Making the Best of a Leaky Situation: Zero-Knowledge PCPs from Leakage-Resilient Circuits

Yuval Ishai* Mor Weiss† Guang Yang‡

Abstract

A Probabilistically Checkable Proof (PCP) allows a randomized verifier, with oracle access to a purported proof, to probabilistically verify an input statement of the form "$x \in L$" by querying only few bits of the proof. A zero-knowledge PCP (ZKPCP) is a PCP with the additional guarantee that the view of any verifier querying a bounded number of proof bits can be efficiently simulated given the input x alone, where the simulated and actual views are statistically close.

Originating from the first ZKPCP construction of Kilian et al. (STOC '97), all previous constructions relied on locking schemes, an unconditionally secure oracle-based commitment primitive. The use of locking schemes makes the verifier *inherently* adaptive, namely, it needs to make at least two rounds of queries to the proof.

Motivated by the goal of constructing non-adaptively verifiable ZKPCPs, we suggest a new technique for compiling standard PCPs into ZKPCPs. Our approach is based on leakage-resilient circuits, which are circuits that withstand certain "side-channel" attacks, in the sense that these attacks reveal nothing about the (properly encoded) input, other than the output. We observe that the verifier's oracle queries constitute a side-channel attack on the wire-values of the circuit verifying membership in L, so a PCP constructed from a circuit resilient against such attacks would be ZK. However, a leakage-resilient circuit evaluates the desired function *only if* its input is properly encoded, i.e., has a specific structure, whereas by generating a "proof" from the wire-values of the circuit on an *ill-formed* "encoded" input, one can cause the verification to accept inputs $x \notin L$ *with probability 1*. We overcome this obstacle by constructing leakage-resilient circuits with the additional guarantee that ill-formed encoded inputs are detected. Using this approach, we obtain the following results:

- We construct the first *witness-indistinguishable* PCPs (WIPCP) for NP with non-adaptive verification. WIPCPs relax ZKPCPs by only requiring that different witnesses be indistinguishable. Our construction combines strong leakage-resilient circuits as above with the PCP of Arora and Safra (FOCS '92), in which queries correspond to side-channel attacks by shallow circuits, and with correlation bounds for shallow circuits due to Lovett and Srivinasan (RANDOM '11).
- Building on these WIPCPs, we construct non-adaptively verifiable *computational* ZKPCPs for NP in the common random string model, assuming that one-way functions exist.
- As an application of the above results, we construct *3-round* WI and ZK proofs for NP in a distributed setting in which the prover and the verifier interact with multiple servers of which t can be corrupted, and the total communication involving the verifier consists of $\mathsf{poly}\log(t)$ bits.

*Department of Computer Science, Technion, Haifa, Israel, and Department of Computer Science, UCLA, LA, California, USA.

†Department of Computer Science, Technion, Haifa, Israel.

‡Institue for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China.

Contents

1 Introduction

In this work we study probabilistically checkable proofs with zero-knowledge properties, and establish a connection between such proofs and leakage-resilient circuits. Before describing our main results, we first give a short overview of these objects.

Probabilistically Checkable Proof (PCP) systems [1, 2] are proof systems that allow an efficient randomized verifier, with oracle access to a purported proof generated by an efficient prover (that is also given the witness), to probabilistically verify claims of the form "$x \in L$" (for an NP-language L) by probing only few bits of the proof. The verifier accepts the proof of a true claim with probability 1 (the *completeness* property), and rejects false claims with high probability (the probability that the verifier accepts a false claim is called *the soundness error*). The celebrated PCP theorem [1, 2, 8] asserts that any NP language admits a PCP system with soundness error $1/2$ in which the verifier reads only a *constant* number of proof bits (soundness can be amplified using repetition). Moreover, the verifier is *non-adaptive*, namely its queries are determined solely by his randomness (a verifier is *adaptive* if each of his queries may also depend on the oracle answers to previous queries).

A very different kind of proofs are zero-knowledge (ZK) proofs [15], namely proofs that carry no extra knowledge other than being convincing. Combining the advantages of ZK proofs and PCPs, a *zero-knowledge PCP* (ZKPCP) is defined similarly to a traditional PCP, except that the proof is also randomized and there is the additional guarantee that the view of any (possibly malicious) verifier who makes a bounded number of queries can be efficiently simulated up to a small statistical distance.

Previous ZKPCP constructions [24, 19, 21] are obtained from standard (i.e., non-ZK) PCPs in two steps. First, the standard PCP is transformed into a PCP with a weaker "honest-verifier" ZK guarantee (which is much easier to achieve than full-fledged ZK). Then, this "honest-verifier" ZKPCP is combined with an unconditionally secure oracle-based commitment primitive called a "locking scheme" [24, 19]. This transformation yields ZKPCPs for NP with statistical ZK against *query-bounded malicious* verifiers, namely ones who are only limited to asking at most $p(|x|)$ queries, for some *fixed* polynomial p that is much smaller than the proof length, but can be much bigger than the (polylogarithmic) number of queries asked by the honest verifier.

A common limitation of all previous ZKPCP constructions is that they require *adaptive* verification, even if the underlying non-ZK PCP can be non-adaptively verified. This raises the natural question of constructing PCPs that can be *non-adaptively* verified, and guarantee ZK against *malicious* verifiers. We note that the adaptivity of the verifier is inherent to any locking-scheme-based ZKPCP, since the unconditional security of locking schemes makes their opening inherently adaptive. Therefore, constructing ZKPCPs that can be verified non-adaptively requires a new approach towards ZKPCP construction. An additional advantage of eliminating the use of locking schemes is the possibility of constructing ZKPCPs preserving the proof length (which is important when these are used for cryptographic applications as described below), since locking schemes inherently incur a polynomial blow-up in the PCP length.

Motivated by these goals, we suggest a new approach for the construction of ZKPCPs. We apply *leakage-resilient circuit compilers* (LRCCs) to construct *witness-indistinguishable PCPs* (WIPCPs) for NP, a weaker variant of ZKPCPs in which the simulation is not required to be efficient. We then apply the so-called "FLS technique" [12] to convert these WIPCPs into *computational* ZKPCPs (CZKPCPs) in the common random string (CRS) model, based on the existence of one-way functions (OWFs). In such a CZKPCP, the view of any query-bounded probabilistic polynomial-time (PPT) verifier can be *efficiently* simulated, in a way which is *computationally indistinguishable* from the actual view.

Informally, an LRCC compiles any circuit into a new circuit that operates on encoded inputs, and

withstands side-channel attacks in the sense that these reveal nothing about the (properly encoded) input, other than what follows from the output. Works on LRCCs obtained information-theoretic security for different classes of leakage functions [20, 11, 30, 10, 17, 28].

Works on LRCC compilers considered different restrictions on the class of leakage functions being tolerated. One line of work, initiated by Ishai et al. [20], restricts the complexity class from which leakage functions are chosen. In these constructions [20, 11, 28] the compiled circuit operates on secret-shares, and its leakage-resilience is reduced to leakage functions being unable to distinguish secret shares of different values. A different approach, initiated by Micali and Reyzin [27], considers leakage that is "local" in the sense that the leakage functions operate on disjoint sets of wires of the circuit (see, e.g., [27, 16, 22, 10, 17]).

Other than the theoretical interest in this question, our study of PCPs with ZK properties is motivated by their usefulness for cryptographic applications. For instance, ZKPCPs are the underlying combinatorial building blocks of succinct zero-knowledge arguments, which have been the subject of a large body of recent work (see, e.g., [3, 4, 5] and references therein).

A more direct application of WIPCPs and ZKPCPs is for implementing efficiently verifiable zero-knowledge proofs in a distributed setting involving a prover, verifier, and multiple (potentially corrupted) servers. In this setting a prover can distribute a ZKPCP between the servers, allowing the verifier to efficiently verify the claim by polling a small random subset of the servers.[1] In this and similar situations, ZKPCPs that only offer security against an honest verifier are not sufficient for protecting against *colluding servers*. We use our non-adaptively verifiable WIPCPs and CZKPCPs for NP to construct *3-round* WI and CZK proofs for NP in this distributed setting, in which the total communication with the verifier is *sublinear* in the input length. The WI proofs are unconditional, whereas the CZK proofs are based on the existence of OWFs. This should be contrasted with standard sublinear ZK arguments, that require at least 4 rounds of interaction, and require the existence of collision resistant hash functions. We refer the reader to, e.g., [19] for additional discussion of ZKPCPs and their applications.

1.1 Our Results and Techniques

We now give a more detailed account of our results, and the underlying techniques.

From LRCCs and PCPs to WIPCPs. Let L be an NP-language with a corresponding NP-relation $\mathcal{R}_L$, and a boolean circuit C verifying $\mathcal{R}_L$. Recall that the prover P in a PCP system for $\mathcal{R}_L$ is given the input x and a witness y for the membership of x in L, and outputs a proof π that is obtained by applying some function f_P to x, y. For our purposes, it would be more convenient to think of f_P as a function of the *entire wire values* w of C, when evaluated on x, y. In a ZKPCP, few bits in the output of f_P should reveal essentially nothing about the wire values w, i.e., C should withstand "leakage" from f_P. In general, we cannot assume that C has this guarantee, but using an LRCC, C can be compiled into a circuit $\hat{C}$ with this property. Informally, an LRCC is associated with a function class $\mathcal{L}$ (the *leakage class*) and a (randomized) input encoding scheme E, and compiles a deterministic circuit C into a deterministic circuit $\hat{C}$, that emulates C, but operates on an encoded input. It is leakage-resilient in the following sense: for any input z for C, and any $\ell \in \mathcal{L}$, the output of ℓ on the wire values of $\hat{C}$, when evaluated on $\mathsf{E}(z)$, reveals nothing other than $C(z)$. This is formalized in the simulation-based paradigm (i.e., the wire-values of $\hat{C}$ can be efficiently simulated given only $C(z)$).

We establish a connection between ZKPCPs and LRCCs. Assume the existence of an LRCC associated with a leakage class $\mathcal{L}$, such that any restriction $f_P^{\mathcal{I}}$ of f_P to a "small" subset $\mathcal{I}$ of its

[1] Unlike the ZKPCP model, the answers of malicious servers may depend on the identity of the verifier's queries, but this can be overcome using techniques of [21].

outputs satisfies $f_P^{\mathcal{I}} \in \mathcal{L}$. Then the oracle answers to the queries of a query-bounded verifier V correspond to functions in $\mathcal{L}$, since for every possible set $\mathcal{I}$ of oracle queries, the answers are $f_P^{\mathcal{I}}(w)$. Therefore, if w is the wire values of a *leakage-resilient* circuit then the system is ZK. This gives a general method of transforming standard PCPs into ZKPCPs: P, V replace $C_x = C(x, \cdot)$ (i.e., C with x hard-wired into it) with $\hat{C}_x$; and P proves that $\hat{C}_x$ is satisfiable by generating the PCP π from the wire values of $\hat{C}_x$.

This transformation crucially relies on the fact that $\hat{C}_x$ emulates C_x (e.g., if $\hat{C}_x$ always outputs 1 then the resultant PCP system is not sound). However, in current constructions of LRCCs (e.g., [20, 11, 28]), this holds *only if the encoded input of* $\hat{C}_x$ *was honestly generated*. Moreover, there always exists a choice of an *ill-formed* "encoding" that satisfies $\hat{C}_x$ (i.e., causes it to output 1). In our case the *prover* generates the encoded input of $\hat{C}_x$ (the verifier does not know this input), so that a malicious prover is able to pick an ill-formed "encoding" that satisfies $\hat{C}_x$, causing the verifier to accept *with probability 1*. Therefore, soundness requires that if C_x is not satisfiable, then there exists *no* satisfying input for $\hat{C}_x$ (either well- or ill-formed), a property which we call *SAT-respecting*. The main tool we use are *SAT-respecting* LRCCs, which we construct based on the LRCC of Faust et al. [11]. To describe our construction, we first need to delve deeper into their construction.

The LRCC of [11] transforms a circuit C into a circuit $\hat{C}$ that operates on encodings generated by a linear encoding scheme, and emulates the operations of C on these encodings. Leakage-resilience against functions in a restricted function class $\mathcal{L}$ is obtained by "refreshing" the encoded intermediate values of the computation after every operation, using encodings of 0. (The LRCCs of [20, 28] operate essentially in the same way.) The input of $\hat{C}$ includes sufficiently many encodings of 0 to be used for the entire computation.[2] However, by providing $\hat{C}$ also with 1-encodings (i.e., encodings of 1), one can change the functionality emulated by $\hat{C}$. (In particular, if the encoding "refreshing" the output gate is a 1-encoding, the output is flipped.) This is not just an artifact of the construction, but rather is *essential* for their leakage-resilience argument. Concretely, to simulate the wire values of $\hat{C}$ *without knowing its input*, the simulator sometimes uses 1-encodings, which rules out the natural solution of verifying that the encodings used for "refreshing" are 0-encodings. We observe that if C were emulated twice, *it would suffice to know that at least one copy used only 0-encodings*, since then $\hat{C}$ is satisfiable only if the honestly-evaluated copy is satisfiable (i.e., C is satisfiable). At first, this may seem as no help at all, but it turns out that by emulating C twice, we can construct what we call a *relaxed* LRCC, which is similar to an LRCC, except that the simulator is *not* required to be efficient. Specifically, assume that before compiling C into $\hat{C}$, we would replace it with a circuit C' that computes C twice, and outputs the AND of both evaluations. Then $\hat{C}'$ (complied from C') would be relaxed leakage-resilient, since an unbounded simulator could simulate the wire values of $\hat{C}'$ by finding a satisfying input z_S for C, and honestly evaluating $\hat{C}'$ on a pair of encodings of z_S. Using a hybrid argument, we prove that functions in $\mathcal{L}$ cannot distinguish the simulated wire values $\mathcal{W}_S$ from the actual wire values $\mathcal{W}_R$ of $\hat{C}'$ when evaluated on a satisfying input z_R. Indeed, we can first replace the input in the *first* copy from z_R to z_S (using the leakage-resilience of the LRCC of [11] to claim that functions in $\mathcal{L}$ cannot distinguish this hybrid distribution from $\mathcal{W}_R$), then do the same in the *second* copy. By replacing the inputs one at a time, we only need to use 1-encodings in a *single* copy.[3] However, holding two copies of the original circuit still does not guarantee that the

[2] Actually, [11] consider a model of *continuous* leakage, in which the circuit is invoked multiple times on different inputs, and maintains a secret state. Their construction uses tamper-proof hardware (called *opaque gates*) to generate the encodings of 0 used for refreshing. We consider the simpler model of *one-time* leakage on circuits that operated on *encoded* inputs [20, 28], and as a result we can incorporate the necessary encodings (used for refreshing) into the encoded input.

[3] This technique is reminiscent of the "2-key trick" of [29], used to convert a CPA-secure encryption scheme into a CCA-secure one.

evaluation in at least one of them uses only 0-encodings.

The natural solution would again be to add a sub-circuit verifying that the encodings used are 0-encodings, but this sub-circuit should hide the identity of the "correctly evaluated" copy. This is because the hybrid argument described above first uses 1-encodings in the first copy (and 0-encodings in the second), and then uses 1-encodings in the second copy (and only 0-encodings in the first). Therefore, if functions in $\mathcal{L}$ could determine which copy uses only 0-encodings, they could also distinguish between the hybrids. Instead, we describe an "oblivious" checker $\mathcal{T}_0$, which at a high-level operates as follows. To check that *either* the first *or* the second copy use only 0-encodings, it checks that for every pair of encodings, one from the first copy, and one from the second, the product of the encoded values is 0. To guarantee that leakage on $\mathcal{T}_0$ reveals no information regarding *which* copy uses only 0-encodings, we use the LRCC of [11] to compile $\mathcal{T}_0$ into a leakage-resilient circuit $\hat{\mathcal{T}}_0$. This introduces the additional complication that now we must also verify the encodings used to "refresh" the computation in $\hat{\mathcal{T}}_0$ (otherwise 1-encodings may be used, potentially changing the functionality of $\hat{\mathcal{T}}_0$ and rendering it useless). However, since $\hat{\mathcal{T}}_0$ does not operate directly on the *inputs to* $\hat{C}'$ (it operates only on the encodings used for "refreshing"), we show that the "refreshing" encodings used in $\hat{\mathcal{T}}_0$ can be checked directly (by decoding the encoded values and verifying that they are 0). Additional technicalities arise since introducing these additional components prevents us from using the LRCC of [11] as a black box (see Section 3 for additional details on the analysis). Finally, we note that our circuit-compiler is *relaxed*-leakage-resilient because in all hybrids, we need the honestly-evaluated copy to be satisfied, so the simulator needs to find a satisfying input for C. This is also the reason that we get *WI*PCPs instead of *ZK*PCPs. If we had a SAT-respecting LRCC, the transformation described above would give a ZKPCP. However, we show in Section 6 that known LRCCs withstanding global leakage [20, 11, 28] cannot be transformed into SAT-respecting *non-relaxed* LRCCs (i.e., LRCCs with an *efficient* simulator), unless $\mathsf{NP} \subseteq \mathsf{BPP}$. Intuitively, this is because these constructions admit a simulator which is *universal* in the sense that it simulates the wire values of the compiled circuit *without knowing the leakage function*, and the simulated values "fool" *all* functions in $\mathcal{L}$. Combining such a SAT-respecting LRCC with PCPs for NP (through the transformation described above) would give a BPP algorithm of deciding any NP-language.

Constructing WIPCPs for NP. Recall that our general transformation described above relied on f_P being in the function class $\mathcal{L}$ that is associated with the SAT-respecting relaxed-LRCC. We observe that the PCP system of Arora and Safra [2] has the property that every "small" subset of proof bits can be generated using a low-depth circuit of polynomial size over the operations $\wedge, \vee, \neg, \oplus$, with "few" $\oplus$ gates. We use recent correlation bounds of Lovett and Srivinasan [26], which roughly state that such circuits have negligible correlation with the boolean function that counts the number of 1's modulo 3 in its input, to construct a SAT-respecting circuit compiler that is relaxed leakage-resilient with respect to this function class. Combining this relaxed LRCC with our general transformation, we prove the following, where NA-WIPCP denotes the class of all NP-languages that have a PCP system with a negligible soundness error, polynomial-length proofs, a non-adaptive honest verifier that queries poly-logarithmically many proof bits, and guarantee WI against (adaptive) malicious verifiers querying a fixed polynomial number of proof bits.

Theorem 1.1 (NA-WIPCPs for NP)**.** $\mathsf{NP} = \mathsf{NA} - \mathsf{WIPCP}$.

Constructing CZKPCPs for NP. Using a general technique of Feige et al. [12], and assuming the existence of OWFs, we transform our WIPCP into a CZKPCP in the CRS model, in which the PCP prover and verifier both have access to a common random string. Concretely, we prove the following result, where NA-CZKPCP corresponds exactly to the class NA-WIPCP, except that the WI property is replaced with CZK in the CRS model.

Corollary 1.2 (NA-CZKPCPs for NP)**.** *Assume that OWFs exist. Then* $\mathsf{NP} = \mathsf{NA} - \mathsf{CZKPCP}$.

At the end of Section 4 we describe a simple alternative approach for constructing CZKPCPs, which applies a PCP on top of a standard non-interactive zero-knowledge (NIZK) proof. However, this alternative relies on stronger assumptions (e.g. existence of trapdoor functions [12]) than our main construction which only relies on a OWF.

2 Preliminaries

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 $F \circ G := \{f \circ g : f \in F, g \in G\}$. Vectors will be denoted by boldface letters (e.g., $\mathbf{a}$). 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. For a function family $\mathcal{L}$, we sometimes use the term "leakage family $\mathcal{L}$", or "leakage class $\mathcal{L}$". In the following, n usually denotes the input length, m usually denotes the output length, d, s denote depth and size, respectively (e.g., of circuits, as defined below), t is used to count $\oplus$ gates, and σ is a security parameter. We assume that standard cryptographic primitives (e.g., OWFs) are secure against non-uniform adversaries.

Definition 2.1 (Leakage-indistinguishability of distributions)**.** Let D, D' be finite sets, $\mathcal{L} = \{\ell : D \to D'\}$ be a family of leakage functions, and $\epsilon > 0$. We say that two distributions X, Y over D are $(\mathcal{L}, \epsilon)$-leakage-indistinguishable, if for any function $\ell \in \mathcal{L}$, $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$.

Remark 2.2. In case $\mathcal{L}$ consists of functions over a union of domains, we say that X, Y over D are $(\mathcal{L}, \epsilon)$-leakage-indistinguishable if $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$ for every function $\ell \in \mathcal{L}$ *with domain* D.

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 probabilistic polynomial-time (PPT) algorithm that given a message $x \in \Sigma^n$ outputs an encoding $\hat{x} \in \Sigma^{\hat{n}}$ for some $\hat{n} = \hat{n}(n)$; and the *decoding algorithm* Dec is a deterministic algorithm, that given an $\hat{x}$ of length $\hat{n}$ in the image of Enc, outputs an $x \in \Sigma^n$. Moreover, $\Pr[\mathsf{Dec}(\mathsf{Enc}(x)) = x] = 1$ for every $x \in \Sigma^n$. We say that E is *onto*, if Dec is defined for every $x \in \Sigma^{\hat{n}(n)}$.

An encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ over $\mathbb{F}$ is *linear* if for every n, n divides $\hat{n}(n)$, and there exists a decoding vector $\mathbf{r}^{\hat{n}(n)} \in \mathbb{F}^{\hat{n}(n)/n}$ such that the following holds for every $x \in \mathbb{F}^n$. First, every encoding $\mathbf{y}$ in the support of $\mathsf{Enc}(x)$ can be partitioned into n equal-length parts $\mathbf{y} = (\mathbf{y}^1, ..., \mathbf{y}^n)$. Second, $\mathsf{Dec}(\mathbf{y}) = (\langle \mathbf{r}^{\hat{n}(n)}, \mathbf{y}^1 \rangle, ..., \langle \mathbf{r}^{\hat{n}(n)}, \mathbf{y}^n \rangle)$ (where "$\langle \cdot, \cdot \rangle$" denotes inner product). Given an encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ over $\mathbb{F}$, and $n \in \mathbb{N}$, we say that a vector $\mathbf{v} \in \mathbb{F}^{\hat{n}(n)}$ is *well-formed* if $\mathbf{v} \in \mathsf{Enc}(0^n)$.

Parameterized encoding schemes. We consider encoding schemes in which the encoding and decoding algorithms are given an additional input 1^σ, which is used as a security parameter. Concretely, the encoding length depends also on σ (and not only on n), i.e., $\hat{n} = \hat{n}(n, \sigma)$, and for every σ the resultant scheme is an encoding scheme (in particular, for every $x \in \Sigma^n$ *and every* $\sigma \in \mathbb{N}$, $\Pr[\mathsf{Dec}(\mathsf{Enc}(x, 1^\sigma), 1^\sigma) = x] = 1$). We call such schemes *parameterized*. A parameterized encoding scheme is *onto* if it is onto for every σ. It is linear if it is linear for every σ (in particular, there exist decoding vectors $\{\mathbf{r}^{\hat{n}(n,\sigma)}\}$). For $n, \sigma \in \mathbb{N}$, a vector $\mathbf{v} \in \mathbb{F}^{\hat{n}(n,\sigma)}$ is *well-formed* if $\mathbf{v} \in \mathsf{Enc}(0^n, 1^\sigma)$. We will only consider parameterized encoding schemes, and therefore when we say "encoding scheme" we mean a *parameterized* encoding scheme.

Definition 2.3 (Leakage-indistinguishability of functions and encodings)**.** Let $\mathcal{L}$ be a family of leakage functions, and $\epsilon > 0$. 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 E is $(\mathcal{L}, \epsilon)$-*leakage-indistinguishable* if for every large enough $\sigma \in \mathbb{N}$, $\mathsf{Enc}(\cdot, 1^\sigma)$ is $(\mathcal{L}, \epsilon)$-leakage indistinguishable.

CIRCUITS. We consider arithmetic circuits C over the field $\mathbb{F}$ and the set $X = \{x_1, ..., x_n\}$ of variables. C is a directed acyclic graph whose vertices are called *gates* and whose edges are called *wires*. The wires of C are labled with functions over X. Every gate in 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 $\alpha \in \mathbb{F}$ and referred to as a const_α *gate*. Following [11], all other gates are labeled by one of the following functions $+, -, \times, \mathsf{copy}$ or id, where $+, -, \times$ are the addition, subtraction, and multiplication operations of the field (i.e., the outcoming 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; copy vertices have fan-in 1 and fan-out 2, where the labels of the outcoming edges carry the same function as the incoming edge; and id vertices that have fan-in and fan-out 1, and the label of the outcoming edge is the same as the incoming edge. We write $C : \mathbb{F}^n \to \mathbb{F}^m$ to indicate that C is an arithmetic circuit over $\mathbb{F}$ with n inputs and m outputs. The *size* of a circuit C, denoted $|C|$, is the number of wires in C, together with input and output gates. $\mathsf{Shallow}(d, s)$ denotes the class of all depth-d, size-s, arithmetic circuits over $\mathbb{F}$. Similarly, $\mathsf{ShallowB}(d, s)$ denotes the class of all depth-d, size-s, boolean circuits with $\wedge, \vee$ gates (replacing the $+, -, \times$ gates of arithmetic circuits), id, copy, const_0, and const_1 gates (with fan-in and fan-out as specified above), and $\neg$ gates with fan-in and fan-out 1. We will sometimes be interested in the input and output lengths of these circuit families. Therefore, we denote circuits in $\mathsf{Shallow}(d, s)$ with input length n and output length m by $\mathsf{Shallow}(n, m, d, s)$. Similarly, we use $\mathsf{ShallowB}(n, m, d, s)$ to denotes circuits in $\mathsf{ShallowB}(d, s)$ with input length n and output length m. We also use the notations $\mathsf{Shallow}(n, d, s) = \cup_{m \in \mathbb{N}} \mathsf{Shallow}(n, m, d, s)$, and $\mathsf{ShallowB}(n, d, s) = \cup_{m \in \mathbb{N}} \mathsf{ShallowB}(n, m, d, s)$. Somewhat abusing notation, we use the same notations to denote the *families of functions* computable by circuits in the respective class of circuits. AC^0 denotes all constant-depth and polynomial-sized boolean circuits over *unbounded fan-in and fan out* $\wedge, \vee, \neg, \mathsf{const}_0$ and const_1 gates.

Definition 2.4. For $\mathbb{F} = \mathbb{F}_2$, a circuit $C : \mathbb{F}^n \to \mathbb{F}$ over $\mathbb{F}_2$ is *satisfiable* if there exists an $x \in \mathbb{F}^n$ such that $C(x) = 1$. For $\mathbb{F} \neq \mathbb{F}_2$, C is *satisfiable* if there exists an $x \in \mathbb{F}^n$ such that $C(x) = 0$.

2.1 Circuit Compilers

We define the notion of a circuit compiler. Informally, it consists of an encoding scheme and a compiler algorithm, that compiles a given circuit into a circuit operating on encodings, and emulating the original circuit. Formally,

Definition 2.5 (Circuit compiler over $\mathbb{F}$)**.** 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 is a PPT encoding algorithm that given a vector $x \in \mathbb{F}^n$, and 1^σ, outputs a vector $\hat{x}$. We assume that $\hat{x} \in \mathbb{F}^{\hat{n}}$ for some $\hat{n} = \hat{n}(n, \sigma)$.
- Comp is a polynomial-time algorithm that given an arithmetic circuit C over $\mathbb{F}$ outputs an arithmetic circuit $\hat{C}$.

We require that $(\mathsf{Comp}, \mathsf{E})$ satisfy the following *correctness* requirement. For any arithmetic circuit C, and any input x for C, we have $\Pr\left[\hat{C}(\hat{x}) = C(x)\right] = 1$, where $\hat{x}$ is the output of $\mathsf{Enc}\left(x, 1^{|C|}\right)$.

A *boolean circuit compiler* is a circuit compiler over $\mathbb{F}_2$.

We consider circuit compilers that are also "sound", meaning that satisfying (possibly *ill formed*) inputs for the compiled circuit exist only if the original circuit is satisfiable.

Definition 2.6 (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 $\hat{C} = \mathsf{Comp}(\mathsf{C})$ is satisfiable then C is satisfiable, i.e., if $\hat{C}(\hat{x}^*) = 0$ for some $\hat{x}^* \in \mathbb{F}^{\hat{n}}$, then there exists an $x \in \mathbb{F}^n$ such that $C(x) = 0$. (For $\mathbb{F} = \mathbb{F}_2$, we require that if $\hat{C}$ outputs 1 on some input, then so does C.)

2.2 Leakage-Resilient Circuit Compilers (LRCCs)

We consider circuit compilers whose outputs are *leakage resilient* for a class $\mathcal{L}$ of functions, in the following sense. For every "not too large" circuit C, and every input x for C, the wire values of the compiled circuit $\hat{C}$, when evaluated on a random encoding $\hat{x}$ of x, can be simulated given only the output of C; and functions in $\mathcal{L}$ cannot distinguish between the actual and simulated wire values.

Notation 2.7. For a Circuit C, a leakage function $\ell : \mathbb{F}^{|C|} \to \mathbb{F}^m$ for some natural m, 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 $[C, x]$.

Definition 2.8 (Relaxed LRCC)**.** Let $\mathbb{F}$ 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 $(\mathcal{L}, \epsilon(n), \mathsf{S}(n))$-relaxed leakage-resilient if there exists an algorithm Sim such that the following holds. For all sufficiently large n's, 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 $\mathsf{SD}\left(\ell\left[\mathsf{Sim}\left(C, C(x)\right)\right], \ell\left[\hat{C}, \hat{x}\right]\right) \leq \epsilon(|x|)$, where $\hat{x} \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$.

Definition 2.8 is relaxed in the sense that (unlike [20, 11, 28]) Sim is not required to be efficient.

The error in Definitions 2.6 and 2.8 is defined with relation to the input length n. Both definitions can be naturally extended such that the compiler is also given a security parameter κ, and the error depends on κ (and possibly also n).

3 SAT-Respecting Relaxed LRCC

In this section we construct a SAT-respecting relaxed LRCC. We first describe a relaxed LRCC over any finite field $\mathbb{F} \neq \mathbb{F}_2$, then use its instantiation over $\mathbb{F}_3$ to construct a boolean relaxed LRCC (which we later use to construct WIPCPs and CZKPCPs). Our starting point is the circuit-compiler of Faust et al. [11], which we denote by $\left(\mathsf{Comp}^{\mathrm{FRRTV}}, \mathsf{E}^{\mathrm{FRRTV}}\right)$. They present a general circuit-compiler that guarantees correctness, and a stronger notion of leakage-resilience (informally, that the wire values of the compiled circuit can be *efficiently* simulated). However, the correctness of their construction relies on the assumption that the inputs to the compiled circuit are honestly encoded. Therefore, their construction is not SAT-respecting, since by using ill-formed encoded inputs one can cause the compiled circuit to output arbitrary values, *even if other than that the compiler was honestly applied to the original circuit*. We describe a method of generalizing their construction such that the circuit-compiler is also SAT-respecting. We first give a high-level overview of the compiler of [11]. (See Appendix A for a more detailed description of this LRCC.)

Gadgets. On input a circuit C, our compiler, and that of $\mathsf{Comp}^{\mathrm{FRRTV}}$, replace every wire of C with a *bundle* of wires, and every gate in C with a *gadget*. More specifically, 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 G is associated with a set M_G of "well-formed" masking input bundles (e.g., in the circuit compiler of [11], M_G consists of sets of 0-encodings). For every standard input x, on input a bundle $\mathbf{x}$ encoding x, and *any* masking input bundles $\mathsf{m} \in M_G$, the output of the gadget G should be consistent with the operation on x. For example, if G computes the operation $\times$, then for every standard input $x = (x_1, x_2)$, for every bundle encoding $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ of x according to E, and for every masking input bundles $\mathsf{m} \in M_G$, $G(\mathbf{x}, \mathsf{m})$ is a bundle encoding $x_1 \times x_2$ according to E. Since all the encoding schemes that we consider are onto, we may think of the masking input bundles m as encoding some set mask of values, in which case we say that G takes $|\mathsf{mask}|$ masking inputs. 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_G, *regardless* of the actual values encoded by the masking input bundles.

GADGET-BASED CIRCUIT-COMPILERS. $\hat{C} = \mathsf{Comp}^{\mathrm{FRRTV}}(C)$ is a circuit in which every gate is replaced with the corresponding gadget, and output gates are followed by decoding sub-circuits (computing the decoding function of E). Recall that the gadgets also have masking inputs. These are provided as part of the encoded input of $\hat{C}$, in the following way. $\mathsf{E}^{\mathrm{FRRTV}}$ uses an "inner" encoding scheme $\mathsf{E}^{\mathrm{in}} = \left(\mathsf{Enc}^{\mathrm{in}}, \mathsf{Dec}^{\mathrm{in}}\right)$, where $\mathsf{Enc}^{\mathrm{FRRTV}}$ uses $\mathsf{Enc}^{\mathrm{in}}$ to encode the inputs of C, concatenated with 0^κ for a "sufficiently large" κ (these 0-encodings will be the masking inputs of the gadgets, that are used to achieve privacy); and $\mathsf{Dec}^{\mathrm{FRRTV}}$ uses $\mathsf{Dec}^{\mathrm{in}}$ to decode its input, and discards the last κ symbols.

