs 1 if $\widehat{x_1}, \ldots, \widehat{x_m}$ are valid encodings, otherwise it outputs 0.)
  - output the circuit $\widehat{C}$ such that

$$\widehat{C}\left(\widehat{x_1}, \ldots, \widehat{x_m}, \widehat{r_1}, \ldots, \widehat{r_m}, \widehat{r}\right) = \widehat{C'}\left(\widehat{x_1}, \ldots, \widehat{x_m}, \widehat{r}\right) \cdot \widehat{C''}\left(\widehat{x}_1, \ldots, \widehat{x_n}, \widehat{r_1}, \ldots, \widehat{r_m}\right)$$

$$+1 - \widehat{C''}\left(\widehat{x}_1, \ldots, \widehat{x_n}, \widehat{r_1}, \ldots, \widehat{r_m}\right).$$

    (Notice that the output is $\widehat{C'}\left(\widehat{x_1}, \ldots, \widehat{x_m}, \widehat{r}\right)$ if $\widehat{C''}\left(\widehat{x}_1, \ldots, \widehat{x_n}, \widehat{r_1}, \ldots, \widehat{r_m}\right) = 1$, otherwise it is 1.)

Next, we analyze Construction 4.5 and show it is leakage-resilient and SAT-respecting.

Lemma 4.6. *Construction 4.5 is correct and SAT-respecting.*

Proof. Notice first that if the inputs of $\widehat{C}$ are valid degree-t Shamir encodings then by the correctness of the ADF-LRCC, $\widehat{C}$ emulates C, which implies the construction is correct. In particular, $\widehat{C}$ outputs 0 if and only if there exists a corresponding input for C (specifically, the inputs encoded in $\widehat{C}$'s inputs) that causes C to output 0.

We show that if the inputs of $\widehat{C}$ are not well-formed then $\widehat{C}$ does not output 0. By Observation 4.4, if any of $\widehat{C}$'s inputs is not a valid degree-t Shamir encoding, then C_{inp} output 0, in which case $\widehat{C''}$ outputs 0, so $\widehat{C}$ outputs 1. □

Lemma 4.7. *Construction 4.5 is t-probing secure.*

Proof. We prove security by reduction to the t-probing security of the ADF-LRCC, by using the ADF-LRCC simulator to simulate any t-probing attack on $\widehat{C}$. First, notice that any probing of wires in $\widehat{C'}$, or wires of $\widehat{C''}$ which are part of the computation of Steps 1 and 2 in one of the copies of the input checker C_{inp} of Construction 4.3, can be perfectly simulated by the ADF-LRCC simulator. Indeed, though these were computed in Construction 4.5 as two separate circuits, one can think of them as consisting of a single circuit $\widetilde{C}$ consisting of C, as well as the multiplication gates of the form $x_i \cdot x_i$ for every input $1 \leq i \leq n$, where the outputs of these multiplication gates are never used. Given this description, $\widehat{C'}$ together with the first two steps of each C_{inp} copy are obtained by applying the ADF-LRCC to $\widetilde{C}$. Here, we also use the fact, discussed below, that the encoding $\widehat{d}$ of 1 used in the construction can be any arbitrary encoding.

It remains to explain how one can simulate the wires in Steps 3 and 4 of each C_{inp} (Construction 4.3), as well as the multiplication gate between the outputs of $\widehat{C'}, \widehat{C''}$. The simulator simply simulates these wire values by setting every a_{2t+1} in Step 3 of Construction 4.3 to 0, and the output of $\widehat{C'}$ to be the output of C which it obtained for the simulation, and outputs the wires probed by

[14] We note that the gadgets of $\widehat{C}$ uses $(s+n)\,M$ random bits.

the leakage function. This is a perfect simulation because in an honest execution (i.e., when the inputs of $\widehat{C}$ are valid degree-t encodings), $a_{2t+1} = 0$ in all the copies of C_{inp}, and $\widehat{C}$ emulates C so in particular its output is the same as C. □

We now describe the remaining gadgets of the ADF-LRCC. The gadgets use an encoding $\widehat{d}$ of 1 which is hard-wired into the compiled circuit (all gadgets use the same encoding).[15] In [ADF16], $\widehat{d}$ is sampled once and for all as a *random* encoding of 1, and hard-wired into the circuit. In the context of constructing ZK-PCPs, this is problematic since the "LRCC to ZK-PCP" transformation of [IWY16] requires a *deterministic* procedure of compiling a circuit C into its leakage-resilient version $\widehat{C}$.[16] One possible approach is to have $\widehat{C}$ take $\widehat{d}$ as input, but for SAT-respecting this would entail verifying that $\widehat{d}$ is a valid encoding, and moreover that it encodes 1. We provide a much simpler solution, by simply having $\widehat{d}$ be a *fixed* encoding of 1, e.g., obtained as $\mathsf{Enc}_S\left(1, 1^t; \vec{0}\right)$ (i.e., when encoding uses the all-0 string as randomness). We observe that this preserves leakage-resilient since the security of the ADF-LRCC anyway does not depend on $\widehat{d}$ being private. Indeed, the ADF-LRCC guarantees a strong notion of probing security which, roughly, protects against probing t wires *in each gadget*.[17] Thus if, for example, the circuit contains sufficiently many ($\approx m/t$) addition gates then by probing t wires of $\widehat{d}$ in each gadget, the leakage function can leak the entire encoding $\widehat{d}$, so security cannot rely on it being secret.

Andrychowicz et a. [ADF16] design addition and multiplication gadgets, as well as gadgets emulating randomness-sampling gates that output a random field element, and copy gates that are used to handle large fan-outs.

Construction 4.8 (Gadgets of ADF-LRCC [ADF16])**.** Let $\widehat{d} = \mathsf{Enc}_S\left(1, 1^t; \vec{0}\right)$, let $\mathcal{G}_\times$ denote the multiplication gadget of Construction 4.2, and let M denote the number of random field elements used in $\mathcal{G}_\times$.

The addition gadget: takes as input $\widehat{a} \in \mathsf{Enc}_S\left(a, 1^t\right)$ and $\widehat{b} \in \mathsf{Enc}_S\left(b, 1^t\right)$, as well as masking inputs $\vec{r} \in \mathbb{F}^M$. It outputs $\widehat{c} \in \mathsf{Enc}_S\left(a \cdot b, 1^t\right)$ computed as follows. Let $\widehat{z} = \widehat{a} + \widehat{b}$, then output $\widehat{c} = \mathcal{G}_\times\left(\widehat{z}, \widehat{d}; \vec{r}\right)$.

The randomness gadget: takes masking inputs $\vec{r} \in \mathbb{F}^M$, and outputs $\widehat{c} \in \mathsf{Enc}_S\left(r, 1^t\right)$ for a random $r \in_R \mathbb{F}$, computed as follows. Compute $(\widehat{r}', \widehat{r}'') = \mathsf{RandSamp}\left(1^t; \vec{r}\right)$ and outputs $\widehat{r}'$.

The copy gadget: takes as input $\widehat{a} \in \mathsf{Enc}_S\left(a, 1^t\right)$, and masking inputs $\vec{r}, \vec{r}' \in \mathbb{F}^M$, and outputs $\widehat{b}, \widehat{c} \in \mathsf{Enc}_S\left(a, 1^t\right)$, computed as follows. Compute $\widehat{b} = \mathcal{G}_\times\left(\widehat{a}, \widehat{d}; \vec{r}\right)$ and $\widehat{c} = \mathcal{G}_\times\left(\widehat{a}, \widehat{d}; \vec{r}'\right)$, and output $\left(\widehat{b}, \widehat{c}\right)$.

Further Extensions and Future Directions. We have described a SAT-respecting variant of the ADF-LRCC using Shamir's secret sharing. The ADF-LRCC of [ADF16] can use AG codes

[15] Specifically, the ADF-LRCC uses the multiplication gadget to "refresh" the encodings at the outputs of these gadgets, i.e., replace them with "fresh" encodings of the same value. This is done by using the multiplication gadget to multiply the encoding at the output of the gadget with the encoding $\widehat{d}$.

[16] This is because in the transformation, the claim "C is satisfiable" (i.e., is in the NP-complete language of circuit satisfiability) is reduced to the claim "the leakage-resilient version $\widehat{C}$ of C is satisfiable". To verify this claim, the verifier needs to know the structure of $\widehat{C}$, since this is the NP statement being verified. If the compilation of C into $\widehat{C}$ is deterministic then it can be locally applied by the prover and verifier. However, if the compilation is randomized then the verifier does not know what randomness the prover used to generate $\widehat{C}$, and consequently does not know the structure of $\widehat{C}$ and cannot verify its satisfiability.

[17] The actual security guarantee is even stronger, but the weaker, and much simpler, description provided here is sufficient for our needs.

as the underlying encoding scheme, and we leave it to future work to design an input checker for AG codes. (One possible approach is to apply the parity check matrix of the code to the input encodings, but this significantly diverges from the ADF-LRCC, so would require proving leakage-resilience of this variant.) We note also that using an LRCC to construct a ZK-PCP requires the LRCC to operate over boolean circuits. In the ADF-LRCC using AG encoding, this can be easily obtained by implementing the field operations using constant-sized boolean circuits, since the field $\mathbb{F}$ can have constant size. However, the Shamir-based construction requires using large fields (of size $O(t)$) so implementing the field operations using boolean circuits in the natural way would cause a loss in security. Finally, we note that our input checker C_{inp} (Construction 4.5), when implemented using gates of fan-in 2, has depth $\log q = O(\log t)$ due to the use of Fermat's little theorem in Step 4, which might be significant (e.g., when C has constant depth). It might be possible to design a different input check that avoids this blowup.

5 Obstacles in Extending the GIMSS-LRCC [GIM+16] to Resist $\mathsf{AC}^0[\oplus]$-Leakage

As discussed in Section 1.2.1, the GIMSS-LRCC cannot be used to construct ZK-PCPs because the leakage class it resists does not contain any PCP. A natural direction would then be to try to prove that the GIMSS-LRCC, or simple adaptations of it, resists leakage from a function class that does contain a PCP, such as $\mathsf{AC}^0[\oplus]$. In this section, we explain why the GIMSS-LRCC is insecure against $\mathsf{AC}^0[\oplus]$-leakage, and show that making it $\mathsf{AC}^0[\oplus]$-leakage-resilient would entail making the ISW-LRCC leakage-resilient which, as discussed in Section 3 we do not currently know how to do.

To explain the limits of the GIMSS-LRCC, we first need to explain how it works. Goyal et al. [GIM+16] design information-theoretically secure protocols in the OT-hybrid model that allow a user, aided by a pair of "honest-but-curious" servers, to compute a function of her input while preserving the privacy of the input and output even under Bounded Communication Leakage (BCL) on the internals of the servers. Genkin et al. [GIW17] observed that one can obtain a BCL-secure LRCC by implementing the server programs as circuits, and implementing the OT calls using constant-sized sub-circuits. At a high level, given a circuit C, its leakage-resilient version is constructed in three steps, as we now describe.

First, C is compiled into a "parity-resilient" circuit $C_{\oplus}$, which emulates C on encoded inputs and resists leakage from the class of all parity function (namely, all functions that output the parity of a subset of wires). More precisely, the inputs to $C_{\oplus}$ are *small-bias* encodings of constant length (over $\mathbb{F}_2$) of C's inputs.[18] Roughly, such encodings fool linear distinguishers in the sense that a linear distinguisher obtains only a small advantage in distinguishing between encodings of 0 and 1. (We do not provide a formal definition of small-bias encoding because the exact properties of the encoding are unimportant to us.) $C_{\oplus}$ uses a constant-sized gadget $\mathcal{G}$ to emulate C's operation over the small-bias encodings. We elaborate on the construction of the parity-resilient $C_{\oplus}$ below.

The second step is to use a GMW-style 2-party protocol π to emulate $C_{\oplus}$, gate-by-gate, on additive secret shares of the input. This protocol uses an oracle to the functionality computed by the gadget $\mathcal{G}$, and outputs additive secret shares of the output. The third and final step is to replace each oracle call to $\mathcal{G}$ with a constant number of OT calls, and convert the resultant 2-party protocol into a boolean circuit (in which OT calls are implemented using a constant number of gates).

Let C' denote the resultant circuit, which operates on encoded inputs, and returns encoded

[18] More specifically, every input of x is individually encoded using a bit string of length c for some constant c.

outputs. Specifically, the encoding of a bit first encodes the bit using the small-bias encoding, then additively secret shares these encodings into two shares.

Notice that the final two steps in the construction provide no protection against $\mathsf{AC}^0[\oplus]$-leakage. That is, C' resists $\mathsf{AC}^0[\oplus]$ leakage only if $C_\oplus$ does. Intuitively, this is because $\mathsf{AC}^0[\oplus]$ circuits can decode the small-bias encoding and the secret sharing and thus recover the original wire values of $C_\oplus$. More formally, assume $C_\oplus$ is not $\mathsf{AC}^0[\oplus]$-leakage-secure, and we show how to attack C' using a leakage function in $\mathsf{AC}^0[\oplus]$. Let $\ell \in \mathsf{AC}^0[\oplus]$ be a leakage function that takes as input the wire values of $C_\oplus$, and breaks $\mathsf{AC}^0[\oplus]$-leakage-security of $C_\oplus$. Consider the function $\ell' \in \mathsf{AC}^0[\oplus]$ that takes as input the wire values of C', uses them to generate the wire values of $C_\oplus$ in the underlying execution, and applies ℓ to the wire values. More specifically, each wire w of $C_\oplus$ is generated as follows. If w is an input or output of $\mathcal{G}$ then its additive secret sharing appears in ℓ''s input, and the value can be reconstructed from the shares using an $\mathsf{AC}^0[\oplus]$ circuit. Otherwise, w is an internal wire of $\mathcal{G}$, in which case an $\mathsf{AC}^0[\oplus]$ circuit can reconstruct the inputs and outputs of $\mathcal{G}$ from their additive secret shares, and then use them to generate the entire internal wire values of $\mathcal{G}$ (because $\mathcal{G}$ is a constant-sized circuit). Thus, $\ell' \in \mathsf{AC}^0[\oplus]$, and obtains the same distinguishing advantage as ℓ, so C' is not $\mathsf{AC}^0[\oplus]$-leakage-resilient.

Consequently, to obtain an $\mathsf{AC}^0[\oplus]$-leakage-resilient C', the underlying parity-resilient circuit $C_\oplus$ must be $\mathsf{AC}^0[\oplus]$-leakage-resilient. We therefore turn our attention to the construction of parity-resilient circuits, which are constructed in [GIM+16] in two steps. First, C is replaced with its private counterpart C_p, namely a probing-secure circuit that emulates C over encodings. Second, every wire w of C_p is replaced with a bundle carrying a constant-length small-bias encoding of w, and the gates of C_p are replaced with a constant-sized gadget $\mathcal{G}_\oplus$ emulating the gate operation over the small-bias encodings. (The fact that the encodings and $\mathcal{G}_\oplus$ have constant size is crucial for obtaining leakage-resilience, since applying constant-sized computations to the wires of the circuits doesn't affect BCL leakage-resilience.)

Since the small-bias encodings, and $\mathcal{G}_\oplus$, have constant size, a similar argument to the one used above shows that $C_\oplus$ is $\mathsf{AC}^0[\oplus]$-leakage-resilient only if C_p is. Thus, in general, proving that the GIMSS-LRCC is $\mathsf{AC}^0[\oplus]$-leakage-resilient reduces to constructing an $\mathsf{AC}^0[\oplus]$-leakage-resilient circuit C_p. Moreover, the GIMSS-LRCC compiler instantiates the underlying private circuit with the ISW-LRCC which, as discussed in Section 1.2.1, is *not* $\mathsf{AC}^0[\oplus]$-leakage-resilient, so the actual LRCC constructed in [GIM+16] is *not* $\mathsf{AC}^0[\oplus]$-leakage-resilient. In summary, it seems that using the GIMSS-LRCC gives no advantage (in the context of constructing ZK-PCPs) over directly constructing a SAT-respecting $\mathsf{AC}^0[\oplus]$-leakage-resilient circuit compiler.