3.1 The Construction

Let $C : \mathbb{F}^n \to \mathbb{F}$ be the circuit to be compiled. In the following, let $\mathsf{r} = \mathsf{r}(\sigma)$ denote the number of masking inputs used in a circuit compiled according to $\mathsf{Comp}^{\mathrm{FRRTV}}$. Recall that our compiler, given a circuit C, generates two copies C_1, C_2 of C (that operate on two copies of the inputs); compiles C_1, C_2 into circuits $\hat{C}_1, \hat{C}_2$ using $\mathsf{Comp}^{\mathrm{FRRTV}}$; generates the circuit $\hat{C}'$ that outputs the AND of $\hat{C}_1, \hat{C}_2$; generates a circuit $\mathcal{T}_0$ verifying that at least one of the copies $\hat{C}_1, \hat{C}_2$ uses well-formed masking inputs (i.e., its masking inputs are well-formed vectors); compiles $\mathcal{T}_0$ into $\hat{\mathcal{T}}_0$ using $\mathsf{Comp}^{\mathrm{FRRTV}}$; and finally verifies "in the clear" that $\hat{\mathcal{T}}_0$ uses well-formed masking inputs. We now describe these ingredients in more detail.

Our first ingredient checks the validity of the masking inputs used in the compiled circuit $\hat{C}'$. If $\mathsf{m}^1, \mathsf{m}^2$ are masking inputs used in the first and second copies $\hat{C}_1, \hat{C}_2$ in $\hat{C}'$, respectively (i.e., these copies are given encodings of $\mathsf{m}^1, \mathsf{m}^2$), then we compute $v_{ij} = \mathsf{m}^1_i \times \mathsf{m}^2_j$ for every $i, j \in [\mathsf{r}]$, and check that all the v_{ij}'s are zero. To make this check easier, we will use the following "binarization" sub-circuit, which outputs 1 if its input is 0, and outputs 0 on all other input values.

Construction 3.1 ("Binarization" sub-circuit $\mathcal{T}$)**.** $\mathcal{T} : \mathbb{F} \to \mathbb{F}$ is defined as $\mathcal{T}(z) = -\prod_{0 \neq a \in \mathbb{F}} (z - a)$, computed using $O(|\mathbb{F}|)$ many $\times$ and constant gates arranged in $O(\log |\mathbb{F}|)$ layers.

Observation 3.2. *$\mathcal{T}(0) = 1$, and for every $0 \neq z \in \mathbb{F}$, $\mathcal{T}(z) = 0$.*

The sub-circuit $\mathcal{T}_0$ described next checks the masking inputs $\mathsf{m}^1, \mathsf{m}^2$ used in the copies of $\hat{C}$, and outputs 1 if and only if at least one of $\mathsf{m}^1, \mathsf{m}^2$ is the all-zero string. It computes all products of the form $\mathsf{m}^1_i \times \mathsf{m}^2_j$, then applies $\mathcal{T}$ to every product, and computes the products of all these outputs.

Construction 3.3 (Oblivious mask-checking sub-circuit $\mathcal{T}_0$)**.** $\mathcal{T}_0 : \mathbb{F}^{\mathsf{r}} \times \mathbb{F}^{\mathsf{r}} \to \mathbb{F}$ is defined as follows. $\mathcal{T}_0(y, z) = \prod_{i,j \in [\mathsf{r}]} \mathcal{T}(y_i \times z_j)$, computed using a multiplication tree of size $O(\mathsf{r})$ and depth $O(\log \mathsf{r})$ (on top of the multiplication trees used to compute $\mathcal{T}$).

Observation 3.4. *Since the outputs of $\mathcal{T}$ are in $\{0,1\}$, $\mathcal{T}_0(y,z)=1$ if and only if for every $i,j\in[\mathsf{r}]$, $\mathcal{T}(y_i,z_j)=1$ (which by Observation 3.2 happens if and only if $y_i\times z_j=0$), otherwise it outputs 0.*

Our final ingredient is a sub-circuit $\mathcal{T}_V$ checking the masking inputs used in the compiled sub-circuit $\hat{\mathcal{T}}_0$. At a high level, $\mathcal{T}_V$ decodes every masking input; uses $\mathcal{T}$ to map the decoded values into $\{0,1\}$ such that only 0 is mapped to 1; and multiplies all these values, to verify that all the masking inputs are well-formed. In the following, $\mathsf{r}_0=\mathsf{r}_0(\sigma)$ denotes the number of masking inputs used in $\hat{\mathcal{T}}_0$.

Construction 3.5 (Non-oblivious mask-checking sub-circuit $\mathcal{T}_V$)**.** Let $n,\sigma,\kappa\in\mathbb{N}$, $\hat{n}=\hat{n}(n+\kappa,\sigma)$, and $\left\{\mathbf{d}^{\hat{n}}\right\}$ be the decoding vectors of E^{in}. We define the *decoding sub-circuit* $\mathcal{D}_V:\mathbb{F}^{\hat{n}}\to\mathbb{F}$ corresponding to $\mathbf{d}^{\hat{n}}$ as follows: $\mathcal{D}_V(\mathbf{v})=\langle\mathbf{d}^{\hat{n}},\mathbf{v}\rangle$, where $\langle\cdot,\cdot\rangle$ denotes inner-product. $\mathcal{D}_V$ is computed using any correct decoding circuit with $O(\hat{n})$ gates arranged in $O(\log\hat{n})$ layers.

We define $\mathcal{T}_V:\left(\mathbb{F}^{\hat{n}}\right)^{\mathsf{r}_0}\to\mathbb{F}$ as follows: for $\mathbf{R}=(\mathbf{r}_1,...,\mathbf{r}_{\mathsf{r}_0})$ where $\mathbf{r}_i\in\mathbb{F}^{\hat{n}}$ for every $1\leq i\leq\mathsf{r}_0$, $\mathcal{T}_V(\mathbf{R})=\prod_{i\in[\mathsf{r}_0]}\mathcal{T}(\mathcal{D}_V(\mathbf{r}_i))$. $\mathcal{T}_V$ is computed using $O(\mathsf{r}_0)$ many $\times$ gates, arranged in a tree of depth $O(\log\mathsf{r}_0)$ (on top of the sub-circuits $\mathcal{T}\circ\mathcal{D}_V$).

Observation 3.6. *Let $\boldsymbol{R}=(\boldsymbol{r}_1,...,\boldsymbol{r}_{\mathsf{r}_0})\in\left(\mathbb{F}^{\hat{n}}\right)^{\mathsf{r}_0}$, then for every $i\in[\mathsf{r}_0]$, $\mathcal{D}_V(\boldsymbol{r}_i)=v_i$, where v_i is the value that $\boldsymbol{r}_i$ encodes. Since the outputs of $\mathcal{T}$ are in $\{0,1\}$, $\mathcal{T}(\mathcal{D}_V(\boldsymbol{r}_i))=1$ if and only if $v_i=0$, so $\mathcal{T}_V=1$ if and only if all $\boldsymbol{r}_i$'s are well-formed, otherwise it outputs 0.*

Our circuit-compiler (Construction 3.7) uses the ingredients described above. Comp first compiles 2 copies of C, i.e. C_1,C_2, and $\mathcal{T}_0$, into $\hat{C}_1,\hat{C}_2,\hat{\mathcal{T}}_0$ (respectively), using $\mathsf{Comp}^{\mathrm{FRRTV}}$. Then, it generates a flag bit indicating whether $\hat{C}_1,\hat{C}_2$ have the same output, and the masking inputs used in $\hat{C}_1,\hat{C}_2,\hat{\mathcal{T}}_0$ are well-formed. If so, the output is that of $\hat{C}_1$, otherwise it is 1. (Recall that an arithmetic circuit is satisfied iff its output is 0.) The encodings scheme generates encoded inputs for both copies $\hat{C}_1,\hat{C}_2$, as well as sufficient masking inputs to be used in $\hat{C}_1,\hat{C}_2,\hat{\mathcal{T}}_0$.

Construction 3.7 ($(\mathcal{L},\epsilon(n),\mathsf{S}(n))$-LRCC over $\mathbb{F}$)**.** The circuit compiler $(\mathsf{Comp},\mathsf{E}=(\mathsf{Enc},\mathsf{Dec}))$ is defined as follows. Let $\mathsf{r}=\mathsf{r}(\sigma),\mathsf{r}_0=\mathsf{r}_0(\sigma):\mathbb{N}\to\mathbb{N}$ be parameters whose value will be set later.

Let $\mathsf{E}^{\mathsf{in}}=\left(\mathsf{Enc}^{\mathsf{in}},\mathsf{Dec}^{\mathsf{in}}\right)$ be a linear encoding scheme over $\mathbb{F}$, with encodings of length $\hat{n}_{\mathrm{in}}=\hat{n}_{\mathrm{in}}(n,\sigma)$, and decoding vectors $\{\mathbf{d}^{\hat{n}_{\mathrm{in}}}\}$. Then $\mathsf{Enc}(x,1^\sigma)=(\hat{x}_1,\hat{x}_2)$, where $\hat{x}_i\leftarrow\mathsf{Enc}^{\mathsf{in}}\left((x,0^{\mathsf{r}+\mathsf{r}_0}),1^\sigma\right)$; and $\mathsf{Dec}\left((\hat{x}_1,\hat{x}_2),1^\sigma\right)$ computes $\mathsf{Dec}^{\mathsf{in}}(\hat{x}_1,1^\sigma)$, and discards the last $\mathsf{r}+\mathsf{r}_0$ symbols. We use $\hat{n}=\hat{n}(n,\sigma)$ to denote the length of encodings output by Enc, and $\hat{n}_1=\hat{n}_1(\sigma):=\hat{n}(1,\sigma)$. (Notice that $\hat{n}(n,\sigma)=2\hat{n}_{\mathrm{in}}(n+\mathsf{r}+\mathsf{r}_0,\sigma)$.) For $(\hat{x}_1,\hat{x}_2)\leftarrow\mathsf{Enc}(x,1^\sigma)$, we denote $\hat{x}_i=\left(\hat{x}_i^{\mathrm{in}},\mathbf{R}_i,\mathbf{R}_i^0\right)$, where $\hat{x}_i^{\mathrm{in}}$ is the encoding of x, and $\mathbf{R}_i,\mathbf{R}_i^0$ are encodings of $0^{\mathsf{r}},0^{\mathsf{r}_0}$, respectively. ($\mathbf{R}_2^0$ is not used in the construction, but it is part of $\hat{x}_2$ because the same internal encoding scheme $\mathsf{Enc}^{\mathsf{in}}$ is used to generate $\hat{x}_1,\hat{x}_2$.)

Let $\left(\mathsf{Comp}^{\mathrm{FRRTV}},\mathsf{E}^{\mathrm{FRRTV}}\right)$ denote the circuit compiler of [11]. Comp on input a circuit $C:\mathbb{F}^n\to\mathbb{F}$, outputs the circuit $\hat{C}:\mathbb{F}^{\hat{n}(n,|C|)}\to\mathbb{F}$ defined as follows.

- Let C_1,C_2 be two copies of C, $\hat{C}_i=\mathsf{Comp}^{\mathrm{FRRTV}}(C_i)$ for $i=1,2$, and $\hat{\mathcal{T}}_0=\mathsf{Comp}^{\mathrm{FRRTV}}(\mathcal{T}_0)$.
- Let $\mathsf{f}\left(\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1,\mathbf{R}_1^0\right),\left(\hat{x}_2^{\mathrm{in}},\mathbf{R}_2,\mathbf{R}_2^0\right)\right):=\mathcal{T}\left(\hat{C}_1\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1\right)-\hat{C}_2\left(\hat{x}_2^{\mathrm{in}},\mathbf{R}_2\right)\right)\times\hat{\mathcal{T}}_0\left(\left(\mathbf{R}_1,\mathbf{R}_2\right),\mathbf{R}_1^0\right)\times\mathcal{T}_V\left(\mathbf{R}_1^0\right)$. ($\mathsf{f}=1$ if $\hat{C}_1,\hat{C}_2$ have the same output, and in addition the masking inputs used in $\hat{\mathcal{T}}_0$, and at least one of $\hat{C}_1,\hat{C}_2$, are well-formed. Otherwise, $\mathsf{f}=0$.) Then:

$$\hat{C}\left(\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1,\mathbf{R}_1^0\right),\left(\hat{x}_2^{\mathrm{in}},\mathbf{R}_2,\mathbf{R}_2^0\right)\right)=\left(1-\mathsf{f}\left(\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1,\mathbf{R}_1^0\right),\left(\hat{x}_2^{\mathrm{in}},\mathbf{R}_2,\mathbf{R}_2^0\right)\right)\right)$$
$$+\mathsf{f}\left(\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1,\mathbf{R}_1^0\right),\left(\hat{x}_2^{\mathrm{in}},\mathbf{R}_2,\mathbf{R}_2^0\right)\right)\cdot\hat{C}_1\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1,\mathbf{R}_1^0\right)$$

(Notice that the output is $\hat{C}_1\left(\hat{x}_1^{\mathrm{in}},\mathbf{R}_1,\mathbf{R}_1^0\right)$ if $\mathsf{f}=1$, otherwise it is 1.)

Let $\mathsf{r}^{\mathrm{FRRTV}} = \mathsf{r}^{\mathrm{FRRTV}}(\sigma)$ denote the maximal number of masking inputs used in a gadget of $\mathsf{Comp}^{\mathrm{FRRTV}}$, and $\mathsf{S}_0(\mathsf{r})$ denote the size of $\mathcal{T}_0$. Then $\mathsf{r}(\sigma) = \sigma \cdot \mathsf{r}^{\mathrm{FRRTV}}$ and $\mathsf{r}_0(\sigma) = \mathsf{r}^{\mathrm{FRRTV}} \cdot \mathsf{S}_0\left(\mathsf{r}^{\mathrm{FRRTV}}\right)$.

We will show that if the underlying "inner" encoding scheme E^{in} is leakage-indistinguishable against a leakage family $\mathcal{L}_{\mathsf{E}}$, then Construction 3.7 is a SAT-respecting relaxed LRCC against a slightly weaker leakage family $\mathcal{L}$. Formally,

Proposition 3.8 (SAT-respecting relaxed LRCC over $\mathbb{F}$)**.** *Let $\mathcal{L}, \mathcal{L}_{\mathsf{E}}$ be families of functions, $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. Let $\mathsf{E}^{\mathrm{in}} = \left(\mathsf{Enc}^{\mathrm{in}}, \mathsf{Dec}^{\mathrm{in}}\right)$ be a linear, onto, $(\mathcal{L}_{\mathsf{E}}, \epsilon(n))$-leakage-indistinguishable encoding scheme with parameters $n = 1$, σ and $\hat{n} = \hat{n}(\sigma)$, such that $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(7, O\left(\hat{n}^4 \left(\mathsf{S}(n)\right) \cdot \mathsf{S}(n)\right)\right)$. Then there exists a SAT-respecting, $(\mathcal{L}, 8\epsilon(n) \cdot \mathsf{S}(n), \mathsf{S}(n))$-relaxed-LRCC over $\mathbb{F}$. Moreover, For every $C : \mathbb{F}^n \to \mathbb{F}$, the compiled circuit $\hat{C}$ has size $\left|\hat{C}\right| = O\left(|\mathbb{F}| \cdot \hat{n}^5 \left(\mathsf{S}(n)\right) \cdot |C|^2\right)$.*

3.1.1 Roadmap Towards Proving Proposition 3.8

We will show that Construction 3.7 satisfies the requirements of Proposition 3.8. We first analyze the SAT-respecting property, showing that if $\hat{C}$ is satisfiable, then so is C. At a high level, if C is not satisfiable, then one could *potentially* satisfy $\hat{C}$ by providing ill-formed masking inputs to one of the copies $\hat{C}_1, \hat{C}_2$, or to the oblivious masking-checking circuit $\hat{\mathcal{T}}_0$. However, if the masking inputs of $\hat{\mathcal{T}}_0$ are ill-formed, then $\mathcal{T}_V$ resets the flag, so the output is 1 (i.e., $\hat{C}$ is not satisfied). Conditioned on $\hat{\mathcal{T}}_0$ having well-formed masking inputs, the correctness of $\mathsf{Comp}^{\mathrm{FRRTV}}$ (applied to $\hat{\mathcal{T}}_0$), guarantees that if the masking inputs of *both* $\hat{C}_1, \hat{C}_2$ are ill-formed then the flag is reset. Finally, if at least one of $\hat{C}_1, \hat{C}_2$ has well-formed masking inputs, and $\hat{C}$ is satisfied (in particular, the flag is not reset), then there exists an $x \in \mathbb{F}^n$ that satisfies the correctly evaluated copy, and therefore also satisfies C. We note that the encoding scheme should be onto, otherwise computations in compiled circuits may *not* correspond to computations in the original circuits (since the "encoded" input may not correspond to a *valid* input for the original circuit). This intuition is formalized in the following lemma.

Lemma 3.9. *If E is linear and onto, then Construction 3.7 is SAT-respecting. That is, if $\hat{C}(\hat{x}) = 0$ for some $\hat{x} \in \mathbb{F}^{\hat{n}}$, then $C(x) = 0$ for some $x \in \mathbb{F}^n$.*

Proof. Assume that $\hat{C}(\hat{x}) = 0$ for some $\hat{x} \in \mathbb{F}^{\hat{n}}$, and denote $\hat{x} = \left(\left(\hat{x}_1^*, \mathbf{R}_1, \mathbf{R}_1^0\right), \left(\hat{x}_2^*, \mathbf{R}_2, \mathbf{R}_2^0\right)\right)$. Then $\mathsf{f}(\hat{x}) = 1$ and $\hat{C}_1\left(\hat{x}_1^{\mathrm{in}}; \mathbf{R}_1\right) = 0$ by the definition of $\hat{C}$. Therefore, $\hat{C}_1, \hat{C}_2$ have the same output, and $\mathcal{T}_V, \mathcal{T}_0$ output 1. Consequently, according to Observation 3.6, $\mathbf{R}_1^0$ is well-formed, so by the correctness of $\mathsf{Comp}^{\mathrm{FRRTV}}$, $\hat{\mathcal{T}}_0$ emulates $\mathcal{T}_0$. (Here, we also use the fact that $\mathcal{T}_V$ is independent of all other components of, and inputs to, $\hat{C}$.) Moreover, since the encoding scheme is onto then $\mathbf{R}_1, \mathbf{R}_2$ define inputs to $\mathcal{T}_0$, on which $\mathcal{T}_0$ outputs 1 (because $\hat{\mathcal{T}}_0$ outputs 1). By observation 3.4, at least one of $\mathbf{R}_1, \mathbf{R}_2$ is well-formed. Assuming (without loss of generality) that $\mathbf{R}_1$ is well-formed, then the correctness of $\mathsf{Comp}^{\mathrm{FRRTV}}$ guarantees that $\hat{C}_1$ emulates C, so $0 = \hat{C}_1\left(\hat{x}_1^*; \mathbf{R}_1\right) = C(x)$, where $x = (x_1, ..., x_n) \in \mathbb{F}^n$ is $x = \mathsf{Dec}^{\mathrm{in}}\left(\hat{x}_1^*\right)$ (x is well-defined because E^{in} is onto). □

Next, we analyse the relaxed leakage-resilience property of Construction 3.7, describing a simulator Sim that, given a circuit C, and its output on some input x, generates simulated wire values for C, such that leakage functions cannot distinguish the simulated wires from the actual wire values of C. At a high level, the simulator operates as follows. On input $C : \mathbb{F}^n \to \mathbb{F}$, and $C(x)$ for $x \in \mathbb{F}^n$, Sim finds a $y \in \mathbb{F}^n$ such that $C(y) = C(x)$ (this is the reason that Sim is unbounded); generates $\hat{C} = \mathsf{Comp}(C)$ and $\hat{y} \leftarrow \mathsf{Enc}\left(y, 1^{|C|}\right)$; honestly evaluates $\hat{C}$ on $\hat{y}$; and outputs the wire values of

$\hat{C}$. We show that if E^{in} is leakage-indistinguishable for a leakage class which is "somewhat stronger" than $\mathcal{L}$, then for every $\ell \in \mathcal{L}$, $\mathsf{SD}\left(\ell\left[\hat{C},\hat{x}\right],\ell\left[\hat{C},\hat{y}\right]\right) \leq \epsilon\left(n\right)$, where $\hat{x} \leftarrow \mathsf{Enc}\left(x,1^{|C|}\right)$. Informally, this follows from a hybrid argument, where we first replace the input of $\hat{C}_1$ from $\hat{x}$ to $\hat{y}$, and then do the same for $\hat{C}_2$. (This is also the reason that we do not explicitly verify that $\hat{C}_1,\hat{C}_2$ are evaluated on encodings of the same input.)

To show that each adjacent pair of hybrids is leakage-indistinguishable, we first use an argument similar to that of [11], where we first replace the bundles of $\hat{C}_1$ or $\hat{C}_2$ (depending on the pair of hybrids in question) that are external to the gadgets (i.e., bundles that correspond to wires of the original circuit C) with random encoding of the "correct" values; and then replace the bundles internal to the gadgets of $\hat{C}_1$ (or $\hat{C}_2$) with simulated values. However, our compiled circuit $\hat{C}$ consists also of $\hat{\mathcal{T}}_0,\mathcal{T}_V$, so the analysis in our case is more complex, and in particular we cannot use the leakage-resilience analysis of [11] as a black box. To explain the difficulty in generating these wires values, we need to take a closer look at their leakage-resilience analysis.

Recall that the leakage-indistinguishability proof for every pair of adjacent hybrids contains in itself two series of hybrid arguments, one replacing external bundles, and the other replacing internal bundles. In the first case, leakage-indistinguishability is reduced to that of the underlying encoding scheme E^{in}, whereas in the second it is reduced to the leakage-indistinguishability of the actual and simulated wire values of a single gadget. Specifically, the leakage function ℓ^{in} in the reduction is given either an encoding of a single field element, or the wire values of a single gadget; uses its input to generate *all the wire values of the compiled circuit*; and then evaluates ℓ on these wire values. Thus, if originally we could withstand leakage from some function class $\mathcal{L}^{\mathrm{in}}$, and the additional wires can be generated by a function class $\mathcal{L}_R$, then after the reduction we can withstand leakage from any function class $\mathcal{L}$ such that $\mathcal{L}\circ\mathcal{L}_R \subseteq \mathcal{L}^{\mathrm{in}}$. In particular, if $\mathcal{L}^{\mathrm{in}}$ consists of functions computable by low-depth circuits, and computing the internal wires of $\hat{\mathcal{T}}_0,\mathcal{T}_V$ require deep circuits (consequently, $\mathcal{L}_R$ necessarily contains functions whose computation requires deep circuits), then we have no leakage-resilience. To overcome this, we show how to simulate these additional wires using shallow circuits. This is possible because (due to the way in which the hybrids are defined) the masking inputs in at least one copy are well-formed. Specifically, the structure of $\hat{\mathcal{T}}_0,\mathcal{T}_V$ guarantees that *conditioned on the masking inputs of $\hat{C}_2$ being well-formed*, these wire values can be computed by shallow circuits. When the masking inputs of $\hat{C}_2$ are *ill-formed*, we are guaranteed that the masking inputs of $\hat{C}_1$ are *well-formed*. Conditioned on this event, we show an *alternative* method of computing the internal wires of $\hat{\mathcal{T}}_0,\mathcal{T}_V$, which can be done by shallow circuits.

3.1.2 The Relaxed Leakage-Resilience Property of Construction 3.7

In this section we show that Construction 3.7 is relaxed leakage-resilient. Let $\mathsf{S}\left(n\right):\mathbb{N}\to\mathbb{N}$ be a size function. Assume that E is $\left(\mathcal{L}_{\mathsf{E}},\epsilon_{\mathsf{E}}\left(n\right)\right)$-leakage-indistinguishable for some family $\mathcal{L}_{\mathsf{E}}$ of leakage functions, and some $\epsilon_{\mathsf{E}}\left(n\right):\mathbb{N}\to\mathbb{R}^{+}$, and let $\mathcal{L}$ be a family of functions such that $\mathcal{L}_{\mathsf{E}} = \mathcal{L}\circ\mathsf{Shallow}\left(7,O\left(\hat{n}_1^4\left(\mathsf{S}\left(n\right)\right)\cdot\mathsf{S}\left(n\right)\right)\right)$. We will show that for an appropriate choice of $\epsilon' > 0$, Construction 3.7 is $\left(\mathcal{L}.\epsilon',\mathsf{S}\left(n\right)\right)$-relaxed leakage-resilient. Towards that end, let $C:\mathbb{F}^n\to\mathbb{F}$ be a circuit of size $|C|\leq\mathsf{S}\left(n\right)$, $\ell\in\mathcal{L}$ be of input length $\left|\hat{C}\right|$, and $x\in\mathbb{F}^n$. Recall that the simulator Sim is given C and $C\left(x\right)$, finds a $y\in\mathbb{F}^n$ such that $C\left(x\right)=C\left(y\right)$, generates $\hat{C}=\mathsf{Comp}\left(C\right)$, evaluates $\hat{C}$ on an honestly-generated encoding of y, and outputs the wire values of $\hat{C}$. Let $\hat{x}\leftarrow\mathsf{Enc}\left(x,1^{|C|}\right)$, $\hat{y}\leftarrow\mathsf{Enc}\left(y,1^{|C|}\right)$ such that $C\left(x\right)=C\left(y\right)=0$, where $\hat{x}=\left(\left(\hat{x}_1,\mathbf{R}_1,\mathbf{R}_1^0\right),\left(\hat{x}_2,\mathbf{R}_2,\mathbf{R}_2^0\right)\right)$ and $\hat{y}=\left(\left(\hat{y}_1,\mathbf{R}_1,\mathbf{R}_1^0\right),\left(\hat{y}_2,\mathbf{R}_2,\mathbf{R}_2^0\right)\right)$, then we need to bound $\mathsf{SD}\left(\ell\left[\hat{C},\hat{x}\right],\ell\left[\hat{C},\hat{y}\right]\right)$. We do so by a sequence of hybrids, in which we first replace the input of the first copy $\hat{C}_1$ from an encoding of x

to an encoding of y, then do the same in the second copy $\hat{C}_2$. Thus, we use the following hybrids:

$$H^x := \left(\left[\hat{C}_1; (\hat{x}_1, \mathbf{R}_1)\right], \left[\hat{C}_2, (\hat{x}_2; \mathbf{R}_2)\right], \left[\hat{\mathcal{T}}_0, \left((\mathbf{R}_1, \mathbf{R}_2); \mathbf{R}_1^0\right)\right], \left[\mathcal{T}_V\left(\mathbf{R}_1^0\right)\right]\right)$$

$$H^{y,x} := \left(\left[\hat{C}_1; (\hat{y}_1, \mathbf{R}_1)\right], \left[\hat{C}_2, (\hat{x}_2; \mathbf{R}_2)\right], \left[\hat{\mathcal{T}}_0, \left((\mathbf{R}_1, \mathbf{R}_2); \mathbf{R}_1^0\right)\right], \left[\mathcal{T}_V\left(\mathbf{R}_1^0\right)\right]\right)$$

$$H^y := \left(\left[\hat{C}_1; (\hat{y}_1, \mathbf{R}_1)\right], \left[\hat{C}_2, (\hat{y}_2; \mathbf{R}_2)\right], \left[\hat{\mathcal{T}}_0, \left((\mathbf{R}_1, \mathbf{R}_2); \mathbf{R}_1^0\right)\right], \left[\mathcal{T}_V\left(\mathbf{R}_1^0\right)\right]\right)$$

Next, we show that both $\ell\left(H^x\right), \ell\left(H^y\right)$ are close to the intermediate distribution $\ell\left(H^{y,x}\right)$. As noted above, to do so we use additional hybrids, in which we first replace the values of bundles external to gadgets from their distribution according to either $\ell\left(H^x\right)$ or $\ell\left(H^y\right)$, to random encodings of the "correct" values, and then replace the bundles internal to the gadgets with simulated bundles values.

Bounding $\mathsf{SD}\left(\ell\left(H^x\right), \ell\left(H^{y,x}\right)\right)$. We use the intermediate distributions $H^x_{\text{ext}}, H^x_{\text{mid}}$ in which the external and internal bundles (respectively) are replaced with simulated bundles. More specifically, H^x_{ext} is the hybrid distribution obtained by evaluating $\hat{C}$ honestly on $\hat{x} \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$; picking local reconstructors for all gadgets of $\hat{C}_1$ (see Lemma A.4), and re-computing their internal wires using the reconstructors; and re-evaluating $\hat{\mathcal{T}}_0$ on the new masking inputs generated for the gadgets of $\hat{C}_1$.[4] H^x_{mid} represents the following mental experiment. Unlike the actual simulation, x *is* given as input to Sim, but Sim uses it *only in the second copy* $\hat{C}_2$, i.e., it generates the wire values of $\hat{C}$ as follows.

- Generating the wires of $\hat{C}_2$: Sim generates an encoding $\hat{x} = \left(\left(\hat{x}_1, \mathbf{R}_1, \mathbf{R}_1^0\right), \left(\hat{x}_2, \mathbf{R}_2, \mathbf{R}_2^0\right)\right) \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$ of x, and honestly evaluates $\hat{C}_2$ on $\hat{x}_2$ with masking inputs $\mathbf{R}_2$.
- Generating the wires of $\hat{C}_1$: Sim picks a random encoding $O \leftarrow \mathsf{Enc}^{\text{in}}\left(1, 1^{|C|}\right)$ for the output of $\hat{C}_1$, and honestly computes the wires of the output decoder. Then, Sim picks $z \in_R \mathbb{F}^n$ and generates $\hat{z}_1 \leftarrow \mathsf{Enc}^{\text{in}}\left(z, 1^{|C|}\right)$, and picks random encodings (according to Enc^{in}) for the outputs of all gadgets (except the gadgets whose outputs "touch" the output decoder, since the outputs of these gadgets have already been determined), which effectively determines the standard inputs, and outputs, of all gadgets in $\hat{C}_1$. Next, Sim picks a local reconstructor for every gadget of $\hat{C}_1$, and uses the reconstructors to compute the internal wires of the gadgets in $\hat{C}_1$. The reconstructors determine the (possibly ill-formed) masking inputs of the gadgets, which (together with $\mathbf{R}_2$) form the standard inputs of $\hat{\mathcal{T}}_0$.
- Generating the wires of $\hat{\mathcal{T}}_0$: Sim honestly evaluates $\hat{\mathcal{T}}_0$ on $\mathbf{R}_1, \mathbf{R}_2$, with masking inputs $\mathbf{R}_1^0$.
- Generating the wires of $\mathcal{T}_V$: Sim honestly evaluates $\mathcal{T}_V$ on $\mathbf{R}_1^0$.
- Finally, Sim uses the outputs of $\hat{C}_1, \hat{C}_2, \hat{\mathcal{T}}_0, \mathcal{T}_V$ to generate the flag f, and the output of $\hat{C}$.

We need the following result [11, Lemma 8] regarding preservation of leakage-resilience under computation.

Lemma 3.10 ([11]). *Let $n \in \mathbb{N}$, $\mathcal{W}_0, \mathcal{W}_0'$ be distributions over $\mathbb{F}^n$, $\mathcal{L}, \mathcal{L}_0$ be families of functions, and $\epsilon > 0$. Let F be a distribution over n-input functions in $\mathcal{L}$. For $f \leftarrow F$, let $\mathcal{W}_1 := f\left(\mathcal{W}_0\right), \mathcal{W}_1' := f\left(\mathcal{W}_0'\right)$. If $\mathcal{W}_0, \mathcal{W}_0'$ are $(\mathcal{L}_0, \epsilon)$-leakage-indistinguishable, then $\mathcal{W}_1, \mathcal{W}_1'$ are $(\mathcal{L}_1, \epsilon)$-leakage-indistinguishable for any family $\mathcal{L}_1$ of leakage functions such that $\mathcal{L}_1 \circ \mathcal{L} \subseteq \mathcal{L}_0$.*

[4] Since re-evaluating $\hat{\mathcal{T}}_0$ does not influence its masking inputs, this does not influence the computation in $\mathcal{T}_V$.

The following notation is used to identify gadgets of $\hat{\mathcal{T}}_0$ (resp. $\mathcal{T}_V$) that are correlated to gadgets of $\hat{C}_1, \hat{C}_2$ (resp. $\hat{\mathcal{T}}_0$).