6 Extensions of the MV-LRCC [MV13]

In this section we describe approaches towards making the MV-LRCC of [MV13] SAT-respecting. As noted in Section 1.2.2, this entails checking that (1) the input encodings encode bits, and (2) the structured randomness used in the construction is "valid", for an appropriate definition of "valid". We describe a method of checking (1) (as discussed in Section 1.2.2, we do not currently know how to check (2)). We first describe the MV-LRCC.

The MV-LRCC. The compiler is designed over the alternating group A_5, namely the group of even permutations of a set of size 5, where group operations are implemented using boolean circuits of constant size. For simplicity of the exposition, we follow [MV13] and describe the construction using A_5 operations. The MV-LRCC uses the following special elements of A_5 (using cycle notation): $\alpha = (1,2,3,4,5)\,, \beta = (1,4,2,3,5)$ and $\gamma = (1,2,3,5,4)$. These elements satisfy

$\beta = \gamma\alpha\gamma^{-1}$ and $\beta\alpha\beta\alpha^{-1}\beta^{-1}\beta^{-1} = \alpha$. We use id to denote the identity element of A_5.

We first describe the encoding scheme used in the construction.

Definition 6.1 (Group encoding scheme)**.** Let $t \in \mathbb{N}$ be a security parameter. The *group encoding scheme* $\mathsf{E}_g = (\mathsf{Enc}_g, \mathsf{Dec}_g)$ over A_5 is a parameterized encoding scheme defined as follows:

- For every input $x \in \{0,1\}$, and security parameter $t \in \mathbb{N}$, $\mathsf{Enc}_g\left(x, 1^t\right) = (x_1, \ldots, x_t) \in A_5^t$, where $x_1, \ldots, x_t$ are random subject to the constraint that $\prod_{i=1}^t x_i = \mathsf{id}$ if $x = 0$, otherwise $\prod_{i=1}^t x_i = \alpha$.[19]
- For every $t \in \mathbb{N}$, and every $(x_1, \ldots, x_t) \in A_5^t$, $\mathsf{Dec}_g\left(x_1, \ldots, x_t\right)$ computes $\prod_{i=1}^t x_i$, outputs 0 if the product is id, otherwise it outputs 1.

Following [MV13], we assume without loss of generality that the circuit $C : \{0,1\}^n \to \{0,1\}$ to be compiled consists solely of NAND gates. Miles and Viola [MV13] design a NAND gadget that emulates NAND gates, as well as a randomization gadget $\mathcal{G}_{\mathsf{r}}$ that is used to "refresh" the encodings at the output of NAND gadgets (replacing them with "fresh" encodings of the same values). We note that the NAND gadget is a straight-forward emulation of the NAND gate over A_5-encodings, and it provides no leakage-resilience guarantee. The leakage-resilience of the circuit is obtained by applying the randomization gadget $\mathcal{G}_{\mathsf{r}}$ to the encoding at the output of each NAND gadget. Miles and Viola [MV13] rely on the fact that $\mathcal{G}_{\mathsf{r}}$ outputs a fresh encoding (of the same value) to prove correctness and leakage resilience.

Our input-checking gadget $\mathcal{G}_{\mathsf{inp}}$. The input-checking gadget we construct verifies that its input is a valid encoding of a bit. We follow the technique of [DPS12], who check conditions of the form $\prod_i x_i \in \{\mathsf{id}, \alpha\}$ for group elements x_i through a series of equations. (We note, however, that since [DPS12] use equations over S_5, whereas we work over the group A_5, we need to use different equations.) Our construction relies on the following fact.

Fact 6.2. *Let $E_1, E_2 : A_5 \to A_5$ be defined as $E_1(g) = g \cdot \alpha \cdot g^{-1} \cdot \alpha^{-1}$ and $E_2(g) = (g_1 \cdot g)^3$, where $g_1 = (5,4,3)$. Then for every $g \in A_5$, $E_1(g) = E_2(g) = \mathsf{id}$ if and only if $g \in \{\mathsf{id}, \alpha\}$.*

Proof. For every $g \in A_5$, $E_1(g) = \mathsf{id}$ if and only if g is a power of α, i.e. if and only if $g \in \left\{\mathsf{id}, \alpha, \alpha^2, \alpha^3, \alpha^4\right\}$ (since α has order 5). Therefore, it suffices to verify that for $g \in \left\{\mathsf{id}, \alpha, \alpha^2, \alpha^3, \alpha^4\right\}$, $E_2(g) = \mathsf{id}$ if and only if $g \in \{\mathsf{id}, \alpha\}$. This holds since $g \cdot \mathsf{id} = g$ has order 3 and so does $g \cdot \alpha$, while $g \cdot \alpha^2, g \cdot \alpha^3, g \cdot \alpha^4$ have orders 5,2,5 respectively, which do not divide 3. □

Construction 6.3 (Input-checking gadget)**.** The gadget takes as input $\widehat{x} \in A_5^t$, and outputs $\widehat{z} = \left(\widehat{z}^{(1)}, \widehat{z}^{(2)}\right) \in A_5^{2t}$ such that $\mathsf{Dec}_g\left(\widehat{z}^{(1)}, 1^t\right) = \mathsf{Dec}_g\left(\widehat{z}^{(2)}, 1^t\right) = \mathsf{id}$ if and only if $x \in \mathsf{Enc}_g\left(0, 1^t\right) \cup \mathsf{Enc}_g\left(1, 1^t\right)$. It operates as follows.

1. Set $\widehat{u} := (x_1, x_2, ..., x_{t-1}, x_t \cdot \alpha)$.
2. Compute $\widehat{y} \in (A_5)^t$ as $\widehat{y} := \left(u_t^{-1}, ..., u_1^{-1} \cdot \alpha^{-1}\right)$.
3. Compute $\widehat{z}' \in (A_5)^{2t}$ as $\widehat{z} = (\widehat{u}, \widehat{y})$. (This gives $\prod_i \widehat{z}'_i = \left(\prod_i x_i\right) \cdot \alpha \cdot \left(\prod_i x_i\right)^{-1} \cdot \alpha^{-1}$).
4. Compute $\widehat{z}^{(1)} \in (A_5)^t$ as $\widehat{z}^{(1)} := \left(z'_1 \cdot z'_2, z'_3 \cdot z'_4, ..., z'_{2t-1} \cdot z'_{2t}\right)$.

[19] Syntactically, Enc_g is not an encoding procedure because its input and output alphabets are different. However, we use the A_5 notation for clarity, where in fact each group element is represented using a bit-string, so the input and output alphabets of Enc_g are both $\{0,1\}$.

5. Compute $\widehat{u} \in (A_5)^t$ as $\widehat{u} := (g_1 \cdot x_1, x_2, ..., x_t)$.
6. Compute $\widehat{u}' \in (A_5)^{3t}$ as $\widehat{u}' = (\widehat{u}, \widehat{u}, \widehat{u})$. (This gives $\prod_i \widehat{u}'_i = (g_1 \prod_i x_i)^3$.)
7. Compute $\widehat{z}^{(2)} \in (A_5)^t$ as $\widehat{z}^{(2)} := \left(u'_1 \cdot u'_2 \cdot u'_3, u'_4, ..., u'_{2t-2} \cdot u'_{2t-1} \cdot u'_{2t}\right)$.
8. Compute and output $\widehat{z} := \left(\widehat{z}^{(1)}, \widehat{z}^{(2)}\right)$.

We note that the correctness of the MV-LRCC relies on the fact that for every bundle in the leakage-resilient circuit, the encoding $\widehat{x} = (x_1, \ldots, x_t)$ carried on the bundle satisfies the following *multiplication property*: $\prod_{i=1}^{t} x_i \in \{\mathsf{id}, \alpha\}$. This property is not satisfied by our input-checking gadget $\mathcal{G}_{\mathsf{inp}}$. However, this will not pose a problem in the construction because, as described below, the outputs of $\mathcal{G}_{\mathsf{inp}}$ will only be used to verify input consistency, and the leakage-resilient circuit will contain a special decoding sub-circuit for decoding the outputs of $\mathcal{G}_{\mathsf{inp}}$, that will deal with encodings that do not satisfy the multiplication property.

Input-Checking MV-LRCC variant. We now describe our variant of the MV-LRCC. Roughly, the leakage-resilient circuit $\widehat{C}$ it outputs emulates the leakage-resilient circuit of MV-LRCC, but includes additional checks (using the input-checking gadget $\mathcal{G}_{\mathsf{inp}}$ of Construction 6.3) to verify that its inputs encode bits. The encodings at the outputs of the check gadgets are refreshed using the randomization gadget $\mathcal{G}_{\mathsf{r}}$ (similar to how it is used in [MV13] to refresh the encodings at the output of NAND gadgets).

Construction 6.4 (Input-checking variant of the MC-LRCC)**.** Let $t, t_{\text{in}} \in \mathbb{N}$ be security parameters, and $n \in \mathbb{N}$ be an input length parameter. Let $\mathsf{E}_g = (\mathsf{Enc}_g, \mathsf{Dec}_g)$ be the encoding scheme of Definition 6.1. The LRCC is defined as follows.

- The encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ operates as follows:
  - for every $x \in \{0,1\}^n$, $\mathsf{Enc}\left(x, 1^t, 1^{t_{\text{in}}}\right) = \left(\mathsf{Enc}_g\left(x, 1^t\right), \vec{r}\right)$, where $\vec{r} = \left(\vec{r}^{(1)}, \vec{l}^{(1)}, \ldots, \vec{r}^{(M)}, \vec{l}^{(M)}\right) \in \left(A_5^{2t}\right)^M$ for $M = n + t_{\text{in}}$, and for every $1 \leq i \leq M$, $\vec{r}^{(i)}, \vec{l}^{(i)} \in A_5^t$ are random subject to the constraint $\prod_{j=1}^{t} r_j^{(i)} \prod_{j=1}^{t} l_j^{(i)} = \mathsf{id}$. (Intuitively, $\vec{r}$ provides the randomness needed for the $\mathcal{G}_{\mathsf{r}}$ gadgets in the compiled circuit.)
  - $\mathsf{Dec}\left(\left(\widehat{x}, \vec{r}\right), 1^t, 1^{t_{\text{in}}}\right) = \mathsf{Dec}_g\left(x, 1^t\right)$.
- The compiler, given a circuit $C : \{0,1\}^n \to \{0,1\}$ with s wires, outputs the circuit $\widehat{C} : \{0,1\}^{n \cdot l} \times \{0,1\}^{(s+2n) \cdot 2t \cdot l} \to \{0,1\}$ (where l denotes the length of the bit-representation of elements in A_5, see Remark 6.5 below) constructed as follows:
  - Let $\widehat{C'}$ denote the circuit obtained from C by the MV-LRCC of [MV13]. Concretely, every NAND gate with m output wires is replaced with a NAND gadget, followed by m parallel randomization gadgets $\mathcal{G}_r$ (the output bundle of the NAND gadget is fed as input to each of these randomization gadgets). Moreover, for the randomization gadget $\mathcal{G}$ following the (single) output wire of the NAND gate computing the output of C, the output $\widehat{o} = (o_1, \ldots, o_t)$ of $\mathcal{G}$ is fed into a decoding sub-circuit that simply computes $\prod_{i=1}^{t} o_i$ in some arbitrary way. We note that the random field elements used by the randomization gadgets are taken from $\widehat{C}$'s second input.
  - Let $\widehat{C''} : \{0,1\}^{n \cdot l} \times \{0,1\}^{2n \cdot 2t \cdot l} \to \{0,1\}$ be the circuit that on input $\left(\widehat{x}^{(1)}, \ldots, \widehat{x}^{(n)}\right)$ and masking inputs $\left(\vec{r}^{(1)}, \vec{r}^{(1)\prime}, \ldots, \vec{r}^{(n)}, \vec{r}^{(n)\prime}\right)$ operates as follows.

* For every $1 \leq i \leq n$, computes $\left(\widehat{y}^{(i)}, \widehat{y}^{(i)\prime}\right) := \mathcal{G}_{\mathsf{inp}}\left(\widehat{x}^{(i)}\right)$.
* For every $1 \leq i \leq n$, computes $\widehat{z}^{(i)} := \mathcal{G}_{\mathsf{r}}\left(\widehat{y}^{(i)}; \vec{r}^{(i)}\right)$, and $\widehat{z}^{(i)\prime} := \mathcal{G}_{\mathsf{r}}\left(\widehat{y}^{(i)\prime}; \vec{r}^{(i)\prime}\right)$.
* For every $1 \leq i \leq n$, uses any correct tree of multiplication gates to compute $w^{(i)} := \prod_{j=1}^{t} z_j^{(i)}$ and $w^{(i)\prime} := \prod_{j=1}^{t} z_j^{(i)\prime}$.
* Let $\widetilde{C} : \{0,1\}^l \to \{0,1\}$ be any (constant-sized) correct circuit that outputs 1 if its input is id (represented as a bit string), otherwise it outputs 0. Then output $\wedge_{i=1}^{n}\left(\widetilde{C}\left(w^{(i)}\right) \wedge \widetilde{C}\left(w^{(i)\prime}\right)\right)$.

– output the circuit $\widehat{C}$ such that

$$\widehat{C}\left(\widehat{x_1}, \ldots, \widehat{x_n}, \vec{r_1}, \vec{r_1}' \ldots, \vec{r_n}, \vec{r_n}', \vec{r}\right) = \widehat{C'}\left(\widehat{x_1}, \ldots, \widehat{x_n}, \vec{r}\right) \cdot \widehat{C''}\left(\widehat{x}_1, \ldots, \widehat{x_n}, \vec{r_1}, \vec{r_1}', \ldots, \vec{r_n}, \vec{r_n}'\right)$$

(Notice that the output is $\widehat{C'}\left(\widehat{x_1}, \ldots, \widehat{x_n}, \vec{r}\right)$ if $\widehat{C''}\left(\widehat{x}_1, \ldots, \widehat{x_n}, \vec{r_1}, \vec{r_1}', \ldots, \vec{r_n}, \vec{r_n}'\right) = 1$, otherwise it is 0.)

Remark 6.5 (representing group elements as bit strings)**.** Construction 6.4 represents elements of A_5 as l-bit strings. We note that not all l-bit strings correspond to an element of A_5, so to guarantee SAT-respecting, $\widehat{C}$ should also check that each length-l bit string given as part of its input or masking input corresponds to a valid element of A_5. Since l is constant, that can be done using a constant-sized boolean circuit, and will not cause any loss in leakage-resilience.

We now show that if the masking inputs used in $\widehat{C}$ are well-formed (i.e., of the form $\left(\vec{r}, \vec{l}\right)$ such that $\prod_i r_i \prod_i l_i = \mathsf{id}$) then $\widehat{C}$ is satisfiable only if C is satisfiable.

Lemma 6.6. *Assume all masking inputs $\left(\vec{r}, \vec{l}\right)$ used in the circuit $\widehat{C}$ of Construction 6.4 satisfy $\prod_i r_i \prod_i l_i = \mathsf{id}$, then for every input $\left(\widehat{x_1}, \ldots, \widehat{x_n}\right)$ that satisfy $\widehat{C}$, there exists $(x_1, \ldots, x_n) \in \{0,1\}^n$ such that $C(x_1, \ldots, x_n) = 1$.*

Proof. First, notice that by the definition of $\widehat{C}$, if it outputs 1 then for every $1 \leq i \leq n$, $w^{(i)} = w^{(i)\prime} = \mathsf{id}$, and moreover $\widehat{C'}\left(\widehat{x}^{(1)}, \ldots, \widehat{x}^{(n)}\right) = 1$.