Notation 3.11. We say that a gadget G_i in $\hat{C}_i$ (for $i = 1, 2$) and an $\times$ gadget G_0 in the first layer of $\hat{\mathcal{T}}_0$ (i.e., gadgets corresponding to $\times$ gates that are evaluated before $\mathcal{T}$ is called) are *connected*, if a masking input (i.e., a vector encoding that masking input) of G_i is one of the inputs to G_0. Similarly, we say that a decoding sub-circuit $\mathcal{D}_V$ (in $\mathcal{T}_V$) and a gadget G_0 of $\hat{\mathcal{T}}_0$ are connected, if the input to $\mathcal{D}_V$ is a masking input of G_0.

In Lemma 3.12 below, we bound the statistical distance between $\ell(H^x)$ and $\ell(H^x_{\text{ext}})$. In Lemma 3.14, we bound the statistical distance between $\ell(H^x_{\text{ext}})$ and $\ell(H^x_{\text{mid}})$. (The proofs of both of these lemmas uses an additional sequence of hybrids.) Then, in Remark 3.15, we show that $\ell(H^x_{\text{mid}})$ and $\ell(H^{y,x})$ are statistically close, so $\ell(H^x)$ is statistically close to $\ell(H^{y,x})$. The proofs of these lemmas rely on the fact that the inputs used in the second copy $\hat{C}_2$ are well formed.

Lemma 3.12. *Let $\mathcal{L}_G, \mathcal{L}$ be families of functions, $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. If every gadget G of $\hat{C}_1$ is $(\mathcal{L}_G, \epsilon(n))$-reconstructible, and $\mathcal{L}_G = \mathcal{L} \circ \mathsf{Shallow}\left(|G|, 2, O\left(\hat{n}_1^4(\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$, then $\mathsf{SD}\left(\ell(H^x), \ell(H^x_{\text{ext}})\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for every $\ell \in \mathcal{L}$.*

Proof. Let $M \leq \mathsf{S}(n)$ denote the number of gadgets in $\hat{C}_1$, then we define a fixed ordering on these gadgets, and a sequence $H_0, ..., H_M$ of hybrids, where in H_i, $\hat{C}$ is honestly evaluated with input x, and then the internal wires of *the first i gadgets of* $\hat{C}_1$ are recomputed using the gadget reconstructors, and the wires of $\hat{C}_0$ influenced by these computations are also recomputed. Then $H_0 = H^x, H_M = H^x_{\text{ext}}$. If $\mathsf{SD}\left(\ell(H^x), \ell(H^x_{\text{ext}})\right) > \epsilon(n) \cdot \mathsf{S}(n)$ for some $\ell \in \mathcal{L}$ then $\mathsf{SD}\left(\ell(H_m), \ell(H_{m-1})\right) > \epsilon(n)$ for some $m \in [M]$. Denote the m'th gadget by G, then G is necessarily a $\times$ gadget. (Indeed, conditioned on their inputs and output, Lemma A.4 guarantees that the internal wires of the reconstructors of all other gadgets are distributed identically to the internal wires in an honest evaluation of the gadget.) Using an averaging argument, Lemma 3.13 (which we can use because $\hat{C}_2, \hat{\mathcal{T}}_0$ are honestly evaluated), and the fact that the masking inputs of G are used (as standard inputs) only in G_0 gadgets (in $\hat{\mathcal{T}}_0$) connected to G, we can fix all the wires in H_m, H_{m-1} except for the masking inputs, and internal wires, of G; and the wires of B_0, U_0, and the internal wires in the computation of $B_0, U_0, \mathbf{q}_0$, in every gadget G_0 (in $\hat{\mathcal{T}}_0$) connected to G (the subscript "0" here is used to denote wires internal to G_0). (Notice that the inputs to $\mathcal{T}_V$ are the columns of the masking inputs of such G_0. As these masking inputs are fixed, then the entire computation in $\mathcal{T}_V$ can also be fixed.)

Let $\mathcal{W}_0^R$ ($\mathcal{W}_0^S$) denote the real-world (reconstructed) wire assignment to the internals of G. We construct a pair of distributions $\mathcal{W}_1^R, \mathcal{W}_1^S$, computable from $\mathcal{W}_0^R, \mathcal{W}_0^S$ by a function $f \in \mathsf{Shallow}\left(|G|, 2, O\left(\hat{n}_1^4(\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$. Given the (either real or reconstructed) internals of G (with the inputs $\mathbf{a}, \mathbf{b}$, and the output $\mathbf{c}$, that were hard-wired into H_m, H_{m-1}), f "fills in the holes" in the wire assignment of $\hat{C}$, i.e., uses its input to honestly evaluate the missing wires in $\hat{C}_0$. Then $f \in \mathsf{Shallow}\left(|G|, 2, O\left(\hat{n}_1^4(\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$ because by Lemma 3.13, every gadget G_0 in $\hat{C}_0$ connected to G can be evaluated in $\mathsf{Shallow}\left(2, O\left(\hat{n}_1^2(\mathsf{S}(n))\right)\right)$ (recall that f need only compute the missing wires B_0, U_0, and the internal values in the computation of B_0, U_0 and the (fixed) $\mathbf{q}_0$); there are at most $O\left(\hat{n}_1^2(\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)$ such gadgets G_0; and we can evaluate them in parallel.

By the lemma's assumptions, the $\times$ gadget is $(\mathcal{L}_G, \epsilon(n))$-reconstructible, so $\mathcal{W}_0^R, \mathcal{W}_0^S$ are $(\mathcal{L}_G, \epsilon(n))$-leakage-indistinguishable. Using Lemma 3.10 (here, F is the distribution that always returns the function f), $\mathcal{W}_1^R, \mathcal{W}_1^S$ are $(\mathcal{L}, \epsilon(n))$-leakage-indistinguishable for every family $\mathcal{L}$ of leakage functions such that $\mathcal{L} \circ \mathsf{Shallow}\left(|G|, 2, O\left(\hat{n}_1^4(\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{L}_G$. In particular,

$$\mathsf{SD}\left(\ell(H_m), \ell(H_{m-1})\right) = \mathsf{SD}\left(\ell\left(\mathcal{W}_1^S\right), \ell\left(\mathcal{W}_1^R\right)\right) \leq \epsilon(n).$$

□

The proof of Lemma 3.12 used the following fact: the internal wires of every $\times$ gadget G_0 in $\hat{\mathcal{T}}_0$ that is connected to gadgets of $\hat{C}_1, \hat{C}_2$, are computable (from the inputs and output of G_0) by a shallow circuit, provided that its *second* input is well-formed. Formally,

Lemma 3.13. *Let G_0 be a gadget of $\hat{\mathcal{T}}_0$ connected to gadgets of $\hat{C}_1$ and $\hat{C}_2$, with inputs $\boldsymbol{a}_0, \boldsymbol{b}_0$, and let $B_0, S_0, U_0, \boldsymbol{q}_0, \boldsymbol{r}_0, \boldsymbol{c}_0$ denote the internal wires of G_0. If $\boldsymbol{b}_0$ and the masking inputs of G_0 are well-formed, then $\boldsymbol{q}_0, \boldsymbol{c}_0$ are well-formed, and independent of $\boldsymbol{a}_0, \boldsymbol{b}_0$. Moreover, if $\boldsymbol{b}_0, S_0$ are fixed, then given $\boldsymbol{q}_0, \boldsymbol{c}_0, \boldsymbol{a}_0, \boldsymbol{r}_0$, the internal and output wires of G_0 are computable in* $\mathsf{Shallow}\left(2, O\left(\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)\right)\right)$.

Proof. Let $\mathbf{c}_0$ denote the output of G_0, and let $B_0, U_0, S_0, \mathbf{q}_0, \mathbf{r}_0 := \mathbf{r}^{\hat{n}_1+1}$ denote its internal wires. Since $\mathbf{b}_0$ encodes 0, then $\mathbf{q}_0$ is independent of $\mathbf{a}_0, \mathbf{b}_0$.[5] Since every column of S_0 is well-formed and independent of $\mathbf{a}_0, \mathbf{b}_0$ then $\mathbf{q}_0$ is also well-formed. Finally, since $\mathbf{r}_0$ is well-formed and independent of $\mathbf{a}_0, \mathbf{b}_0$, then so is $\mathbf{c}_0$.

Next, we show that for fixed $\mathbf{b}_0, S_0$, and given $\mathbf{a}_0, \mathbf{q}_0, \mathbf{c}_0$, the (remaining) internals of G_0 are computable in $\mathsf{Shallow}\left(2, O\left(\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)\right)\right)$. Notice that we only need to reconstruct B_0, U_0 and the internal wires in the computation of $B_0, U_0, \mathbf{q}_0$. B_0 is computable in a single layer using $\hat{n}_1\left(\mathsf{S}\left(n\right)\right)$ constant gates (for the coordinates of $\mathbf{b}_0$), $\hat{n}_1\left(\mathsf{S}\left(n\right)\right)$ input gates (for the coordinates of $\mathbf{a}_0$), and $\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)$ $\times$ gates (computing the products $a_{0,i}b_{0,j}$). Once B_0 is known, U_0 is computable in a single layer with $\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)$ constant gates (for the coordinates of S_0) and $\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)$ + gates (computing $B_{0,i} + S_{0,i}$). The internal values in the computation of every $q_{0,i}$ are $a_i \sum_{k=1}^{j} b_{0,k}d_k + \sum_{k=1}^{j} s_{0,i,k}d_k$, where $j = 1, ..., \hat{n}_1\left(\mathsf{S}\left(n\right)\right)$, and $\mathbf{d}$ is the decoding vector of E^{in}, so for fixed $\mathbf{b}_0, S_0, \mathbf{d}$, these internal wires can be computed with $O\left(\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)\right)$ gates organized in two layers: the first with $\hat{n}_1\left(\mathsf{S}\left(n\right)\right)$ constant gates (for the values $\sum_{k=1}^{j} b_{0,k}d_k, j = 1, ..., \hat{n}_1\left(\mathsf{S}\left(n\right)\right)$) and $\hat{n}_1\left(\mathsf{S}\left(n\right)\right)$ $\times$ gates (computing $a_i \sum_{k=1}^{j} b_{0,k}d_k, j = 1, ..., \hat{n}_1\left(\mathsf{S}\left(n\right)\right)$), and the second with $\hat{n}_1\left(\mathsf{S}\left(n\right)\right)$ constant gates (for the values $\sum_{k=1}^{j} s_{i,k}d_k, j = 1, ..., \hat{n}_1$) and $\hat{n}_1$ + gates (computing $a_i \sum_{k=1}^{j} b_{0,k}d_{t,k} + \sum_{k=1}^{j} s_{i,k}d_k, j = 1, ..., \hat{n}_1\left(\mathsf{S}\left(n\right)\right)$). Since the internal wires of $\mathbf{q}_0$ can be computed in parallel to B_0, U_0, the entire computation is in $\mathsf{Shallow}\left(2, O\left(\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)\right)\right)$. □

Lemma 3.14. *Let $\mathcal{L}_{\mathsf{E}}, \mathcal{L}$ be families of functions, $\mathsf{S}\left(n\right) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon\left(n\right) : \mathbb{N} \to \mathbb{R}^+$. If E^{in} is $(\mathcal{L}_{\mathsf{E}}, \epsilon\left(n\right))$-leakage-indistinguishable, and $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(\hat{n}_1\left(\mathsf{S}\left(n\right)\right), 4, O\left(\hat{n}_1^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$, then $\mathsf{SD}\left(\ell\left(H^x_{\mathrm{mid}}\right), \ell\left(H^x_{\mathrm{ext}}\right)\right) \leq \epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$ for every $\ell \in \mathcal{L}$.*

Proof. Define a fixed ordering on the set of inputs bundles of $\hat{C}_1$, and bundles at the output of gadgets in $\hat{C}_1$ that do not touch the decoder, and let $M \leq \mathsf{S}\left(n\right)$ denote the number of such bundles. We define hybrids $H_0, ..., H_M$, where H_i is generated by drawing an assignment to the outputs of all gadgets in $\hat{C}_1$, according to the values of the corresponding wires in $C\left(x\right)$; replacing the first i bundles with random encodings of random values; computing the internal wires of the gadgets of $\hat{C}_1$ using the gadget reconstructors (see Lemma A.4); and evaluating $\hat{C}_2, \hat{\mathcal{T}}_0, \mathcal{T}_V$ honestly (in particular, the input to $\hat{C}_2$ is a random encoding of x). Notice that $H_0 = H^x_{\mathrm{ext}}$ and $H_M = H^x_{\mathrm{mid}}$. If $\mathsf{SD}\left(\ell\left(H^x_{\mathrm{mid}}\right), \ell\left(H^x_{\mathrm{ext}}\right)\right) > \epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$ for some $\ell \in \mathcal{L}$, then $\mathsf{SD}\left(\ell\left(H_m\right), \ell\left(H_{m-1}\right)\right) > \epsilon\left(n\right)$ for some $m \in [M]$. Let G_o (G_i) denote the gadget whose output (input) is the m'th bundle. (If the m'th bundle is an input bundle, then we consider only the gadget G_i.) Using an averaging argument, we can fix all wires in H_m, H_{m-1} (while preserving the statistical distance) except for the following: the m'th bundle; the masking inputs, outputs, and internal wires of G_o; the masking inputs, and internal wires, of G_i; the input wire of G_i that corresponds to the m'th bundle; and the wires correspond to

[5] Indeed, For every $1 \leq i \leq \hat{n}_1$, $q_{0,i} = U_{0,i} \cdot \mathbf{d} = a_{0,i} \sum_{j=1}^{\hat{n}_1} b_{0,j}d_j + \sum_{j=0}^{\hat{n}_1} S_{0,i,j}d_j \overset{(1)}{=} a_{0,i} \cdot 0 + \sum_{j=0}^{\hat{n}_1} S_{0,i,j}d_j = \sum_{j=0}^{\hat{n}_1} S_{0,i,j}d_j$, where $U_{0,i}$ denotes the i'th row of U_0, and $\mathbf{d}$ denotes the decoding vector of E (the equality denoted (1) holds because $\mathbf{b}_0$ encodes 0).

B_0, U_0 and to the computation of $B_0, U_0, \mathbf{q}_0$, inside every gadget G_0 (in $\hat{C}_0$) connected to G_o or G_i (see Lemma 3.13).

Let b denote the value that the m'th bundle encodes in H^x_{ext}. Let $\mathcal{W}^R_0 = \mathsf{Enc}^{\text{in}}\left(b, 1^{\mathsf{S}(n)}\right)$, and $\mathcal{W}^S_0 = \mathsf{Enc}^{\text{in}}\left(r, 1^{\mathsf{S}(n)}\right)$ for $r \in_R \{0,1\}$ (i.e., $\mathcal{W}^R_0$ is the distribution of the m'th bundle in H_{m-1}, and $\mathcal{W}^S_0$ is its distribution in H_m), then by the lemma's assumption, $\mathcal{W}^R_0, \mathcal{W}^S_0$ are $(\mathcal{L}_{\mathsf{E}}, \epsilon(n))$-leakage-indistinguishable. Let $\mathcal{W}^R_1 := f\left(\mathcal{W}^R_0\right), \mathcal{W}^S_0 := f\left(\mathcal{W}^S_1\right)$ where f is chosen according to the following distribution F over $\mathsf{Shallow}\left(\hat{n}_1\left(\mathsf{S}(n)\right), 4, O\left(\hat{n}^4_1\left(\mathsf{S}(n)\right) \cdot \mathsf{S}(n)\right)\right)$. F chooses functions $\mathsf{rec}_o, \mathsf{rec}_i$ according to the distribution over reconstructors for G_o, G_i, respectively, and the obtained function f, on input $\mathbf{e} \in \mathbb{F}^{\hat{n}_1}$: evaluates rec_o on the inputs $\mathbf{a}_o, \mathbf{b}_o$ (that were hard-wired into H_m, H_{m-1}), and output $\mathbf{e}$ (thus reconstructing the masking inputs, and internal wires, of G_o); evaluates rec_i on $\mathbf{e}$ as one of the inputs, and the other input, and output, according to the hard-wired values (thus generating the masking inputs, and internal wires, of G_i); and for every G_0 gadget in $\hat{C}_0$ connected to G_o or G_i, honestly re-evaluates G_0. Notice that $\mathcal{W}^R_1 \equiv H_{m-1}$, and $\mathcal{W}^S_1 \equiv H_m$, because the bundles are re-randomizing (re-randomization guarantees that these equivalences hold *although* some of the values were fixed in advance). Moreover, $f \in \mathsf{Shallow}\left(\hat{n}_1\left(\mathsf{S}(n)\right), 4, O\left(\hat{n}^4_1\left(\mathsf{S}(n)\right) \cdot \mathsf{S}(n)\right)\right)$. Indeed, by Lemma A.4, $\mathsf{rec}_o, \mathsf{rec}_i$ are in $\mathsf{Shallow}\left(2, O\left(\hat{n}^2_1\left(\mathsf{S}(n)\right)\right)\right)$ (and given $\mathbf{e}$ they can be evaluated in parallel), and given the internals of G_o, G_i, the missing wires of every G_0 connected to G_o or G_i are computable in $\mathsf{Shallow}\left(2, O\left(\hat{n}^2_1\left(\mathsf{S}(n)\right)\right)\right)$ (see the proof of Lemma 3.12 for a more detailed analysis). As there are at most $O\left(\hat{n}^2_1\left(\mathsf{S}(n)\right)\right) \cdot \mathsf{S}(n)$ such gadgets, which can be evaluated in parallel, $f \in \mathsf{Shallow}\left(\hat{n}_1\left(\mathsf{S}(n)\right), 4, O\left(\hat{n}^4_1\left(\mathsf{S}(n)\right) \cdot \mathsf{S}(n)\right)\right)$. By Lemma 3.10, $\mathcal{W}^R_1, \mathcal{W}^S_1$ are $(\mathcal{L}, \epsilon(n))$-leakage-indistinguishable, and in particular, $\mathsf{SD}\left(\ell\left(H_{m-1}\right), \ell\left(H_m\right)\right) = \mathsf{SD}\left(\ell\left(\mathcal{W}^R_1\right), \ell\left(\mathcal{W}^S_1\right)\right) \leq \epsilon(n)$. □

Remark 3.15. Let $H^{y,x}_{\text{ext}}$ denote the hybrid distribution obtained by evaluating $\hat{C}$, when $\hat{C}_1, \hat{C}_2$ are honestly evaluated on y, x, respectively (i.e., picking random encodings of $\left(y, 0^{\mathsf{r}_0(|C|)+\mathsf{r}(|C|)}\right), \left(x, 0^{\mathsf{r}_0(|C|)+\mathsf{r}(|C|)}\right)$ according to Enc^{in} etc.); re-computing the internal wires of all gadgets in $\hat{C}_1$ using their reconstructors; and re-evaluating $\hat{\mathcal{T}}_0$. Then under the conditions of Lemma 3.14, $\mathsf{SD}\left(\ell\left(H^x_{\text{mid}}\right), \ell\left(H^{y,x}_{\text{ext}}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$. Indeed, as long as in both hybrids the input to $\hat{C}_2$ encodes the same value, the proof was independent of the actual values whose encodings were the inputs of $\hat{C}_1, \hat{C}_2$ (because in $\hat{C}_2$ the same value is used, and in $\hat{C}_1$ the value is compared to a random value). The proof of Lemma 3.12 also relied only on the inputs to $\hat{C}_1, \hat{C}_2$ encoding the same values in both distributions, and was independent of the *actual* encoded values, so the same proof can be used to show that under the conditions of Lemma 3.12, $\mathsf{SD}\left(\ell\left(H^{y,x}_{\text{ext}}\right), \ell\left(H^{y,x}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for every leakage function $\ell \in \mathcal{L}$. Consequently, $\mathsf{SD}\left(\ell\left(H^x\right), \ell\left(H^{y,x}\right)\right) \leq 4\epsilon(n) \cdot \mathsf{S}(n)$.

Bounding $\mathsf{SD}\left(\ell\left(H^{y,x}\right), \ell\left(H^y\right)\right)$. The argument in this case is somewhat more complex, because here $\hat{C}_2$ may not have been honestly evaluated, whereas Lemmas 3.12 and 3.14 crucially relied on the fact that the *second* input $\mathbf{b}_0$ of every $\times$ gadget G_0 in $\hat{\mathcal{T}}_0$ (connected to gadgets of $\hat{C}_1, \hat{C}_2$) is well-formed. In particular, when only the masking inputs of $\hat{C}_1$ are guaranteed to be well-formed then we cannot fix the internal wires of G_0, and consequently generating the internal wires of $\hat{\mathcal{T}}_0, \mathcal{T}_V$ may require deep circuits. Therefore, we present an alternative method of generating these wires using shallow circuits. We call these alternative methods *right-reconstructors*, to emphasize that they are used *only* when the simulator in the mental experiment simulates the internal wires of the *right* copy $\hat{C}_2$. More specifically, we describe (Construction 3.16) a right-reconstructor for every $\times$ gadget in first layer of $\hat{\mathcal{T}}_0$ (recall that these are the gadgets connected to gadgets of $\hat{C}_1, \hat{C}_2$), and a right-reconstructor for every decoding sub-circuit $\mathcal{D}_V$ of $\mathcal{T}_V$ (Lemma 3.18). We note that unlike the reconstructors of Lemma A.4, the right-reconstructors are only used when one of the inputs, and the output, are well formed (in the case of $\hat{\mathcal{T}}_0$); or if the inputs are well formed, and the output is zero (in the case of $\mathcal{T}_V$).

Construction 3.16 (Right reconstructor, $\hat{C}_0$)**.** We define a set of functions $\left\{\mathsf{rec}_0^{\mathbf{r}^1,...,\mathbf{r}^{\hat{n}}}\right\}$ for $\mathbf{r}^1,...,\mathbf{r}^{\hat{n}_1} \in \mathbb{F}^{\hat{n}_1}$ (recall that $\hat{n}_1 = \hat{n}_1(\mathsf{S}(n))$). Given standard inputs $\mathbf{r}_0^1, \mathbf{r}_0^2$, and output $\mathbf{c}_0$, $\mathsf{rec}_0^{\mathbf{r}^1,...,\mathbf{r}^{\hat{n}}}$ operates as follows.

1. Computes $B_0 \leftarrow \mathbf{r}_0^1 \left(\mathbf{r}_0^2\right)^T = \left(r_{0,i}^1 \times r_{0,j}^2\right)_{i,j\in\hat{n}_1}$ (using $\hat{n}_1^2$ many $\times$ gates).
2. Sets the columns of U_0 to $\mathbf{r}^1,...,\mathbf{r}^{\hat{n}_1}$.
3. Computes $S_0 \leftarrow U_0 - B_0$ (using $\hat{n}_1^2$ many $+$ gates and $\hat{n}_1^2$ many constant gates).
4. Computes a vector $\mathbf{q}_0$, where $q_{0,i}$ is the decoding of the i'th row of U_0. (Notice that $\mathbf{q}_0$ depends only on U_0, which is independent of $\mathbf{r}_0^1, \mathbf{r}_0^2$.)
5. Computes $\mathbf{r}^{\hat{n}_1+1} \leftarrow \mathbf{c}_0 - \mathbf{q}_0$.
6. $\mathsf{rec}_0^{\mathbf{r}^1,...,\mathbf{r}^{\hat{n}}}$ outputs $\mathbf{r}_0^1, \mathbf{r}_0^2, \mathbf{c}_0, \mathbf{r}^1, ..., \mathbf{r}^{\hat{n}+1}, B_0, S_0, U_0, \mathbf{q}_0$.

The distribution REC is defined by picking a function $\mathsf{rec}_0^{\mathbf{r}^1,...,\mathbf{r}^{\hat{n}}}$ where $\mathbf{r}^1,...,\mathbf{r}^{\hat{n}_1}$ are chosen uniformly at random from $\mathsf{Enc}^{\mathrm{in}}\left(0,1^{\mathsf{S}(n)}\right)$.

Lemma 3.17. *Let* REC *denote the distribution defined in Construction 3.16, then* $\mathrm{supp}(\mathsf{REC}) \subseteq \mathsf{Shallow}\left(3\hat{n}_1(\mathsf{S}(n)), 2, O\left(\hat{n}_1^2(\mathsf{S}(n))\right)\right)$*. Moreover, for every plausible pair* $\left(\left(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2\right), \boldsymbol{c}_0\right)$ *(according to Definition A.3) for an* $\times$ *gadget in the first layer of* $\hat{\mathcal{T}}_0$ *such that* $\boldsymbol{r}_0^1 \in \mathsf{Enc}^{\mathrm{in}}\left(0,1^{|C|}\right)$*, if* $\mathsf{rec} \leftarrow \mathsf{REC}$ *then* $\mathsf{rec}\left(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2, \boldsymbol{c}_0\right)$ *is indistinguishable from the wire distribution in a real-world evaluation of the* $\times$ *gadget in the first layer of* $\hat{\mathcal{T}}_0$*, conditioned on the input-output pair* $\left(\left(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2\right), \boldsymbol{c}_0\right)$.

Proof. First, $\mathrm{supp}(\mathsf{REC}) \in \mathsf{Shallow}\left(3\hat{n}_1(\mathsf{S}(n)), 2, O\left(\hat{n}_1^2(\mathsf{S}(n))\right)\right)$, since every function $\mathsf{rec}^{\mathbf{r}^1,...,\mathbf{r}^{\hat{n}}} \in \mathrm{supp}(\mathsf{REC})$ uses $O\left(\hat{n}_1^2\right)$ gates which can be arranged in two layers: the first layer contains the $O\left(\hat{n}_1^2\right)$ $\times$ gates of step 1, the $\hat{n}_1^2$ constant gates of step 2 (since $\mathbf{r}^1,...,\mathbf{r}^{\hat{n}}$ are fixed), $O\left(\hat{n}_1^2\right)$ constant gates (for the internal wires in the computation of $\mathbf{q}_0$, which is also fixed), and $O(\hat{n}_1)$ $-$ gates (for the computation of $\mathbf{r}^{\hat{n}_1+1}$); and the second contains the $\hat{n}_1^2$ many $-$ gates of step 3.

Second, we claim that when $\mathbf{r}_0^1$ is well formed, and $\mathsf{rec} \leftarrow \mathsf{REC}$, then the wire distribution generated by rec is indistinguishable from the real-world wire assignment, conditioned on $\left(\left(\mathbf{r}_0^1, \mathbf{r}_0^2\right), \mathbf{c}_0\right)$. The only difference between the $\times$ gadget and the output of rec is that in the real world, the columns of S_0 are uniform well-formed vectors, whereas in the output of rec this holds for U_0. However, since the columns of B_0 are well-formed (because $\mathbf{r}_0^1$ is well-formed), E^{in} is linear, and $S_0 = U_0 - B_0$, then even in the reconstructed wire assignment, the columns of S_0 are uniform well-formed vectors, and $U_0 = B_0 + S_0$, (i.e., S_0, U_0 are equally distributed in the real-world and the simulated wire assignment). As all other computations in rec (conditioned on S_0) imitate the computation in the $\times$ gadget, we conclude that the distributions are identical. □

Next, we describe a right-reconstructor for $\mathcal{T}_V$.

Lemma 3.18. *Let* $\boldsymbol{r}_0^1, \boldsymbol{c}_0, \boldsymbol{r}^1, ..., \boldsymbol{r}^{\hat{n}_1} \in \mathsf{Enc}^{\mathrm{in}}\left(0,1^{|C|}\right)$*,* G_0 *be a* $\times$ *gadget in the first layer of* $\hat{\mathcal{T}}_0$*, and* $\left(\left(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2\right), \boldsymbol{c}_0\right)$ *be a plausible pair for* G_0 *(according to Definition A.3). Then there exists a function* $\mathsf{rec}_V \in \mathsf{Shallow}\left(\hat{n}_1(\mathsf{S}(n)), 2, O\left(\hat{n}_1(\mathsf{S}(n))\right)\right)$*, such that* $\mathsf{rec}_V\left(\mathsf{rec}_0^{\boldsymbol{r}^1,...,\boldsymbol{r}^{\hat{n}}}\left(\left(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2\right), \boldsymbol{c}_0\right)\right)$ *is the wire assignment to the decoding sub-circuits* $\mathcal{D}_V$ *of* $\mathcal{T}_V$ *that are given as input* $\boldsymbol{r}^1, ..., \boldsymbol{r}^{\hat{n}_1+1}$ *(where* $\boldsymbol{r}^{\hat{n}_1+1}$ *is as defined by* $\mathsf{rec}^{\boldsymbol{r}_1,...,\boldsymbol{r}^{\hat{n}_1}}\left(\left(\boldsymbol{r}_0^1, \boldsymbol{r}_0^2\right), \boldsymbol{c}_0\right)$*).*

Proof. The function rec_V decodes $S_0 = U_0 - B_0$, and $\mathbf{r}^{\hat{n}+1} = \mathbf{c}_0 - \mathbf{q}_0$, where $B_0 = \mathbf{r}_0^1 \cdot \left(\mathbf{r}_0^2\right)^T$ for a fixed well-formed $\mathbf{r}_0^1$; $\mathbf{c}_0$ is a fixed well-formed vector; $U_0 = \left(\mathbf{r}^1, ..., \mathbf{r}^{\hat{n}_1}\right)$ is fixed; and consequently $\mathbf{q}_0$, obtained by decoding the rows of U_0, is also fixed. Let $\mathbf{d}$ denote the decoding vector of $\mathsf{Enc}^{\text{in}}\left(\cdot, 1^{\mathsf{S}(n)}\right)$, then the values of the wires of $\mathcal{D}_V\left(\mathbf{r}^{\hat{n}_1+1}\right)$ are $\sum_{j=1}^{k} d_j \times \left(\mathbf{c}_{0,j} - \mathbf{q}_{0,j}\right), k = 1, 2, ..., \hat{n}_1\left(\mathsf{S}(n)\right)$. These values are all fixed, so they are computable by a circuit containing $\hat{n}_1\left(\mathsf{S}(n)\right)$ constant gates, arranged in a single layer. The values of the wires of $\mathcal{D}_V\left(S_{0,i}\right)$ (where $S_{0,i}$ denotes the i'th column of S_0) are

$$\sum_{j=1}^{k} d_j \times \left(U_{0,j,i} - B_{0,j,i}\right) = \sum_{j=1}^{k} d_j \times \left(U_{0,j,i} - r_{0,j}^1 \times r_{0,i}^2\right) =$$

$$= \sum_{j=1}^{k} d_j \times U_{0,j,i} - r_{0,i}^2 \times \sum_{j=1}^{k} d_j \times r_{0,j}^1$$

for $k = 1, 2, ..., \hat{n}_1\left(\mathsf{S}(n)\right)$. $U_i = \mathbf{r}^i$, $\mathbf{r}$, and $\mathbf{r}_0^1$ are fixed, so for every $1 \leq k \leq \hat{n}_1\left(\mathsf{S}(n)\right)$, $\sum_{j=1}^{k} d_j \times U_{0,j,i}$, $\sum_{j=1}^{k} d_j \times r_{0,j}^1$ are fixed, and so the internal wires of $\mathcal{D}_V\left(S_{0,i}\right)$ are computable by a depth-2 circuit whose first layer contains $\hat{n}_1\left(\mathsf{S}(n)\right)$ constant gates (for the values $\sum_{j=1}^{k} d_j \times r_{0,j}^1$, $k = 1, ..., \hat{n}_1\left(\mathsf{S}(n)\right)$) and $\hat{n}_1\left(\mathsf{S}(n)\right)$ $\times$ gates (for the values $r_{0,i}^2 \times \sum_{j=1}^{k} d_j \times r_{0,j}^1$); and the second layer contains $\hat{n}_1\left(\mathsf{S}(n)\right)$ constant gates (for the values $\sum_{j=1}^{k} d_j \times U_{0,j,i}$, $k = 1, ..., \hat{n}_1\left(\mathsf{S}(n)\right)$) and $\hat{n}_1\left(\mathsf{S}(n)\right)$ $-$ gates (for the values $\sum_{j=1}^{k} d_j \times \left(U_{0,j,i} - r_{0,j}^1 \times r_{0,i}^2\right)$, $k = 1, ..., \hat{n}_1\left(\mathsf{S}(n)\right)$). Since rec_V performs the same computation as $\mathcal{D}_V$, its output is the wire assignment of $\mathcal{D}_V$, conditioned on the values $\mathbf{r}_0^1, \mathbf{c}_0, \mathbf{r}^0, ..., \mathbf{r}^{\hat{n}}$, and $\mathbf{r}_0^2$. □

Using the right-reconstructors, we can now bound the statistical distance between $\ell\left(H^y\right)$ and $\ell\left(H_{\text{ext}}^{y,x}\right)$. To that effect, we first define the distributions $H_{\text{mid}}^y, H_{\text{ext}}^y$, that are similar to $H_{\text{mid}}^x, H_{\text{ext}}^x$ except that y is the input used for the computation, and the wires of $\hat{C}_1$ remain unchanged. More formally, let H_{mid}^y be the intermediate distribution defined by a mental experiment in which Sim is given the input y, but uses it only in the *first* copy $\hat{C}_1$. Specifically, Sim generates $\hat{y} = \left(\left(\hat{y}_1, \mathbf{R}_1, \mathbf{R}_1^0\right), \left(\hat{y}_2, \mathbf{R}_2, \mathbf{R}_2^0\right)\right) \leftarrow \mathsf{Enc}\left(y, 1^{|C|}\right)$, honestly evaluates $\hat{C}_1$ on $\hat{y}_1$ with masking inputs $\mathbf{R}_1$, simulates the computation in $\hat{C}_2$ (by picking a random input, random values for the outputs of the gadgets, and reconstructors for all gadgets, and generating the internal wires of the gadgets using the reconstructors), and then uses the right-reconstructor of $\hat{\mathcal{T}}_0$, and the function rec_V of Lemma 3.18, to generate an assignment to the internal wires of these components. H_{ext}^y is obtained by evaluating $\hat{C}$ honestly on $\hat{y} \leftarrow \mathsf{Enc}\left(y, 1^{|C|}\right)$, then picking reconstructors for all gadgets of $\hat{C}_2$, and re-computing their internal wires using the reconstructors; re-evaluating all wires of $\hat{\mathcal{T}}_0$ using its right-reconstructor; and re-computing the internal wires of $\mathcal{T}_V$ using the function rec_V of Lemma 3.18. In Lemma 3.19 below, we show that $\ell\left(H^y\right), \ell\left(H_{\text{ext}}^y\right)$ are statistically close. Then, in Lemma 3.20, we show that $\ell\left(H_{\text{ext}}^y\right), \ell\left(H_{\text{mid}}^y\right)$ are statistically close. In Remark 3.21 we show that $\ell\left(H_{\text{mid}}^y\right), \ell\left(H^{y,x}\right)$ are statistically close, so $\ell\left(H^y\right), \ell\left(H^{y,x}\right)$. We conclude that $\ell\left(H^y\right), \ell\left(H^x\right)$ are statistically close.