Second, conditioned on all masking inputs being well-formed, for every $\widehat{y}^{(i)}, \widehat{y}^{(i)\prime}$ and $\widehat{z}^{(i)}, \widehat{z}^{(i)\prime}$ computed in $\widehat{C''}$ it holds that $\prod_{j=1}^{t} y_j^{(i)} = \prod_{j=1}^{t} z_j^{(i)}$ and $\prod_{j=1}^{t} y_j^{(i)\prime} = \prod_{j=1}^{t} z_j^{(i)\prime}$ (from the correctness of the randomization gadget when it uses well-formed masking inputs). Thus, by Fact 6.2, for every $1 \leq i \leq n$, $\prod_{j=1}^{t} \widehat{y}^{(i)} = \prod_{j=1}^{t} \widehat{y}^{(i)\prime} = \mathsf{id}$ implies that $\widehat{x}^{(i)}$ is a valid encoding of some $x_i \in \{0,1\}$.

Finally, since the masking inputs used in $\widehat{C}$ are well-formed then $\widehat{C'}$ on input $\widehat{x}^{(1)}, \ldots, \widehat{x}^{(n)}$ emulates C on input $x_1, \ldots, x_n$, so $C(x_1, \ldots, x_n) = 1$. □

Next, we claim that Construction 6.4 is as leakage-resilient as the MV-LRCC of [MV13].

Lemma 6.7. *Let $n \in \mathbb{N}$ be an input parameter, and $\mathcal{L}$ be a leakage class. For a circuit C, let $\widehat{C}_{\mathsf{MV}}$ denote its leakage-resilient variant obtained through the LRCC of [MV13]. Then for every circuit $C : \{0,1\}^n \to \{0,1\}$, if $\widehat{C}_{\mathsf{MV}}$ is $(\mathcal{L}, \epsilon)$-leakage-resilient for some ϵ, then the circuit $\widehat{C}$ of Construction 6.4 is $(\mathcal{L}, \epsilon(1+2n))$-leakage-resilient.*

The proof of Lemma 6.7 is similar to the leakage-resilience proof of [MV13]. At a high level, the leakage-resilience of the original MV-LRCC, and of our modified version in Construction 6.4, follows from the randomization gadget. The randomization gadget $\mathcal{G}_{\mathsf{r}}$ of [MV13] is both *locally reconstructible* and *rerandomizing*. A gadget is rerandomizing if its output is a fresh encoding of

the value encoded in its input. Local reconstruction means that given the input and output encodings $(\widehat{x}, \widehat{y})$ of the gadget, there exists a "weak" simulator (specifically, whose computational power is much weaker than the leakage functions) that can generate the entire wire values of the gadget, and the simulated wires are indistinguishable from the actual wire values of the gadget when using random well-formed masking inputs, conditioned on its input and output being $\widehat{x}, \widehat{y}$, respectively. (See Definitions 7.8 and 7.9 in Section 7.1 for the formal definitions.) The randomization gadgets in [MV13] randomize the outputs of the NAND gadgets and thus effectively "break" the correlations between consecutive NAND gates in the circuit (indeed, in the leakage-resilient circuit the corresponding NAND gadgets will be separated by a randomization gadget). This allows the simulator to arbitrarily fix the outputs of all NAND gadgets, and later "account" for these values consistently using the local reconstruction of $\mathcal{G}_{\mathsf{r}}$.

Proof of Lemma 6.7 (sketch). Let $\mathcal{L}$ be a leakage class against which the LRCC of [MV13] is secure, and let Sim' be the corresponding simulator. We describe the simulator Sim that simulates $\mathcal{L}$ leakage on the circuit $\widehat{C}$ of Construction 6.4, by simulating the entire wire values of $\widehat{C}$. Sim obtains the output $y = C(x_1, \ldots, x_n)$. It picks a random encoding of y for the output of $\widehat{C'}$, and for every $1 \leq i \leq n$ it picks $\widehat{z}^{(i)}, \widehat{z}^{(i)\prime}$ at random subject to $\prod_{j=1}^{t} z_j^{(i)} = \prod_{j=1}^{t} z_j^{(i)} = \mathsf{id}$. Then, it honestly computes the internal wires of the decoder that decodes the output of $\widehat{C'}$, the wires in the decoding of each $\widehat{z}^{(i)}, \widehat{z}^{(i)\prime}$, and the wires of $\widetilde{C}$.

Next, for every $1 \leq i \leq n$, Sim chooses uniformly at random the values of the bundles encoding x_i, and the output of every randomization gadget (except for the gadgets following $\widehat{C''}$'s output gate, and the randomization gadgets used in $\widehat{C''}$ - the outputs of these gadgets have already been determined during the simulation). Then, Sim computes the values on the internal wires and the output wires of every NAND gadget and every $\mathcal{G}_{\mathsf{inp}}$ gadget, by honestly evaluating the gadget. (This is possible because the input to these gadgets is always the output of a randomization gadget, or an input encoding, so the values of all the input wires to these gadgets have already been determined.) This determines the inputs and outputs of all randomization gadgets. Finally, Sim uses the local reconstructor of $\mathcal{G}_{\mathsf{r}}$ (whose existence is guaranteed by [MV13, Lemma 2.1]) to simulate the internal wires of $\mathcal{G}_{\mathsf{r}}$.

The remainder of the proof follows identically to [MV13, Theorem 1.4] (and is therefore omitted). We only explain why the distinguishing advantage of our construction is larger. The real and simulated distributions over the wire values are proven indistinguishable using a hybrid argument that shows they are both indistinguishable from a hybrid distribution in which all wires, except the internal wires of $\mathcal{G}_{\mathsf{r}}$ gadgets, are distributed identically to the real wire values, and the internals of $\mathsf{r}_{\mathcal{G}}$ are generated using the local reconstructor.

The indistinguishability of the real and hybrid distributions is proven by performing a hybrid over the $\mathcal{G}_{\mathsf{r}}$ gadgets, replacing the internals of one gadget at a time, and reducing indistinguishability to the local reconstruction of $\mathcal{G}_{\mathsf{r}}$. Thus, if the reconstruction error of the local reconstructor of $\mathcal{G}_{\mathsf{r}}$ is ϵ, then the real and hybrid distributions are $\epsilon \cdot |C|$-close, since $\widehat{C}_{\mathsf{MV}}$ contains at most $|C|$ randomization gadgets. For us, this becomes $\epsilon \cdot (|C| + 2n)$, since our input-checking gadgets use a pair or randomization gadgets per input bit.

The indistinguishability of the hybrid and simulated distributions is proven by a hybrid over the input bundles and the bundles at the output of $\mathcal{G}_{\mathsf{r}}$ gadgets, replacing these bundles one at a time, and reducing indistinguishability to the leakage-resilience of the encoding scheme. Thus, if the underlying encoding scheme is ϵ-leakage resilient, then the simulated and hybrid distributions are $\epsilon \cdot |C|$-close, since there are at most $|C|$ such encodings. For us, this becomes $\epsilon \cdot (|C| + 2n)$, since again our input-checking gadgets introduce two additional encodings per input bit. □

7 Non-Adaptively Verifiable WI-PCPs of Proximity

In this section we describe our construction of WI-PCPs of Proximity (WI-PCPPs) that can be non-adaptively verified. More specifically, we reduce the task of constructing such WI-PCPPs to the task of designing a code with "good" distance that resists $\mathsf{AC}^0[\oplus]$ leakage.

The high-level idea is to generalize the "LRCC-to-WI-PCP" transformation of [IWY16] to also apply to PCPPs. As noted in Section 1.3, this turns out to be highly non-trivial, and requires overcoming several obstacles. Recall that Ishai et a. [IWY16] construct WI-PCPs for an NP language L with a corresponding NP relation $\mathcal{R}_L$ by reducing L to the NP-complete language of circuit satisfiability. Specifically, if C is the verification circuit of $\mathcal{R}_L$, then to prove a claim of the form "$x \in L$" the prover and verifier operates a follows. First, they hard-wire x into C, obtaining a circuit C_x, which is satisfiable if and only if $x \in L$. Then, they use a SAT-respecting (relaxed) LRCC to obtain the leakage-resilient version $\widehat{C_x}$ of x, and the prover then uses a standard PCP for circuit satisfiability to prove that $\widehat{C_x}$ is satisfiable. A successful proof implies, by the soundness of the PCP, that with high probability $\widehat{C_x}$ is indeed satisfiable, which by the SAT-respecting property of the LRCC implies that C_x is satisfiable, i.e., that $x \in L$.

The first obstacle we face in generalizing this construction to work for PCPPs is that the PCPP verifier does not know x, so we cannot hard-wire x into C. Thus, we have the compiled circuit take as input (encodings of) both the input and witness. Recall from Section 1.3 that soundness then requires that the verifier check consistency of the encoded input $\widehat{x}$ (used in $\widehat{C}$) with its own implicit input x. More specifically, since [IWY16] encode bit-strings by encoding each bit separately, it suffices to show how to check that $\widehat{b}$ encodes a given bit b, and we do so by providing the verifier with an additional (non-ZK) PCPP proof that $\widehat{b}$ encodes b. Since an accepted proof only guarantees that $\widehat{b}$ is close to an encoding of b, to rule-out the possibility that $\widehat{b}$ actually encodes $1-b$ the encoding much have "good" distance.

An encoding scheme with good distance (in fact, with any non-trivial distance) is not onto, meaning not all vectors of the "right" length are valid encodings. The SAT-respecting property of the LRCC of [IWY16] crucially relies on the underlying encoding being onto. Therefore, we cannot provide x to $\widehat{C}$ in encoded form. This immediately implies that we cannot use the SAT-respecting LRCC of [IWY16] "as-is", since its leakage-resilience relied on the fact that its inputs were encoded.

Our first observation is that the SAT-respecting LRCC of [IWY16] has the following property. It can take its inputs in two parts (x, w) where x is given in the clear, and w is encoded. (We use the notation x, w here because in the final construction the input x will constitute the first, un-encoded, part of the input; and the witness will be the second, encoded, part of the input.) In this case, any leakage from $\mathcal{L}$ (where $\mathcal{L}$ is the leakage class which the original LRCC of [IWY16] resists, e.g., $\mathsf{AC}^0[\oplus]$ leakage) applied to the internal wires of the leakage-resilient circuit can be simulated by a simulator that is given x and the output $C(x, w)$. Indeed, the (inefficient) simulator of [IWY16], given $C(x, w)$, finds an input (x', w') such that $C(x, w) = C(x', w')$, and uses it to generate the entire wire values of the leakage-resilient version $\widehat{C}$. The simulator can use *any* input (x', w') satisfying $C(x, w) = C(x', w')$. In particular, given x, the simulator can find a w'' such that $C(x, w) = C(x, w'')$ (such a w'' always exists) and use (x, w'') in the simulation. This is formalized in Lemma 7.16 below.

Thus, we can replace the circuit C with the circuit C' that takes as input an *encoding* $\mathsf{Enc}(x)$ *of* x, and w, decodes x, and then emulates C (C' is described in Definition 7.18 below). Applying the modified LRCC described above to C' gives a circuit $\widehat{C'}$ such that leakage from $\mathcal{L}$ on the internals of $\widehat{C'}$ can be simulated given only $\mathsf{Enc}(x)$.

This modified construction is still insufficient to obtain a WI-PCPP since no security is guaranteed for the first part of the input which corresponds to the input x of the PCPP, whereas in

a ZK-PCPP most of x *should* remain private. We solve this issue in two steps. First, we use a *leakage-resilient* encoding scheme Enc to encode the bits of x. This guarantees that leakage on the inputs of C' reveals no information about the underlying input x. As noted above, this is the "missing piece of the puzzle" which we currently do not know how to construct.

Second, we need the underlying LRCC to guarantee some sort of privacy even for the *un-encoded* part of its input, roughly, that leakage on the internals of the circuit can be simulated given only leakage (from a related leakage class) on its first input x. When using a leakage-resilient encoding of x, this would guarantee x's privacy. Of course, such a guarantee is impossible in general since the circuit can perform arbitrary computations over its inputs which can "help" the leakage function. (For example, if the leakage function simply probes few bits, but the circuit computes the XOR of all bits of x, then by probing the outcome of this internal computation the leakage outputs a value that cannot be simulated given any subset of bits of x, let alone few of them.) To resolve the issue we resort to an "average-case" leakage-resilience property which is nonetheless sufficient for our needs. Roughly, average-case $\mathcal{L}$-leakage-resilience with relation to encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ guarantees that as long as the first input of the circuit consists of *random* encodings, according to E, of x's bits, then leakage from $\mathcal{L}$ on the internals of the circuit (including its first input) can be simulated given only its output and a small subset of input encodings, and the joint distribution of the simulated leakage and this subset of input encodings is indistinguishable from the real-world distribution. (See Definition 7.19 for the formal definition.) We note that this is sufficient since in the WI-PCPP construction described below, the prover will provide random encodings of the bits of x to $\widehat{C'}$.

The final issue is that the WI-PCP of [IWY16] uses the PCP of Arora and Safra [AS92], whereas to obtain a WI-PCPP we need a PCPP in a similar complexity class. We observe that the construction described in [AS92] can in fact give a PCPP, if one encodes the input oracle as part of the proof.

Given these observations and building blocks, the WI-PCPP is then obtained in the following way. The prover and verifier both construct the leakage-resilient version $\widehat{C'}$ of the circuit C' described above. Recall that $\widehat{C'}$'s input is divided into two parts, one is given under some leakage-resilient encoding with good distance, and the other is given under the (onto) encoding used by the LRCC. The prover then generates random encodings of x, w (each bit x_i is encoded separately into an encoding $\mathsf{Enc}(x_i)$) and uses these encodings to generate the entire wire values $\mathcal{W}$ of $\widehat{C'}$. The prover then uses $\mathcal{W}$ to generate an AS-PCPP π for the claim "$\widehat{C'}$ is satisfiable". Notice that this proof does not yet guarantee that the satisfying input is consistent with x. Next, for every input bit x_i of x, the prover generates a (standard, non-WI) PCPP $\pi^{(i)}$ for the claim "$\mathsf{Enc}(x_i)$ encodes x_i". The WI-PCPP consists of π, $\pi^{(1)}, \ldots, \pi^{(n)}$, and the encodings $\mathsf{Enc}(x_,), \ldots, \mathsf{Enc}(x_n)$. The verifier runs the AS-PCPP to verify π, then picks $O(\lambda)$ (where λ is the security parameter) random indices $i \in [n]$, and for each of them uses $\pi^{(i)}$ to verify that $\mathsf{Enc}(x_i)$ is a valid encoding of the bit x_i reported in its input oracle. The verifier accepts if all tests pass.

This intuition is formalized in the next sections, where we first establish needed notations and definitions.

7.1 Notations and Definitions.

ZK- and WI-PCPs of Proximity. A ZK-PCP of Proximity (ZK-PCPP) is a generalization of a ZK-PCP in which the verifier queries only few input bits. In particular, the verifier can only check whether the input is "close" or "far" from a given language L. Thus, we first formalize the notion of "closeness" which we use. Specifically, we measure distance in terms of the relative Hamming distance: for $x, y \in \{0,1\}^n$, the relative Hamming distance $\Delta(x, y)$ is defined as

$\Delta(x,y) := \frac{|\{i\in[n] \,:\, x_i=y_i\}|}{n}$. For a subset $S \subseteq \{0,1\}^n$, and a distance parameter $\delta \in (0,1)$, we say that x is *δ-close to S* if $\Delta(x,y) \leq \delta$ for some $y \in S$, otherwise x is *δ-far from S*. For a language $L \subseteq \{0,1\}^*$, we say that $x \in \{0,1\}^n$ is *δ-close to L* if x is δ-close to $L \cap \{0,1\}^n$, otherwise x is δ-far from L.