Lemma 3.19. *Let $\mathcal{L}_G, \mathcal{L}$ be families of functions, $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. If every gadget G of $\hat{C}_2$ is $(\mathcal{L}_G, \epsilon(n))$-reconstructible, and $\mathcal{L}_G = \mathcal{L} \circ \mathsf{Shallow}\left(|G|, 4, O\left(\hat{n}_1^4\left(\mathsf{S}(n)\right) \cdot \mathsf{S}(n)\right)\right)$, then $\mathsf{SD}\left(\ell\left(H^y\right), \ell\left(H_{\text{ext}}^y\right)\right) \leq \epsilon(n)\,\mathsf{S}(n)$ for every $\ell \in \mathcal{L}$.*

Proof. The proof is similar to that of Lemma 3.12, with the additional complication that we need to reconstruct the internal wires of $\hat{\mathcal{T}}_0, \mathcal{T}_V$ (using their right-reconstructors). Assume towards negation that the claim does not hold, and we define a fixed ordering on the $M \leq \mathsf{S}(n)$ gadgets *of* $\hat{C}_2$, and

the hybrids $H_i, i = 0, ..., M$ are obtained by evaluating $\hat{C}$ honestly *with input* y; recomputing the internal wires of the first i gadgets *of* $\hat{C}_2$, using their reconstructors; *using the right-reconstructor*, re-computing the internals of $\hat{\mathcal{T}}_0$ that are influenced by the internals of the first i gadgets of $\hat{C}_2$; and (using the function rec_V of Lemma 3.18) *recomputing the internal wires of* $\mathcal{T}_V$ that were influenced by re-evaluating $\hat{\mathcal{T}}_0$. Then $H_0 = H^y, H_M = H^y_{\text{ext}}$, so let $H_m, H_{m-1}, m \in [M]$ be the neighboring hybrids such that $\mathsf{SD}\left(\ell\left(H_m\right), \ell\left(H_{m-1}\right)\right) > \epsilon\left(n\right)$ for some $\ell \in \mathcal{L}$, and denote the m'th gadget by G. Notice that we can fix all wire values in H_m, H_{m-1}, except for the following: the internal wires of G; for every gadget G_0 in $\hat{\mathcal{T}}_0$ that is connected to G, the wires of B_0, S_0, and the wire corresponding to the computation of B_0, S_0; and for every decoding sub-circuit $\mathcal{D}_V$ in $\mathcal{T}_V$ that is connected to some G_0, the internal wires corresponding to the computation of $\mathbf{q}_V$. Indeed, the computation in $\hat{C}_1$ is the same in both hybrids, so we can fix the wire assignment of $\hat{C}_1$. Consequently, the right input $\mathbf{r}_1^0$ of every G_0 gadget of $\hat{\mathcal{T}}_0$ that is connected to G is well-formed and fixed, so its output $\mathbf{c}_0$ is well-formed. Since G_0 is evaluated using the right-reconstructor then the columns of U_0 can also be fixed to well-formed vectors, so we can also fix the internal wires in the computation of $\mathbf{q}_0$ (since these depend only on U_0), and $\mathbf{c}_0$ is therefore independent of the inputs $\mathbf{r}_0^1, \mathbf{r}_0^2$ and can also be fixed. Consequently, $\mathbf{r}^{\hat{n}_1+1}$ can also be fixed. Therefore, the only non-fixed internals in G_0 are B_0, S_0, and the wires corresponding to their computation. Recall that $\mathbf{r}^{\hat{n}_1+1}$, and the columns $S_{0,i}$ of S_0, constitute the inputs to a decoding sub-circuit $\mathcal{D}_V$ of $\mathcal{T}_V$ (that is connected to G_0), and $\mathcal{D}_V$ outputs the decoding of $\mathbf{r}^{\hat{n}_1+1}, S_{0,i}$ (respectively), which is zero (for $\mathbf{r}_1^{\hat{n}_1+1}$ this is because $\mathbf{q}_0, \mathbf{c}_0$ are well-formed, and for $S_{0,i}$ this is because $\mathbf{r}_0^1$, and all columns of U_0, are well-formed). Therefore, the outputs of these sub-circuits $\mathcal{D}_V$, and all values computed from them, can be fixed.

Let $\mathcal{W}_0^R$ ($\mathcal{W}_0^S$) denote the real-world (reconstructed) wire assignment to the internals of G after we have fixed the values described above, and let $\mathcal{W}_1^R = f\left(\mathcal{W}_0^R\right), \mathcal{W}_1^S = f\left(\mathcal{W}_0^S\right)$, where f is chosen according to the following distribution F over $\mathsf{Shallow}\left(4, O\left(\hat{n}_1^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$. For every gadget G_0 of $\hat{\mathcal{T}}_0$ that is connected to G, F chooses a function rec_{G_0} chosen according to the distribution REC (see Construction 3.16). The obtained function f, on input $E \in \mathbb{F}^{|G|}$ evaluates rec_{G_0} (for evert G_0 connected to G) on the corresponding masking inputs of G, and the hard-wired values, thus generating the masking inputs, and internal wires, of G_0; and uses the function rec_V of Lemma 3.18 (with the values that are hard-wired into the rec_{G_0}'s) to reconstruct the internals of every decoding sub-circuit $\mathcal{D}_V$ connected to one such G_0. Notice that f is well-defined, because once the masking inputs of every G_0 have been fixed, then we can apply the function rec_V. $f \in \mathsf{Shallow}\left(4, O\left(\hat{n}_1^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$ because by Lemma 3.17 the right-reconstructor of every G_0 in $\hat{\mathcal{T}}_0$ that is connected to G is in $\mathsf{Shallow}\left(2, O\left(\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right)\right)\right)$ (there are at most $O\left(\hat{n}_1^2\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)$ such gadgets G_0, and their reconstructors can be applied in parallel); and the function rec_V used to reconstruct the internals of every decoding sub-circuit $\mathcal{D}_V$ of $\mathcal{T}_V$ that is connected to G_0, is in $\mathsf{Shallow}\left(2, O\left(\hat{n}_1\left(\mathsf{S}\left(n\right)\right)\right)\right)$ (there are at most $O\left(\hat{n}_1^3\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)$ such $\mathcal{D}_V$'s, since every G_0 is connected to $\hat{n}_1\left(\mathsf{S}\left(n\right)\right) + 1$ decoding sub-circuits $\mathcal{D}_V$, and for all of them, rec_V can be applied in parallel). By the lemma's assumption, $\mathcal{W}_0^R, \mathcal{W}_0^S$ are $(\mathcal{L}_G, \epsilon\left(n\right))$-leakage-indistinguishable, so Lemma 3.10 guarantees that $\mathcal{W}_1^R, \mathcal{W}_1^S$ are $(\mathcal{L}, \epsilon\left(n\right))$-leakage-indistinguishable for any $\mathcal{L}$ such that $\mathcal{L} \circ \mathsf{Shallow}\left(4, O\left(\hat{n}_1^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right) \subseteq \mathcal{L}_G$. In particular, $\mathsf{SD}\left(\ell\left(H_m\right), \ell\left(H_{m-1}\right)\right) = \mathsf{SD}\left(\ell\left(\mathcal{W}_1^S\right), \ell\left(\mathcal{W}_1^R\right)\right) \leq \epsilon\left(n\right)$. (In the left equality we also use Lemmas 3.17 and 3.18 which guarantees that the wire values generated by the right-reconstructors are leakage-indistinguishable from the actual wire values.) □

Lemma 3.20. *Let* $\mathcal{L}_{\mathsf{E}}, \mathcal{L}$ *be families of functions,* $\mathsf{S}\left(n\right) : \mathbb{N} \to \mathbb{N}$ *be a size function, and* $\epsilon\left(n\right) : \mathbb{N} \to \mathbb{R}^+$. *If* E^{in} *is* $(\mathcal{L}_{\mathsf{E}}, \epsilon\left(n\right))$*-leakage-indistinguishable, and* $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(\hat{n}_1\left(\mathsf{S}\left(n\right)\right), 6, O\left(\hat{n}_1^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$, *then* $\mathsf{SD}\left(\ell\left(H^y_{\text{mid}}\right), \ell\left(H^y_{\text{ext}}\right)\right) \leq \epsilon\left(n\right)\mathsf{S}\left(n\right)$ *for every* $\ell \in \mathcal{L}$.

Proof. The proof is similar to that of Lemma 3.14, with the additional complication that we need to

reconstruct the internal wires of $\hat{\mathcal{T}}_0, \mathcal{T}_V$ (using their right-reconstructors). Assume towards negation that the claim does not hold, and define a fixed ordering on the the $M \leq \mathsf{S}(n)$ bundles that are either input bundles of $\hat{C}_2$, or output bundles of gadgets in $\hat{C}_2$ (that do not touch the decoder). We define the hybrids $H_0, ..., H_M$, where H_i is generated from H^y_{ext} by replacing the first i bundles with random encodings of random values; recomputing the internal wires of the first i gadgets *of* $\hat{C}_2$ using the gadget reconstructors (see Lemma A.4); reconstructing the internal wires of the $\times$ gadgets G_0 in the first layer of $\hat{\mathcal{T}}_0$ (that touch the first i gadgets of $\hat{C}_2$) *using the right reconstructor of Lemma 3.17*, and *re-evaluating the decoding sub-circuits* $\mathcal{D}_V$ *of* $\mathcal{T}_V$ that touch these gadgets G_0 using the function rec_V of Lemma 3.18. Thus, $H_0 = H^y_{\text{ext}}, H_M = H^y_{\text{mid}}$, and let H_m, H_{m-1} be the neighboring hybrids such that $\mathsf{SD}\left(\ell\left(H_m\right), \ell\left(H_{m-1}\right)\right) > \epsilon\left(n\right)$ for some $\ell \in \mathcal{L}$. We can fix all wire values in H_m, H_{m-1} (while preserving the statistical distance), except for the following: the m'th bundle; the masking inputs, the output, and internal wires of the gadget G_o whose output is the m'th bundle; the masking inputs, and internal wires, of the gadget G_i such that one of its inputs is the m'th bundle, and its input bundle that corresponds to the m'th bundle; for every gadget G_0 in $\hat{\mathcal{T}}_0$ that is connected to G_o or G_i, the internal wires of G_0 that correspond to B_0, S_0, and the computation of these values; and for every decoding sub-circuit $\mathcal{D}_V$ in $\mathcal{T}_V$ that is connected to such a G_0, the internal wires of $\mathcal{D}_V$ that correspond to the computation of $\mathbf{q}_V$. (This holds due to the same arguments used in the proof of lemma 3.19.)

Let b denote the value encoded by the m'th bundle in H^y_{ext}. Let $\mathcal{W}^R_0 = \mathsf{Enc}^{\text{in}}\left(b, 1^{\mathsf{S}(n)}\right), \mathcal{W}^S_0 = \mathsf{Enc}^{\text{in}}\left(r, 1^{\mathsf{S}(n)}\right)$ for $r \in_R \{0,1\}$, then by the lemma's assumption, $\mathcal{W}^R_0, \mathcal{W}^S_0$ are $(\mathcal{L}_{\mathsf{E}}, \epsilon\left(n\right))$-leakage-indistinguishable. Let $\mathcal{W}^R_1 := f\left(\mathcal{W}^R_0\right), \mathcal{W}^S_0 := f\left(\mathcal{W}^S_1\right)$, where f is chosen according to the following distribution F over $\mathsf{Shallow}\left(6, O\left(\hat{n}^4_1\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$. F chooses a function rec_o according to the distribution over reconstructors for G_o; chooses a function rec_i from the set of reconstructors for G_i; and for every gadget G_0 of $\hat{\mathcal{T}}_0$ that is connected to G_o or G_i, a function rec_{G_0} chosen according to the distribution REC (see Construction 3.16). The obtained function f, on input $\mathbf{e} \in \mathbb{F}^{\hat{n}_1(\mathsf{S}(n))}$ evaluates rec_o on the inputs $\mathbf{a}_o, \mathbf{b}_o$ (that were hard-wired into H_m, H_{m-1}), and output $\mathbf{e}$ (thus reconstructing the masking inputs, and internal wires, of G_o); evaluates rec_i on $\mathbf{e}$ as one of the inputs, and the other input, and output, according to the hard-wired values (thus generating the masking inputs, and internal wires, of G_i); for every G_0 gadget in $\hat{\mathcal{T}}_0$ connected to G_o or G_i, evaluates rec_{G_0} on the corresponding masking inputs (as determined by $\mathsf{rec}_o, \mathsf{rec}_i$, respectively, and the hard-wired values), thus generating the masking inputs, and internal wires, of G_0; and uses rec_V (with the values that are hard-wired into the rec_{G_0}'s) to reconstruct the internals of every decoding sub-circuit $\mathcal{D}_V$ connected to one such G_0. Then $\mathcal{W}^R_1 \equiv H_{m-1}$ and $\mathcal{W}^S_1 \equiv H_m$ (because the bundles are re-randomizing), and $f \in \mathsf{Shallow}\left(6, O\left(\hat{n}^4_1\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\right)\right)$ because $\mathsf{rec}_o, \mathsf{rec}_i \in \mathsf{Shallow}\left(2, O\left(\hat{n}^2_1\left(\mathsf{S}\left(n\right)\right)\right)\right)$ (see Lemma A.4), the $O\left(\hat{n}^2_1\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)$ right-reconstructors rec_{G_0} are in $\mathsf{Shallow}\left(2, O\left(\hat{n}^2_1\left(\mathsf{S}\left(n\right)\right)\right)\right)$ (and can be evaluated in parallel), and the $\hat{n}^3_1\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)$ functions rec_V are in $\mathsf{Shallow}\left(2, O\left(\hat{n}_1\left(\mathsf{S}\left(n\right)\right)\right)\right)$ and can be evaluated in parallel (see the proof of Lemma 3.19 for a more detailed analysis). Therefore, by Lemma 3.10, H_{m-1}, H_m are $(\mathcal{L}, \epsilon\left(n\right))$-leakage-indistinguishable for every family $\mathcal{L}$ of leakage functions such that $\mathcal{L} \circ \mathsf{Shallow}\left(6, O\left(\hat{n}^4_1\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right) \subseteq \mathcal{L}_{\mathsf{E}}$. □

Remark 3.21. Let $H^{y,x}_{\text{ext},r}$ denote the hybrid distribution obtained by evaluating $\hat{C}$, when $\hat{C}_1, \hat{C}_2$ are honestly evaluated on y, x, respectively (i.e., picking random encodings of $\left(y, 0^{\mathsf{r}_0(|C|)+\mathsf{r}(|C|)}\right), \left(x, 0^{\mathsf{r}_0(|C|)+\mathsf{r}(|C|)}\right)$ according to Enc^{in} etc.); re-computing the internal wires of all gadgets in $\hat{C}_2$ using their reconstructors; re-evaluating all $\times$ gadgets G_0 (in the first layer of $\hat{\mathcal{T}}_0$) that touch gadgets of $\hat{C}_2$ using their right-reconstructors; and then re-computing $\mathcal{T}_V$. Then under the conditions of Lemma 3.20, $\mathsf{SD}\left(\ell\left(H^y_{\text{mid}}\right), \ell\left(H^{y,x}_{\text{ext},r}\right)\right) \leq \epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$, since as long as the value whose encoding was the input of $\hat{C}_1$ is the same in both hybrids, the proof was independent of the

actual values whose encodings were the inputs of $\hat{C}_1, \hat{C}_2$. Moreover, the proof of Lemma 3.19 relied only on the inputs to $\hat{C}_1, \hat{C}_2$ encoding the same values in both distributions, and was independent of the *actual* encoded values, so the same proof can be used to show that under the conditions of Lemma 3.19, $\mathsf{SD}\left(\ell\left(H^{y,x}_{\text{ext},r}\right), \ell\left(H^{y,x}\right)\right) \leq \epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$ for every leakage function $\ell \in \mathcal{L}$. Consequently, $\mathsf{SD}\left(\ell\left(H^{y}\right), \ell\left(H^{y,x}\right)\right) \leq 4\epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$.

We are finally ready to prove Proposition 3.8.

Proof. Soundness follows from Lemma 3.9, because E^{in} is linear and onto. As for relaxed leakage-resilience, by Lemma A.4, if E^{in} is $(\mathcal{L}_{\mathsf{E}}, \epsilon\left(n\right))$-reconstructible, then all the gadgets of $\hat{C}$ are $(\mathcal{L}_G, \hat{n}\left(\mathsf{S}\left(n\right)\right) \cdot \epsilon\left(n\right))$-reconstructible, for every family $\mathcal{L}_G$ of leakage functions such that $\mathcal{L}_{\mathsf{E}} = \mathcal{L}_G \circ \mathsf{Shallow}\left(3, O\left(\hat{n}\left(\mathsf{S}\left(n\right)\right)\right)\right)$. Since $\mathsf{Shallow}\left(4, O\left(\hat{n}\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right) \circ \mathsf{Shallow}\left(3, O\left(\hat{n}^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right) \subseteq \mathsf{Shallow}\left(7, O\left(\hat{n}^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$, then using the union bound, Lemmas 3.12 and 3.14, and Remark 3.15, we know that for every $\mathcal{L}$ such that $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(7, O\left(\hat{n}^4\left(\mathsf{S}\left(n\right)\right) \cdot \mathsf{S}\left(n\right)\right)\right)$, and for every $\ell \in \mathcal{L}$, $\mathsf{SD}\left(\ell\left(H^{x}\right), \ell\left(H^{y,x}\right)\right) \leq 4\epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$. Similarly, by Lemmas 3.19 and 3.20, and Remark 3.21, $\mathsf{SD}\left(H^{y,x}, H^{y}\right) \leq 4\epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$. Therefore, $\mathsf{SD}\left(\ell\left(H^{x}\right), \ell\left(H^{y}\right)\right) \leq 8\epsilon\left(n\right) \cdot \mathsf{S}\left(n\right)$.

Regarding the size of the compiled circuit, $\hat{C}$ contains two copies of C, where in each copy each gate (out of at most $|C|$) is replaced with a gadget whose size is at most $O\left(\hat{n}^2\left(\mathsf{S}\left(n\right)\right)\right)$, So $\left|\hat{C}_1\right|, \left|\hat{C}_2\right| \leq O\left(\hat{n}^2\left(\mathsf{S}\left(n\right)\right)|C|\right)$. $\hat{\mathcal{T}}_0$ contains $O\left(\hat{n}^4\left(\mathsf{S}\left(n\right)\right)|C|^2\right)$ "binarization" sub-circuits $\mathcal{T}$, each of size at most $O\left(|\mathbb{F}|\right)$, so $\left|\hat{\mathcal{T}}_0\right| \leq O\left(|\mathbb{F}| \cdot \hat{n}^4\left(\mathsf{S}\left(n\right)\right)|C|^2\right)$. As for $\mathcal{T}_V$, it contains a decoding sub-circuit for each of the (at most $O\left(\hat{n}\left(\mathsf{S}\left(n\right)\right)\right)$) masking inputs used in the (at most $O\left(|\mathbb{F}|\,\hat{n}^4\left(\mathsf{S}\left(n\right)\right)|C|^2\right)$) gadgets of $\hat{\mathcal{T}}_0$. The decoding of each masking input requires $\hat{n}\left(\mathsf{S}\left(n\right)\right) \times$ gates followed by $\hat{n}\left(\mathsf{S}\left(n\right)\right) +$ gates. In addition, we have $O\left(\hat{n}^4\left(\mathsf{S}\left(n\right)\right)|C|^2\right)$ constant-sized binarization circuits, followed by $O\left(\hat{n}^4\left(\mathsf{S}\left(n\right)\right)|C|^2\right) \times$ gates, so $|\mathcal{T}_V| \leq O\left(|\mathbb{F}| \cdot \hat{n}^5\left(\mathsf{S}\left(n\right)\right)|C|^2\right)$. Therefore, $\left|\hat{C}\right| \leq O\left(|\mathbb{F}|\,\hat{n}^5\left(\mathsf{S}\left(n\right)\right)|C|^2\right)$. □

3.2 A SAT-Respecting Relaxed LRCC Over $\mathbb{F}_2$

In this section we describe a relaxed LRCC over $\mathbb{F}_2$. Our starting point is the circuit-compiler of Construction 3.7 over the field $\mathbb{F}$, which we apply to an "arithmetic version" of the boolean circuit. At a high-level, we construct our circuit compiler over $\mathbb{F}_2$ as follows: we represent field elements of $\mathbb{F}$ using bit-strings; and operations $+, -, \times, \mathsf{id}, \mathsf{copy}, \mathsf{const}_\alpha, \alpha \in \mathbb{F}$ as functions over $\lceil \log |\mathbb{F}| \rceil$-bit strings. (For now, we assume that there exist gates operating on $\lceil \log |\mathbb{F}| \rceil$-bit strings and computing these operations.) We "translate" boolean circuits into arithmetic circuits with such operations, and apply the circuit-compiler of Construction 3.7 (where the field operations are implemented using the boolean operations described in Section 2) to the "translated" circuit. (We note that leakage-resilience deteriorates when an arithmetic compiler is transformed to a boolean one, but only by a constant factor in the depth and size of circuits computing the leakage functions.) Concretely, we set $\mathbb{F} = \mathbb{F}_3$.

FROM BOOLEAN CIRCUITS TO ARITHMETIC CIRCUITS. Our boolean circuit-compiler operates on *boolean circuits*, but employs an arithmetic circuit-compiler operating on *arithmetic circuits* over $\mathbb{F}$. Therefore, we first transform the boolean circuit into an equivalent arithmetic circuit in the natural manner (i.e., representing every bit operation as a polynomial over the arithmetic field):

Definition 3.22 (Boolean-to-arithmetic "translator" T')**.** Given a boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$, the algorithm T' transforms it into a functionally equivalent arithmetic circuit $C' : \mathbb{F}^n \to \mathbb{F}^m$

(where by "functionally-equivalent" we mean that for every $x \in \{0,1\}^n$, $C'(x) = C(x)$). This is done by replacing the gates of C as follows.

- The negation operation $\neg x$ is replaced with $1 - x$ (replacing a single boolean gate with 2 arithmetic gates, i.e. a const_1 gate and a $-$ gate).
- The operation $x \wedge y$ is replaced with $x \cdot y$ (replacing a single boolean gate with a single arithmetic gate).
- Using De-Morgan's laws, the operation $x \vee y$ is replaced with $1 - ((1-x) \cdot (1-y))$ (replacing a single boolean gate with 7 arithmetic gates).
- $\mathsf{id}, \mathsf{copy}, \mathsf{const}_0, \mathsf{const}_1$ remain unchanged.

Observation 3.23. *For every $x \in \{0,1\}^n$, $C'(x) = C(x)$.*

Representing field elements as bit strings. We can use any transformation $E_b : \mathbb{F}_3 \to \{0,1\}^2$ such that every bit string is associated with a field element. This is required for the SAT-respecting property, to guarantee that whatever values are carried on the wires of the boolean circuit, they can be "translated" into wires of the arithmetic circuit over $\mathbb{F}_3$, and is achieved by defining a "reverse" mapping E_b^{-1}. Concretely, we use the following mapping.

Definition 3.24 (mod-3 mapping E_b)**.** The mod-3 mapping $E_b : \mathbb{F}_3 \to \{0,1\}^2$ is defined as follows: $E_b(0) = 00, E_b(1) = 01$, and $E_b(2) = 11$. The "reverse" mapping $E_b^{-1} : \{0,1\}^2 \to \mathbb{F}_3$ is defined as follows: $E_b^{-1}(00) = 0, E_b^{-1}(01) = E_b^{-1}(10) = 1$, and $E_b^{-1}(11) = 2$. E_b, E_b^{-1} naturally extend to longer strings, where for $\mathbf{v} = (v_1, ..., v_m) \in \mathbb{F}_3^m$, $E_b(\mathbf{v}) = (E_b(v_1), ..., E_b(v_m))$, and for $(b_{1,1}, b_{1,2}, ..., b_{m,1}, b_{m,2}) \in \{0,1\}^{2m}$, $E_b^{-1}(b_{1,1}, ..., b_{m,2}) = \left(E_b^{-1}(b_{1,1}, b_{1,2}), ..., E_b^{-1}(b_{m,1}, b_{m,2})\right)$.

Note that the string "10" is never used as long as the compiler is honestly applied to the arithmetic circuit then.

Implementing field operations. The compiled arithmetic circuit uses the field operations $+, -, \times$, and also $\mathsf{copy}, \mathsf{id}$ and $\mathsf{const}_\alpha, \alpha \in \mathbb{F}_3$. These operations are represented using bit operations over bit strings generated by E_b. Specifically, we think of every field operation as a boolean function with 4 inputs (a pair of 2-bit strings representing the pair of input field elements) and 2 outputs (a 2-bit string representing the output field element). We stress that though an honest construction over bits uses only 3 of the 4 possible 2-bit strings encoding field elements (i.e., only the strings in the image of E_b as defined, for example, in Definition 3.24), the function representing a field operation in $\mathbb{F}_3$ should be defined to output the correct values on *all* 2-bit strings. The truth table of each output bit has constant size, and can be represented by a constant-size, depth-3 boolean circuit. $\mathsf{copy}, \mathsf{id}$ and const_α gates are handled similarly. Therefore, the size (depth) of each gadget (and consequently, of the entire compiled circuit) increases by a constant multiplicative factor (specifically, by a factor of 3).

Notice that representing boolean circuits using arithmetic circuits introduces the following obstacle. For a satisfiable circuit $\hat{C}$, we are only guaranteed the existence of an $x \in \mathbb{F}^n$ satisfying the original *arithmetic* circuit, whereas for boolean circuits we require that $x \in \{0,1\}^n$. Therefore, we need an additional "input checker" sub-circuit to guarantee that the inputs to the compiled circuit encode binary strings.

Definition 3.25 (Input-checker $\mathcal{T}^{\text{in}}$)**.** $\mathcal{T}^{\text{in}} : \mathbb{F} \to \mathbb{F}$ is defined as follows: $\mathcal{T}^{\text{in}}(z) = \mathcal{T}\left(z^2 - z\right)$.

Observation 3.26. *For every $z \in \mathbb{F}_3$, $\mathcal{T}^{\text{in}}(z) \in \{0,1\}$, and $\mathcal{T}^{\text{in}}(z) = 1$ if and only if $z \in \{0,1\}$.*

We now use an arithmetic SAT-respecting relaxed LRCC (Construction 3.7) to construct a boolean circuit compiler with similar properties.

Construction 3.27 (SAT-respecting relaxed LRCC)**.** Let E_b, E_b^{-1} be the mappings of Definition 3.24. Let T' be the algorithm of Definition 3.22 over $\mathbb{F}_3$, and $(\mathsf{Comp}, \mathsf{E} = (\mathsf{Enc}, \mathsf{Dec}))$ be the circuit compiler over $\mathbb{F}_3$ of Construction 3.7. The circuit compiler over $\mathbb{F}_2$ is $(\mathsf{Comp}_b, \mathsf{E}_b = (\mathsf{Enc}_b, \mathsf{Dec}_b))$, where:

- $\mathsf{Enc}_b = E_b \circ \mathsf{Enc}$ and $\mathsf{Dec}_b = \mathsf{Dec} \circ E_b^{-1}$.
- Comp_b on input $C : \{0,1\}^n \to \{0,1\}$:
  - Uses T' to transform C into an equivalent arithmetic circuit $C' : \mathbb{F}_3^n \to \mathbb{F}_3$.
  - Constructs the circuit $C'' : \mathbb{F}_3^n \to \mathbb{F}_3$ such that $C''(x_1, ..., x_n) = 1 - \left(C'(x_1, ..., x_n) \times \left(\times_{i=1}^n \mathcal{T}^{\text{in}}(x_i)\right)\right)$. (Notice that $C''(x_1, ..., x_n)$ outputs 0 if and only if $C'(x_1, ..., x_n) = 1$ and $x_1, ..., x_n \in \{0,1\}$.)
  - Computes $\hat{C}'' = \mathsf{Comp}(C'')$.
  - Replaces every gate in $\hat{C}''$ with a constant-size, depth-3 boolean circuit computing the truth table of the gate operation. Comp_b can use any correct circuit, as long as these circuits are used consistently (i.e., for every gate the same circuit is used to replace all appearances of such gate in $\hat{C}''$).
  - Denote the output of $\hat{C}''$ by $e \in \mathbb{F}_3$, represented by the string $(e_1, e_2) \in \{0,1\}^2$. Then Comp_b outputs the circuit $\hat{C}_b$ obtained from $\hat{C}''$ by applying an $\vee$ gate, followed by a $\neg$ gate, to the output of $\hat{C}''$. (This reduces the output string of $\hat{C}''$ to a single bit, and flips the output of $\hat{C}''$, which is required due to the negation added in C''.)

  We use $\hat{C}_{1,b}, \hat{C}_{2,b}, \hat{\mathcal{T}}_{0,b}, \mathcal{T}_{V,b}$ to denote the components of $\hat{C}_b$ corresponding to $\hat{C}_1, \hat{C}_2, \hat{\mathcal{T}}_0, \mathcal{T}_V$, respectively.

Observation 3.28. *$\hat{C}_b(\hat{x}) \in \{0,1\}$ for every $\hat{x}$. Moreover, $\hat{C}_b(\hat{x}) = 1$ if and only if $\hat{C}''(\hat{x}) = 0$. If Comp is SAT-respecting, then this guarantees that $C''(x) = 0$ for some $x \in \mathbb{F}_3$. The definition of C'', and the correctness of T', guarantees that $x \in \{0,1\}^n$, and that $C'(x) = C(x) = 1$.*

Next, we show that Construction 3.27 "inherits" the properties of the underlying arithmetic circuit compiler. We first show that Construction 3.27 is SAT-respecting.

Lemma 3.29. *If Construction 3.7 is SAT-respecting then Construction 3.27 is also SAT-respecting, i.e. if $\hat{C}_b$ is satisfiable then so is C.*

Proof. Assume that $\hat{C}_b(\hat{x}_b) = 1$ for some $\hat{x}_b \in \{0,1\}^{2\hat{n}}$, where $\hat{C}_b = \mathsf{Comp}_b(C)$. Then because $\hat{C}_b$ computes the negation of the OR of the (2-bit) output of $\hat{C}'$, then the output of $\hat{C}''$ was 0. Therefore, by the definition of Comp_b, the correctness of the implementation of field operations using bit operations, and since E_b^{-1} is on $\mathbb{F}_3$, $\hat{C}''\left(E_b^{-1}(\hat{x}_b)\right) = 0$ (where $\hat{C}'' = \mathsf{Comp}(C'')$). Since Comp is SAT-respecting, there exists an $x \in \mathbb{F}_3^n$ such that $C''(x) = 0$ which, by the definition of C'', is possible if and only if $x \in \{0,1\}^n$ and $C'(x) = 1$. By the correctness of the transformation T' (Definition 3.22), $C(x) = C'(x) = 1$. □

Next, we show that Construction 3.27 is relaxed-leakage-resilient.