Definition 7.1 (ZK- and WI-PCPPs, [IW14])**.** A probabilistic proof system (P,V) is a *Zero-Knowledge Probabilistically Checkable Proof of Proximity (ZK-PCPP)* system for an NP-relation $\mathcal{R}_L = \mathcal{R}_L(x,w)$, if the following holds.

- *Syntax.* The prover P has input $\epsilon, \delta, 1^{q^*}, x, w$, and outputs a proof π for (x,w). The verifier V has input $\epsilon, \delta, q^*, |x|$, and oracle access to x, π, and outputs either acc or rej.

  We associate with P, V as above the same efficiency measures as in a ZK-PCP system (Definition 2.5), except that all measures also depend on δ.

- *Semantics.* (P,V) should have the following properties.

  - *Completeness.* For every $(x,w) \in \mathcal{R}$ and every proof $\pi \in P\left(\epsilon, \delta, 1^{q^*}, x, w\right)$, $\Pr\left[V^{x,\pi}(\epsilon,\delta,q^*,|x|) = \mathsf{acc}\right] = 1$, where the probability is over the randomness of V.
  - *Soundness.* For every x which is δ-far from $L_{\mathcal{R}}$, and every π^*, $\Pr\left[V^{\pi^*,x}(\epsilon,\delta,q^*,|x|) = \mathsf{acc}\right] \leq \epsilon$.
  - *(ϵ, q^*)-Zero-Knowledge (ZK).* For every (possibly malicious) verifier V^* that reads at most q^* symbols from his input and proof oracles there exists a PPT simulator Sim such that for every $(x,w) \in \mathcal{R}_L$, $\mathsf{SD}\left(\left(\mathsf{V}_{V^*,P}(\epsilon,\delta,q^*,x,w), q_V\right), \left(\mathsf{Sim}^x\left(\epsilon,\delta,1^{q^*},|x|\right), q_S\right)\right) \leq \epsilon$ where $\mathsf{V}_{V^*,P}(\epsilon,\delta,q^*,x,w)$ denotes the view of V^* when given oracle access to x, and a proof π generated by $P\left(\epsilon,\delta,1^{q^*},x,w\right)$, q_V denotes the number of symbols V^* reads from x,π, and q_S denotes the number of bits Sim reads from x.

    If the above holds with an *inefficient* simulator, we say the PCPP system is (ϵ, q^*)-*witness indistinguishable*.

Notation 7.2. We use $\mathsf{PCPP}[r, q, \epsilon, \delta, \ell]$ to denote the class of NP-languages that admit an NP-relation $\mathcal{R}_L$ with a (non-ZK) PCPP in which the prover outputs proofs of length ℓ, the verifier tosses $O(r)$ coins, queries $O(q)$ proof bits, and rejects claims that are δ-far from the language with probability at most ϵ.

Complexity of a PCP system. Roughly, the "complexity" of a PCP system is the complexity required to generate a small portion of the proof. This notion is useful due to the connection between zero-knowledge and leakage-resilience: the view of the verifier depends on few proof bits, and so can be viewed as "leakage" on the witness.

Definition 7.3. Let $\mathcal{L}$ be a function family, $q^* \in \mathbb{N}$ be a length parameter, and (P,V) be a PCP (or PCPP) system for relation $\mathcal{R}_L = \mathcal{R}_L(x,w)$. We say that (P,V) *has complexity* $(\mathcal{L}, q^*)$ if for every $(x,w) \in \mathcal{R}_L$, every q^* bits in every proof $\pi \in P(x,w)$ can be generated by a function in $\mathcal{L}$.

Representing computations as 3CNFs. Following [IWY16], we use boolean formulas to represent computations of boolean circuits. The description of the representation presented here is taken verbatim from [IWY16].

Definition 7.4 (Canonical 3CNFs representing boolean circuits)**.** A 3CNF is a conjunction of clauses, where each clause contains exactly 3 literals (a literal is a variable or its negation). Given a circuit C, we define the *canonical 3CNF representing C*, denoted φ_C, as follows.

- For every input gate g_i of C we introduce a variable x_i.
- For every gate g of C, with input wires a, b and output wire c:
  - We introduce a variable x_c. (Notice that if the gates are traversed from the input gates to the output gate, then the variables x_a, x_b corresponding to a, b have already been defined.)
  - We define a 3CNF φ_g as follows:
    - g is an $\wedge$ gate, $c = a \wedge b$: $\varphi(x_a, x_b, x_c) = (x_c \vee \neg x_a \vee \neg x_b) \wedge (\neg x_c \vee x_a \vee \neg x_c) \wedge (\neg x_c \vee x_b \vee \neg x_c)$.[20]
    - g is an $\vee$ gate, $c = a \vee b$: $\varphi(x_a, x_b, x_c) = (\neg x_c \vee x_a \vee x_b) \wedge (x_c \vee \neg x_a \vee x_c) \wedge (x_c \vee \neg x_b \vee x_c)$.
    - g is a $\neg$ gate, $c = \neg a$: $\varphi(x_a, x_c) = (\neg x_c \vee \neg x_a \vee \neg x_c) \wedge (x_c \vee x_a \vee x_c)$.
- For the output gate g_o of C, with output wire o, we concatenate the clause $(x_o \vee x_o \vee x_o)$ to φ_{g_o}, i.e., we obtain a new 3CNF $\varphi_{g_o} \wedge (x_o \vee x_o \vee x_o)$.
- $\varphi_C = \wedge_g \varphi_g$, where the conjunction is over all gates except input gates.

Example 7.5. Let $C : \{0,1\}^2 \to \{0,1\}$, $C(y, z) = (y \wedge z) \vee (\neg y)$. Let $g_\wedge, g_\vee, g_\neg$ denote the $\wedge, \vee, \neg$ gates of C, and notice that $g_\vee$ is also the output gate. Then:

- The variables of φ_C are x_y, x_z (corresponding to the input gates of C), and $x_\wedge, x_\vee, x_\neg$ (corresponding to the output wires of $g_\wedge, g_\vee, g_\neg$, respectively).
- $\varphi_{g_\wedge}(x_y, x_z, x_\wedge) = (x_\wedge \vee \neg x_y \vee \neg x_z) \wedge (\neg x_\wedge \vee x_y \vee \neg x_\wedge) \wedge (\neg x_\wedge \vee x_z \vee \neg x_\wedge)$.
- $\varphi_{g_\neg}(x_y, x_\neg) = (\neg x_\neg \vee \neg x_y \vee \neg x_\neg) \wedge (x_\neg \vee x_y \vee x_\neg)$.
- $\varphi_{g_\vee}(x_\wedge, x_\neg, x_\vee) = (\neg x_\vee \vee x_\wedge \vee x_\neg) \wedge (x_\vee \vee \neg x_\wedge \vee x_\vee) \wedge (x_\vee \vee \neg x_\neg \vee x_\vee) \wedge (x_\vee \vee x_\vee \vee x_\vee)$ (the last clause of $\varphi_{g_\vee}$ was inserted because $g_\vee$ is also the output gate).
- $\varphi_C(x_y, x_z, x_\wedge, x_\neg, x_\vee) = \varphi_{g_\wedge} \wedge \varphi_{g_\neg} \wedge \varphi_{g_\vee}$.

Notice that the variables of φ_C correspond to the wires of C. φ_C represents C in the sense that a wire assignment to C (which is also an assignment to the variables of φ_C) satisfies φ_C only if it corresponds to the evaluation of C on a satisfying input, as stated in the next fact.

Fact 7.6. *Let C be a boolean circuit, and let φ_C be the canonical 3CNF representing C (as in Definition 7.4). Then a wire assignment W to C, which is also an assignment to the variables of φ_C, satisfies φ_C if and only if W is the assignment to the wires of C when evaluated on a satisfying input x. Moreover, φ_C can be constructed from C in linear time, so $|\varphi_C| = O(|C|)$, where $|\varphi|$ denotes the number of clauses in φ.*

[20] Notice that some variables appear twice in the same clause. This is not needed for the functionality of the formula, but is required for φ to be a *3CNF*. Instead, we could have introduced new variables, where a clause of the form $a \vee b$ would have been replaced with the 3CNF $(a \vee b \vee z) \wedge (a \vee b \vee \neg z)$, where z is a new variable. The alternative transformation has the advantage that each variable appears at most once in every clause, but increases the number of variables. As we will not require that every variable appears at most once in each clause, we have chosen the first transformation, which has the advantage that a wire assignment to C is also an assignment to φ_C.

Circuit Classes. Recall that to construct ZK-PCPs and ZK-PCPPs, we are interested in protecting the prover computation from leakage that is computable in low depth. More formally, we denote by $\mathsf{Shallow}(n, m, d, s)$ the class of all depth-d, size-s, arithmetic circuits over $\mathbb{F}$ with input length n and output length m. We Denote $\mathsf{Shallow}(n, d, s) = \cup_{m \in \mathbb{N}} \mathsf{Shallow}(n, m, d, s)$. Somewhat abusing notation, we use the same notations to denote the *families of functions* computable by circuits in the respective class of circuits.

Leakage-Resilience of Encodings. We define the notion of leakage-resilience for encoding schemes.

Definition 7.7 (Leakage-indistinguishability of functions and encodings, [IWY16])**.** Let D, D' be finite sets, $\mathcal{L}_D = \{\ell : D \to D'\}$ be a family of leakage functions, and $\epsilon > 0$. We say that two distributions X, Y over D are $(\mathcal{L}_D, \epsilon)$-*leakage-indistinguishable*, if for any function $\ell \in \mathcal{L}_D$, $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$. In case $\mathcal{L}_D$ consists of functions over a union of domains, we say that X, Y over D are $(\mathcal{L}_D, \epsilon)$-leakage-indistinguishable if $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$ for every function $\ell \in \mathcal{L}$ *with domain* D.

Let $\mathcal{L}$ be a family of leakage functions. We say that a randomized function $f : \Sigma^n \to \Sigma^m$ is $(\mathcal{L}, \epsilon)$-*leakage-indistinguishable* if for every $x, y \in \Sigma^n$, the distributions $f(x), f(y)$ are $(\mathcal{L}, \epsilon)$-leakage-indistinguishable. We say that an encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ is $(\mathcal{L}, \epsilon)$-*leakage-indistinguishable* if for every large enough $t \in \mathcal{N}$, $\mathsf{Enc}(\cdot, 1^t)$ is $(\mathcal{L}, \epsilon)$-leakage indistinguishable.

Gadgets of an LRCC. The LRCCs described in the next sections will be gadget-based, and their leakage-resilience will depend on the underlying gadgets satisfying two properties. First, the gadgets are *re-randomizing*, namely the encodings at the output of each gadget are uniformly random subject to encoding the "correct" value. Specifically, recall that a gadget takes both standard inputs (encoded according to some encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$) and masking inputs, then re-randomization holds when the masking inputs are chosen according to the "correct" distribution for masking inputs of the gadget. Formally,

Definition 7.8 (Re-randomization, [FRR+10])**.** A gadget $\mathcal{G}$ is *re-randomizing* if for every standard input $\widehat{x} = \mathsf{Enc}(x)$, and every masking input m chosen according to the distribution of masking inputs to $\mathcal{G}$, $\mathcal{G}(\widehat{x}, \mathsf{m})$ is random subject to encoding the correct output (as determined by x, and the operation which $\mathcal{G}$ emulates).

The second property is that the gadgets are locally reconstructible, i.e., given any encoding of a "legal" input-output pair, the internal wires of the gadget (as determined by the encoding of the inputs and outputs, and the masking inputs) can be simulated in a low complexity class. Formally,

Definition 7.9 (Local reconstructibility, [FRR+10])**.** Let $\mathcal{G}$ be a gadget. A pair $(\widehat{x}, \widehat{y})$ of encodings is *plausible* for $\mathcal{G}$ if for some well-formed masking input m, $\mathcal{G}$ on input $(\widehat{x}, \mathsf{m})$ outputs $\widehat{y}$. Given a gadget $\mathcal{G}$, $\epsilon > 0$, and families $\mathcal{L}, \mathcal{L}_{\mathcal{G}}$ of functions, $\mathcal{G}$ is $(\mathcal{L}, \epsilon)$-*reconstructible by* $\mathcal{L}_{\mathcal{G}}$ if the following holds. There exists a distribution REC over functions rec that take as input the standard inputs of $\mathcal{G}$, and its output, and output simulated values for the masking inputs, and internal wires of $\mathcal{G}$, such that for every plausible pair $(\widehat{x}, \widehat{y})$: $\mathrm{supp}(\mathsf{REC}) \subseteq \mathcal{L}_{\mathcal{G}}$; and if rec is chosen according to REC then $\mathsf{rec}(\widehat{x}, \widehat{y})$ is $(\mathcal{L}, \epsilon)$-leakage-indistinguishable from the actual distribution of the wires of $\mathcal{G}$ (as determined by the distribution of the masking inputs), conditioned on $\widehat{x}, \widehat{y}$.

The leakage resilience of our constructions will rely on the fact that the gadgets of Faust et al. [FRR+10] satisfy these properties. Specifically, [FRR+10] show that the gadgets are re-

randomizing because the outputs of the gadgets are masked with random and independent 0-encodings. They also show (Lemma 9 in the full version of [FRR+10]) that the gadgets are locally reconstructible in a low complexity class:

Lemma 7.10 (Gadgets are locally reconstructible, [FRR+10])**.** *Let $t \in \mathbb{N}$ be a security parameter, $\mathcal{L}, \mathcal{L}_{\mathsf{E}}$ be families of functions, and $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$. Let E^{in} denote the internal encoding scheme used in the LRCC of [FRR+10] (see Section 2.2), and let $\hat{n}_1^{\mathrm{in}}$ denote the length of encodings which it outputs when encoding a single field element. Then the following holds for the gadgets of the LRCC.*

- *$+$ and $-$ gadgets are $(\mathcal{L}, 0)$-reconstructible by $\mathsf{Shallow}\left(3\hat{n}_1^{\mathrm{in}}, 2, O\left(\hat{n}_1^{\mathrm{in}}\right)\right)$.*
- *copy gadgets are $(\mathcal{L}, 0)$-reconstructible by $\mathsf{Shallow}\left(3\hat{n}_1^{\mathrm{in}}, 1, O\left(\hat{n}_1^{\mathrm{in}}\right)\right)$.*
- *mask gadgets are $(\mathcal{L}, 0)$-reconstructible by $\mathsf{Shallow}\left(2\hat{n}_1^{\mathrm{in}}, 1, O\left(\hat{n}_1^{\mathrm{in}}\right)\right)$.*
- *const_α gadgets are $(\mathcal{L}, 0)$-reconstructible by $\mathsf{Shallow}\left(\hat{n}_1^{\mathrm{in}}, 1, O\left(\hat{n}_1^{\mathrm{in}}\right)\right)$.*
- *If E^{in} is $(\mathcal{L}_{\mathsf{E}}, \epsilon(t))$-leakage-indistinguishable, and $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(3\hat{n}_1^{\mathrm{in}}, 3, O\left(\hat{n}_1^{\mathrm{in}}\right)\right)$, then the $\times$ gadget is $\left(\mathcal{L}, \hat{n}_1^{\mathrm{in}} \cdot \epsilon(t)\right)$-reconstructible by $\mathsf{Shallow}\left(3\hat{n}_1, 2, O\left(\left(\hat{n}_1^{\mathrm{in}}\right)^2\right)\right)$.*

Finally, we describe the copy gadget $\mathcal{G}_c$ of [FRR+10], because it is used in a non-black-box way by the LRCC with public inputs of Section 7.2.

Construction 7.11 (Copy gadget $\mathcal{G}_c$, Figure 3 in the full version of [FRR+10])**.** Let $\mathsf{E}^{\mathrm{in}} = \left(\mathsf{Enc}^{\mathrm{in}}, \mathsf{Dec}^{\mathrm{in}}\right)$ denote the internal encoding scheme used in the LRCC of [FRR+10]. The copy gadget $\mathcal{G}_c$ takes as input $\widehat{a} \in \mathsf{Enc}^{\mathrm{in}}\left(a, 1^t\right)$ and masking inputs $\vec{r}, \vec{r}'$, and operates as follows.

1. Computes $\widehat{a}' = \widehat{a} + \vec{r}$.
2. Computes $\widehat{a}'' = \widehat{a} + \vec{r}'$.
3. Outputs $(\widehat{a}', \widehat{a}'')$.

It is immediate from the construction that $\mathcal{G}_c \in \mathsf{Shallow}\left(3\widehat{n}^{\mathrm{in}}(1,t), 1, O\left(\widehat{n}^{\mathrm{in}}(1,t)\right)\right)$.

7.2 LRCCs with public inputs

We formalize the notion of an LRCC in which part of the input is given in the clear, and for which no privacy is guaranteed. We start with the definition of a circuit compiler with public inputs.

Definition 7.12 (Circuit compiler with public inputs)**.** A circuit compiler over $\mathbb{F}$ is a pair $(\mathsf{Comp}, \mathsf{E})$ of algorithms with the following syntax.

- $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ is an encoding scheme with the same syntax as in Definition 2.1.
- Comp is a polynomial-time algorithm that given an arithmetic circuit C over $\mathbb{F}$, and 1^t, outputs an arithmetic circuit $\hat{C}$.

We require that $(\mathsf{Comp}, \mathsf{E})$ satisfy the following *correctness* requirement. There exists a negligible function $\epsilon(t) = \mathsf{negl}(t)$ such that for any arithmetic circuit C, and any inputs x, w for C, we have $\Pr\left[\hat{C}(x, \hat{w}) = C(x, w)\right] = 1$, where $\hat{w} \leftarrow \mathsf{Enc}\left(w, 1^t, 1^{|C|}\right)$.

A boolean circuit compiler with public inputs is defined similarly, except that it operates on boolean circuits C.

Definition 7.13 (LRCC with public inputs)**.** Let t be a security parameter, and $\mathbb{F}$ be a finite field. For a function class $\mathcal{L}$, $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$, and a size function $\mathsf{S}(n,m) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, we say that $(\mathsf{Comp}, \mathsf{E})$ is $(\mathcal{L}, \epsilon(t), \mathsf{S}(n,m))$-*leakage-resilient with public inputs* if there exists a PPT algorithm Sim such that the following holds. For all sufficiently large t, every arithmetic circuit C over $\mathbb{F}$ of input length $n+m$ and size at most $\mathsf{S}(n,m)$, every $\ell \in \mathcal{L}$ of input length $\left|\hat{C}\right|$, and every $x \in \mathbb{F}^n, w \in \mathbb{F}^m$, we have $\mathsf{SD}\left(\ell\left[\mathsf{Sim}\left(C, C(x,w), x\right)\right], \ell\left[\hat{C}, x, \hat{w}\right]\right) \leq \epsilon(t)$, where $\hat{w} \leftarrow \mathsf{E}\left(w, 1^{|C|}\right)$.

If the above holds with an inefficient simulator Sim, then we say that $(\mathsf{Comp}, \mathsf{E})$ is $(\mathcal{L}, \epsilon(t), \mathsf{S}(n,m))$-*relaxed leakage-resilient with public inputs*.

We show how to modify the LRCC of [IWY15, Construction 3.7] to obtain a relaxed LRCC with public inputs. The construction uses the LRCC in a non-black-box way. It also uses (as a black-box) the copy gadget $\mathcal{G}_c$ and multiplication gadget $\mathcal{G}_\times$ of [FRR+10] to "refresh" encodings. (Similar methods of "refreshing" encodings using a multiplication gadget were used before, e.g., in [ADF16].) Specifically, since $\mathcal{G}_\times$ is re-randomizing then one can refresh an encoding $\widehat{b}$ of a bit b by first computing two copies $\widehat{b}', \widehat{b}''$ of $\widehat{b}$ using the copy gadget $\mathcal{G}_c$, and then refreshing the encoding by computing $\mathcal{G}_\times\left(\widehat{b}', \widehat{b}''\right)$, where output encoding will be a random encoding of b. In the following, we use $r_\times, r_c$ to denote the length of masking inputs used by $\mathcal{G}_\times, \mathcal{G}_c$ (respectively).

Construction 7.14 (Relaxed LRCC with public inputs)**.** Let $\mathbb{F}$ be a finite field, $t, t_{\text{in}} \in \mathbb{N}$ be security parameters, and $n, m \in \mathbb{N}$ be input length parameters. Let $\left(\mathsf{Comp}^{\mathsf{IWY}}, \mathsf{E}^{\mathsf{IWY}} = \left(\mathsf{Enc}^{\mathsf{IWY}}, \mathsf{Dec}^{\mathsf{IWY}}\right)\right)$ denote the LRCC of [IWY15, Construction 3.7]. We define an LRCC with public inputs $(\mathsf{Comp}, \mathsf{E})$, as follows.

- $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$, where $\mathsf{Dec} = \mathsf{Dec}^{\mathsf{IWY}}$. As for Enc, recall that $\mathsf{E}^{\mathsf{IWY}}$ uses an internal encoding scheme $\mathsf{E}^{\text{in}} = \left(\mathsf{Enc}^{\text{in}}, \mathsf{Dec}^{\text{in}}\right)$, where $\mathsf{Enc}^{\mathsf{IWY}}\left(x, 1^t, 1^{t_{\text{in}}}\right)$ computes $\widehat{x}^b \leftarrow \mathsf{Enc}^{\text{in}}\left(x, 1^t, 0^{R(t,t_{\text{in}})}\right)$ for $b \in \{0,1\}$ and some function $R(t, t_{\text{in}})$, and outputs $\left(\widehat{x}^0, \widehat{x}^1\right)$. Let $R'(n,t,t_{\text{in}}) = R(t,t_{\text{in}}) + n \cdot (r_\times + r_c)$, then $\mathsf{Enc}\left(x, 1^t, 1^{t_{\text{in}}}\right) = \mathsf{Enc}^{\text{in}}\left(x, 1^t, 0^{R'(n,t,t_{\text{in}})}\right)$. (Intuitively, $\mathsf{Enc}^{\mathsf{IWY}}$ concatenates to its input sufficiently many 0-encodings to be used as masking inputs in the leakage-resilient circuit. The $n \cdot (r_\times + r_c)$ additional encodings are needed for the n copies of C^{inp} introduced in the construction of $\widehat{C}$ below.) Let $\widehat{n}^{\text{in}} = \widehat{n}^{\text{in}}(n,t,t_{\text{in}})$ and $\widehat{n} = \widehat{n}(n,t,t_{\text{in}})$ denote the lengths of encodings which $\mathsf{Enc}^{\text{in}}, \mathsf{Enc}$ output, respectively.
- Comp on input a circuit $C : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}$ outputs the circuit $\widehat{C}$ defined as follows:
  - $\widehat{C}$ interprets its input as $x \in \mathbb{F}^n$, and an encoding $\widehat{w} \in \mathbb{F}^{\widehat{n}(m,t,t_{\text{in}})}$.
  - Define $C^{\mathsf{inp}} : \mathbb{F} \times \mathbb{F}^{\widehat{n}^{\text{in}}(r_\times + r_c, t, t_{\text{in}})} \to \mathbb{F}$, which on input $z \in \mathbb{F}$, and masking inputs $\vec{r}_\times, \vec{r}_c \in \mathbb{F}^{\widehat{n}^{\text{in}}(r_\times + r_c, t, t_{\text{in}})}$:
    - Concatenates z with $0^{\hat{n}_1^{\text{in}} - 1}$, where $\hat{n}_1^{\text{in}} = \widehat{n}^{\text{in}}(1,t,t_{\text{in}})$, and let z' denote this concatenation. (Intuitively, z' is a valid, though not random, encoding of z according to E.)
    - Uses $\mathcal{G}_c$ with $\vec{r}_c$ as the masking inputs, to compute two copies $\tilde{z}, \tilde{z}'$ of z'.
    - Uses $\mathcal{G}_\times$ with $\vec{r}_\times$ as the masking inputs, to compute the multiplication $\tilde{z} \times \tilde{z}'$, and let z'' denote the output of the gadget. (Intuitively, this refreshes the encoding of z.)
  - For every $1 \leq i \leq n$ and $b \in \{0,1\}$, let $\widehat{x_i}^b$ denote the output of $C_{\mathsf{inp}}\left(x_i, \vec{r}_i^{\,b}\right)$, where $\vec{r}_i^{\,b}$ is taken from the masking inputs given as input to $\widehat{C}$ as part of the encoding of $\widehat{w}$. Let $\widehat{x}^b = \left(\widehat{x_1}^b, \ldots, \widehat{x_n}^b\right)$.

– Let C^{IWY} denote the circuit $\mathsf{Comp}^{\mathsf{IWY}}\left(C, 1^t\right)$ obtained through the LRCC of Construction 3.7 in the full version of [IWY16]. Recall that C^{IWY} emulates the execution of C twice in parallel, on two copies of its inputs, and additionally checks the validity of the masking inputs used in these executions. $\widehat{C}$ emulates C^{IWY} on inputs $\left(\widehat{x}^0, \widehat{x}^1, \widehat{w}\right)$, where $\widehat{x}^b$ is used as input to the b'th copy of C in C^{IWY}. We stress that the masking inputs used in the copies of C_{inp} are checked together with the other masking inputs in C^{IWY}. (Specifically, the masking inputs used to compute the $\widehat{x}_i^b$'s are checked together with the masking inputs of the b'th copy of C.)

Next, we prove that Construction 7.14 is SAT-respecting and relaxed LR.

Lemma 7.15. *Construction 7.14 is SAT-respecting.*

Proof sketch for Lemma 7.15. Ishai et al. [IWY16] prove that C^{IWY} is SAT-respecting. Moreover, their proof shows that if C^{IWY} is satisfied then at least one of the copies of C uses only well-formed masking inputs (i.e., encodings of 0). Since the masking inputs used in C^{inp} are checked as part of C^{IWY}, it follows that for at least one $b \in \{0,1\}$, $\widehat{x}^b$ was generated using well-formed masking inputs, i.e., for every $1 \leq i \leq n$, $\widehat{x}_i{}^b$ encodes x_i^2. Moreover, C^{IWY} checks that its input symbols are bits, so $x \in \{0,1\}^n$, and in particular $x_i^2 = x_i$ for every $1 \leq i \leq n$. Therefore, C^{IWY} emulates C on (x, w), so $\widehat{C}$ is satisfiable only if C is. □

Lemma 7.16. *Let $\mathcal{L}$ be a leakage class, and $\epsilon(t), \epsilon_\times(t) : \mathbb{N} \to \mathbb{R}^+$. If the LRCC of Construction 3.7 in the full version of [IWY16] is $(\mathcal{L}, \epsilon(t))$-(relaxed) leakage-resilient, and the multiplication gadget $\mathcal{G}_\times$ of [FRR+10] is $(\mathcal{L}, \epsilon_\times(t))$-leakage-resilient, then Construction 7.14 is $(\mathcal{L}', \epsilon'(t))$-(relaxed) leakage-resilient, where $\mathcal{L}' = \mathcal{L} \circ \mathsf{Shallow}\left(6n \cdot \widehat{n}\left(1, t, t_{\text{in}}\right), 3, O\left(n \cdot \widehat{n}^2\left(1, t, t_{\text{in}}\right)\right)\right)$ and $\epsilon'(t) = \epsilon(t) + 4n\epsilon_\times(t)$.*

The proof uses the following observation regarding the simulator $\mathsf{Sim}^{\mathsf{IWY}}$ of [IWY16]. For any satisfying input (x, w) for C, If $\mathsf{Sim}^{\mathsf{IWY}}$ is given an honestly-generate encoding of $\widehat{x}$ then it can successfully simulate the wire values of C^{IWY} consistently with $\widehat{x}$ (i.e., $\widehat{x}$ is the encoded input used in the simulated wires). Similarly, it can successfully simulate the wire values even when some of the masking inputs are fixed to random 0-encodings.

Proof sketch for Lemma 7.16. We prove the claim by reduction to the leakage-resilience of Construction 3.7 in the full version of [IWY16]. We describe a simulator Sim that uses the simulator $\mathsf{Sim}^{\mathsf{IWY}}$ of [IWY16] in a non-black-box manner. Sim on input x and $C(x, w)$ honestly emulates the copies of C^{inp}, with well-formed masking inputs, to generate the input encoding $\widehat{x}$ for C^{IWY}. It then uses $\mathsf{Sim}^{\mathsf{IWY}}$ to simulate the entire wire values of C^{IWY} consistently with $\widehat{x}$ and the masking inputs used in the copies of C^{inp}.

We show that the simulated leakage is $\epsilon'(t)$-statistically close to the real-world leakage. We use the fact, proven in [FRR+10] (and cited in Lemma 7.10) that the multiplication gadget $\mathcal{G}_\times$ is locally reconstructible in $\mathsf{Shallow}\left(3 \cdot \widehat{n}^{\text{in}}\left(1, t, t_{\text{in}}\right), 2, O\left(\left(\widehat{n}^{\text{in}}\left(1, t, t_{\text{in}}\right)\right)^2\right)\right)$, and that the copy gadget $\mathcal{G}_c$ is in $\mathsf{Shallow}\left(3 \cdot \widehat{n}^{\text{in}}\left(1, t, t_{\text{in}}\right), 1, O\left(\widehat{n}^{\text{in}}\left(1, t, t_{\text{in}}\right)\right)\right)$.

Let $\mathcal{W}_S, \mathcal{W}_R$ denote the wire values in the simulation described above, and in the real-world execution, respectively. We show that for any $\ell' \in \mathcal{L}'$, $\mathsf{SD}\left(\ell'\left(\mathcal{W}_S\right), \ell'\left(\mathcal{W}_R\right)\right) \leq \epsilon'(t)$. We define a pair of hybrid distributions, $\mathcal{H}_S, \mathcal{H}_R$ which are obtained from $\mathcal{W}_S, \mathcal{W}_R$ (respectively) by replacing the internal wire values of the $\mathcal{G}_\times$ gadgets in the copies of C_{inp} with the simulated wire values generated by the local reconstructor of $\mathcal{G}_\times$. Let ℓ denote the leakage function that has x hard-wired into it, and given the wire values $\mathcal{W}$ of C^{IWY}, honestly computes the copy gadgets $\mathcal{G}_c$ on the bits

of x, then uses the obtained values, together with the encodings of x reported in $\mathcal{W}$, to generate the wire values of all copies of C_{inp} (using the local reconstructor of $\mathcal{G}_\times$). Then $\ell \in \mathcal{L}$ (because there are $2n$ such gadgets, which can be evaluated in parallel), and by a standard hybrid argument over the local reconstruction property of $\mathcal{G}_\times$, $\mathsf{SD}\left(\ell'\left(\mathcal{W}_R\right), \ell'\left(\mathcal{H}_R\right)\right), \mathsf{SD}\left(\ell'\left(\mathcal{W}_S\right), \ell'\left(\mathcal{H}_S\right)\right) \leq 2n \cdot \epsilon_\times$. (More formally, this follows from a standard hybrid argument where we replace the internal wires of one of the $2n$ $\mathcal{G}_\times$ gadgets at a time.)