Lemma 3.30. *Let $\mathcal{L}, \mathcal{L}_b$ be families of functions, $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. If $\mathcal{L} = \mathcal{L}_b \circ \mathsf{ShallowB}\left(12, O\left(\left(\hat{n}^5\left(1, \mathsf{S}(n)\right) \cdot \mathsf{S}(n)^2\right)\right)\right)$, and $(\mathsf{Comp}, \mathsf{E})$ of Construction 3.7 is an $(\mathcal{L}, \epsilon(n), 49\mathsf{S}(n))$-relaxed-LRCC over $\mathbb{F}_3$, then $(\mathsf{Comp}_b, \mathsf{E}_b)$ of Construction 3.27 is an $(\mathcal{L}_b, \epsilon(n), \mathsf{S}(n))$-relaxed-LRCC over $\mathbb{F}_2$. Moreover, $\left|\hat{C}_b\right| = O\left(\hat{n}^5\left(1, \mathsf{S}(n)\right)|C|^2\right)$.*

Proof. Notice that the existence of a non-efficient simulator in Definition 2.8 is equivalent to requiring that leakage functions cannot distinguish evaluations of $\hat{C}_b$ on encodings of two inputs on which C has the same output. We will use this alternative definition to prove the lemma. Assume towards negation that $(\mathsf{Comp}_b, \mathsf{E}_b)$ is *not* $(\mathcal{L}_b, \epsilon(n))$-leakage-resilient. Then there exist a pair of inputs $y_b, z_b \in \{0,1\}^n$ for C such that $C(y_b) = C(z_b)$, and a leakage function $\ell_b \in \mathcal{L}_b$, such that $\mathsf{SD}\left(\ell_b\left(\hat{C}_b(\hat{y}_b)\right), \ell_b\left(\hat{C}_b(\hat{z}_b)\right)\right) > \epsilon(n)$, where $\hat{y}_b, \hat{z}_b$ are random encodings of y_b, z_b, respectively, according to $\mathsf{E}_b\left(\cdot, 1^{|C|}\right)$. We show that $\mathsf{SD}\left(\ell\left(\hat{C}(\hat{y})\right), \ell\left(\hat{C}(\hat{z})\right)\right) > \epsilon(n)$ for some leakage function $\ell \in \mathcal{L}$, where $\hat{y}, \hat{z}$ are random encodings of y_b, z_b, respectively, according to $\mathsf{E}\left(\cdot, 1^{|C|}\right)$, and $\hat{C} = \mathsf{Comp}(C'')$ (recall that we assume that field elements are encoded using bit strings; and that we have "gate sub-circuits", operating on bit strings, that emulate the field operations). This contradicts the witness-indistinguishability of $(\mathsf{Comp}, \mathsf{E})$, because $C''(y) = C''(z)$ (because in both cases the input-checker outputs 1), and $|C''| \leq 49|C| \leq 49\mathsf{S}(n)$. (Indeed, the transformation from C to C' blows up the circuit by a factor of at most 7, and the transformation from C' to C'' adds at most 7 gates for every input gate, so it blows up the circuit by a factor of at most 7.)

We distinguish between two "kinds" of wires in $\hat{C}_b$: *external* wires, that appear also in $\hat{C}$ (these are the wires between the gates of $\hat{C}$), and *internal* wires (these are the wires in the sub-circuits of $\hat{C}_b$ that emulate the gates of $\hat{C}$). The function ℓ is given as input a wire assignment for $\hat{C}$, which is a sequence of field elements, encoded into 2-bit strings using *some* correct encoding. ℓ first "translates" every 2-bit string into the 2-bit string encoding the same field element under the mapping E_b of Definition 3.24. This is computable by a constant-size, depth-3 circuit, and since these computations can be done in parallel, this "translation" is computable in $\mathsf{ShallowB}\left(3, O\left(\left|\hat{C}\right|\right)\right)$. This defines the wire values of all the external wires of $\hat{C}_b$, encoded into bit-string using the same mapping that Comp_b uses. Next, ℓ computes the internal wires of $\hat{C}_b$. Recall that the internal wires are organized in constant-sized, depth-3 "groups", where every such "group" corresponds to the computation of a single gate of $\hat{C}$. Therefore, the wires in every "group" are computable in $\mathsf{ShallowB}(9, O(1))$ (since these wires can be computed sequentially given the input to the gate, where every wire is computable from the previous by a depth-3, constant-size circuit). As all "groups" can be evaluated in parallel, the internal wires are computable from the external wires in $\mathsf{ShallowB}\left(9, O\left(\left|\hat{C}\right|\right)\right)$. Then, ℓ evaluates ℓ_b on the wire values that it generated. As $\left|\hat{C}\right| \leq O\left(\hat{n}^5\left(1, \mathsf{S}(n)\right)|C''|^2\right) \leq O\left(\hat{n}^5\left(1, \mathsf{S}(n)\right) \cdot \mathsf{S}(n)^2\right)$, then $\ell \in \mathcal{L}_b \circ \mathsf{ShallowB}\left(12, O\left(\left(\hat{n}^5\left(1, \mathsf{S}(n)\right) \cdot \mathsf{S}(n)^2\right)\right)\right) = \mathcal{L}$. Moreover, $\ell_b\left(\hat{C}_b(\hat{y}_b)\right) = \ell\left(\hat{C}(\hat{y})\right)$ and $\ell_b\left(\hat{C}_b(\hat{z}_b)\right) = \ell\left(\hat{C}(\hat{z})\right)$, so $\mathsf{SD}\left(\ell\left(\hat{C}(\hat{y})\right), \ell\left(\hat{C}(\hat{z})\right)\right) > \epsilon(n)$. □

Combining Lemmas 3.29 and 3.30 with Proposition 3.8, we have the following.

Proposition 3.31 (SAT-respecting relaxed LRCC over $\mathbb{F}_2$)**.** *Let $\mathcal{L}, \mathcal{L}_{\mathsf{E}}$ be families of functions, $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. Let E^{in} be a linear, onto encoding scheme over $\mathbb{F}_3$ with parameters $n = 1$, σ and $\hat{n} = \hat{n}(\sigma)$, that is $(\mathcal{L}_{\mathsf{E}}, \epsilon(n))$-leakage-indistinguishable, and $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{ShallowB}\left(33, O\left(\hat{n}^5\left(\mathsf{S}(n)\right) \cdot \mathsf{S}(n)^2\right)\right)$. Then there exists a constant $c > 0$, and a SAT-respecting, $(\mathcal{L}, c \cdot \epsilon(n) \cdot \mathsf{S}(n), \mathsf{S}(n))$-relaxed-LRCC over $\mathbb{F}_2$. Moreover, $\left|\hat{C}_b\right| = O\left(\hat{n}^5\left(\mathsf{S}(n)\right)|C|^2\right)$.*

Proof. We show that Construction 3.27 has the required properties, when E is the encoding scheme of Construction 3.7 (using the inner encoding scheme E^{in}), and $49\mathsf{S}(n)$ is its size parameter. Lemmas 3.29 and 3.9 guarantees that the compiled circuit is SAT-respecting. As noted in the proof of Lemma 3.30, if the wires of the compiled *arithmetic* circuit are already encoded into bit-strings using *the same mapping that* Comp_b *uses*, then every arithmetic gate can be implemented with a depth-3 boolean circuit of constant size. Therefore, functions computable in $\mathsf{Shallow}(d,s)$ are also computable in $\mathsf{ShallowB}(3d, O(s))$ (since all sub-circuits corresponding to gates in a single layer can be evaluated in parallel). (Recall that all field operations are implemented using boolean operations over bit strings. Therefore, in essence this means that leakage functions over $\mathbb{F}_3$ can be *simulated* using somewhat deeper boolean circuits.) In particular, if E is $(\mathcal{L}_{\mathsf{E}}, \epsilon)$-leakage-indistinguishable and $\mathcal{L}_{\mathsf{E}} = \mathcal{L}' \circ \mathsf{ShallowB}\left(21, O\left(\hat{n}^4 (\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$, then by Proposition 3.8, Construction 3.7 is an arithmetic SAT-respecting $(\mathcal{L}', 8\epsilon \cdot 49\mathsf{S}(n), 49\mathsf{S}(n))$-relaxed LRCC. Therefore, Lemma 3.30 guarantees that $(\mathsf{Comp}_b, \mathsf{E}_b)$ is $(\mathcal{L}, 392\epsilon(n) \cdot \mathsf{S}(n), \mathsf{S}(n))$-leakage-resilient as long as $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{ShallowB}\left(21, O\left(\hat{n}^4 (\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \circ \mathsf{ShallowB}\left(12, O\left(\hat{n}^5 (\mathsf{S}(n)) \cdot \mathsf{S}(n)^2\right)\right)$. Moreover, $\left|\hat{C}_b\right| = O\left(\hat{n}^5 (\mathsf{S}(n)) |C|^2\right)$ because $\left|\hat{C}\right| = O\left(\hat{n}^5 (\mathsf{S}(n)) |C''|^2\right)$, and since the blowup in the transformations from C to C'', and from $\hat{C}$ to $\hat{C}_b$, is constant. □

Taking E^{in} to be the parity encoding in the previous proposition, and using a result of Håstad [18], we obtain an LRCC secure against leakage from AC^0 circuits (namely, constant-depth and polynomial-sized boolean circuits with unbounded fan-in $\wedge, \vee$ and $\neg$ gates).

Corollary 3.32. *There exists a SAT-respecting* $\left(\mathsf{AC}^0, \mathsf{negl}(n), \mathsf{poly}(n)\right)$*-relaxed-LRCC over* $\mathbb{F}_2$.

More specifically, we instatiate E^{in} with the parity encoding scheme:

Definition 3.33 (Parity encoding scheme E_p)**.** The parity encoding scheme $\mathsf{E}_p = (\mathsf{Enc}_p, \mathsf{Dec}_p)$ is defined as follows: for $b \in \{0,1\}$ and $\sigma \in \mathbb{N}$, $\mathsf{Enc}_p(b, 1^\sigma)$ outputs $\mathbf{e} \in \{0,1\}^\sigma$ which is uniform over the set $\{\mathbf{v} \in \{0,1\}^\sigma : \sum_{i=1}^{\sigma} v_i = b \mod 2\}$; and $\mathsf{Dec}_p(\mathbf{e}, 1^\sigma)$ outputs $\sum_{i=1}^{\sigma} e_i \mod 2$.

We use the following result of Håstad [18] (as cited in [25, Corollary 1]) that AC^0 circuits cannot distinguish parity encodings of 0 and 1:

Theorem 3.34 ([18])**.** *Let* $d \in \mathbb{N}$ *be a constant, and* $\sigma \in \mathbb{N}$*. The parity encoding scheme* E_p *of Definition 3.33 is* $\left(\mathsf{ShallowB}\left(\sigma, 1, d, 2^{\sigma^{\frac{1}{d}}}\right), 2^{-\sigma^{\frac{1}{d}+1}}\right)$*-leakage-indistinguishable.*

Corollary 3.32 now follows from combining Proposition 3.31 with Theorem 3.34.

Proof of Corollary 3.32. We use the circuit compiler of Proposition 3.31, instantiating E^{in} with the parity encoding E_p. Given a circuit $|C| : \{0,1\}^n \to \{0,1\}$ of size s (notice that $s \geq n$), E_p is applied with security parameter $\sigma = s$, to obtain the compiled circuit $\hat{C}$. Fix some AC^0 circuit C^{leak} (recall that this is actually a family of AC^0 circuits, defined for every input length n of the family C of circuits). Then C^{leak} has constant depth d and polynomial size $p(s)$ for some constant d and polynomial p. Taking $d' = d + 33$, Theorem 3.34, guarantees that $\mathsf{Enc}_p(\cdot, 1^s)$ is $\left(\mathsf{ShallowB}\left(s, 1, d', 2^{s^{\frac{1}{d'}}}\right), 2^{-s^{\frac{1}{d'}+1}}\right)$-leakage-indistinguishable. For a large enough n, $2^{s^{\frac{1}{d'}}} \geq p(s) + s^7$, which means that the composition of C^{leak} with *any* boolean circuit of size s^7 and depth 33, obtains at most an advantage of $2^{-s^{\frac{1}{d'}+1}}$ in distinguishing parity encodings of 0 and 1. By proposition 3.31, the simulated and actual wire values of $\hat{C}$ are $c \cdot 2^{-s^{\frac{1}{d'}+1}} \cdot s = \mathsf{negl}(s) = \mathsf{negl}(n)$ statistically close. □

Remark 3.35 (Withstanding leakage functions with long outputs)**.** Using a result of Dubrov et al. ([9, Theorem 3.3], cited as Theorem 3.36 below), the parity encoding of Definition 3.33 is leakage-indistinguishable against a broader family of leakage-functions, that can output many bits (as long as the output is asymptotically shorter than the encoding length). Repeating the proof of Corollary 3.32, and replacing Theorem 3.34 with Theorem 3.36, we can prove the existence of a boolean SAT-respecting $\left(\mathsf{AC}^0_{1/2}, \mathsf{negl}\left(n\right), \mathsf{poly}\left(n\right)\right)$-relaxed LRCC, where $\mathsf{AC}^0_{1/2}$ denotes AC^0 circuits with $n^{\frac{1}{2}}$ output bits (n denotes the input length). We can replace $1/2$ with any constant $\delta \in (0,1)$, as long as δ is fixed in advance (in particular, before the depth d is determined).

Theorem 3.36 ([9])**.** *For a natural $d > 1$, and $\delta \in (0,1)$, the parity encoding E_p of Definition 3.33 with security parameter σ is $\left(\mathsf{ShallowB}\left(\sigma, \sigma^{\delta}, d, 2^{O\left(\sigma^{\frac{1-\delta}{d}}\right)}\right), 2^{-\Omega\left(\sigma^{\frac{1-\delta}{d}}\right)}\right)$-leakage-indistinguishable.*

3.3 Withstanding Leakage from AC^0 Circuits with $\oplus$ Gates

In this section we extend the results of Section 3.2, and present a boolean SAT-respecting circuit compiler that is relaxed leakage-resilient against AC^0 circuits (namely, constant-depth, polynomial-sized boolean circuits over unbounded fan-in and fan-out $\wedge, \vee, \neg$ gates), *augmented with a sublinear number of $\oplus$ gates of unbounded fan-in and fan-out.* This circuit compiler will be used in Section 4 to construct WIPCPs and CZKPCPs.

The high level idea is to use Construction 3.27, where the underlying arithmetic LRCC over $\mathbb{F}_3$ is instantiated with the encoding scheme E^{in} that maps an element $\gamma \in \mathbb{F}_3$ into a vector $v \in \{0,1\}^k$ (for some natural k), which is random subject to the constraint that the number of 1's in v is congruent to γ modulo 3. We show, by reduction to correlation bounds of [26], that AC^0 circuits, augmented with a sublinear number of $\oplus$ gates, have a negligible advantage in distinguishing between random encodings of 0 and 1 according to E^{in}. Using the leakage-indistinguishability of E^{in}, we construct a SAT-respecting circuit compiler withstanding leakage from AC^0 circuits that have several output bits and are augmented with a sublinear number of $\oplus$ gates:

Theorem 3.37 (SAT-respecting relaxed LRCC for AC^0 with $\oplus$ gates)**.** *For input length parameter n, leakage length bound $\hat{n} = \hat{n}\left(n\right)$, size bound $s = s\left(n\right)$, output length bound $m = m\left(n\right)$, parity gate bound $t = t\left(n\right)$, and depth bound d, let $\mathcal{L}^{m}_{\hat{n},d,s,\oplus t} = \bigcup_{n\in\mathbb{N}} \mathcal{L}^{m(n)}_{\hat{n}(n),d,s(n),\oplus t(n)}$, where $\mathcal{L}^{m_0}_{\hat{n}_0,d_0,s_0,\oplus t_0}$ denotes the class of boolean circuits of input length $\hat{n}_0$ over $\neg$ gates and unbounded fan-in $\wedge, \vee, \oplus$ gates, whose depth, size, output length, and number of parity gates are bounded by d_0, s_0, m_0, t_0, respectively. Then for every positive constants d, c, polynomials m, t, and polynomial size bound $s' = s'\left(n\right)$, there exists a polynomial $l\left(n\right)$, such that there exists a SAT-respecting $\left(\mathcal{L}^{m}_{l,d,l^c,\oplus t}, 2^{-n^c}, s'\left(n\right)\right)$-relaxed LRCC over $\mathbb{F}_2$, which on input a circuit $C : \{0,1\}^n \to \{0,1\}$ of size $|C| \leq s'\left(n\right)$ outputs a circuit $\hat{C}$ of size $|\hat{C}| \leq l\left(n\right)$.*

The remainder of the section is organized as follows. In Section 3.3.1 we exhibit an encoding scheme that is leakage-indistinguishable against AC^0 circuits augmented with a sublinear number of $\oplus$ gates. Then, in Section 3.3.2 we prove Theorem 3.37.

3.3.1 A Leakage-Indistinguishable Encoding Scheme

In this section we use correlation bounds of [26] to show that a certain encoding scheme is leakage-indistinguishable against leakage computable by AC^0 circuits, augmented with few $\oplus$ gates. This

encoding scheme will be used in Section 3.3.2 to construct a SAT-respecting relaxed-LRCC withstanding leakage computed by AC^0 circuits with few $\oplus$ gates. We first define the encoding scheme which we use.

Notation 3.38. For $\gamma \in \{0,1,2\}$ and $n \in \mathbb{N}$, U_γ^n denotes the uniform distribution over $\left\{v \in \{0,1\}^{3n} : \#_1(v) \equiv \gamma \mod 3\right\}$; $\#_1(v)$ denotes the number of 1's in v; and $U_{1,2}^n$ denotes the uniform distribution over $\left\{v \in \{0,1\}^{3n} : \#_1(v) \not\equiv 0 \mod 3\right\}$.

Definition 3.39. We define an encoding scheme $\mathsf{E}_3 = (\mathsf{Enc}_3, \mathsf{Dec}_3)$ over $\mathbb{F}_3$ such that for every $e \in \mathbb{F}_3$, $\mathsf{Enc}_3(e, 1^n)$ is distributed according to U_e^n,[6] and $\mathsf{Dec}_3(v)$ returns $(\#_1(v) \mod 3)$. Notice that E_3 is linear, with decoding vectors $\left\{1^{3n}\right\}$, and consequently also onto.

We will be interested in withstanding leakage computed by AC^0 circuits, augmented with few $\oplus$ gates". This leakage class is formalized in the next Definition.

Definition 3.40 ($\mathcal{L}_{n,d,s,\oplus t}^m$ leakage family)**.** Let $n \in \mathbb{N}$ be a length parameter, $d \in \mathbb{N}$ be a depth parameter, $s \in \mathbb{N}$ be a size parameter, and $t \in \mathbb{N}$ be a parity gate bound. The family $\mathcal{L}_{n,d,s,\oplus t}$ consists of all functions computable by a boolean circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most s and depth d, with unbounded fan-in and fan-out $\wedge, \vee, \neg, \oplus$ gates, out of which at most t are $\oplus$ gates. The family $\mathcal{L}_{d,s,\oplus t}$ of functions is defined as $\mathcal{L}_{d,s,\oplus t} = \cup_{n\in\mathbb{N}} \mathcal{L}_{n,d,s,\oplus t}$.

For a length parameter $m \in \mathbb{N}$, and a function $f : \{0,1\}^n \to \{0,1\}^m$, let $f_i(x_1, ..., x_n), i \in [m]$ denote the i'th output bit of f. We use the following notation: $\mathcal{L}_{n,d,s,\oplus t}^m = \{f : \{0,1\}^n \to \{0,1\}^m : \forall 1 \le i \le m, f_i \in \mathcal{L}_{n,d,s,\oplus t}\}$, and $\mathcal{L}_{d,s,\oplus t}^m := \cup_{n\in\mathbb{N}}\left(\mathcal{L}_{n,d,s,\oplus t}^m\right)$.

We use a correlation bound of Lovett and Srinivasan [26, Theorem 6] which, informally, states that AC^0 circuits, augmented with "few" $\oplus$ gates, have negligible correlation with the boolean function MOD_3 where $\mathrm{MOD}_3(v) = 0$ if and only if $\#_1(v) \equiv 1 \mod 3$. (Their result is more general, but we state a weaker and simpler version that suffices for our needs.) We first define the notion of correlation.

Definition 3.41 (Correlation)**.** Let $n \in \mathbb{N}$, $g, f : \{0,1\}^n \to \{0,1\}$, and let $\mathcal{D}$ be a distribution over $\{0,1\}^n$. The *correlation of g and f in relation to $\mathcal{D}$* is $\mathsf{Corr}_{\mathcal{D}}(g, f) = 2\left|\frac{1}{2} - \Pr_{x\leftarrow\mathcal{D}}[g(x) = f(x)]\right|$.

For a class $\mathcal{G}$ of functions, $\mathsf{Corr}_{\mathcal{D}}(\mathcal{G}, f) = \max_{g\in\mathcal{G}} \mathsf{Corr}_{\mathcal{D}}(g, f)$.

We are interested in correlations with the following function:

Notation 3.42 (MOD_s function)**.** Let $s \in \mathbb{N}$. The function $\mathrm{MOD}_s^n : \{0,1\}^{3n} \to \{0,1\}$ is defined as $\mathrm{MOD}_s^n(x) = 0$ if and only if $\sum_{i=1}^{3n} x_i \equiv 0 \mod s$. We use MOD_s to denote the family of functions $\cup_{n\in\mathbb{N}}\mathrm{MOD}_s^n$.

Theorem 3.43 ([26], Theorem 6 (rephrased))**.** *For every constant depth parameter $d \in \mathbb{N}$ there exist constants $c, \epsilon \in (0,1)$, such that for every constant $l \in \mathbb{N}$ there exists a minimal length parameter $n_0 \in \mathbb{N}$ such that for every $n \ge n_0$, $\mathsf{Corr}_{\mathcal{D}_3^n}\left(\mathcal{L}_{3n,d,n^l,\oplus n^\epsilon}, \mathrm{MOD}_3^n\right) \le 2^{-n^c}$, where $\mathcal{D}_3^n$ is the distribution induced by the following process: first pick a random bit $b \in_R \{0,1\}$; if $b = 0$ pick $x \in \{0,1\}^{3n}$ according to the distribution U_0^n, otherwise pick $x \in \{0,1\}^{3n}$ according to $U_{1,2}^n$.*

Next, we use Theorem 3.43 to show that AC^0 circuits, augmented with "few" $\oplus$ gates, have a negligible advantage in distinguishing between random encodings of 0,1, and 2 according to the encoding scheme of Definition 3.39. Formally:

[6] Enc_3 can be computed efficiently by repeating the following procedure n^2 times. Pick $v \in \{0,1\}^{3n}$ uniformly at random, compute $t := \#_1(v)$, and if $t = e$ then return v. If all iterations fail, return a fixed $v_e \in \{0,1\}^n$ such that $\#_1(v) = e$. Then the output of Enc_3 is statistically close to U_e^n.

Corollary 3.44. *For every constant depth parameter $d \in \mathbb{N}$ there exist constants $c, \epsilon \in (0,1)$, such that for every constant $l \in \mathbb{N}$ there exists a minimal length parameter $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$ the encoding scheme $\mathsf{Enc}_3(\cdot, 1^n)$ of Definition 3.39 is $\left(\mathcal{L}_{3n,d,n^l,\oplus n^\epsilon}, 2^{-n^c}\right)$-leakage-indistinguishable.*

We proceed to prove Corollary 3.44 in two steps. First, we show that Theorem 3.43 implies that AC^0 circuits, augmented with "few" $\oplus$ gates, cannot distinguish between random encodings of 0, and random encodings of either 1 or 2. Second, we show that this implies indistinguishability of encodings of every pair of values in $\{0,1,2\}$. The first step follows from the next lemma.

Lemma 3.45. *Let $\epsilon \in (0,1)$, $n \in \mathbb{N}$, and $\mathcal{G}$ be a class of functions from $\{0,1\}^{3n}$ to $\{0,1\}$. If $\mathsf{Corr}_{\mathcal{D}_3^n}\left(\mathcal{G}, \mathrm{MOD}_3^n\right) \leq \epsilon$ then $U_0^n, U_{1,2}^n$ are $(\mathcal{G}, \epsilon)$-leakage-indistinguishable, where $\mathcal{D}_3^n$ is the distribution defined in Theorem 3.43.*

Proof. Let $g \in \mathcal{G}$. We first establish the connection between the probability $p_g := \Pr_{x \leftarrow \mathcal{D}_3^n}\left[g(x) = \mathrm{MOD}_3^n(x)\right]$ that g computes MOD_3^n correctly, and the distinguishing advantage of g:

$$p_g = \Pr_{x \leftarrow \mathcal{D}_3^n}\left[g(x) = \mathrm{MOD}_3^n(x) \,|\mathrm{MOD}_3^n(x) = 0\right] \cdot \Pr_{x \leftarrow \mathcal{D}_3^n}\left[\mathrm{MOD}_3^n(x) = 0\right] +$$
$$+ \Pr_{x \leftarrow \mathcal{D}_3^n}\left[g(x) = \mathrm{MOD}_3^n(x) \,|\mathrm{MOD}_3^n(x) = 1\right] \cdot \Pr_{x \leftarrow \mathcal{D}_3^n}\left[\mathrm{MOD}_3^n(x) = 1\right]$$

observing that for $x \leftarrow \mathcal{D}_3^n$, $\mathrm{MOD}_3^n(x)$ is 0 (or 1) with probability half, and that

$$\Pr_{x \leftarrow \mathcal{D}_3^n}\left[g(x) = \mathrm{MOD}_3^n(x) \,|\mathrm{MOD}_3^n(x) = 0\right] = \Pr_{x \leftarrow U_0^n}\left[g(x) = 0\right]$$

$$\Pr_{x \leftarrow \mathcal{D}_3^n}\left[g(x) = \mathrm{MOD}_3^n(x) \,|\mathrm{MOD}_3^n(x) = 1\right] = \Pr_{x \leftarrow U_{1,2}^n}\left[g(x) = 1\right]$$

we get:

$$p_g = \frac{1}{2} + \frac{1}{2}\left(\Pr_{x \leftarrow U_{1,2}^n}\left[g(x) = 1\right] - \Pr_{x \leftarrow U_0^n}\left[g(x) = 1\right]\right).$$

By the assumption of the lemma,

$$2\left|\frac{1}{2} - p_g\right| = \mathsf{Corr}_{\mathcal{D}_3^n}\left(g, \mathrm{MOD}_3^n\right) \leq \epsilon.$$

Therefore, we get:

$$\left|\Pr_{x \leftarrow U_{1,2}^n}\left[g(x) = 1\right] - \Pr_{x \leftarrow U_0^n}\left[g(x) = 1\right]\right| \leq \epsilon.$$

□ □

Next, we establish a connection between the distinguishing advantage of circuits between the following pairs of distributions: $U_0^{2n}, U_{1,2}^{2n}$ (over $6n$-*bit vectors*); $U_0^n, U_{1,2}^n$; and U_0^n, U_1^n (over $3n$-*bit vectors*).

Lemma 3.46. *Let $d, s, t \in \mathbb{N}$, and $c \in (0,1)$ be a constant. If there exists an $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$, $U_0^n, U_{1,2}^n$ are $(\mathcal{L}_{3n,d,s,\oplus t}, \epsilon)$-leakage-indistinguishable for $\epsilon = 2^{-n^c}$, and $U_0^{2n}, U_{1,2}^{2n}$ are $(\mathcal{L}_{6n,d+1,2s+1,\oplus 2t}, \epsilon)$-leakage-indistinguishable, then there exists an n_0' such that for every $n \geq n_0'$, U_0^n, U_1^n are $\left(\mathcal{L}_{3n,d,s,\oplus t}, \sqrt{7\epsilon}\right)$-leakage-indistinguishable.*

We will need the following notation and observation regarding the connection between U_1^n, U_2^n and $U_{1,2}^n$.

Notation 3.47. Let $n \in \mathbb{N}$. For $\gamma \in \{0,1,2\}$, we use $\mathcal{S}_\gamma^n$ to denote $\mathsf{supp}\left(U_\gamma^n\right)$, $\mathcal{S}_{1,2}^n$ to denote $\mathsf{supp}\left(U_{1,2}^n\right)$, and k_γ^n to denote $\left|\mathcal{S}_\gamma^n\right|$.

Observation 3.48. *For every $n \in \mathbb{N}$, and every function $g : \{0,1\}^{3n} \to \{0,1\}$, by the law of total probability, and since $\Pr_{x \leftarrow U_{1,2}^n}\left[x \in \mathcal{S}_1^n\right] = \Pr_{x \leftarrow U_{1,2}^n}\left[x \in \mathcal{S}_2^n\right] = \frac{1}{2}$,*

$$\Pr_{x \leftarrow U_{1,2}^n}\left[g\left(x\right) = 1\right] = \frac{1}{2}\left(\Pr_{x \leftarrow U_1^n}\left[g\left(x\right) = 1\right] + \Pr_{x \leftarrow U_2^n}\left[g\left(x\right) = 1\right]\right).$$

Proof of Lemma 3.46. If the lemma does not hold, then there exist infinitely many n's, for each of which U_0^n, U_1^n are *not* $\left(\mathcal{L}_{3n,d,s,\oplus t}, \sqrt{7\epsilon}\right)$-leakage-indistinguishable. Let $\epsilon' = \epsilon'\left(n\right) > \sqrt{7\epsilon}$ denote the maximal distinguishing advantage between U_0^n, U_1^n, let $\hat{D} = \left\{\hat{D}_n\right\}$ be a family of distinguishers obtaining this advantage, and let $\mathcal{N}$ be the infinite set of n's for which $\hat{D}$ obtains this advantage. For $\gamma \in \{0,1,2\}$, let $p_\gamma^n := \Pr_{x \leftarrow U_\gamma^n}\left[\hat{D}_n\left(x\right) = 1\right]$. Assume first that $p_0^n > p_1^n$ for infinitely many n's in $\mathcal{N}$. There are two possible cases: either for infinitely many n's in $\mathcal{N}$, $p_0^n \geq p_2^n$; or $p_0^n < p_2^n$ for infinitely many n's in $\mathcal{N}$. In the first case, $\hat{D}$ has advantage at least $\frac{\epsilon'}{2} > \frac{\sqrt{7\epsilon}}{2} > \frac{\sqrt{4\epsilon}}{2} \overset{\epsilon \leq 1}{\geq} \epsilon$ in distinguishing between $U_0^n, U_{1,2}^n$, for every n such that $p_0^n \geq p_2^n$ and $p_0^n \geq p_1^n + \epsilon'$. Indeed, using Observation 3.48,

$$\left|\Pr_{x \leftarrow U_0^n}\left[\hat{D}_n\left(x\right) = 1\right] - \Pr_{x \leftarrow U_{1,2}^n}\left[\hat{D}_n\left(x\right) = 1\right]\right| = \left|p_0^n - \frac{1}{2}\left(p_1^n + p_2^n\right)\right|$$

using the case assumption that $p_0^n \geq p_1^n, p_2^n$, this advantage is equal to:

$$\frac{1}{2}\left(p_0^n - p_1^n\right) + \frac{1}{2}\left(p_0^n - p_2^n\right) \geq \frac{1}{2}\left(p_0^n - p_1^n\right) \geq \frac{\epsilon'}{2}.$$

Therefore, only the second case $p_0^n < p_2^n$ remains, and Lemma 3.49 below shows that there exists an $\hat{n}_0 \in \mathbb{N}$ such that for every such n which is greater than $\hat{n}_0$, $U_0^{2n}, U_{1,2}^{2n}$ are distinguishable by $\mathcal{L}_{6n,d+1,2s+1,\oplus 2t}$ circuits, with advantage at least $\frac{\left(\epsilon'\right)^2}{6} + E\left(n\right) > \frac{\left(\sqrt{7\epsilon}\right)^2}{6} + E\left(n\right) = \epsilon + \frac{\epsilon + E(n)}{6}$, where $E\left(n\right) = O\left(2^{-3n}\right)$. Recall that $\epsilon = 2^{-n^c}$, so $E\left(n\right) = o\left(\epsilon\right)$, and let $n' \in \mathbb{N}$ such that for every $n \geq n'$, $\left|E\left(n\right)\right| \leq \epsilon$ (notice that $E\left(n\right)$ may be negative). Then for every $n \geq \max\left\{n', \hat{n}_0\right\}$ in $\mathcal{N}$ such that $p_2^n > p_0^n \geq p_1^n + \epsilon'$ (there are infinitely many such n's by the case assumption), $\epsilon + \frac{\epsilon + E(n)}{6} \geq \epsilon$, meaning that $U_0^{2n}, U_{1,2}^{2n}$ can be distinguished in $\mathcal{L}_{6n,d+1,2s+1,\oplus 2t}$ with advantage more than ϵ, a contradiction to the assumption of the lemma. Therefore, it cannot be the case that $p_0^n \geq p_1^n + \epsilon'$ for infinitely many n's in $\mathcal{N}$.