It remains to bound $\mathsf{SD}\left(\ell'\left(\mathcal{H}_R\right), \ell'\left(\mathcal{H}_S\right)\right)$. Notice that the internals of all the C_{inp} circuits are identically distributed in $\mathcal{H}_R, \mathcal{H}_S$, so it suffices to bound the statistical distance conditioned on some fixed value v of these wires. Let ℓ be the leakage function that has v hard-wired into it, and that, given the wire values $\mathcal{W}$ of C^{IWY}, runs ℓ' on $(v, \mathcal{W})$. Then $\ell \in \mathcal{L}$. Let $\mathcal{W}'_R, \mathcal{W}'_S$ denote the restriction of $\mathcal{W}_R, \mathcal{W}_S$ to the wires of C^{IWY}, i.e., $\mathcal{W}'_R$ are the wire values of C^{IWY} in an actual execution, whereas $\mathcal{W}'_S$ were generated by $\mathsf{Sim}^{\mathsf{IWY}}$. Therefore, by the leakage-resilience of the LRCC of [IWY16], $\mathsf{SD}\left(\ell\left(\mathcal{H}'_S\right), \ell\left(\mathcal{H}'_R\right)\right) \leq \epsilon\left(t\right)$. We conclude the proof by noting that $\ell\left(\mathcal{H}'_\star\right) = \ell'\left(\mathcal{H}_\star\right)$ for $\star \in \{S, R\}$. □

Remark 7.17 (Efficient simulation)**.** We note that if the simulator $\mathsf{Sim}^{\mathsf{IWY}}$ of the LRCC of Construction 3.7 in the full version of [IWY16] is given a witness w such that $(x, w) \in \mathcal{R}_L$ then the simulation is efficient. This is implicit in [IWY16], since $\mathsf{Sim}^{\mathsf{IWY}}$ can use *any* satisfying witness for the simulation, and finding such a satisfying witness is the only computation $\mathsf{Sim}^{\mathsf{IWY}}$ performs that is not polynomial-time. Consequently, if the simulator Sim described in the proof of Lemma 7.16 is given a satisfying witness w, it can pass it along to $\mathsf{Sim}^{\mathsf{IWY}}$, and overall the simulation would be efficient.

7.3 Average-Case LRCCs

Next, we use LRCCs with public inputs to construct an average-case LRCC. We first define the class of circuits for which we will guarantee average-case leakage-resilience.

Definition 7.18. Let $\mathbb{F}$ be a finite field, and let $C : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}$ be an arithmetic circuit over $\mathbb{F}$. Let $\mathsf{E}_{\mathsf{inp}} = (\mathsf{Enc}_{\mathsf{inp}}, \mathsf{Dec}_{\mathsf{inp}})$ be an encoding scheme that outputs encodings of length $\widehat{n}\left(n, t, t_{\mathrm{in}}\right)$, in which the decoder can be implemented by a circuit C_{Dec}. The circuit $C' : \mathbb{F}^{n \cdot \widehat{n}(1,t,t_{\mathrm{in}})} \times \mathbb{F}^m \to \mathbb{F}$ is defined as follows:

1. C' interprets its input as n encodings $\mathbf{x}^1, \cdots, \mathbf{x}^n$ according to $\mathsf{E}_{\mathsf{inp}}$ of $x_1, \ldots, x_n \in \{0,1\}$, and a witness w.
2. For every $1 \leq i \leq n$, C' uses C_{Dec} to decode $\mathbf{x}_i$ and obtain a bit x_i. Let $b_i \in \mathbb{F}$ be indicator of the event that decoding succeeded, i.e. $b_i = 0$ if and only if the decoding failed.
3. if $\prod_{i=1}^n b_i = 0$ then C' outputs 0. Otherwise, C' outputs $C\left(\left(x_1, \cdots, x_n\right), w\right)$.

Intuitively, C' emulates the operation of C, when the input x is encoded using $\mathsf{E}_{\mathsf{inp}}$. If not all input encodings are valid, namely, the encoded inputs of C' do not correspond to a valid input for C, then C' outputs 0. Otherwise, C' emulates C on the corresponding input.

Let $\mathsf{E}_{\mathsf{inp}} = (\mathsf{Enc}_{\mathsf{inp}}, \mathsf{Dec}_{\mathsf{inp}})$ be an encoding scheme, and C be a circuit taking inputs (x, w). Informally, C is average-case leakage-resilient with respect to $\mathsf{E}_{\mathsf{inp}}$ if for every inputs x, w, and every subset $I \subseteq [n]$ of bits of x, the *joint distribution* of $\{\mathbf{x}_i\}_{i \in I}$, and the leakage on the wire values $C\left[\mathbf{x}, w\right]$ when $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a random encoding of $x = (x_1, \ldots, x_n)$ accordingly to $\mathsf{E}_{\mathsf{inp}}$, can be simulated given only I, $\{\mathbf{x}_i\}_{i \in I}$, and C''s output. Formally:

Definition 7.19 (average-case leakage-resilience)**.** Let t be a security parameter, let $\mathbb{F}$ be a finite field, and let $\mathsf{E}_{\mathsf{inp}} = (\mathsf{Enc}_{\mathsf{inp}}, \mathsf{Dec}_{\mathsf{inp}})$ be an encoding scheme. For a function class $\mathcal{L}$, $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$, and a size function $\mathsf{S}(n,m) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, we say that a circuit compiler with public inputs $(\mathsf{Comp}, \mathsf{E})$ is $(\mathcal{L}, \epsilon(t), \mathsf{S}(n,m))$-*average leakage-resilient with respect to* $\mathsf{E}_{\mathsf{inp}}$ if there exists a PPT algorithm Sim such that the following holds. For all sufficiently large t, every arithmetic circuit C over $\mathbb{F}$ of input length $n+m$ and size at most $\mathsf{S}(n,m)$, every $\ell \in \mathcal{L}$ of input length $\left|\hat{C}'\right|$, every subset $I \subseteq [n]$, and every $x \in \mathbb{F}^n, w \in \mathbb{F}^m$, we have

$$\mathsf{SD}\left(\left(\ell\left[\mathsf{Sim}\left(C, C(x,w), \{\mathbf{x}'_i\}_{i\in I}\right)\right], \{\mathbf{x}'_i\}_{i\in I}\right), \left(\ell\left[\hat{C}', \mathbf{x}, \hat{w}\right], \{\mathbf{x}_i\}_{i\in I}\right)\right) \leq \epsilon(t)$$

where C' is the circuit defined in Definition 7.18, $\hat{w} \leftarrow \mathsf{E}\left(w, 1^t, 1^{|C'|}\right)$, $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{x}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_n)$, and $\mathbf{x}_i, \mathbf{x}'_i \leftarrow \mathsf{Enc}_{\mathsf{inp}}\left(x_i, 1^t\right)$ for every $1 \leq i \leq n$.

If the above holds with an inefficient simulator Sim, then we say that $(\mathsf{Comp}, \mathsf{E})$ is $(\mathcal{L}, \epsilon(t), \mathsf{S}(n,m))$-*relaxed average-case leakage-resilient with respect to* $\mathsf{E}_{\mathsf{inp}}$.

Next, we show that the relaxed LRCC with public inputs of Construction 7.14 is relaxed average-case leakage-resilient with respect to a leakage-resilient encoding scheme.

Lemma 7.20. *Let $\mathcal{L}$ be a leakage class, $\mathsf{S}(n,m)$ be a size function, and $\epsilon(t), \epsilon'(t), \epsilon_\times(t) : \mathbb{N} \to \mathbb{R}^+$. If Construction 7.14 is $(\mathcal{L}, \mathsf{S}(n,m), \epsilon(t))$-relaxed leakage-resilient with an $(\mathcal{L}, \epsilon'(t))$-leakage-indistinguishable encoding scheme E, the gadgets of [FRR+10] are re-randomizing and $(\mathcal{L}, \epsilon_\times(t))$-reconstructible by $\mathsf{Shallow}\left(O\left(\widehat{n}(1,t,t_{\text{in}})\right), 2, O\left(\widehat{n}^2(1,t,t_{\text{in}})\right)\right)$, and $\mathsf{E}_{\mathsf{inp}}$ is $(\mathcal{L}, \epsilon'(t))$-leakage-indistinguishable, then for every circuit C of size $s \leq \mathsf{S}(n,m) - n \cdot |C_{\mathsf{Dec}}| - n$, Construction 7.14 is $(\mathcal{L}', \epsilon''(t))$-relaxed average-case leakage-resilient, where $\mathcal{L} = \mathcal{L}' \circ \mathsf{Shallow}\left(O\left(s \cdot \widehat{n}(1,t,t_{\text{in}})\right), 3, O\left(s \cdot \widehat{n}^2(1,t,t_{\text{in}})\right)\right)$, and $\epsilon''(t) = 2(2n+s) \cdot \epsilon_\times + 6n \cdot \epsilon'(t) + \epsilon(t)$.*

Proof sketch of Lemma 7.20. Let Sim' denote the simulator for the LRCC of Construction 7.14, and we describe a simulator Sim for average-case leakage-resilience. Sim is given $C(x,w)$ and $\{\mathbf{x}'_i\}_{i\in I}$. For every $i \in I$, it decodes $\mathbf{x}'_i$ to recover the encoded bit x_i. It finds (x', w') such that $C(x,w) = C(x', w')$, and $\{x_i\}_{i\in I} = \{x'_i\}_{i\in I}$. Then, for every $1 \leq i \leq n, i \notin I$, it computes $\mathbf{x}'_i \leftarrow \mathsf{Enc}_{\mathsf{inp}}\left(x'_i, 1^t\right)$, and runs Sim' on input $\mathbf{x}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_n)$ and $C(x,w)$ to simulate the entire wire values of $\widehat{C'}$ consistently with $\mathbf{x}'$ (honestly generating the decoder wires in C').

Let $\mathcal{W}_S, \mathcal{W}_R$ denote the wire values in the simulation described above, and in the real-world execution, respectively. We show that for any $\ell' \in \mathcal{L}'$, $\mathsf{SD}\left(\left(\ell'(\mathcal{W}_S), \{\mathbf{x}'_i\}_{i\in I}\right), \left(\ell'(\mathcal{W}_R), \{\mathbf{x}_i\}_{i\in I}\right)\right) \leq \epsilon''(t)$. Since $\{\mathbf{x}'_i\}_{i\in I}$ and $\{\mathbf{x}_i\}_{i\in I}$ are identically distributed, it suffices to bound the statistical distance conditioned on an arbitrary choice of these values, in which case it suffices to bound $\mathsf{SD}\left(\ell'(\mathcal{W}_S), \ell'(\mathcal{W}_R)\right)$. We do so through a sequence of hybrids.

Let $\ell' \in \mathcal{L}'$ be a leakage function. We say that a gadget $\mathcal{G}$ of $\widehat{C'}$ *touches* C_{inp} if (at least one of) its input(s) is the output of a $\mathcal{G}_\times$ gadget in one of the C_{inp} copies of Construction 7.14. Notice that there are $s' \leq |C|$ such gadgets. Let $\mathcal{H}_R$ be the distribution obtained from $\mathcal{W}_R$ by replacing the internal wire values of the $\mathcal{G}_\times$ gadgets in the copies of C_{inp}, and all gadgets that touch C_{inp}, with the simulated wire values generated by the gadget local reconstructors. We show that $\mathsf{SD}\left(\ell(\mathcal{W}_R), \ell(\mathcal{H}_R)\right) \leq (2n + |C|)\epsilon_\times(t)$.

Towards that end, fix some arbitrary order on these $s'' := 2n + s'$ gadgets, and define $\mathcal{H}^i_R$ which is obtained from $\mathcal{W}_R$ by replacing the internals of the first i gadgets with the locally reconstructed wires, i.e., $\mathcal{H}^0_R = \mathcal{W}_R$ and $\mathcal{H}^{s''}_R = \mathcal{H}_R$. We show that $\mathsf{SD}\left(\ell'\left(\mathcal{H}^{i+1}_R\right), \ell'\left(\mathcal{H}^i_R\right)\right) \leq \epsilon_\times(t)$ for every $0 \leq i < s''$. Let ℓ be the function that has all wires of $\widehat{C'}$, except the internals of the $i+1$'th gadget, hard-wired into it. It is given as input the wire values of the $i+1$'th gadget, uses the hard-wired

values to generate the entire wire values of $\widehat{C'}$, and runs ℓ' on these wire values. Then $\ell \in \mathcal{L}$, and so by the local-reconstruction property $\mathsf{SD}\left(\ell'\left(\mathcal{H}_R^{i+1}\right), \ell'\left(\mathcal{H}_R^{i}\right)\right) \leq \epsilon_{\times}(t)$.

Next, define $\mathcal{H}'_R$ which is obtained from $\mathcal{H}_R$ by first replacing the encodings at the outputs of the $\mathcal{G}_{\times}$ gadgets of C_{inp} with random encodings (of random values), and then generating the internal wire values of these gadgets, as well as all the gadgets that touch C_{inp}, using their local reconstructors. We show that $\mathsf{SD}\left(\ell\left(\mathcal{H}_R\right), \ell\left(\mathcal{H}'_R\right)\right) \leq 2n\epsilon'(t)$ follows from the leakage-indistinguishability of E. Indeed, fix some arbitrary order on the $2n$ encodings at the output of the $\mathcal{G}_{\times}$ gadgets of C_{inp}, and define $\mathcal{H}_R^{i\prime}$ which is obtained from $\mathcal{H}_R$ by replacing the first i encodings, i.e., $\mathcal{H}_R^{0\prime} = \mathcal{H}_R$ and $\mathcal{H}_R^{2n\prime} = \mathcal{H}'_R$. We show that $\mathsf{SD}\left(\ell'\left(\mathcal{H}_R^{i+1\prime}\right), \ell'\left(\mathcal{H}_R^{i\prime}\right)\right) \leq \epsilon'(t)$ for every $0 \leq i < 2n$. Let ℓ be the function that has all wires of $\widehat{C'}$, except the $i+1$'th encoding and the internal wires of all gadgets that touch it (i.e., the $\mathcal{G}_{\times}$ gadget of C_{inp} which outputs it, and the at most s gadgets that touch C_{inp} which take the $i+1$'th encoding as input), hard-wired into it. It is given as input the $i+1$'th encoding, and uses the gadget local reconstructors to generate the internal wire values of these gadgets. Then, it concatenates these values to the hard-wired values, and applies ℓ'. Then $\ell \in \mathcal{L}$ (here, we also use the fact that $n \leq s$), and by the leakage-indistinguishability of E, $\mathsf{SD}\left(\ell'\left(\mathcal{H}_R^{i+1\prime}\right), \ell'\left(\mathcal{H}_R^{i\prime}\right)\right) \leq \epsilon'(t)$.

Let $\mathcal{H}''_R$ be the distribution obtained from $\mathcal{H}'_R$ by replacing the input encodings $\mathbf{x}$ of x to encodings $\mathbf{x}'$ of x', then generating the internal wire values of the copy gadgets $\mathcal{G}_c$ of C_{inp} by honestly evaluating the gadgets, and generating the internal wires of the multiplication gadgets $\mathcal{G}_{\times}$ of C_{inp} using the local reconstructor. (More accurately, we only replace the gadgets whose inputs are encoding of the i'th input bit for some $i \notin I$.) Let $k := n - |I|$. For $1 \leq i \leq k$, let $\mathcal{H}_R^{i\prime\prime}$ be obtained from $\mathcal{H}'_R$ by replacing the first i input bits of x for $i \notin I$ with the corresponding bits of x'. We show that $\mathsf{SD}\left(\ell'\left(\mathcal{H}_R^{i+1\prime\prime}\right), \ell'\left(\mathcal{H}_R^{i\prime\prime}\right)\right) \leq \epsilon'(t)$ by the leakage-indistinguishability of $\mathsf{E}_{\mathsf{inp}}$. Indeed, let ℓ be the leakage function that has all wires of $\widehat{C'}$, except the internal wires of the $\mathcal{G}_c$ and $\mathcal{G}_{\times}$ gadgets that takes the $i+1$'th input bit as input. ℓ is given as input an encoding $\mathbf{b}$ of either x_{i+1} or x'_{i+1}, it honestly emulates $\mathcal{G}_c$ to generate its internal and output wires, and uses the local reconstructor of the $\mathcal{G}_{\times}$ gadget to generate the internal wires of the gadget. Then, it concatenates these wire values to the hard-wired values, and applies ℓ'. Then $\ell \in \mathcal{L}$ and so $\mathsf{SD}\left(\ell'\left(\mathcal{H}_R^{i+1\prime\prime}\right), \ell'\left(\mathcal{H}_R i\prime\prime\right)\right) \leq \epsilon'(t)$, so $\mathsf{SD}\left(\ell'\left(\mathcal{H}''_R\right), \ell'\left(\mathcal{H}_R\prime\right)\right) \leq k\epsilon'(t) \leq n \cdot \epsilon'(t)$.

Finally, let $\mathcal{W}'_R$ denote the distribution over the wire values in a real-world execution with input x'. Notice that $\mathcal{H}''_R$ is obtained from $\mathcal{W}'_R$ by replacing the encodings at the output of the $\mathcal{G}_{\times}$ gadgets with random encodings, and then generating the internal wires of these gadgets, and all gadgets that touch C_{inp}, with their local reconstructors. Therefore, the same arguments above show that $\mathsf{SD}\left(\ell'\left(\mathcal{W}'_R\right), \ell'\left(\mathcal{H}\right)\right) \leq (2n+s) \cdot \epsilon_{\times}(t) + 3n \cdot \epsilon'(t)$. We conclude the proof by noting that by the leakage-resilience with public inputs of Construction 7.14, $\mathsf{SD}\left(\ell'\left(\mathcal{W}'_R\right), \ell'\left(\mathcal{W}_S\right)\right) \leq \epsilon(t)$. □

Remark 7.21 (Efficient simulation)**.** Notice that except for finding $(x', w') \in \mathcal{R}_L$ which are consistent with (x, w), the simulator Sim described in the proof of Lemma 7.20 is efficient, if the simulator Sim' of the LRCC with public inputs is efficient. Using Remark 7.17, Sim' is efficient when given w'. Consequently, when given w', Sim is PPT.

7.4 A PCPP for 3SAT based on [AS92]

We now describe how to modify the AS-PCP [AS92] to get a PCPP. The AS-PCP proves satisfiability of a given 3-CNF, and one ingredient of the proof is an error-correcting encoding (the so called "low degree extension") of the witness, i.e., the satisfying assignment. Part of the verification procedure verifies the purported encoding is indeed close to some valid encoding. In the PCPP setting, we think of the input x and the witness w as two parts of a satisfying assignment for the

3CNF. Thus, we can include the error-correcting encoding of x in the proof (instead of hard-wiring it into the 3CNF as in the PCP setting).

What does including an encoding of x in the proof give us? Recall that soundness only guarantees that if the verifier accepts then x is δ-close to L. The PCPP can be used to verify that the purported encoding of x is $\delta/2$-close to a valid encoding. The other ingredients of the PCPP (specifically, the sum-check part) then verify that the closest encoding to the purported encoding of x is $\delta/2$-close to a satisfying assignment to the 3CNF. Thus, if x is δ-far from L then at least one of these tests will fail with high probability, and the verifier will reject. Specifically, using [AS98, Theorem 2], we have the following (using Notation 7.2):

Theorem 7.22 (PCPPs for 3SAT, implicit in [AS92])**.** *For any* $\delta \in (0,1)$*,*

$$3SAT \in PCPP\left[\log n, \log^2 n \cdot \log\frac{1}{\delta}, \frac{1}{2}, \delta, \text{poly}\,(n)\right].$$

7.5 The WI-PCPP System

We now describe the transformation from a(relaxed) average-case LRCC to a WI-PCPP system. We will need the following notation.

Notation 7.23 (L_{E})**.** Let $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ be a parameterized encoding scheme, and $b \in \{0,1\}$. We use L^b_{E} to denote all encodings of b according to E. That is: $L^b_{\mathsf{E}} = \cup_{t\in\mathbb{N}}\mathsf{Supp}\left(\mathsf{Enc}\left(b, 1^t\right)\right)$.

Construction 7.24. Let $t \in \mathbb{N}$ be a security parameter, and let $\mathcal{R}_L = \mathcal{R}_L(x,w)$ be an NP-relation with verification circuit C.[21] (More precisely, C is a family $\{C_n\}$ of circuits, where C_n is applied to inputs x of length n. To simplify notations, we denote all circuits in the family by C.) The Wi-PCPP system (P,V) uses the following building blocks.

- A PCPP system $(P_{\text{in}}, V_{\text{in}})$ for 3SAT with complexity $(\mathcal{L}, q^*)$ and a non-adaptive verifier V.
- A SAT-respecting $\mathcal{L}$-average-case (relaxed) LRCC $\left(\mathsf{Comp}, \mathsf{E}' = \left(\mathsf{Enc}', \mathsf{Dec}'\right)\right)$.
- An $\mathcal{L}$-leakage-resilient encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ with relative distance δ, in which the decoder Dec can be implemented by a circuit C_{Dec}.
- For $b \in \{0,1\}$, a PCPP system $\left(P^b_{\mathsf{E}}, V^b_{\mathsf{E}}\right)$ for the language L^b_{E} of Notation 7.23, where V^b_{E} is non-adaptive.

Prover algorithm. On input $(x,w) \in \mathcal{R}_L$ where $n = |x|$, and 1^t,[22] P:

1. Uses C to construct the circuit C' of size $s := |C'|$ defined in Definition 7.18.
2. Computes $\widehat{C}(\cdot) = \mathsf{Comp}(C')$.
3. For every $1 \le i \le n$, P samples a random encoding $\mathbf{x}_i \leftarrow \mathsf{Enc}\left(x_i, 1^t\right)$ of x_i, and let $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$.

[21] We note that since C takes the witness w as input, its description reveals $|w|$. To avoid this, we can use the (polynomial) bound p on the witness length which $\mathcal{R}_L$ is guaranteed to have. Specifically, C will take inputs of length $n + p + \log p$, where the first n bits are the input, the following p bits are the witness padded to length p, and the last $\log p$ bits are the witness length in binary representation padded to length $\log p$.

[22] Formally, P take as input $\epsilon, 1^{q^*}$, and not 1^t. However, these are used to determine which t to use for the underlying LRCC and encoding scheme E, so for simplicity we choose to describe the construction using t.

4. Samples a random encoding $\widehat{w} \leftarrow \mathsf{Enc}'\left(w, 1^t, 1^s\right)$ of the witness.
5. Evaluates $\widehat{C}$ on input $(\mathbf{x}, \widehat{w})$, and let $\mathcal{W}$ denote the wire values of $\widehat{C}$ in this evaluation.
6. Construct the canonical 3CNF φ representing $\widehat{C}$.
7. Runs P_{in} on input φ, and witness $\mathcal{W}$, to generate a proof π^C for the claim "$\varphi \in \mathsf{3SAT}$".
8. For every $1 \leq i \leq n$, Runs $P_{\mathsf{E}}^{x_i}$ to generate a PCPP π^i for the claim "$\mathbf{x}_i \in L_{\mathsf{E}}^{x_i}$".
9. Outputs the proof $\pi = \pi^C \circ \pi^1 \circ \cdots \circ \pi^n \circ \mathbf{x}_1 \circ \cdots \circ \mathbf{x}_n$.

Verifier algorithm. On input $1^t, 1^n$ (where $n = |x|$), and given oracle access to input oracle x, and proof $\pi = \pi^C \circ \pi^1 \circ \cdots \circ \pi^n \circ \mathbf{x}_1 \circ \cdots \circ \mathbf{x}_n$, the verifier V:

1. Constructs the leakage-resilient circuit $\widehat{C} = \mathsf{Comp}\left(C'\right)$, and its canonical 3CNF φ. (Here, C' denotes the circuit defined in Definition 7.18.)
2. *(Verifying that $\widehat{C}$ is satisfiable.)* Runs V_{in} with input φ, implicit input $\mathbf{x}_1 \circ \ldots \circ \mathbf{x}_n$, and proof π^C. If V_{in} rejects then V rejects.
3. *(Verifying consistency of $\widehat{C}$'s input with x.)* Repeats the following t independent times: picks a random $i \in_R [n]$, reads x_i from its input oracle, and runs $V_{\mathsf{E}}^{x_i}$ with input oracle $\mathbf{x}_i$, and proof oracle π^i. If $V_{\mathsf{E}}^{x_i}$ rejects, then V rejects. If all t iterations succeeded, V accepts.

Next, we analyze the properties of Construction 7.24. Perfect completeness follows directly from the perfect completeness of the encodings schemes E, E', the perfect completeness of the LRCC $(\mathsf{Comp}, \mathsf{E}')$, and the perfect completeness of the three underlying PCPP systems.

Informally, soundness follows from the SAT-respecting property of $(\mathsf{Comp}, \mathsf{E})$, and from the soundness of the underlying PCPP systems. This is formalized in the following lemma.

Lemma 7.25. *Let $\epsilon, \epsilon' \in (0,1)$ be error parameters, and $\delta' \in (0,1)$ be a proximity parameter. Let $\widehat{n}(n,t), \widehat{n'}(n,t)$ denote the length of encodings output by the encoding schemes E, E' used in Construction 7.24, respectively. If:*

- *E has relative distance δ,*
- *$\left(P_{\mathsf{E}}^b, V_{\mathsf{E}}^b\right)$ (for $b \in \{0,1\}$) has soundness error ϵ with proximity parameter δ, and*
- *$(P_{\text{in}}, V_{\text{in}})$ has soundness error ϵ_{in} with proximity parameter $\delta_{\text{in}} = \min\left\{\frac{\delta'}{4\widehat{n}(1,t)}, \frac{\delta\delta'}{4}\right\}$,*

then Construction 7.24 has soundness error $\max\left\{\epsilon_{\text{in}}, \left(\left(1 - \frac{\delta'}{4}\right) + \frac{\delta'}{4} \cdot \epsilon\right)^t\right\}$ with proximity parameter δ'.

Proof. Let x be δ'-far from $L_{\mathcal{R}_L}$, and denote $n = |x|$. Denote $\widehat{n'_1} = \widehat{n'_1}(t) = \widehat{n'}(1,t)$ and $\widehat{n_1} = \widehat{n_1}(t) = \widehat{n}(1,t)$. Let $\pi^* = \pi^{C,*} \circ \pi^{1,*} \cdots \circ \pi^{n,*} \circ \mathbf{x}_1^* \circ \cdots \circ \mathbf{x}_n^*$ denote the (possibly ill-formed) "proof".

Consider the following mental experiment, in which instead of generating $\widehat{C}$ from C', P generates $\widehat{C}$ from $C'_{\mathbf{x}^*} = C'(\mathbf{x}^*, \cdot)$ (namely, C' with $\mathbf{x}^*$ hard-wired into it). To distinguish it from the actual leakage-resilient circuit generated by P in the real world, we use $\widehat{C'}$ to denote the leakage-resilient circuit generated in the mental experiment. Notice that for any (possibly ill-formed) encoding $\widehat{w}^*$, $\left[\widehat{C'}, \widehat{w}^*\right]$ and $\left[\widehat{C}, (\mathbf{x}^*, \widehat{w}^*)\right]$ are identically distributed. In particular, the SAT-respecting property of $(\mathsf{Comp}, \mathsf{E}')$ guarantees that $\widehat{C'}$ is satisfiable if and only if $C'(\mathbf{x}^*, \cdot)$ is satisfiable.

Let $\mathcal{E}_1$ denote the event that for at least a $\frac{\delta'}{4}$-fraction of the i's, $\mathbf{x}_i^*$ is *not* a valid encoding (according to E) of some x_i^*. Let $\mathcal{E}_2$ denote the event that $\mathcal{E}_1$ does not occur, and additionally $x^* = (x_1^*, \cdots, x_n^*)$ is $\frac{\delta}{2}$-close to x, where for every i such that $\mathbf{x}_i^*$ is not a valid encoding according to E, we set $x_i^* = x_i$. We consider three possible cases.

First, if $\mathcal{E}_1$ occurs, then there exists a set $|\mathcal{I}| \subseteq [n]$ of size $\mathcal{I} \geq \frac{\delta'}{4}n$ such that for every $i \in \mathcal{I}$, $\mathbf{x}_i^*$ is not a valid encoding according to E, i.e., $b_i = 0$ (where b_i is the field element computed in Step 2 of the circuit C' of Definition 7.18), and so $C'_{\mathbf{x}^*}$ is not satisfiable. Moreover, to satisfy $C'_{\mathbf{x}^*}$, one needs to change at least $\frac{\delta'}{4}n$ bits of $\mathbf{x}^*$ (at least one bit of $\mathbf{x}_i^*$, for every $i \in \mathcal{I}$). As noted above, this implies that $\widehat{C'}$ is not satisfiable, and moreover to make it satisfiable, one needs to change at least $\frac{\delta'}{4}n$ bits of the hard-wired inputs to $\widehat{C'}$. Since $|\mathbf{x}^*| = n \cdot \widehat{n_1}$, this implies that $(\varphi, \mathbf{x}^*)$ is $\frac{\delta' n}{4n \cdot \widehat{n_1}} = \frac{\delta'}{4\widehat{n_1}}$-far from 3SAT. Therefore, the soundness of $(P_{\text{in}}, V_{\text{in}})$ guarantees that V_{in} (and consequently also V) rejects except with probability ϵ_{in}.

Second, if $\mathcal{E}_2$ occurs, then for at least $\left(1 - \frac{\delta'}{4}\right) n$ of the $i \in [n]$, $\mathbf{x}_i^*$ is a valid encoding (according to E) of some x_i (recall that for all invalid encodings, the corresponding bit is set to be consistent with x), and in addition $x^* = (x_1^*, \cdots, x_n^*)$ is $\frac{\delta'}{2}$-close to x. Therefore, x^* is $\frac{\delta'}{2}$-far from $L_{\mathcal{R}_L}$, i.e., to obtain an $x' \in L_{\mathcal{R}_L}$, one needs to change at least $\frac{\delta' n}{2}$ bits of x^*. Moreover, since at most $\frac{\delta' n}{4}$ $\mathbf{x}_i^*$ are *not* valid encodings (according to E'), then to obtain an encoding $\mathbf{x}' = (\mathbf{x}'_1, \cdots, \mathbf{x}'_n)$ of some $x' \in L_{\mathcal{R}_L}$, one needs to change at least $\frac{\delta' n}{4}$ encodings $\mathbf{x}_i^*$ from a valid encoding of x_i^* to a valid encoding of $\bar{x}_i^*$ which (by the δ relative distance of E) requires changing at least $\delta\widehat{n_1}$ bits of the encodings. The same arguments used in the first case now imply that $(\varphi, \mathbf{x}^*)$ is $\frac{\delta' n \cdot \delta' \cdot \widehat{n}_1/4}{n \cdot \widehat{n_1}} = \frac{\delta\delta'}{4}$-far from 3SAT, and again by the soundness of $(P_{\text{in}}, V_{\text{in}})$, V rejects except with probability ϵ_{in}.