Assume now that $p_0^n \geq p_1^n$ only for finitely many n's in $\mathcal{N}$, i.e., $p_1^n \geq p_0^n$ for infinitely many n's in $\mathcal{N}$. If for infinitely many n's in $\mathcal{N}$, $p_2^n \geq p_0^n$ and $p_1^n > p_0^n$, then the advantage of $\hat{D}_n$ in distinguishing between $U_0^n, U_{1,2}^n$ is at least

$$\left|p_0^n - \frac{p_1^n + p_2^n}{2}\right| = \frac{p_1^n - p_0^n}{2} + \frac{p_2^n - p_0^n}{2} \geq \frac{p_1^n - p_0^n}{2} \geq \frac{\epsilon'}{2}.$$

The second case, where $p_2^n < p_0^n < p_1^n$ for infinitely many n's, follows from Lemma 3.49 in the same manner as before. □

We now prove the lemma used in the proof of Lemma 3.46, for the case $p_2^n > p_0^n > p_1^n$ (or $p_1^n > p_0^n > p_2^n$) for infinitely many n's. Notice that Lemma 3.49 uses the distributions $U_0^{2n}, U_{1,2}^{2n}$ over $6n$-*bit vectors*, and distinguishers over $3n$-*bit vectors*.

Lemma 3.49. *Let $n, d, s, t \in \mathbb{N}$, $\epsilon > 0$, and $\{D_n \in \mathcal{L}_{3n,d,s,\oplus t}\}_{n\in\mathbb{N}}$. For $\gamma \in \{0,1,2\}$, denote $p_\gamma^n := \Pr_{x\leftarrow U_\gamma^n}[D_n(x) = 1]$. Then there exist error terms $E^+(n), E^-(n) = O\left(2^{-3n}\right)$, and an $n_0 \in \mathbb{N}$, such that the following holds for every $n_0 \leq n \in \mathbb{N}$. If $p_2^n > p_0^n > p_1^n$ and $p_0^n - p_1^n \geq \epsilon$, then $U_0^{2n}, U_{1,2}^{2n}$ are $\left(\mathcal{L}_{6n,d+1,2s+1,\oplus 2t}, \frac{\epsilon^2}{6} + E^+(n)\right)$-distinguishable; and if $p_2^n < p_0^n < p_1^n$ and $p_1^n - p_0^n \geq \epsilon$, then $U_0^{2n}, U_{1,2}^{2n}$ are $\left(\mathcal{L}_{6n,d+1,2s+1,\oplus 2t}, \frac{\epsilon^2}{6} + E^-(n)\right)$-distinguishable.*

Proof. Let D'_n be the distinguisher that interprets its input as a pair (x, y) of $3n$-bit vectors, and outputs $D_n(x) \wedge D_n(y)$. Notice that if $D_n \in \mathcal{L}_{3n,d,s,\oplus t}$, then $D'_n \in \mathcal{L}_{6n,d+1,2s+1,\oplus 2t}$. We now analyze the advantage of D'_n in distinguishing between $U_0^{2n}, U_{1,2}^{2n}$. Using Lemma 3.51, $\Pr_{(x,y)\leftarrow U_0^{2n}}[D'_n(x,y) = 1] = \frac{\left(p_0^n\right)^2 + 2p_1^n p_2^n}{3} + E_0(n) + E'_0(n) \cdot p_2^n$, where $E_0(n), E'_0(n)$ are error terms, and $|E_0(n)|, |E'_0(n)| = O\left(2^{-3n}\right)$. Using Lemma 3.52, $\Pr_{(x,y)\leftarrow U_{1,2}^{2n}}[D'_n(x,y) = 1] = \frac{2p_0^n p_1^n + \left(p_1^n\right)^2 + 2p_0^n p_2^n + \left(p_2^n\right)^2}{6} + E_{1,2}(n) + E'_{1,2}(n) \cdot p_2^n + E''_{1,2}(n) \cdot \left(p_2^n\right)^2$, where $E_{1,2}(n), E'_{1,2}(n), E''_{1,2}(n)$ are error terms, and $|E_{1,2}(n)|, \left|E'_{1,2}(n)\right|, \left|E''_{1,2}(n)\right| = O\left(2^{-3n}\right)$. Therefore,

$$\mathcal{E}_{D'_n} := \Pr_{x\leftarrow U_{1,2}^{2n}}\left[D'_n(x,y) = 1\right] - \Pr_{x\leftarrow U_0^{2n}}\left[D'_n(x,y) = 1\right] =$$

$$= \frac{2p_0^n p_1^n + \left(p_1^n\right)^2 + 2p_0^n p_2^n + \left(p_2^n\right)^2 - 2\left(p_0^n\right)^2 - 4p_1^n p_2^n}{6} +$$

$$+ E(n) + E'(n) \cdot p_2^n + E''(n) \cdot \left(p_2^n\right)^2$$

where $E(n), E'(n), E''(n)$ are error terms, and $|E(n)|, |E'(n)|, |E''(n)| = O\left(2^{-3n}\right)$. Thinking of $\mathcal{E}_{D'_n}$ as a function of p_2^n, there exists an n_0 such that for every $n \geq n_0$, the minimal value of $\mathcal{E}_{D'_n}(p_2^n)$ is obtained when $p_2^n = \frac{2p_1^n - p_0^n - 3E'(n)}{1+6E''(n)} \approx 2p_1^n - p_0^n$. Let $n \geq n_0$, and assume first $p_2^n > p_0^n > p_1^n$ and $p_0^n - p_1^n \geq \epsilon$. Then $\frac{2p_1^n - p_0^n - 3E'(n)}{1+6E''(n)} \approx 2p_1^n - p_0^n < p_0$, and in the domain $z \geq \frac{2p_1^n - p_0^n - 3E'(n)}{1+6E''(n)}$, $\mathcal{E}_{D'}$ is monotonically increasing, so the minimal value of $\mathcal{E}_{D'}$ in this section is obtained when $p_2^n = p_0^n$ (since by the case assumption, $p_2^n \geq p_0^n$), in which case $\mathcal{E}_{D'_n}|_{p_2^n = p_0^n} = \frac{\left(p_0^n - p_1^n\right)^2}{6} + E(n) + E'(n) \cdot p_0^n + E''(n) \cdot \left(p_0^n\right)^2 \geq \frac{\epsilon^2}{6} + E(n) + E'(n) \cdot p_0^n + E''(n) \cdot \left(p_0^n\right)^2 =^{p_0^n \in (0,1)} \frac{\epsilon^2}{6} + E^+(n)$, where $E^+(n) = O\left(2^{-3n}\right)$, so D'_n obtaining advantage $\delta^+ := \frac{\epsilon^2}{6} + E^+(n)$ in distinguishing between $U_0^{2n}, U_{1,2}^{2n}$, where $E^+(n) = O\left(2^{-3n}\right)$.

Second, assume that $p_2^n < p_0^n < p_1^n$ and $p_1^n - p_0^n \geq \epsilon$. Then $\frac{2p_1^n - p_0^n - 3E'(n)}{1+3E''(n)} \approx 2p_1^n - p_0^n > p_0$. Since by the case assumption $p_2^n < p_0^n$ then in the domain $z \leq \frac{2p_1^n - p_0^n - 3E'(n)}{1+3E''(n)}$ the function is monotonically decreasing, so the minimal advantage is obtained when $p_0^n = p_2^n$, and the rest of the analysis follows as in the previous case. □

We now state and prove the lemmas that were used in the proof of Lemma 3.49. We will need the following result about the values of k_0^n, k_1^n, k_2^n.

Lemma 3.50. *Let $n \in \mathbb{N}$. Then $k_1^n = k_2^n = \frac{2^{3n} + (-1)^{n-1}}{3}$, and $k_0^n = \frac{2^{3n} + 2\cdot(-1)^n}{3}$.*

Proof. First notice that $k_1^n = |\mathcal{S}_1^n| = |\mathcal{S}_2^n| = k_2^n$, because the transformation that flips all the bits in the vector is a bijection between $\mathcal{S}_1^n$ and $\mathcal{S}_2^n$. Second, notice that $k_0^n = 2^{3n} - 2k_1^n$, and $\frac{2^{3n}+2\cdot(-1)^n}{3} = 2^{3n} - 2\cdot\frac{2^{3n}+(-1)^{n-1}}{3}$, so it suffices to prove the claim for k_1^n, which we do by induction on n. The base case, for $n = 1$, holds because $k_1 = 3$ (since $\mathcal{S}_1^n = \{(0,0,1),(0,1,0),(1,0,0)\}$). For the step, assume that the claim holds for $i \le n$ and we prove the claim for $i = n+1$. Notice that the vectors in $\mathcal{S}_1^{n+1}$ can be divided into 3 subsets: vectors of the form "a vector in $\mathcal{S}_0^n$, concatenated with a vector in $\mathcal{S}_1^1$" (there are $k_0^{n-1}\cdot 3$ such vectors); vectors of the form "a vector in $\mathcal{S}_1^n$, concatenated with a vector in $\mathcal{S}_0^1$" (there are $k_1^{n-1}\cdot 2$ such vectors, because $\mathcal{S}_0^1 = \{(0,0,0),(1,1,1)\}$); and vectors of the form "a vector in $\mathcal{S}_2^n$, concatenated with a vector in $\mathcal{S}_2^1$" (there are $k_2^{n-1}\cdot 3$ such vectors). Using the observations that $k_1^n = k_2^n$ and $k_0^n = 2^{3n} - 2k_1^n$, we get:

$$k_1^{n+1} = 3k_0^n + 2k_1^n + 3k_2^n = 3\cdot\left(2^{3n} - 2k_1^n\right) + 5k_1^n = 3\cdot 2^{3n} - k_1^n.$$

Using the induction hypothesis, $k_1^n = \frac{2^{3n}+(-1)^{n-1}}{3}$, so:

$$k_1^{n+1} = 3\cdot 2^{3n} - k_1^n = 3\cdot 2^{3n} - \frac{2^{3n}+(-1)^{n-1}}{3} = \frac{1}{3}\cdot\left(8\cdot 2^{3n} - (-1)^{n-1}\right) = \frac{2^{3(n+1)}+(-1)^{(n+1)-1}}{3}.$$

□

Lemma 3.51. *Let $D'_n, p_0^n, p_1^n, p_2^n$ be as defined in the proof of Lemma 3.49. Then $\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1\right] = \frac{\left(p_0^n\right)^2+2p_1^np_2^n}{3} + E_0(n) + E'_0(n)\cdot p_2^n$, where $E_0(n), E'_0(n)$ are error terms, and $|E_0(n)|, |E'_0(n)| = O\left(2^{-3n}\right)$.*

Proof. Since

$$\mathcal{S}_0^{2n} = \left\{(x,y):\ x,y\in\{0,1\}^{3n} \wedge (x,y\in\mathcal{S}_0^n \vee x\in\mathcal{S}_1^n, y\in\mathcal{S}_2^n \vee x\in\mathcal{S}_2^n, y\in\mathcal{S}_1^n)\right\}$$

then by the law of total probability, $\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1\right]$ is equal to:

$$\begin{aligned}
&\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1 | x,y\in\mathcal{S}_0^n\right]\cdot\Pr_{(x,y)\leftarrow U_0^{2n}}\left[x,y\in\mathcal{S}_0^n\right] + \\
&+\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1 | x\in\mathcal{S}_1^n, y\in\mathcal{S}_2^n\right]\cdot\Pr_{(x,y)\leftarrow U_0^{2n}}\left[x\in\mathcal{S}_1^n, y\in\mathcal{S}_2^n\right] + \\
&+\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1 | x\in\mathcal{S}_2^n, y\in\mathcal{S}_1^n\right]\cdot\Pr_{(x,y)\leftarrow U_0^{2n}}\left[x\in\mathcal{S}_2^n, y\in\mathcal{S}_1^n\right] = \\
&= \left(\Pr_{x\leftarrow U_0^n}\left[D(x)=1\right]\right)^2\cdot\frac{|\mathcal{S}_0^n|^2}{\left|\mathcal{S}_0^{2n}\right|} + 2\Pr_{x\leftarrow U_1^n}\left[D(x)=1\right]\cdot\Pr_{x\leftarrow U_2^n}\left[D(x)=1\right]\cdot\frac{|\mathcal{S}_1^n|\cdot|\mathcal{S}_2^n|}{\left|\mathcal{S}_0^{2n}\right|}
\end{aligned}$$

If n is even, then by Lemma 3.50: $k_0^n = |\mathcal{S}_0^n| = \frac{2^{3n}+2}{3}$; $k_0^{2n} = \left|\mathcal{S}_0^{2n}\right| = \frac{2^{6n}+2}{3}$; and $k_1^n = |\mathcal{S}_1^n| = \frac{2^{3n}-1}{3}$. Therefore,

$$\frac{|\mathcal{S}_0^n|^2}{\left|\mathcal{S}_0^{2n}\right|} = \frac{\left(\frac{2^{3n}+2}{3}\right)^2}{\frac{2^{6n}+2}{3}} = \frac{1}{3}\cdot\frac{2^{6n}+2^{3n+2}+4}{2^{6n}+2} = \frac{1}{3}\cdot\left(1+\frac{2^{3n+2}+2}{2^{6n}+2}\right) = \frac{1}{3}+O\left(2^{-3n}\right)$$

$$\frac{|\mathcal{S}_1^n|\cdot|\mathcal{S}_2^n|}{\left|\mathcal{S}_0^{2n}\right|} = \frac{|\mathcal{S}_1^n|^2}{\left|\mathcal{S}_0^{2n}\right|} = \frac{\left(\frac{2^{3n}-1}{3}\right)^2}{\frac{2^{6n}+2}{3}} = \frac{1}{3}\cdot\frac{2^{6n}-2^{3n+1}+1}{2^{6n}+2} = \frac{1}{3}-O\left(2^{-3n}\right)$$

Otherwise, n is odd, and by Lemma 3.50: $k_0^n = |\mathcal{S}_0^n| = \frac{2^{3n}-2}{3}$; $k_0^{2n} = \left|\mathcal{S}_0^{2n}\right| = \frac{2^{6n}+2}{3}$; and $k_1^n = |\mathcal{S}_1^n| = \frac{2^{3n}+1}{3}$. Similar calculations give:

$$\frac{|\mathcal{S}_0^n|^2}{\left|\mathcal{S}_0^{2n}\right|} = \frac{1}{3} - O\left(2^{-3n}\right), \quad \frac{|\mathcal{S}_1^n| \cdot |\mathcal{S}_2^n|}{\left|\mathcal{S}_0^{2n}\right|} = \frac{1}{3} + O\left(2^{-3n}\right)$$

Consequently,

$$\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1\right] = \frac{\left(p_0^n\right)^2 + 2p_1^n p_2^n}{3} + E_0(n) + E'_0(n)\cdot p_2^n$$

where E_0, E'_0 are error terms, and $|E_0(n)|, |E'_0(n)| = O\left(2^{-3n}\right)$. □

Lemma 3.52. *Let $D'_n, p_0^n, p_1^n, p_2^n$ be as defined in the proof of Lemma 3.49. Then $\Pr_{(x,y)\leftarrow U_{1,2}^{2n}}\left[D'_n(x,y)=1\right] = \frac{2p_0^n p_1^n + \left(p_1^n\right)^2 + 2p_0^n p_2^n + \left(p_2^n\right)^2}{6} + E_{1,2}(n) + E'_{1,2}(n)\cdot p_2^n + E''_{1,2}(n)\cdot\left(p_2^n\right)^2$, where $E_{1,2}, E'_{1,2}, E''_{1,2}$ are error terms, and $|E_{1,2}(n)|, \left|E'_{1,2}(n)\right|, \left|E''_{1,2}(n)\right| = O\left(2^{-3n}\right)$.*

Proof. The proof is similar to the proof of Lemma 3.51: $\mathcal{S}_{1,2}^{2n}$ is the disjoint union of the sets

$$\mathcal{S}_1^{2n} = \left\{(x,y)\in\{0,1\}^{6n} :\ x,y\in\{0,1\}^{3n} \wedge \left(x\in\mathcal{S}_0^n, y\in\mathcal{S}_1^n \vee x\in\mathcal{S}_1^n, y\in\mathcal{S}_0^n \vee x,y\in\mathcal{S}_2^n\right)\right\}$$

and

$$\mathcal{S}_2^{2n} = \left\{(x,y)\in\{0,1\}^{6n} :\ x,y\in\{0,1\}^{3n} \wedge \left(x\in\mathcal{S}_0^n, y\in\mathcal{S}_2^n \vee x\in\mathcal{S}_2^n, y\in\mathcal{S}_0^n \vee x,y\in\mathcal{S}_1^n\right)\right\},$$

and a random element in $\mathcal{S}_{1,2}^{2n}$ belongs to each of these sets with probability $\frac{1}{2}$. Therefore, by the law of total probability, $\Pr_{(x,y)\leftarrow U_{1,2}^{2n}}\left[D'_n(x,y)=1\right]$ is equal to:

$$\frac{1}{2}\left(2\Pr_{x\leftarrow U_0^n}\left[D(x)=1\right]\cdot\Pr_{x\leftarrow U_1^n}\left[D(x)=1\right]\cdot\frac{|\mathcal{S}_0^n|\cdot|\mathcal{S}_1^n|}{\left|\mathcal{S}_1^{2n}\right|} + \left(\Pr_{x\leftarrow U_2^n}\left[D(x)=1\right]\right)^2\cdot\frac{|\mathcal{S}_2^n|^2}{\left|\mathcal{S}_1^{2n}\right|}\right) +$$

$$+\frac{1}{2}\left(2\Pr_{x\leftarrow U_0^n}\left[D(x)=1\right]\cdot\Pr_{x\leftarrow U_2^n}\left[D(x)=1\right]\cdot\frac{|\mathcal{S}_0^n|\cdot|\mathcal{S}_2^n|}{\left|\mathcal{S}_2^{2n}\right|} + \left(\Pr_{x\leftarrow U_1^n}\left[D(x)=1\right]\right)^2\cdot\frac{\left|\mathcal{S}_2^1\right|^2}{\left|\mathcal{S}_2^{2n}\right|}\right)$$

If n is even, then by Lemma 3.50: $k_0^n = |\mathcal{S}_0^n| = \frac{2^{3n}+2}{3}$; $k_1^n = |\mathcal{S}_1^n| = \frac{2^{3n}-1}{3}$; and $k_1^{2n} = \left|\mathcal{S}_1^{2n}\right| = k_2^{2n} = \left|\mathcal{S}_2^{2n}\right| = \frac{2^{6n}-1}{3}$. Therefore,

$$\frac{|\mathcal{S}_0^n|\cdot|\mathcal{S}_2^n|}{\left|\mathcal{S}_1^{2n}\right|} = \frac{|\mathcal{S}_0^n|\cdot|\mathcal{S}_1^n|}{\left|\mathcal{S}_1^{2n}\right|} = \frac{\frac{2^{3n}+2}{3}\cdot\frac{2^{3n}-1}{3}}{\frac{2^{6n}-1}{3}} = \frac{1}{3}\cdot\frac{2^{6n}+2^{3n}-2}{2^{6n}-1} = \frac{1}{3}\cdot\left(1+\frac{2^{3n}-1}{2^{6n}-1}\right) = \frac{1}{3}+O\left(2^{-3n}\right)$$

and

$$\frac{|\mathcal{S}_2^n|^2}{\left|\mathcal{S}_2^{2n}\right|} = \frac{|\mathcal{S}_1^n|^2}{\left|\mathcal{S}_2^{2n}\right|} = \frac{\left(\frac{2^{3n}-1}{3}\right)^2}{\frac{2^{6n}-1}{3}} = \frac{1}{3}\cdot\frac{2^{6n}-2^{3n+1}+1}{2^{6n}-1} = \frac{1}{3}\cdot\left(1-\frac{2^{3n+1}-2}{2^{6n}-1}\right) = \frac{1}{3}-O\left(2^{-3n}\right)$$

Otherwise, n is odd, and by Lemma 3.50: $k_0^n = |\mathcal{S}_0^n| = \frac{2^{3n}-2}{3}$; $k_1^n = |\mathcal{S}_1^n| = \frac{2^{3n}+1}{3}$; and $k_1^{2n} = \left|\mathcal{S}_1^{2n}\right| = k_2^{2n} = \left|\mathcal{S}_2^{2n}\right| = \frac{2^{6n}-1}{3}$. Therefore,

$$\frac{|\mathcal{S}_0^n|\cdot|\mathcal{S}_1^n|}{\left|\mathcal{S}_1^{2n}\right|} = \frac{|\mathcal{S}_0^n|\cdot|\mathcal{S}_2^n|}{\left|\mathcal{S}_1^{2n}\right|} = \frac{\frac{2^{3n}-2}{3}\cdot\frac{2^{3n}+1}{3}}{\frac{2^{6n}-1}{3}} = \frac{1}{3}\cdot\frac{2^{6n}-2^{3n}-2}{2^{6n}-1} = \frac{1}{3}\cdot\left(1-\frac{2^{3n}+1}{2^{6n}-1}\right) = \frac{1}{3}-O\left(2^{-3n}\right)$$

and

$$\frac{|\mathcal{S}_1^n|^2}{|\mathcal{S}_2^{2n}|} = \frac{|\mathcal{S}_2^n|^2}{|\mathcal{S}_2^{2n}|} = \frac{\left(\frac{2^{3n}+1}{3}\right)^2}{\frac{2^{6n}-1}{3}} = \frac{1}{3}\cdot\frac{2^{6n}+2^{3n+1}+1}{2^{6n}-1} = \frac{1}{3}\cdot\left(1+\frac{2^{3n+1}+2}{2^{6n}-1}\right) = \frac{1}{3}+O\left(2^{-3n}\right)$$

Consequently,

$$\Pr_{(x,y)\leftarrow U_{1,2}^{2n}}\left[D'_n(x,y)=1\right] = \frac{2p_0^np_1^n+(p_1^n)^2+2p_0^np_2^n+(p_2^n)^2}{6}+E_{1,2}(n)+E'_{1,2}(n)\cdot p_2^n+E''_{1,2}(n)\cdot(p_2^n)^2$$

where $E_{1,2}(n), E'_{1,2}(n), E''_{1,2}(n)$ are error terms, and $|E_{1,2}(n)|, \left|E'_{1,2}(n)\right|, \left|E''_{1,2}(n)\right| = O\left(2^{-3n}\right)$. □

According to Lemma 3.46, indistinguishability of $U_0^n, U_{1,2}^n$ and $U_0^{2n}, U_{1,2}^{2n}$ implies indistinguishability of U_0^n, U_1^n. We now show that this implies that E_3 is leakage indistinguishable (against a slightly weaker family of leakage functions).

Lemma 3.53. *Let $n,d,s,t\in\mathbb{N}$, and $\epsilon=\epsilon(n)>0$. If there exists an $n_0\in\mathbb{N}$ such that for every $n\geq n_0$, U_0^n, U_1^n are $(\mathcal{L}_{3n,d,s,\oplus t},\epsilon)$-leakage-indistinguishable, then for every $n\geq n_0$, $\mathsf{E}_3(\cdot,1^n)$ is $(\mathcal{L}_{3n,d-1,s-3n,\oplus t},2\epsilon)$-leakage-indistinguishable.*

Proof. We show first that $\mathsf{Enc}_3(0,1^n)$,$\mathsf{Enc}_3(2,1^n)$ are $(\mathcal{L}_{3n,d-1,s-3n,\oplus t},\epsilon)$-leakage-indistinguishable for every $n\geq n_0$. Otherwise, there exists an $n>n_0$, and a distinguisher $D_n\in\mathcal{L}_{3n,d-1,s-3n,\oplus t}$ that achieves advantage $\epsilon'>\epsilon$ in distinguishing between the distributions $\mathsf{Enc}_3(0,1^n)$, $\mathsf{Enc}_3(2,1^n)$. We define D'_n to apply negation gates on its inputs, and run D_n. Then $D'_n\in\mathcal{L}_{3n,d,s,\oplus t}$, and notice that since the encoding length is divisible by 3, and the transformation $v\to\bar{v}$ is 1:1 and onto (where $\bar{v}$ denotes the vector obtained by coordinate-wise negating v) then: if $v\leftarrow\mathsf{Enc}_3(0,1^n)$ then $\bar{v}\leftarrow\mathsf{Enc}_3(0,1^n)$; and if $v\leftarrow\mathsf{Enc}_3(1,1^n)$ then $\bar{v}\leftarrow\mathsf{Enc}_3(2,1^n)$. Therefore, $|\Pr[D'_n(\mathsf{Enc}(0,1^n))=1]-\Pr[D'_n(\mathsf{Enc}(1,1^n))=1]| = |\Pr[D_n(\mathsf{Enc}(0,1^n))=1]-\Pr[D_n(\mathsf{Enc}(2,1^n))=1]| = \epsilon'>\epsilon$, contradicting the assumption of the lemma. Second, since for every $n\geq n_0$, $\mathsf{Enc}_3(0,1^n)$, $\mathsf{Enc}_3(2,1^n)$ are $(\mathcal{L}_{3n,d-1,s-3n,\oplus t},\epsilon)$-leakage-indistinguishable, and $\mathsf{Enc}_3(0,1^n)$,$\mathsf{Enc}_3(1,1^n)$ are $(\mathcal{L}_{3n,d,s,\oplus t},\epsilon)$-leakage-indistinguishable, then using the triangle inequality $\mathsf{Enc}_3(1,1^n)$,$\mathsf{Enc}_3(2,1^n)$ are $(\mathcal{L}_{3n,d-1,s-3n,\oplus t},2\epsilon)$-leakage-indistinguishable. □

We are finally ready to prove Corollary 3.44.

Proof of Corollary 3.44. Let $d'=d+2$, let ϵ,c be the constants for which Theorem 3.43 holds for depth parameter d', and we set $c'=\frac{c}{2}$, and $\epsilon'=\frac{\epsilon}{2}$. Given l, let $l'=l+1$, and let n_0 be the minimal length parameter for which Theorem 3.43 holds with parameters d',l'. Let n'_0 be such that for every $n\geq n'_0$, $2\left(n^l+3n\right)+1\leq n^{l'}$, $2n^{\epsilon'}\leq n^{\epsilon}$, and $2\sqrt{7}\cdot 2^{-\frac{n^c}{2}}\leq 2^{-n^{c'}}$. Let n''_0 be the minimal length parameter whose existence is guaranteed in Lemma 3.46 for the length parameter $\max\{n_0,n'_0\}$, constant c, depth parameter $d+2$, size parameter $s=n^l+3n$, and parity gate bound $t=n^{\epsilon'}$. Let $\tilde{n}_0=\max\{n_0,n'_0,n''_0\}$. We show that the corollary holds for minimal length parameter $\tilde{n}_0$ and constants c',ϵ'. Indeed, for every $n\geq\tilde{n}_0$ Theorem 3.43 guarantees that $\mathsf{Corr}_{\mathcal{D}_3^n}\left(\mathcal{L}_{3n,d+2,2\left(n^l+3n\right)+1,\oplus 2n^{\epsilon'}},\mathrm{MOD}_3^n\right)\leq 2^{-n^c}$ (since $n\geq n_0$ and $n\geq n'_0$). By Lemma 3.45, this implies that for every $n\geq\tilde{n}_0$, $U_0^n, U_{1,2}^n$ are $\left(\mathcal{L}_{3n,d+1,n^l+3n,\oplus n^{\epsilon'}},2^{-n^c}\right)$-leakage-indistinguishable, and $U_0^{2n}, U_{1,2}^{2n}$ are $\left(\mathcal{L}_{6n,d+2,2\left(n^l+3n\right)+1,\oplus 2n^{\epsilon'}},2^{-n^c}\right)$-leakage-indistinguishable. By Lemma 3.46, for

every $n \geq \tilde{n}_0$, U_0^n, U_1^n are $\left(\mathcal{L}_{3n,d+1,n^l+3n,\oplus n^{\epsilon'}}, \sqrt{7} \cdot 2^{-\frac{n^c}{2}}\right)$-leakage-indistinguishable (because $n \geq n_0''$). By Lemma 3.53, $\mathsf{E}_3(\cdot, 1^n)$ is $\left(\mathcal{L}_{3n,d,n^l,\oplus n^{\epsilon'}}, 2\sqrt{7} \cdot 2^{-\frac{n^c}{2}}\right)$-leakage-indistinguishable. Since $\tilde{n}_0 \geq n_0'$, $\mathsf{E}_3(\cdot, 1^n)$ is $\left(\mathcal{L}_{3n,d,n^l,\oplus n^{\epsilon'}}, 2^{-n^{c'}}\right)$-leakage-indistinguishable □

3.3.2 A Proof of Theorem 3.37

In this section we prove Theorem 3.37. The proof combines Proposition 3.31 and Corollary 3.44. More specifically, we instantiate the encoding scheme E of Proposition 3.31 with an "appropriate" encoding scheme. Recall that Proposition 3.31 uses the compiler of Construction 3.27, which is based on the arithmetic circuit compiler of Construction 3.7, and is obtained by mapping an encoding according to a linear encoding scheme over $\mathbb{F}_3$ to an encoding over $\{0,1\}$ in which every field element is represented by a 2-bit string (e.g., using the mapping E_b of Definition 3.24). The next lemma states that the leakage-indistinguishability of an encoding is preserved under this mapping (up to a constant additive loss in the depth, and a constant multiplicative loss in the size, of the circuits computing the leakage functions).

Lemma 3.54. *Let $n, m, d, s \in \mathbb{N}$, $\epsilon > 0$, and $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ be an $\left(\mathcal{L}^m_{n,d+1,s+4n,\oplus t}, \epsilon\right)$-leakage-indistinguishable encoding scheme. Then the encoding scheme $\mathsf{E}_b = \left(E_b \circ \mathsf{Enc}, \mathsf{Dec} \circ E_b^{-1}\right)$ is $\left(\mathcal{L}^m_{2n,d,s,\oplus t}, \epsilon\right)$-leakage-indistinguishable.*[7]

Proof. Denote $\mathsf{Enc}_b = E_b \circ \mathsf{Enc}$. If $\mathsf{E}_b = \left(E_b \circ \mathsf{Enc}, \mathsf{Dec} \circ E_b^{-1}\right)$ is *not* $\left(\mathcal{L}^m_{2n,d,s,\oplus t}, \epsilon\right)$-leakage-indistinguishable, then there exists a pair $w_1, w_2 \in \mathbb{F}_3$, and a leakage function $\ell_b \in \mathcal{L}^m_{2n,d,s,\oplus t}$, such that $\mathsf{SD}\left(\ell_b(\mathbf{w}_1), \ell_b(\mathbf{w}_2)\right) > \epsilon$, where $\mathbf{w}_i \leftarrow \mathsf{Enc}_b(w_i, 1^n)$ for $i = 1, 2$ (notice that the output length of $\mathsf{Enc}_b(\cdot, 1^n)$ is $2n$). We show a leakage function $\ell \in \mathcal{L}^m_{n,d+1,s+4n,\oplus t}$ that achieves advantage more that ϵ in distinguishing between $\mathbf{v}_1 \leftarrow \mathsf{Enc}(w_1, 1^n)$ and $\mathbf{v}_2 \leftarrow \mathsf{Enc}(w_2, 1^n)$. On input $\mathbf{z} = (z_1, ..., z_n) \in \{0,1\}^n \subseteq \mathbb{F}_3^n$, ℓ first computes the string $\mathbf{z}' = (0, z_1, 0, z_2, ..., 0, z_n)$ (this can be done in $\mathcal{L}_{n,1,4n,0}$ by adding n constant gates and $2n$ output gates), and outputs $\ell_b(\mathbf{z}')$. Then $\ell \in \mathcal{L}^m_{n,d+1,s+4n,\oplus t}$, and notice that if $\mathbf{z}$ is distributed according to $\mathsf{Enc}(\gamma, 1^n)$ for some $\gamma \in \mathbb{F}_3$, then $\mathbf{z}'$ is distributed according to $\mathsf{Enc}_b(\gamma, 1^n)$. Therefore, $\mathsf{SD}\left(\ell(\mathbf{v}_1), \ell(\mathbf{v}_2)\right) = \mathsf{SD}\left(\ell_b(\mathbf{v}_1'), \ell_b(\mathbf{v}_2')\right) = \mathsf{SD}\left(\ell_b(\mathbf{w}_1), \ell_b(\mathbf{w}_2)\right) > \epsilon$. □

The proof of Theorem 3.37 now follows from Corollary 3.44, Proposition 3.31, and Lemma 3.55 by an appropriate choice of the parameters.

Proof of Theorem 3.37. Let $C, |C| \leq s'$ be the circuit to be compiled. Let $d' = d + 33$, and denote by $\tilde{c}, \tilde{\epsilon}$ the constants whose existence is guaranteed by Corollary 3.44 for the depth parameter $d' + 1$. Let $\mathsf{Enc}_b(\cdot, 1^\sigma) = E_b \circ \mathsf{Enc}_3(\cdot, 1^\sigma)$, $\mathsf{Dec}_b = \mathsf{Dec}_3 \circ E_b^{-1}$ (here, E_b, E_b^{-1} are the mapping and reverse mapping of Definition 3.24, and $\mathsf{Enc}_3, \mathsf{Dec}_3$ are the encoding and decoding algorithms of the encoding scheme of Definition 3.39). Since $\mathsf{E}_3 = (\mathsf{Enc}_3, \mathsf{Dec}_3)$ is linear and onto, then Proposition 3.31 guarantees that there exist constants $z, c' \in \mathbb{N}$ such that when Construction 3.27 is instantiated with $\mathsf{E}_b := (\mathsf{Enc}_b, \mathsf{Dec}_b)$ as the underlying encoding scheme, σ is the security parameter used in Enc_b, and s' is the size parameter, then the following holds for every $\epsilon(n) > 0$, and every families $\mathcal{L}_3, \mathcal{L}$ of leakage

[7] E_b is used to encode field elements in $\mathbb{F}_3$ through bit strings. Formally, an encoding scheme is based on a *single* alphabet over which both the messages, and the encodings, are defined. For E_b to be consistent with the syntactic definition, and with our applications (for boolean SAT-respecting relaxed LRCCs), the input to its encoding algorithm Enc_b is a 2-bit string representing a field element.

functions such that $\mathcal{L} \circ \mathsf{ShallowB}\left(33, z\sigma^5 \cdot (s')^2\right) \subseteq \mathcal{L}_3$.[8] If E_3 is $(\mathcal{L}_3, \epsilon(n))$-leakage-indistinguishable then $(\mathsf{Comp}_b, \mathsf{E}_b)$ is $(\mathcal{L}, c' \cdot \epsilon(n) \cdot s', s')$-relaxed leakage-resilient. Moreover, there exists a constant z' such that for every circuit C of size at most s', the compiled circuit $\hat{C}$ has size at most $z' |C|^2 \cdot \sigma^5$.[9]

Let $p := 2c + \frac{10c}{\tilde{\epsilon}\tilde{c}} + 2$ and $p' = p + 5c + 10$, then Corollary 3.44 guarantees that there exists a minimal length parameter $\tilde{\sigma}_0$ such that for every $\sigma \geq \tilde{\sigma}_0$, $\mathsf{Enc}_3(\cdot, 1^\sigma)$ is $\left(\mathcal{L}_{3\sigma, d'+1, \sigma^{p'}, \oplus\sigma^{\tilde{\epsilon}}}, 2^{-\sigma^{\tilde{c}}}\right)$-leakage-indistinguishable. Let σ_0' be such that for every $\sigma \geq \sigma_0'$, $z + (z')^c \leq \sigma$, and σ_0'' be such that for every $\sigma \geq \sigma_0''$, $c'\sigma 2^{-\sigma^{\tilde{c}}} \leq 1$.

We take $l(n) := l'(n, m, s', t) := z' \cdot (\max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\})^5 (4n^c m s' t)^{2+\frac{10}{\tilde{\epsilon}\tilde{c}}} = \mathsf{poly}(n, m, s', t) = \mathsf{poly}(n)$ (the last equality holds because $s, t, m = \mathsf{poly}(n)$), and $\sigma = (4n^c m s' t)^{\frac{2}{\tilde{\epsilon}\tilde{c}}} \cdot \max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\}$, and instantiate the SAT-respecting circuit-compiler $(\mathsf{Comp}_b, \mathsf{E}_b = (\mathsf{Enc}_b, \mathsf{Dec}_b))$ of Construction 3.27, with size parameter s', and leakage class $\mathcal{L} = \mathcal{L}^m_{d, l^c(n), \oplus t}$ (ϵ will be set later), where *the security parameter of the encoding scheme* E_b *is taken to be* σ (instead of $|C|$). Then for every C such that $|C| \leq s'$, the compiled circuit $\hat{C}$ satisfies $\left|\hat{C}\right| \leq z' |C|^2 \cdot \sigma^5 \leq z' (s')^2 \cdot (\max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\})^5 (4n^c m s' t)^{\frac{10}{\tilde{\epsilon}\tilde{c}}} \leq z' \cdot (\max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\})^5 (4n^c m s' t)^{2+\frac{10}{\tilde{\epsilon}\tilde{c}}} = l'(n, m, s', t) = l(n)$. Moreover, since the length of encodings generated by the encoding scheme was taken to be $\sigma \geq \max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\} \geq \tilde{\sigma}_0$, $\mathsf{Enc}_3(\cdot, 1^\sigma)$ is $\left(\mathcal{L}_{3\sigma, d'+1, \sigma^{p'}, \oplus\sigma^{\tilde{\epsilon}}}, 2^{-\sigma^{\tilde{c}}}\right)$-leakage-indistinguishable. Notice that

$$l^c(n) + z(s')^2 = (z')^c \cdot \left(\max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\}\right)^{5c} \left(4n^c m s' t\right)^{2c + \frac{10c}{\tilde{\epsilon}\tilde{c}}} + z(s')^2$$

which is upper bounded by

$$\left((z')^c + z\right) \cdot \left(\max\{\tilde{\sigma}_0, \sigma_0', \sigma_0''\}\right)^{5c} \left(4n^c m s' t\right)^{2c + \frac{10c}{\tilde{\epsilon}\tilde{c}} + 2}$$

which (by the choice of σ) is at most $\sigma \cdot \sigma^{5c} \sigma^{p\tilde{c}\tilde{\epsilon}+2} \leq^{\tilde{c}, \tilde{\epsilon} < 1} \sigma^{3+5c+p}$, so

$$m \cdot \left(l^c(n) + z(s')^2 \sigma^5\right) + 1 \leq^{m \leq \sigma} \left(l^c(n) + z(s')^2\right) \cdot \sigma^7 \leq \sigma^{p+5c+10} = \sigma^{p'}.$$

Moreover, $mt + 1 \leq \sigma^{\tilde{\epsilon}}$ (because $\tilde{c} < 1$ and $s' \geq 2$). Using Lemma 3.55, this implies that $\mathsf{Enc}_3(\cdot, 1^\sigma)$ is $\left(\mathcal{L}^m_{3\sigma, d', l^c(n) + z(s')^2 \sigma^5, \oplus t}, 2^{\frac{m}{2} - \sigma^{\tilde{c}}}\right)$-leakage-indistinguishable. Therefore, the compiled circuit is $\left(\mathcal{L}^m_{d, l^c(n), \oplus t}, c' \cdot 2^{\frac{m}{2} - \sigma^{\tilde{c}}} \cdot \sigma\right)$-relaxed-leakage-resilient (here, we also use the fact that $s' \leq \sigma$, which holds because $n, m, t \geq 1$ and $\tilde{\epsilon}, \tilde{c} < 1$). We conclude the proof by noticing that since $\tilde{\epsilon}, \tilde{c} < 1$ then $c' \cdot 2^{\frac{m}{2} - \sigma^{\tilde{c}}} \cdot \sigma \leq 2^{\frac{m}{2} - 2n^c m s' t} \cdot c' \sigma 2^{-\frac{\sigma^{\tilde{c}}}{2}}$. Since $s', t \geq 1$, and $\sigma \geq \sigma_0''$, we can upper bound this expression by 2^{-n^c}. □

The next lemma was used in the proof of Theorem 3.37.

Lemma 3.55. *Let $n, n', m, d, s, t \in \mathbb{N}$, let $\epsilon > 0$, and let $f : \{0,1\}^{n'} \to \{0,1\}^n$ be a randomized function. If f is $(\mathcal{L}_{n, d+1, ms+1, \oplus mt+1}, \epsilon)$-leakage-indistinguishable, then f is $\left(\mathcal{L}^m_{n, d, s, \oplus t}, \epsilon \cdot 2^{\frac{m}{2}}\right)$-leakage-indistinguishable.*

[8] We note that Proposition 3.31 requires that $\mathcal{L} \circ \mathsf{ShallowB}\left(33, z\hat{n}^5(s') \cdot (s')^2\right) \subseteq \mathcal{L}_3$, where s' bounds the size of the circuit that is being compiled, but the dependency on $\hat{n}(s')$ is because in the proof of the proposition, the second input to the encoding scheme was taken to be $1^{|C|}$, where $|C| \leq s$. Here, the second input to E_b is σ, not $|C|$. In addition, in E_b, $\hat{n}(m) = O(m)$ for every m, so we can replace $\hat{n}(\sigma)$ with $O(\sigma)$.

[9] We note that according to Proposition 3.31, $\left|\hat{C}\right| \leq z'' |C|^2 \cdot \hat{n}(s')^5$ for some constant z'', but as the second input to the encoding scheme in our construction is 1^σ, not $1^{|C|}$, and since $\hat{n}(m) = O(m)$ for every m, then we can use the bound $\left|\hat{C}\right| \leq z' |C|^2 \cdot \sigma^5$.

The proof uses the following lemma, which generalized Vazirani's XOR Lemma [31, 14] and can be proved similarly. Intuitively, the lemma states that statistical distance is "somewhat preserved" under linear transformations.

Lemma 3.56 (XOR Lemma)**.** *Let X, Y be distributions over $\{0,1\}^m$ such that $\mathsf{SD}(X,Y) = \epsilon$. Then there exists an $\alpha \in \{0,1\}^m$ such that $\mathsf{SD}\left(\alpha^T X, \alpha^T Y\right) \geq \frac{\epsilon}{2^{\frac{m}{2}}}$.*

Proof of Lemma 3.55. If f is *not* $\left(\mathcal{L}^m_{n,d,s,\oplus t}, \epsilon \cdot 2^{\frac{m}{2}}\right)$-leakage-indistinguishable, then there exist $y, z \in \{0,1\}^n$, and a function $\ell \in \mathcal{L}^m_{n,d,s,\oplus t}$, such that $\mathsf{SD}\left(\ell\left(f\left(y\right)\right), \ell\left(f\left(z\right)\right)\right) > \epsilon \cdot 2^{\frac{m}{2}}$. Since $\ell \in \mathcal{L}^m_{n,d,s,\oplus t}$, then there exist m circuits $C_1, ..., C_m : \{0,1\}^n \to \{0,1\}$ of depth at most d and size at most s, with unbounded fan-in and fan-out $\wedge, \vee, \neg, \oplus$ gates, out of which at most t are $\oplus$ gates, where C_i computes ℓ_i, the i'th output bit of ℓ. Let X_y, X_z be random variables over $\{0,1\}^m$ distributed according to $\ell\left(f\left(y\right)\right), \ell\left(f\left(z\right)\right)$, then $\mathsf{SD}\left(X_y, X_z\right) > \epsilon \cdot 2^{\frac{m}{2}}$. Therefore, by Lemma 3.56 there exists an $\alpha \in \{0,1\}^m$ such that $\mathsf{SD}\left(\alpha^T X_y, \alpha^T X_z\right) > \epsilon$. Let $\hat{\ell} : \{0,1\}^n \to \{0,1\}$ be the function computed by the following circuit C. The first (at most) d layers contain the circuits $C_1, ..., C_m$ in parallel. Layer $d+1$ contains a single $\oplus$ gate, whose inputs are the outputs of the circuits $\{C_i : \alpha_i = 1\}$. Then $\hat{\ell} \in \mathcal{L}_{n,d+1,ms+1,\oplus mt+1}$, and notice that $\hat{\ell}\left(f\left(y\right)\right)$ is distributed according to $\alpha^T X_y$, while $\hat{\ell}\left(f\left(z\right)\right)$ is distributed according to $\alpha^T X_z$. Therefore, $\mathsf{SD}\left(\hat{\ell}\left(f\left(y\right)\right), \hat{\ell}\left(f\left(z\right)\right)\right) > \epsilon$, contradicting the $(\mathcal{L}_{n,d+1,ms+1,\oplus mt+1}, \epsilon)$-leakage-indistinguishability of f. □

4 WIPCPs and CZKPCPs

In this section, we use SAT-respecting relaxed LRCCs to transform standard PCPs into WIPCPs and CZKPCPs in the CRS model, with a *non-adaptive* honest verifier. We first give an overview of probabilistic proof systems, and then discuss WIPCPs (Section 4.1) and CZKPCPs (Section 4.2).

Given a relation $\mathcal{R} = \mathcal{R}(x,w)$, we let $L_{\mathcal{R}} := \{x : \exists w, (x,w) \in \mathcal{R}\}$. A *probabilistic proof system* (P,V) for an NP-relation $\mathcal{R} = \mathcal{R}(x,w)$ consists of a *PPT* prover P that on input (x,w) outputs a proof π (in standard probabilistically checkable proofs the prover is deterministic, but our constructions will crucially rely on the prover being probabilistic), and a probabilistic verifier V that given input x and oracle access to a proof π outputs either accept or reject. We say that V is *q-query-bounded* if V makes at most q queries to π.

4.1 WIPCPs

In this section we define WIPCPs, describe a general transformation from SAT-respecting relaxed LRCCs to WIPCPs, and then use the compiler of Theorem 3.37, together with the PCP of [1], to construct a WIPCP system.

Intuitively, a probabilistic proof system is a WIPCP for an NP-relation $\mathcal{R} = \mathcal{R}(x,w)$ if it satisfies the following. First, given oracle access to an honestly generated proof for $x \in L_{\mathcal{R}}$, the verifier always accepts. Second, given $x \notin L_{\mathcal{R}}$, the verifier rejects except with some probability ϵ_S, *regardless* of its "proof" oracle. Thirdly, for every (possibly malicious, possibly adaptive) q^*-query bounded verifier V^*, every $x \in L_{\mathcal{R}}$, and every pair w_1, w_2 of witnesses for x, the view of V^* when verifying an honestly generated proof for (x, w_1) is ϵ_{ZK}-statistically close to its view when verifying an honestly generated proof for (x, w_2). Formally,

Definition 4.1 (WIPCP)**.** We say that a probabilistic proof system (P,V) is a *witness-indistinguishable probabilistically checkable proof (WIPCP)* system for an NP-relation $\mathcal{R} = \mathcal{R}(x,w)$, if the following holds.

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

  We associate with P, V as above the following efficiency measures. The *alphabet* $\Sigma = \Sigma\left(\epsilon_S, \epsilon_{\text{ZK}}, q^*, |x|\right)$ over which π is defined; The *length* $\ell = \ell\left(\epsilon_S, \epsilon_{\text{ZK}}, q^*, |x|\right)$ of the proof π; the *query complexity* $q = q\left(\epsilon_S, \epsilon_{\text{ZK}}, q^*, |x|\right)$ of V (i.e., the number of queries V makes to his oracle); and the *randomness complexity* $r = r\left(\epsilon_S, \epsilon_{\text{ZK}}, q^*, |x|\right)$ 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 $\pi \in P\left(\epsilon_S, \epsilon_{\text{ZK}}, 1^{q^*}, x, w\right)$, $\Pr\left[V^{\pi}\left(\epsilon_S, \epsilon_{\text{ZK}}, q^*, x\right) = \mathsf{acc}\right] = 1$, where the probability is over the randomness of V.
  - *Soundness.* For every $x \notin L_{\mathcal{R}}$ and every π^*, $\Pr\left[V^{\pi^*}\left(\epsilon_S, \epsilon_{\text{ZK}}, q^*, x\right) = \mathsf{acc}\right] \leq \epsilon_S$.
  - $(\epsilon_{\text{ZK}}, q^*)$*-witness-indistinguishability (WI).* For every (possibly adaptive) q^*-query-bounded verifier V^*, every $x \in L_{\mathcal{R}}$, and every pair w_1, w_2 of witnesses for x (i.e., $(x,w_1), (x,w_2) \in \mathcal{R}$), $\mathsf{SD}\left(\mathsf{Real}_{V^*,P}\left(x,w_1\right), \mathsf{Real}_{V^*,P}\left(x,w_2\right)\right) \leq \epsilon_{\text{ZK}}$, where $\mathsf{Real}_{V^*,P}\left(x,w\right)$ denote the view of V^* on input $\epsilon_S, \epsilon_{\text{ZK}}, q^*, x$, and given oracle access to $\pi \leftarrow P\left(\epsilon_S, \epsilon_{\text{ZK}}, 1^{q^*}, x, w\right)$.

We say that a WIPCP is a *non-adaptive WIPCP (NA-WIPCP)* system for a relation $\mathcal{R} = \mathcal{R}\left(x,w\right)$, if the *honest* verifier is non-adaptive. That is,

Definition 4.2 (Non-adaptive WIPCP)**.** We say that a probabilistic proof system (P, V) is a *non-adaptive WIPCP (NA-WIPCP)* system for an NP-relation $\mathcal{R} = \mathcal{R}\left(x,w\right)$, if it is a WIPCP system in which the honest verifier is non adaptive, i.e., his queries are determined by his inputs and randomness.

Notation 4.3. We use $\mathsf{NA}-\mathsf{WIPCP}\left[r, q, q^*, \epsilon_S, \epsilon_{\text{ZK}}, \ell\right]$ to denote the class of NP-languages that admit an NP-relation $\mathcal{R}$ with a non-adaptive $(\epsilon_{\text{ZK}}, q^*)$-WIPCP, in which the prover outputs proofs of length ℓ, the honest verifier tosses $O\left(r\right)$ coins, queries $O\left(q\right)$ proof bits, and rejects false claims except with probability at most ϵ_S. We use $\mathsf{PCP}\left[r, q, \epsilon, \ell\right]$ to denote the class of NP-languages admitting a standard (i.e., non-WI) PCP system with the same properties, and write $\mathcal{R} \in \mathsf{PCP}\left[r, q, \epsilon, \ell\right]$ to denote that $L_{\mathcal{R}} \in \mathsf{PCP}\left[r, q, \epsilon, \ell\right]$. We denote $\mathsf{NA}-\mathsf{WIPCP} := \mathsf{NA}-\mathsf{WIPCP}[\mathsf{poly}\log n, \mathsf{poly}\log n, \mathsf{poly}\left(n\right), \mathsf{negl}\left(n\right), \mathsf{negl}\left(n\right), \mathsf{poly}\left(n\right)]$.

We describe a transformation from PCPs for 3SAT to NA-WIPCPs for arbitrary NP-relations $\mathcal{R} = \mathcal{R}\left(x,w\right)$, which can be applied to any PCP system for 3SAT in which the proof is obtained from the witness through an "easy" function f_{3SAT} (we formalize this notion below). If $(x,w) \in \mathcal{R}$ then w satisfies $C_{\mathcal{R}}\left(x,\cdot\right)$ (i.e., $C_{\mathcal{R}}$ with x hard-wired into it). If every "small" subset of bits in the output of f_{3SAT} is constitutes a function in a function class $\mathcal{L}$, then the system can be made WI as follows. The prover and verifier both compile $C_{\mathcal{R}}\left(x,\cdot\right)$ into a SAT-respecting circuit $\hat{C}_{\mathcal{R}}$ that is relaxed leakage-resilient against $\mathcal{L}$, and then generate a 3CNF φ that represents $\hat{C}_{\mathcal{R}}$ (see Definition 4.5 below). Notice that by the SAT-respecting property, $\hat{C}_{\mathcal{R}}$ (and consequently, also φ) is satisfiable if and only if $x \in L_{\mathcal{R}}$. The prover then samples a random encoding $\hat{w}$ of w (notice that $\left[\hat{C}_{\mathcal{R}}, \hat{w}\right]$ is a satisfying assignment for φ), and generates the PCP $\pi = f_{\text{3SAT}}\left[\hat{C}_{\mathcal{R}}, \hat{w}\right]$. The verifier probabilistically verifies that φ is satisfiable by reading few symbols of π, which (if the verifier is non-adaptive) correspond to applying a leakage function from $\mathcal{L}$ to the wire values of $\left[\hat{C}_{\mathcal{R}}, \hat{w}\right]$. This gives the following result.

Proposition 4.4. *Let n be a length parameter, $\epsilon_S, \epsilon_{\mathrm{ZK}} \in [0,1]$, $\mathsf{S} = \mathsf{S}(n)$ be a size function, $q^* = q^*(n)$ be a query function, and $g(\cdot)$ be a polynomial. Let $\mathcal{L}$ be a family of leakage functions, such that:*

- *there is a SAT-respecting $(\mathcal{L}, \epsilon_{\mathrm{ZK}}, \mathsf{S})$-relaxed LRCC $(\mathsf{Comp}, \mathsf{E})$ satisfying $|\mathsf{Comp}(\mathsf{C})| \leq g(|C|)$ for every circuit C;*
- *there is a $\mathsf{PCP}[r(n), q(n), \epsilon_S, \ell(n)]$ system for 3SAT, such that for every $(\varphi, W) \in$ 3SAT, every subset $\mathcal{Q}$ of q^* bits of an honestly-generated proof $\pi = \pi(\varphi, W)$ is computable from W by a function $f_{\varphi,\mathcal{Q}} \in \mathcal{L}$.*

Then for every NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ with verification circuit $C^{\mathcal{R}}$ of size at most S, we have that $\mathcal{R} \in \mathsf{NA} - \mathsf{WIPCP}[r(t), q(t), q^*, \epsilon_S, 2\epsilon_{\mathrm{ZK}}, \ell(t)]$, where $t = O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)$, and WI holds against non-adaptive *verifiers.*

Before proving Proposition 4.4, we first explicitly define the WIPCP system, starting with the 3CNF-representation of circuits.

Representing computations as 3CNFs. We will use boolean formulas to represent computations of boolean circuits.

Definition 4.5 (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, $c = a \wedge b$: $\varphi(x_a, x_b, x_c) = (x_c \vee \neg x_a \vee \neg x_b) \wedge (\neg x_c \vee x_a \vee \neg x_c) \wedge (\neg x_c \vee x_b \vee \neg x_c)$.[10]
    - 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 4.6. 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).

[10] 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.

- $\varphi_{g_\wedge}(x_y, x_z, x_\wedge) = (x_\wedge \vee \neg x_y \vee \neg x_z) \wedge (\neg x_\wedge \vee x_y \vee \neg x_\wedge) \wedge (\neg x_\wedge \vee x_z \vee \neg x_\wedge)$.
- $\varphi_{g_\neg}(x_y, x_\neg) = (\neg x_\neg \vee \neg x_y \vee \neg x_\neg) \wedge (x_\neg \vee x_y \vee x_\neg)$.
- $\varphi_{g_\vee}(x_\wedge, x_\neg, x_\vee) = (\neg x_\vee \vee x_\wedge \vee x_\neg) \wedge (x_\vee \vee \neg x_\wedge \vee x_\vee) \wedge (x_\vee \vee \neg x_\neg \vee x_\vee) \wedge (x_\vee \vee x_\vee \vee x_\vee)$ (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 4.7. *Let C be a boolean circuit, and let φ_C be the canonical 3CNF representing C (as in Definition 4.5). 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.*

Using the canonical 3CNF representation of boolean circuits, we can now formally describe the NA-WIPCP system.

Construction 4.8 (NA-WIPCP)**.** Let $(P_{3\text{SAT}}, V_{3\text{SAT}})$ be a PCP system for 3SAT, and let $(\mathsf{Comp}, \mathsf{E} = (\mathsf{Enc}, \mathsf{Dec}))$ be a SAT-respecting $(\mathcal{L}, \epsilon(n), \mathsf{S}(n))$-relaxed LRCC. Let $\mathcal{R} = \mathcal{R}(x, w)$ be an NP-relation with verification circuit $C^{\mathcal{R}}$. (More precisely, $C^{\mathcal{R}}$ is a family $\left\{C_n^{\mathcal{R}}\right\}$ of circuits, where $C_n^{\mathcal{R}}$ is applied to inputs x of length n. To simplify notations, we denote all circuits in the family by $C^{\mathcal{R}}$.) The NA-WIPCP system consists of the prover P and the verifier V.
Prover algorithm. On input $(x, w) \in \mathcal{R}$:

- Let $C^{\mathcal{R}}(x, \cdot)$ denote the circuit $C^{\mathcal{R}}$ with x hard-wired into it, then P computes $\hat{C}_x(\cdot) = \mathsf{Comp}\left(C^{\mathcal{R}}(x, \cdot)\right)$.
- Samples a random encoding $\hat{w} \leftarrow \mathsf{Enc}\left(w, 1^{|C^{\mathcal{R}}|}\right)$, and computes the internal wires of $\hat{C}_x(\hat{w})$. Let $\mathcal{W}$ denote this wire assignment.
- Construct the canonical 3CNF φ_x representing $\hat{C}_x$.
- Outputs a proof $\pi \in_R P_{3\text{SAT}}(\varphi_x, \mathcal{W})$ for the claim "$\varphi_x \in$ 3SAT".

Verifier algorithm. On input x, and given oracle access to π, V computes $\hat{C}_x(\cdot) = \mathsf{Comp}\left(C^{\mathcal{R}}(x, \cdot)\right)$, and constructs the 3CNF formula φ_x. Then, V runs $V_{3\text{SAT}}^{\pi}(\varphi_x)$ (and accepts or rejects according to the output of $V_{3\text{SAT}}$).

Remark 4.9. The prover and verifier of the PCP system for 3SAT may expect additional parameters as part of their input. In such cases, these parameters would also be given as input to the prover and verifier of the NA-WIPCP, who will provide them as part of the input to $P_{3\text{SAT}}, V_{3\text{SAT}}$. However, to make the construction clearer, we have chosen not to explicitly include these additional parameters in Construction 4.8.

Next, we prove Proposition 4.4, that determines the relation between the queries of the verifier, and the family of leakage functions, for which Construction 4.8 would be witness-indistinguishable.

Proof of Proposition 4.4. We show that the system of Construction 4.8, when using $(P_{3\text{SAT}}, V_{3\text{SAT}})$ and $(\mathsf{Comp}, \mathsf{E})$ as the underlying components, has the required properties.

PARAMETERS. The wire assignment $\mathcal{W}$ to $\hat{C}_x$ has size $|\mathcal{W}| = \left|\hat{C}_x\right| \leq g\left(\left|C^{\mathcal{R}}(x,\cdot)\right|\right) \leq g\left(\left|C^{\mathcal{R}}\right|\right)$. V emulates the verification procedure of the inner verifier $V_{3\text{SAT}}$, on input φ_x of size $|\varphi_x| = O\left(\left|\hat{C}_x\right|\right) \leq O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)$. Therefore, the query complexity of V is $O\left(q\left(O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)\right)\right)$, and the randomness complexity is $O\left(r\left(O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)\right)\right)$.[11] The proof is generated from the witness $\mathcal{W}$, that has size at most $O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)$, so the proof has size $\ell\left(O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)\right)$.

PERFECT COMPLETENESS follows directly from the perfect completeness of the underlying systems.

SOUNDNESS. Let $x \notin L_{\mathcal{R}}$, then for every "witness" w, $C(x,w) = 0$, i.e., C_x is not satisfiable. Since $(\mathsf{Comp}, \mathsf{E})$ is SAT-respecting, $\hat{C}_x$ is not satisfiable, i.e., $\varphi_x \notin 3\text{SAT}$. (Notice that here *perfect* SAT-respecting is *crucial*, otherwise there may exist a satisfying input $\hat{w}^*$ for $\hat{C}_x$, even though $C(x,\cdot)$ is not satisfiable, and a malicious prover would use $\hat{w}^*$ to convince the verifier.) Therefore, the soundness of $(P_{3\text{SAT}}, V_{3\text{SAT}})$ guarantees that for *every* π^*, $\Pr\left[V^{\pi^*}(x) = \mathsf{acc}\right] = \Pr\left[V_{3\text{SAT}}^{\pi^*}(\varphi_x) = \mathsf{acc}\right] \leq \epsilon_S$.

WITNESS-INDISTINGUISHABILITY. Let $x \in L_{\mathcal{R}}$, φ_x be the canonical 3CNF representing $\hat{C}_x$, and w_1, w_2 be two witnesses for x. Let V^* be a non-adaptive q^*-query-bounded verifier, and let π_1, π_2 be proofs that were randomly generated by the honest prover P for w_1, w_2, respectively. The entire view of V^* can be reconstructed from the oracle answers to his queries, and since applying a function to a pair of random variables does not increase the statistical distance between them, it suffices to show that for every set $\mathcal{Q}$ of q^* symbols of π_1, π_2, $\mathsf{SD}\left(\pi_1|_{\mathcal{Q}}, \pi_2|_{\mathcal{Q}}\right) \leq 2\epsilon_{\text{ZK}}$, where $\pi_i|_{\mathcal{Q}}$ denotes the restriction of π_i to the coordinates indexed by $\mathcal{Q}$. Since P evaluates $\hat{C}_x$ on random encodings $\hat{w}_i \leftarrow \mathsf{Enc}\left(w_i, 1^{|C(x,\cdot)|}\right)$ of w_i, $i = 1, 2$, and $\left|C^{\mathcal{R}}\right| \leq \mathsf{S}$, then the relaxed leakage-resilience of $(\mathsf{Comp}, \mathsf{E})$ guarantees that for every $f \in \mathcal{L}$, $\mathsf{SD}\left(f\left[\hat{C}_x, \hat{w}_1\right], f\left[\hat{C}_x, \hat{w}_2\right]\right) \leq 2\epsilon_{\text{ZK}}$. (Using the union bound, this holds because both $\pi_1|_{\mathcal{Q}}, \pi_2|_{\mathcal{Q}}$ are ϵ_{ZK}-statistically close to the output of the leakage function on the simulated wire values generated by the simulator of the relaxed LRCC.) Moreover, $f_{\varphi_x,\mathcal{Q}} \in \mathcal{L}$, and $\pi_i|_{\mathcal{Q}} = f_{\varphi_x,\mathcal{Q}}\left[\hat{C}_x, \hat{w}_i\right]$ (recall that $f[C,x]$ denotes the output of f, when given as input the wire assignment of the circuit C when evaluated on input x), so by the $(\epsilon_{\text{ZK}}, \mathcal{L})$-relaxed leakage-resilience of Comp, $\mathsf{SD}\left(\pi_1|_Q, \pi_2|_Q\right) \leq 2\epsilon_{\text{ZK}}$. □

Proposition 4.4 shows that Construction 4.8 is WI against *non-adaptive* (possibly malicious) query-bounded verifiers. Using techniques of [7] (see Theorem 4.11 below), we can generalize the WI property of Proposition 4.4 to *adaptive* verifiers, while increasing the statistical distance of the WI property by a multiplicative factor of roughly ℓ^{q^*} (all other parameters remain unchanged). Formally,

Corollary 4.10. *Let n be a length parameter, $\epsilon_S, \epsilon_{\text{ZK}} \in [0,1]$, $\mathsf{S} = \mathsf{S}(n)$ be a size function, $q^* = q^*(n)$ be a query function, and $g(\cdot)$ be a polynomial. Let $\mathcal{L}$ be a family of leakage functions, such that:*

- *there is a SAT-respecting $(\mathcal{L}, \epsilon_{\text{ZK}}, \mathsf{S})$-relaxed LRCC $(\mathsf{Comp}, \mathsf{E})$ satisfying $|\mathsf{Comp}(\mathsf{C})| \leq g(|C|)$ for every circuit C;*
- *there is a $\mathsf{PCP}\left[r(n), q(n), \epsilon_S, \ell(n)\right]$ system for 3SAT, such that for every $(\varphi, W) \in 3\text{SAT}$, every subset $\mathcal{Q}$ of q^* bits of an honestly-generated proof $\pi = \pi(\varphi, W)$ is computable from W by a function $f_{\varphi,\mathcal{Q}} \in \mathcal{L}$.*

[11] Since $g\left(\left|C^{\mathcal{R}}\right|\right)$ is an upper-bound on the size of the formula, we assume here that q, r, ℓ are non-decreasing, which is without loss of generality.

Then for every NP-relation $\mathcal{R} = \mathcal{R}(x,w)$ *with verification circuit* $C^{\mathcal{R}}$ *of size at most* S*, we have that* $\mathcal{R} \in \mathsf{NA}-\mathsf{WIPCP}\left[r(t), q(t), q^*, \epsilon_S, O\left(\epsilon_{\mathrm{ZK}} \cdot q^* \cdot (\ell(t))^{2q^*}\right) + e^{-\Omega\left(q^* \cdot (\ell(t))^{q^*}\right)}, \ell(t)\right]$*, where* $t = O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)$*.*

Proof. The proof follows by applying Theorem 4.11 to the NA-WIPCP system of Proposition 4.4, and by noting that the proofs in the NA-WIPCP system of Proposition 4.4 have length $\ell(t)$. □

The proof of Corollary 4.10 used the following result of [7].

Theorem 4.11 (Implicit in [7])**.** *Let* (P,V) *be a PCP system which is* (ϵ, q^*)*-WI against* non-adaptive *verifiers, and in which the prover generates proofs of length* ℓ*. Then* (P,V) *is also* $\left(O\left(\epsilon \cdot q^* \cdot \ell^{2q^*}\right) + e^{-\Omega\left(q^* \cdot \ell^{q^*}\right)}, q^*\right)$*-WI against* adaptive *verifiers.*

Using the circuit compiler of Theorem 3.37, we can now state a specific restriction on the function transforming a satisfying assignment for a 3CND into a corresponding PCP, such that a relation $\mathcal{R}$ whose verification circuit is "not too large" would have an NA-WIPCP system.