Finally, if both $\mathcal{E}_1, \mathcal{E}_2$ do not occur, then at most $\frac{\delta' n}{4}$ of the $\mathbf{x}_i^*$'s are *not* valid encodings (according to E), and additionally, $x^* = (x_1^*, \cdots, x_n^*)$ is $\frac{\delta'}{2}$-far from x (recall that for every i such that $\mathbf{x}_i^*$ is not a valid encoding according to E, x_i^* is set to x_i). In particular, at least $\frac{\delta' n}{4}$ of the $\mathbf{x}_i^*$ are *valid* encodings (according to E) of $\bar{x}_i$. Consequently, for each of the t iterations that V performs in Step 3, with probability at least $\frac{\delta'}{4}$ the index i chosen for the iteration is such that $\mathbf{x}_i^*$ is a valid encoding (according to E) of $x_i^* = \bar{x}_i$. Moreover, since E has relative distance δ, then $\mathbf{x}_i^*$ is δ-far from $L_{\mathsf{E}}^{x_i}$. Therefore, the soundness of the PCPP system $\left(P_{\mathsf{E}}^{x_i}, V_{\mathsf{E}}^{x_i}\right)$ guarantees that this iteration fails, except with probability ϵ. Overall, a random iteration in Step 3 of V's algorithm succeeds with probability at most $\epsilon'' = \left(1 - \frac{\delta'}{4}\right) + \frac{\delta'}{4} \cdot \epsilon$. Since V performs t random and independent iterations, the probability that all of them succeeds (which upper bounds the probability that V accepts) is at most $(\epsilon'')^t$. □

Next, we analyze the zero-knowledge property of the construction. We first describe the simulator algorithm which will be used to prove zero-knowledge against non-adaptive (possibly malicious, possibly unbounded) verifiers. Notice that since the verifier is non-adaptive, its entire view can be generated given only the input, its randomness, and the oracle answers to its queries. Therefore, it suffices for the simulator to generate these values.

Construction 7.26 (Simulator for Construction 7.24, non-adaptive V^*)**.** Let Sim_{in} denote the simulator for the (relaxed) LRCC $(\mathsf{Comp}, \mathsf{E})$. The simulator Sim for Construction 7.24 operates as follows. On input $1^t, 1^{q^*}$, and given oracle access to x and a (possibly malicious, possibly unbounded) non-adaptive verifier V^*, Sim:

1. Picks a random string r for V^*, and calls V^* with randomness r to obtain its queries $\mathcal{Q}$ to the input oracle x and the proof. Let $\mathcal{Q}_x, \mathcal{Q}_C, \mathcal{Q}_1, \cdots, \mathcal{Q}_n \subseteq \mathcal{Q}$ denote the subsets of queries from $\mathcal{Q}$ to bits of x, π^C, and $\pi^1 \circ \mathbf{x}_1, \cdots, \pi^n \circ \mathbf{x}_n$, respectively.

2. For every $i \in \mathcal{Q}_x$, queries its oracle on x_i, and uses the oracle answers to generate the answer A_x to the queries in $\mathcal{Q}_x$.

3. Constructs the circuit $\widehat{C} = \mathsf{Comp}(C')$ (where C' is the circuit from Definition 7.18), and the canonical 3CNF φ representing $\hat{C}$.

4. Let $\mathcal{I} = \{i \ : \ \mathcal{Q}_i \neq \emptyset\}$. Then for every $i \in \mathcal{I}$, Sim:

   - Queries its oracle x on i.
   - Samples a random encodings $\mathbf{x}'_i \leftarrow \mathsf{Enc}\left(x_i, 1^t\right)$.
   - Emulates $P_E^{x_i}$ with input $\mathbf{x}'_i$ to generate a proof $\pi^{i\prime}$ for the claim "$\mathbf{x}'_i \in L_{\mathsf{E}}^{x_i}$".
   - Uses $x_i, \mathbf{x}'_i, \pi^{i,\prime}$ to generate the answers A_i to the queries in $\mathcal{Q}_i$.

5. Runs Sim_{in} with input $\{\mathbf{x}'_i\}_{i\in\mathcal{I}}, 1$ (1 is the output of the circuit C) to obtain a simulated wire assignment $\mathcal{W}'$ to the wires of $\widehat{C}$, which is consistent with $\{\mathbf{x}'_i\}_{i\in\mathcal{I}}$.

6. Runs P_{in} on input φ, and witness $\mathcal{W}'$, to generate a proof $\pi^{C\prime}$, and uses $\pi^{C\prime}$ to generate the answers A_C to the queries in $\mathcal{Q}_C$.

Remark 7.27. We note that the only operation that Sim performs which might require super-polynomial time is the emulation of Sim_{in}.

Next, we use the simulator of Construction 7.26 to prove that Construction 7.24 is q^*-witness-indistinguishable against non-adaptive (malicious) verifiers.

Lemma 7.28. *Let $n \in \mathbb{N}$ denote an input length parameter. Let $\mathcal{R}_L = \mathcal{R}_L(x, w)$ be an NP-relation with verification circuit C of size $S(n)$, and $q^* \in \mathbb{N}$ be a zero-knowledge parameter. If:*

- *$\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ is an encoding scheme in which Dec can be implemented by a circuit C_{Dec} of size s,*
- *$(P_{\text{in}}, V_{\text{in}})$ has complexity $(\mathcal{L}, q^*)$, and*
- *$(\mathsf{Comp}, \mathsf{E}' = (\mathsf{Enc}, \mathsf{Dec}))$ is an $(\mathcal{L}, \epsilon, S(n) + n(s+2))$-relaxed average-case LRCC with respect to E,*

then Construction 7.24 has $(q^*, 3\epsilon)$-witness-indistinguishability against non-adaptive verifiers.

Proof. We show the lemma holds with the simulator Sim of Construction 7.26. Let V^* be a (possibly malicious, possibly unbounded) non-adaptive verifier that makes at most q^* queries to its oracles, and let $x \in L_{\mathcal{R}_L}$. Let Real denote the restriction of V^*'s view to x, V^*'s randomness, and the oracle answers to V^*'s queries in the real world execution. Let Ideal denote the output of Sim on input $1^t, 1^{q^*}$, and given oracle access to x and to V^*. First, notice that V^*'s randomness is identically distributed in both $\mathsf{Real}, \mathsf{Ideal}$, and so it suffices to bound the statistical distance when both distributions are conditioned on every possible randomness r. Notice that this conditioning also determines the queries $\mathcal{Q}_x, \mathcal{Q}_c, \mathcal{Q}_1, \ldots, \mathcal{Q}_n$ of V^* to its input and proof oracles. We show that under this conditioning, both distributions are close to the hybrid distribution $\mathcal{H}$ obtained through the mental experiment in which P is run with input $x' \in L_{\mathcal{R}_L}$, and a witness w' such that $(x', w') \in L_{\mathcal{R}_L}$ instead of (x, w), where x' is such that the emulation of Sim_{in} in Step 5 of Construction 7.26 used encodings according to E of x'. Let $Q_x, \mathcal{Q}_C, \mathcal{Q}_1, \cdots, \mathcal{Q}_n, \mathcal{I}$ be as defined in Construction 7.26.

$\mathsf{SD}(\mathsf{Real}, \mathcal{H}) \leq 2\epsilon$. The only difference between $\mathsf{Real}, \mathcal{H}$ is that in Real the prover uses (x, w), whereas in $\mathcal{H}$ the prover uses (x', w'). Notice that for every $i \in \mathcal{I}$, $x_i = x'_i$ and so the distributions induced by $\pi^i, \mathbf{x}_i$ are identical in $\mathsf{Real}, \mathcal{H}$. Moreover, for every $i \notin \mathcal{I}$, the bits of π^i do not appear in $\mathsf{Real}, \mathcal{H}$, and so these are of no interest. Therefore, we can condition $\mathsf{Real}, \mathcal{H}$ on the event that both use the same encodings $\mathbf{x}_i$, and the same proofs π^i, for all $i \in \mathcal{I}$. In this case, the only difference between $\mathsf{Real}, \mathcal{H}$ is in the proof π^C, and the encodings $\mathbf{x}_i$ for $i \notin \mathcal{I}$. Let $\mathcal{W}$ denote the wire values of $\widehat{C}$ in the real world, and $\mathcal{W}_{\mathcal{H}}$ denote the wire values in the hybrid $\mathcal{H}$. Notice that $\mathcal{W}, \mathcal{W}_{\mathcal{H}}$ contain $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and therefore any q^* bits of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ can be generated by probing q^* bits of $\mathcal{W}, \mathcal{W}_{\mathcal{H}}$. Moreover, every q^* bits of π^C can be generated by a function in $\mathcal{L}$. Therefore, the answers to all the verifier queries to π^C can be answered by applying a function from $\mathcal{L}$ to $\mathcal{W}$ (in Real) or $\mathcal{W}_{\mathcal{H}}$ (in $\mathcal{H}$). The $\mathcal{L}$-average-case relaxed leakage-resilience of $(\mathsf{Comp}, \mathsf{E})$ guarantees that in both cases the answers are ϵ-statistically close to the output of the leakage function on the simulated wire values that the simulator Sim_{in} of the LRCC would have generated. Using the triangle inequality, $\mathsf{SD}(\mathsf{Real}, \mathcal{H}) \leq 2\epsilon$.

$\mathsf{SD}(\mathcal{H}, \mathsf{Ideal}) \leq \epsilon$. The only different between $\mathsf{Ideal}, \mathcal{H}$ is that in Ideal the wires $\mathcal{W}'$ of $\widehat{C}$ (including the encodings $\mathbf{x}_i$ for $i \notin \mathcal{I}$) were generated by the simulator Sim_{in} of the LRCC, whereas in $\mathcal{H}$ the wire value $\mathcal{W}_{\mathcal{H}}$ were obtained by evaluating $\widehat{C}$ on encodings generated by the prover. Notice that for every $i \in \mathcal{I}$, the encoding $\mathbf{x}_i$ contained in $\mathcal{W}', \mathcal{W}_{\mathcal{H}}$ are random, honestly-generated encoding of x'_i, so the average-case relaxed leakage-resilience property of the LRCC guarantees that $\mathsf{SD}(\mathcal{H}, \mathsf{Ideal}) \leq \epsilon$.

The claim now follows from the triangle inequality. □

Remark on Witness-Indistinguishability Against *Adaptive* Verifiers. Lemma 7.28 only shows witness-indistinguishability against (possibly malicious) *non-adaptive* verifiers. As in [IWY16], we can combine witness-indistinguishability against non-adaptive verifiers with a result of [CDD+01] to show that Construction 7.24 is in fact witness-indistinguishbale against *adaptive* verifiers. (The error increases by roughly ℓ^{2q^*}, where ℓ is the proof length, but similar to [IWY16] by setting the error of the original WI-PCPP scheme to be sufficiently small, we can obtain a negligibly small error despite this security loss.)

Remark on Efficient Simulation. The simulator Sim of Construction 7.26 is inefficient because it emulates the simulator Sim_{in} of the average-case LRCC, which is inefficient. However, Remark 7.21 guarantees that the emulation of Sim_{in} is efficient when the simulator is given a w' such that $(x, w') \in \mathcal{R}_L$. Consequently, the simulation is efficient if, given a subset $\{x_i\}_{i \in I}$ of bits in an $x \in L_{\mathcal{R}_L}$, one can find in polynomial time $(x', w') \in \mathcal{R}_L$ such that $x'_i = x_i$ for every $i \in I$.

Remark on the Assumptions Used in the WI-PCPP Construction. Our WI-PCPP construction (Construction 7.24) is based on a standard PCPP for languages in P (which is used to prove closeness to the languages $L^0_{\mathsf{E}}, L^1_{\mathsf{E}}$), a PCPP for $\mathsf{3SAT}$ with complexity $(\mathcal{L}, q^*)$, an average-case LRCC with respect to some encoding scheme E that resists leakage from $\mathcal{L}$, and an encoding scheme E that has "good" (e.g., constant) relative distance and resists leakage from $\mathcal{L} \circ \mathsf{Shallow}\left(O\left(s \cdot \widehat{n}\left(1, t, t_{\text{in}}\right)\right), 3, O\left(s \cdot \widehat{n}^2\left(1, t, t_{\text{in}}\right)\right)\right)$ for some $s = \text{poly}(n)$. If we take $\mathcal{L} = \mathsf{AC}^0[\oplus]$ then all of the building blocks, *except the encoding scheme* E, are known (given such an encoding scheme, an average-case LRCC against $\mathcal{L}$-leakage exists by Lemma 7.20). Thus, we have reduced the task of constructing a WI-PCPP with a non-adaptive verifier to the task of constructing the encoding scheme E.

Acknowledgements

The author is extremely grateful to Yuval Ishai and Yael Kalai for endless useful discussions and insightful comments throughout the different stages of this work.

This work was supported in part by ISF grants 1861/16, 1399/17 and 1709/14, AFOSR Award FA9550-17-1-0069, ERC starting grant 259426, ,ERC grant 742754, and BSF grants 2012378 and 2012366.

References

[ADF16] Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $O(1/n \log(n))$ leakage rate. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 586–615, 2016.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 14–23, 1992.

[ARW17] Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36, 2017.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 2–13, 1992.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[BBC+17] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 551–579, 2017.

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 14:1–14:17, 2018.

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.

[BCF+17] Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Zero knowledge protocols from succinct constraint detection. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, pages 172–206, 2017.

[BCG+11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 722–739, 2011.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasi-linear size zero knowledge from linear-algebraic PCPs. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 33–64, 2016.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 31–60, 2016.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 1–10, 2004.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 1–18, 2001.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 521–536, 2006.

[CDD+01] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, pages 262–279, 2001.

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 230–247, 2012.

[Din06] Irit Dinur. The PCP theorem by gap amplification. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 241–250, 2006.

[DPS12] Yvo Desmedt, Josef Pieprzyk, and Ron Steinfeld. Active security in multiparty computation over black-box groups. In *Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings*, pages 503–521, 2012.

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 135–156, 2010.

[GIM+16] Vipul Goyal, Yuval Ishai, Hemanta K. Maji, Amit Sahai, and Alexander A. Sherstov. Bounded-communication leakage resilience via parity-resilient circuits. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 1–10, 2016.

[GIW17] Daniel Genkin, Yuval Ishai, and Mor Weiss. How to construct a leakage-resilient (stateless) trusted party. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, pages 209–244, 2017.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[GR10] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 59–79, 2010.

[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 31–40, 2012.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 463–481, 2003.

[IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 121–145, 2014.

[IWY15] Yuval Ishai, Mor Weiss, and Guang Yang. Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. *IACR Cryptology ePrint Archive*, 2015:1055, 2015.

[IWY16] Yuval Ishai, Mor Weiss, and Guang Yang. Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 3–32, 2016. Full version available at http://eprint.iacr.org/2015/1055.

[JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 41–58, 2010.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 496–505, 1997.

[Mie08] Thilo Mie. Polylogarithmic two-round argument systems. *J. Mathematical Cryptology*, 2(4):343–363, 2008.

[Mil14] Eric Miles. Iterated group products and leakage resilience against NC^1. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 261–268, 2014.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, pages 278–296, 2004.

[MV13] Eric Miles and Emanuele Viola. Shielding circuits with groups. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 251–260, 2013.

[Rot12] Guy N. Rothblum. How to compute under AC^0 leakage without secure hardware. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 552–569, 2012.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.