Corollary 4.12. *Let* n *be a length parameter,* $\epsilon_S, \epsilon_{\mathrm{ZK}} \in [0,1]$*, and* $q^* = q^*(n) = \mathsf{poly}(n)$ *be a query function. Assume that* $3\mathrm{SAT} \in \mathsf{PCP}\left[r(n), q(n), \epsilon_S, \ell(n)\right]$ *with the system* $(P_{3\mathrm{SAT}}, V_{3\mathrm{SAT}})$*, where* $\ell(n) = \mathsf{poly}(n)$*, and for every* $(\varphi, W) \in 3\mathrm{SAT}$*, every bit of an honestly-generated proof* $\pi = \pi(\varphi, W)$ *is computable by* $\mathcal{L}_{O(1),\mathsf{poly}(n),\oplus 1}$ *(i.e., by a constant-depth, poly-sized, boolean circuit with unbounded fan-in and fan-out* $\wedge, \vee, \neg, \oplus$ *gates, out of which only one is an* $\oplus$ *gate, see Definition 3.40). Then* $\mathsf{NP} \subseteq \mathsf{NA}-\mathsf{WIPCP}\left[r(\mathsf{poly}(n)), q(\mathsf{poly}(n)), q^*, \epsilon_S, \mathsf{negl}(n), \ell(\mathsf{poly}(n))\right]$*.*

Proof. The corollary follows from applying Corollary 4.10 to the circuit compiler of Theorem 3.37. More specifically, let $d, c \in \mathbb{N}$ be constants such that every proof bit generated by the honest $P_{3\mathrm{SAT}}$ is computable from the NP-witness in $\mathcal{L}_{d,n^c,\oplus 1}$, where n denotes the witness length. Moreover, proofs generated by the prover have length at most $n^{c''}$, for some constant c''.

Let $\mathcal{R}$ be an NP-relation with verification circuit $C = C_{\mathcal{R}}$, then $|C| = n^{c'}$ for some constant c' (because $\mathcal{R}$ is polynomially bounded and efficiently computable). We use Theorem 3.37 with parameters $d_{\mathsf{Comp}} = d, s'_{\mathsf{Comp}} = |C|, n_{\mathsf{Comp}} = n$, $t_{\mathsf{Comp}} = 1$, $m_{\mathsf{Comp}} = nq^*$, and take $c_{\mathsf{Comp}} \geq c$ to be a large enough constant whose value will be set later (the subscript Comp is used to denote the parameters in the statement of Theorem 3.37, and to differentiate them from the constants mentioned in the proof). Notice that $s'_{\mathsf{Comp}}, t_{\mathsf{Comp}}, m_{\mathsf{Comp}}$ are all polynomial in n. We apply the circuit compiler $(\mathsf{Comp}, \mathsf{E})$ obtained from Theorem 3.37 with these parameters, to C, and obtain a circuit $\hat{C}$ of size $\left|\hat{C}\right| \leq l(n)$ which, because $|C| \leq s'_{\mathsf{Comp}}$, is $\left(\mathcal{L}^{q^*}_{d,l^c(n),\oplus 1}, 2^{-n^{c_{\mathsf{Comp}}}}\right)$-relaxed leakage-resilient (where $l(n)$ is the polynomial whose existence is guaranteed by Theorem 3.37). Let $k \in \mathbb{N}$ be the constant such that $l(n) = n^k$.

Let φ denote the canonical 3CNF representing $\hat{C}$ (see Definition 4.5). Then a wire assignment $\mathcal{W}$ for $\hat{C}$, which is also an assignment to the variables of φ, has size $|\mathcal{W}| = \left|\hat{C}\right| \leq l(n)$. Moreover, every proof bit of a proof generated by $P_{3\mathrm{SAT}}$ for φ can be generated from $\mathcal{W}$ in $\mathcal{L}_{d,l^c(n),\oplus 1}$. Therefore, every q^* proof bits are computable from $\mathcal{W}$ in $\mathcal{L}^{q^*}_{d,l^c(n),\oplus 1}$, and Corollary 4.10 guarantees that the system (P,V) of Construction 4.8 is an NA-WIPCP system for $\mathcal{R}$, with $\left(q^*, O\left(q^* \cdot \ell^{2q^*} \cdot 2^{-n^{c_{\mathsf{Comp}}}}\right) + e^{-\Omega\left(q^* \cdot \ell^{q^*}\right)}\right)$-WI (where ℓ denotes the proof length), perfect completeness, and soundness error ϵ_S. Since $g(|C|) = l(n) = n^k$, then the proof length is $\ell = \ell\left(n^k\right) = n^{kc''}$.

We set c_{Comp} to be large enough, such that the statistical error in the relaxed leakage-resilience property is

$$O\left(q^* \cdot \left(n^{kc''}\right)^{2q^*} \cdot 2^{-n^{c_{\mathsf{Comp}}}}\right) + e^{-\Omega\left(q^* \cdot \ell^{q^*}\right)} = \mathsf{negl}\left(n\right).$$

(Such a choice of c_{Comp} exists since

$$O\left(q^* \cdot \left(n^{kc''}\right)^{2q^*} \cdot 2^{-n^{c_{\mathsf{Comp}}}}\right) + e^{-\Omega\left(q^* \cdot \ell^{q^*}\right)} \leq O\left((q^* n)^{2q^* kc''} \cdot 2^{-n^{c_{\mathsf{Comp}}}}\right) + e^{-\Omega\left(q^* \cdot \ell^{q^*}\right)}$$

where the left had side is equal to $2^{c' \cdot 2q^* kc'' \log(nq^*) - n^{c_{\mathsf{Comp}}}} + 2^{-c' q^* \cdot n^{q^* kc''}}$ for some constant c', because $q^* = \mathsf{poly}\left(n\right)$, and $k, c'', c_{\mathsf{Comp}}$ are constants.) Finally, we note that the honest verifier is non adaptive, tosses $O\left(r\left(\mathsf{poly}\left(nq^*\right)\right)\right) = O\left(r\left(\mathsf{poly}\left(n\right)\right)\right)$ coins and reads $O\left(q\left(\mathsf{poly}\left(nq^*\right)\right)\right) = O\left(q\left(\mathsf{poly}\left(n\right)\right)\right)$ bits of the proof.

□

In Appendix B we show that the PCP system of Arora and Safra [2] for 3SAT has the property that every proof bit is computable from the witness by an AC^0 circuit, augmented with a single $\oplus$ gate of unbounded fan-in and fan-out. Combining this PCP system with Corollary 4.12 yields Theorem 1.1.

Proof of Theorem 1.1. We show that $\mathsf{NP} \subseteq \mathsf{NA} - \mathsf{WIPCP}$. Let $\left(P^{\mathrm{AS}}, V^{\mathrm{AS}}\right)$ denote the PCP system of Arora and Safra [2, Theorem 1] (used to prove that $3\mathrm{SAT} \in \mathsf{PCP}\left[\log n, \log^2 n, \frac{1}{2}, \mathsf{poly}\left(n\right)\right]$). In their system, there exists a constant $c > 0$ such that every proof bit is computable given the witness by a depth-3 boolean circuit, where the first layer contains n^c constant gates, the second layer contains $n^{c'}$ unbounded fan-in $\wedge$ gates, and the third layer consists of a single unbounded fan-in $\oplus$ gate. (See Appendix B for a more detailed description of $\left(P^{\mathrm{AS}}, V^{\mathrm{AS}}\right)$.) In particular, every proof bit is computable, given the witness, by an AC^0 circuit, augmented with a single $\oplus$ gate of unbounded fan-in and fan-out. Applying Corollary 4.12 to $\left(P^{\mathrm{AS}}, V^{\mathrm{AS}}\right)$ with $q^* = \mathsf{poly}\left(n\right)$, and amplifying soundness via repetition, we get that $\mathsf{NP} \subseteq \mathsf{WIPCP}\left[\mathsf{poly}\log\left(n\right), \mathsf{poly}\log\left(n\right), \mathsf{poly}\left(n\right), \mathsf{negl}\left(n\right), \mathsf{negl}\left(n\right), \mathsf{poly}\left(n\right)\right]$. □

4.2 CZKPCPs in the CRS Model

In this section we construct NA-CZKPCPs in the CRS model from NA-WIPCPs. Roughly speaking, a probabilistic proof system is a CZKPCP in the CRS model for an NP-relation $\mathcal{R} = \mathcal{R}\left(x, w\right)$ if the prover and verifier have access to a common random string s; correctness holds for *any* s; soundness holds for a uniformly random s; and there exists a PPT simulator Sim such that for every q^*-query bounded verifier V^*, and every $x \in L_{\mathcal{R}}$, $\mathsf{Sim}\left(x\right)$ is computationally indistinguishable from the joint distribution of a uniformly random s, and the view of V^* given s and oracle access to an honestly generated proof for x. Formally,

Definition 4.13 (Computational ZKPCP in the CRS model)**.** We say that a probabilistic proof system (P, V) is a *computational zero-knowledge probabilistically checkable proof (CZKPCP)* system for an NP-relation $\mathcal{R} = \mathcal{R}\left(x, w\right)$ *in the CRS model*, if the following holds.

- *Syntax.* The prover P has input $\epsilon_S, 1^{q^*}, 1^{\sigma}, x, w$, and access to a common random string (CRS) $s \in \{0,1\}^{\sigma}$, and outputs a proof π for (x, w) (i.e., $P\left(\epsilon_S, 1^{q^*}, 1^{\sigma}, x, w, s\right)$ defines a distribution over proofs for (x, w)). The verifier V has input $\epsilon_S, q^*, 1^{\sigma}, x$, access to s, and oracle access to π, and outputs either acc or rej.

  We associate with P, V as above the following efficiency measures. The *alphabet* $\Sigma = \Sigma\left(\epsilon_S, q^*, \left|x\right|, \sigma\right)$ over which π is defined; The *length* $\ell = \ell\left(\epsilon_S, q^*, \left|x\right|, \sigma\right)$ of the proof π; the

query complexity $q = q(\epsilon_S, q^*, |x|, \sigma)$ of V (i.e., the number of queries that V makes to his oracle); and the *randomness complexity* $r = r(\epsilon_S, q^*, |x|, \sigma)$ of V (namely, the number of random bits that he uses).

- *Semantics.* (P, V) should have the following properties.
  - *Completeness.* For every $(x, w) \in \mathcal{R}$, every CRS $s \in \{0,1\}^\sigma$, and every proof $\pi \in P\left(\epsilon_S, 1^{q^*}, 1^\sigma, x, w, s\right)$, $\Pr\left[V^\pi(\epsilon_S, q^*, 1^\sigma, x, s) = \mathsf{acc}\right] = 1$, where the probability is over the randomness of V.
  - *Soundness.* For every $x \notin L_\mathcal{R}$ and every π^*, if $s \in_R \{0,1\}^\sigma$ then $\Pr\left[V^{\pi^*}(\epsilon_S, q^*, 1^\sigma, x, s) = \mathsf{acc}\right] \leq \epsilon_S$, where the probability is over the choice of s, and the random coins used by V.
  - *q^*-computational zero-knowledge (CZK).* There exists a PPT simulator Sim such that the following holds for every q^*-query-bounded (possibly malicious) PPT verifier V^*, and every $(x, w) \in \mathcal{R}$. The simulator Sim on input $x, 1^\sigma$ generates a simulated CRS s_Sim, and gives s_Sim to V^*. Then, V^* (adaptively) queries the proof, and Sim generates simulated answers to these queries. At the end of this interaction, Sim outputs a simulated view of V^*. Then $(x, s, \mathsf{Real}_{V^*,P}(x, w, s)) \approx \mathsf{Sim}(x, 1^\sigma)$, where $\mathsf{Real}_{V^*,P}(x, w, s)$ denotes the view of V^* on input $\epsilon_S, q^*, x, 1^\sigma$, given access to $s \in_R \{0,1\}^\sigma$ and oracle access to $\pi \in_R P\left(\epsilon_S, 1^{q^*}, 1^\sigma, x, w, s\right)$; $\mathsf{Sim}(x, 1^\sigma)$ denotes the output of Sim on input $x, 1^\sigma$; and $\approx$ denotes computational indistinguishability.

Remark 4.14. It would sometimes be useful to bound the computational distance between $(x, s, \mathsf{Real}_{V^*,P}(x, w, s))$ and $\mathsf{Sim}(x, 1^\sigma)$ precisely. Therefore, we say that a CZKPCP system has $(\epsilon_{\mathrm{ZK}}, q^*)$-CZK if the computational distance between $(x, s, \mathsf{Real}_{V^*,P}(x, w, s))$ and $\mathsf{Sim}(x, 1^\sigma)$ is at most ϵ_{ZK}. In this case, ϵ_{ZK} is given to both P and V, and the parameters of the system (i.e., Σ, ℓ, q, r) may depend on ϵ_{ZK}.

Similar to WIPCPs, we consider non-adaptive CZKPCPs:

Definition 4.15 (Non-adaptive CZKPCP)**.** We say that a probabilistic proof system (P, V) is a *non-adaptive CZKPCP (NA-CZKPCP)* system for an NP-relation $\mathcal{R} = \mathcal{R}(x, w)$, if it has the syntax, completeness, soundness, and CZK of a CZKPCP system (Definition 4.13); and the *honest* verifier is non-adaptive, i.e., his queries are determined by his inputs and randomness.

We use a technique of Fiege et al. [12] to construct CZKPCPs in the CRS model from WIPCPs. Though [12] use this technique to construct non-interactive zero-knowledge proofs (NIZKs) from *non-interactive WI proofs*, the same transformation can also be applied to WIPCPs. The high-level idea is as follows. To construct a CZKPCP for an NP-relation $\mathcal{R}_L$, we use a WIPCP system for a "related" NP-relation, which admits a trapdoor used by the simulator in the simulation (this trapdoor is the reason that a CRS is needed). Formally,

Construction 4.16 (From WIPCP to CZKPCP in the CRS model)**.** Let $G : \{0,1\}^\sigma \to \{0,1\}^{p(\sigma)}$ be a PRG, with some stretch function $p : \mathbb{N} \to \mathbb{N}$. Let L be an NP-language with the corresponding NP-relation $\mathcal{R} = \mathcal{R}(x, w)$, then we define an NP-language L_G as follows: $L_G = \{(x, y) : x \in L \vee y \in \mathrm{Im}(G)\}$. Let $\mathcal{R}_G$ denote the corresponding NP-relation, and let $\left(P^{\mathrm{in}}, V^{\mathrm{in}}\right)$ be a WIPCP system for L_G.[12]

[12] It suffices for $\left(P^{\mathrm{in}}, V^{\mathrm{in}}\right)$ to be a WIPCP system for *any* NP-complete language L^{in}, since the prover and verifier of Construction 4.16 can reduce instances in L_G to instances in L^{in} using a witness-preserving transformation. We choose to use a WIPCP system for L_G because it makes the presentation clearer.

PROVER ALGORITHM. The prover P, on input $(x,w) \in \mathcal{R}$, and given access to a CRS $s \in \{0,1\}^{p(\sigma)}$, runs P^{in} on $(x,s),w$ and outputs the proof π that P^{in} outputs.
VERIFIER ALGORITHM. The verifier V, on input x, given access to the CRS s, and oracle access to a proof π, runs V^{in} on input (x,s), with oracle π, and outputs whatever V^{in} outputs.

The next proposition summarizes the properties of Construction 4.16. We say that a PRG G has $\epsilon(n)$-pseudorandom output against *non-uniform* distinguishers, if for every n, and every (possibly non-uniform) polynomial distinguisher D, $\left|\Pr_{u \leftarrow U_{p(n)}}[D(u)=1] - \Pr_{s \leftarrow U_n}[D(G(s))=1]\right| \leq \epsilon(n)$, where U_n denotes the uniform distribution over $\{0,1\}^n$.

Proposition 4.17. *Let $G : \{0,1\}^{\sigma} \to \{0,1\}^{p(\sigma)}$ be a PRG, whose output is $\epsilon_G(\sigma)$-pseudorandom against non-uniform distinguishers, where $p : \mathbb{N} \to \mathbb{N}$ is a stretch function. Let L_G be as defined in Construction 4.16. If $L_G \in \mathsf{NA}-\mathsf{WIPCP}[r(n), q(n), q^*(n), \epsilon_S(n), \epsilon_{\text{ZK}}(n), \ell(n)]$ with the system $\left(P^{\text{in}}, V^{\text{in}}\right)$, then $\mathcal{R} \in \mathsf{NA}-\mathsf{CZKPCP}[r(n+p(\sigma)), q(n+p(\sigma)), q^*(n+p(\sigma)), \epsilon_S(n+p(\sigma)) + 2^{\sigma-p(\sigma)}, \epsilon_{\text{ZK}}(n+p(\sigma)) + \epsilon_G(\sigma), \ell(n+p(\sigma))]$ with the system of Construction 4.16.*

Proof. We analyze the properties of Construction 4.16.
PARAMETERS. Follows from the fact that the underlying WIPCP is run on instances of length $n+p(\sigma)$.
PERFECT COMPLETENESS. Follows from the perfect completeness of the underlying WIPCP system, and the definition of L_G.
SOUNDNESS. Let $x \notin L$, π^* be the "proof" provided for V, and $s \in_R \{0,1\}^{p(\sigma)}$. If $s \notin \text{Im}(G)$ then $(x,s) \notin L_G$ (because $x \notin L$), so the soundness of $\left(P^{\text{in}}, V^{\text{in}}\right)$ guarantees that V^{in} (and consequently, V) accepts with probability at most $\epsilon_S(n+p(\sigma))$. Since $s \in_R \{0,1\}^{p(\sigma)}$ and $|\text{Im}(G)| \leq 2^{\sigma}$ then $s \notin \text{IM}(G)$ except with probability at most $2^{\sigma-p(\sigma)}$. Therefore, V accepts with probability at most $\epsilon_S(n+p(\sigma)) + 2^{\sigma-p(\sigma)}$.
CZK. Let V^* be a q^*-query bounded PPT verifier, and we describe the simulator Sim. On input x, Sim picks $z \in_R \{0,1\}^{\sigma}$ and computes $s = G(z)$. Then, it runs P^{in} to generate a proof π for (x,s), providing z as the witness to P^{in}. Sim then emulates V^* with input x, CRS s, and proof oracle π, and outputs the view of V^* in this interaction (which includes the CRS s). We prove that $(x, s, \mathsf{Real}_{V^*,P}(x,w,s)) \approx^{\epsilon_{\text{ZK}}(n+p(\sigma))+\epsilon_G(\sigma)} \mathsf{Sim}(x,1^{\sigma})$, where $\approx^{\epsilon}$ denotes computational distance of ϵ. We define a hybrid distribution $\mathcal{H}$, describing a mental experiment in which the simulator is given the witness w, and uses it to generate the proof π, but generates the CRS as the PRG image of a random value. Concretely, the hybrid $\mathcal{H}$ is defined as follows. Sim generates $z \in_R \{0,1\}^{\sigma}$ and $s = G(z)$, runs P^{in} with input $((x,s),w)$ to obtain a proof π, and emulates V^* on input x, CRS s and proof π. Then $(x, s, \mathsf{Real}_{V^*,P}(x,w,s)) \approx^{\epsilon_G(\sigma)} \mathcal{H}$. Indeed, if a PPT distinguisher $\mathcal{D}$ could distinguish between the two with advantage more than $\epsilon_G(\sigma)$, then we would obtain a non-uniform PPT distinguisher $\mathcal{D}_G$ between a random string and the PRG output, achieving distinguishing advantage at least $\epsilon_G(\sigma)$: $\mathcal{D}_G$ on input $s \in \{0,1\}^{p(\sigma)}$ would run P^{in} on input $((x,s),w)$ to generate a proof π, then run V^* on x, s, π, and feed $\mathcal{D}$ with x, w, and the view of V^*. (We note that $\mathcal{D}_G$ is non-uniform since x, w are hard-wired into it.) Second, the $(\epsilon_{\text{ZK}}, q^*)$-WI of the underlying WIPCP guarantees that $\mathsf{Sim}(x,1^{\sigma}) \approx^{\epsilon_{\text{ZK}}(n+p(\sigma))} \mathcal{H}$, since in both cases the verifier is run on an honestly-generated proof for the same instance, only using different witnesses. (We note that the instance is (x,s) and therefore it is not fixed, but if the hybrids are not $\epsilon_{\text{ZK}}(n+p(\sigma))$-computationally close then using an averaging argument we can fix s.) □

The proof of Corollary 1.2 now follows from Proposition 4.17 by instantiating Construction 4.16 with the NA-WIPCP system of Theorem 1.1. Recall that by OWF we mean a function which is

one-way against *non-uniform* adversaries. The existence of such functions implies (by standard reductions) the existence of PRGs that are pseudorandom against non-uniform distinguishers.

Proof of Corollary 1.2. Assuming OWFs exist, there exists a PRG $G : \{0,1\}^{\sigma} \to \{0,1\}^{2\sigma}$ whose output is $\mathsf{negl}(\sigma)$-pseudorandom against non-uniform distinguishers (G can be constructed using standard reductions). We take $\sigma = n$, and $L = 3\mathrm{SAT}$, then by Theorem 1.1 $L_G \in \mathsf{NA}-\mathsf{WIPCP}[\mathsf{poly}\log n, \mathsf{poly}\log n, \mathsf{poly}(n), \mathsf{negl}(n), \mathsf{negl}(n), \mathsf{poly}(n)]$ (because $L_G \in \mathsf{NP}$). Therefore, by Proposition 4.17, $\mathrm{SAT} \in \mathsf{NA}-\mathsf{CZKPCP}[\mathsf{poly}\log n, \mathsf{poly}\log n, \mathsf{poly}(n), \mathsf{negl}(n), \mathsf{negl}(n), \mathsf{poly}(n)]$. □

Alternative CZKPCP constructions in the CRS model. We note that a simple alternative construction of CZKPCP for NP can be obtained by applying a standard PCP on top of a standard NIZK proof [6, 13]. Concretely, the CZKPCP prover generates a PCP for the NP-claim "there exists a NIZK for the claim $x \in L_{\mathcal{R}}$, relative to the CRS s, that would cause the NIZK-verifier to accept", where the witness is the NIZK proof string. Since the NIZK itself is CZK, the resultant PCP is also CZK. However, NIZK proofs for NP are not known to follow from the existence of one-way functions, and can currently be based only on much stronger assumptions such as the existence of trapdoor permutations [12].

5 Distributed Zero-Knowledge and Witness-Indistinguishable Proofs

We use our WIPCPs and CZKPCPs to construct *3-round* distributed WI and CZK proofs (respectively) for NP in a distributed setting, in which the PPT prover P and verifier V are aided by m polynomial-time servers $S_1, ..., S_m$. We call such systems *m-distributed proof systems*. Our constructions *crucially* rely on the *non-adaptivity* of the honest WIPCP (respectively, CZKPCP) verifier, and on the fact that WI (respectively, CZK) holds against *malicious* verifiers.

Our motivation for studying proofs in a distributed setting is to minimize the round complexity, and underlying assumptions, of sublinear ZK proofs. Concretely, it is known that assuming the existence of collision resistant hash functions, there exist 2-party 4-round sublinear ZK arguments for NP [23, 19]. (Arguments guarantee soundness only against *bounded* malicious provers.) We show that in the distributed setting, there exist *3-round* sublinear CZK (respectively, WI) *proofs* for NP, *assuming the existence of OWFs* (respectively, *unconditional*). Thus, the distributed setting allows us to improve previous results in terms of round complexity, underlying assumptions, and soundness type.

The setting. At a high level, an m-distributed proof system allows a prover to convince a verifier of the validity of an NP statement, where the parties are aided by several servers. The proof system should have the standard completeness property (guaranteeing that when all parties are honest, the verifier accepts true claims); and a strong soundness property, guaranteeing that a corrupted prover *cooperating with a small subset of servers* cannot convince the verifier of false claims (except with small probability). We are interested in systems with an additional ZK or WI property, which should hold against a corrupted receiver that *cooperates with a small subset of the servers*.

Definition 5.1 (m-distributed proof system)**.** An *m-distributed proof system* $(P, S_1, ..., S_m, V)$ for an NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ is a protocol executed between a PPT prover P that has input (x, w), m polynomial-time servers $S_1, ..., S_m$ that have no input, and a PPT verifier V that has input x and outputs either accept or reject. The protocol is executed in rounds, where in each round every party may send a message to every other party (over a secure, authenticated, point-to-point channel), and

may also send a message over a broadcast channel. We denote an execution of the protocol, in which P has input (x, w) and V has input x, by $(P(x, w), S_1, ..., S_m, V(x))$. We say that an m-distributed proof system is *sublinear* if the total communication involving V is $\mathsf{poly}\log(n)$.

For $t \in \mathbb{N}$, a *t-adversary* $\mathcal{A}$ non-adaptively corrupts $0 \leq t' \leq t$ servers, and possibly also P or V. The execution of a protocol in the presence of a t adversary $\mathcal{A}$ is carried out as follows. The honest parties follow the protocol, but the corrupted parties may arbitrarily deviate from the protocol (by sending arbitrary messages). The corrupted parties are also rushing (namely, in each round they first receive the messages sent to them, before sending their own messages). We consider both *unbounded* adversaries (in which case the corrupted parties are computationally unbounded), and bounded adversaries (in which case even corrupted parties run in polynomial time). We denote an execution of the protocol in the presence of a t-adversary $\mathcal{A}$, by $(P(y), S_1, ..., S_m, V(x))_{\mathcal{A}}$, where y is either $(x, w) \in \mathcal{R}$ (if P is honest) or $x \notin L_{\mathcal{R}}$ (if P is corrupted).

In Sections 5.1 and 5.2, we use WIPCPs (respectively, CZKPCPs) to construct a *3-round* distributed-WI proof system (respectively, CZK proof system in the CRS model) which, at a high level, operates as follows. In the first round the prover distributes a WIPCP (respectively, a CZKPCP) between the servers, and in the second and third rounds the verifier and servers emulate the WIPCP (respectively, CZKPCP) verification procedure (the verifier sends the proof queries of the WIPCP or CZKPCP verifier, and the servers provide the corresponding proof bits). This overview is an over-simplification of the construction: the verification procedure of the WIPCP (respectively, CZKPCP) cannot be used as-is since it only guarantees soundness when the verification is performed with a proof *oracle*, whereas corrupted servers *can determine their answers after seeing the queries of the verifier*. We overcome this by using techniques of [21].

Our distributed proof systems use the following WIPCP system.

Remark 5.2. For $n \in \mathbb{N}$, and a polynomial $t = t(n)$, let $(P_{\text{PCP}}, V_{\text{PCP}})$ be the proof system of Corollary 4.12, applied to the PCP for 3SAT of [2, Theorem 1] (more specifically, applied to the proof system used to prove that $\text{3SAT} \in \mathsf{PCP}\left[\log n, \log^2 n, \frac{1}{2}, \mathsf{poly}(n)\right]$), with length parameter n and zero-knowledge parameter q^* (see the proof of Theorem 1.1 for details). We amplify the soundness error of the PCP from $\frac{1}{2}$ to $\frac{1}{4}$ by repeating the verification procedure twice. Then by Corollary 4.12, there exist constants c_q, c_ℓ such that the honest verifier V_{PCP} makes $\log^{c_q}(nq^*)$ queries, and the proof has length $(nq^*)^{c_\ell}$. Moreover, this system is WI against *adaptive* verifiers.

5.1 Distributed Witness-Indistinguishable Proof Systems

In this section we construct a *witness-indistinguishable* distributed proof system for NP. We first formally define such systems.

Definition 5.3 ((t, m)-distributed WI proof system)**.** Let $t = t(n), m = m(n)$. An m-distributed proof system as in Definition 5.1 is a (t, m)-*distributed WI proof system* for $\mathcal{R}$ if it has the following properties.

- *Completeness.* For every $(x, w) \in \mathcal{R}$, $V(x)$ outputs accept (with probability 1) in the execution $(P(x, w), S_1, ..., S_m, V(x))$.
- *Soundness.* For every $x \notin L_{\mathcal{R}}$, and every (possibly malicious, possibly unbounded) t-adversary $\mathcal{A}$ corrupting the prover P, and a subset of $0 \leq t' \leq t$ servers, $V(x)$, in an execution $(P(x), S_1, ..., S_m, V(x))_{\mathcal{A}}$ outputs reject except with $\mathsf{negl}(n)$ probability.
- *Witness indistinguishability (WI).* Let $\mathcal{A}$ be a (possibly unbounded) t-adversary corrupting V and a subset $\mathcal{I} \subseteq [m]$ of $t' \leq t$ servers. Then for every $x \in L_{\mathcal{R}}$, and every pair w_1, w_2 of

witnesses for x, $\mathsf{SD}\left(\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}\left(x,w_1\right),\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}\left(x,w_2\right)\right)=\mathsf{negl}\left(n\right)$, where $\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}\left(x,w\right)$ denotes the view of $\mathcal{A}$ (consisting of the views of the verifier and the servers $\{S_i : i\in\mathcal{I}\}$), in the execution $\left(P\left(x,w\right),S_1,...,S_m,V\left(x\right)\right)_{\mathcal{A}}$.

Using WIPCPs with a *non-adaptive* honest verifier, we obtain the following result.

Theorem 5.4 (Sublinear distributed WI proofs)**.** *For every* NP*-relation* $\mathcal{R}$*, and polynomial* $t\left(n\right)$*, there exists a polynomial* $m\left(n\right)>t\left(n\right)$ *such that* $\mathcal{R}$ *has a 3-round sublinear* (t,m)*-distributed WI proof system, where* n *is the input length.*

We first give a high-level idea of the construction used to prove Theorem 5.4. On input (x,w), the prover generates a WIPCP for (x,w), and distributes the proof bits between the servers. Then, the verifier emulates the PCP verifier $V_{\mathrm{PCP}}\left(x\right)$, broadcasting the proof bits queried by V_{PCP}.[13] The servers holding these bits send them to the verifier, who verifies that V_{PCP} accepts. Notice that the verification procedure of the WIPCP cannot be used as-is since in the context of a distributed proof system, the adversary can determine the answers of corrupted servers *after seeing the queries of the verifier*, while the soundness of the WIPCP is only guaranteed when the verification is performed *with oracles* (in particular, the oracle answers are independent of the queries). Therefore, we need to restrict the influence that the adversary has on the verification procedure. To that effect, we have the prover distribute several copies of the proof, where the value of a proof bit queried by V_{PCP} is determined using the majority vote over the corresponding bits in several randomly selected copies. Thus, symbols held by corrupted servers are queried with low probability. This procedure can be thought of as applying a sort of "error correction" to the proof bits, where the "errors" we need to correct occur due to the answers of corrupted servers. We note that the witness-indistinguishability requirement prevents us from using any arbitrary error correction method, since every "encoded" proof bit should have low locality (namely, depend on few bits of the original proof).

Protocol 5.5 (Distributed-WI for NP-relation $\mathcal{R}$)**.** The distributed WI system consists of the prover P, m servers $S_1,...,S_m$, and the verifier V, and uses a WIPCP system $(P_{\mathrm{PCP}},V_{\mathrm{PCP}})$ parameterized by the ZK parameter q^*, and where the honest verifier V_{PCP} is non-adaptive and makes $q\left(n,q^*\right)$ queries to its proof oracle (where n denotes the input length). The system is parameterized by l, which determines the number of proof copies that P generates; and z, which determines how many copies V will use to answer a single query of V_{PCP}. The protocol is executed as follows.

- *Round 1.* P, on input x,w, generates a proof $\pi\leftarrow P_{\mathrm{PCP}}\left(1^{q^*},x,w\right)$ for $q^*=O\left(t+|x|\right)$ (see the proof of Lemma 5.7 for the exact constants). P duplicates the proof l times: $\pi^1,...,\pi^l$, and distributes $\pi^1,...,\pi^l$ between the servers (each server is given a single bit, so every bit of π is now held by l servers).[14]

- *Round 2.* V on input x emulates $V_{\mathrm{PCP}}\left(q^*,x\right)$, and obtains the queries $i_1,...,i_q,q\leq q\left(|x|,q^*\right)$ that V_{PCP} makes to its oracle. Then, for every $1\leq j\leq q$, V picks z random indices $s_1^j,...,s_z^j\in[l]$. V broadcasts $i_1,...,i_q$ and the sets $\left\{s_1^1,...,s_z^1\right\}$,...,$\left\{s_1^q,...,s_z^q\right\}$. If $q\geq q\left(|x|,q^*\right)$, or one of the sets $\left\{s_1^1,...,s_z^1\right\}$,...,$\left\{s_1^q,...,s_z^q\right\}$ has size larger than z, all servers abort the execution.

- *Round 3.* For every $1\leq j\leq q$ and $1\leq s\leq z$, the server holding the bit $\pi_{i_j}^s$ sends it to V. V then reconstructs the bits $\pi_{i_1},...,\pi_{i_q}$ as follows. For every $1\leq j\leq q$, if there exists a bit

[13] Broadcast is used to ensure that the verifier does not contact more servers than allowed by the WI guarantee. Other alternatives are to extract the verification queries from an unpredictable source of public randomness; or have each server which is contacted by the verifier poll a small number of other servers in order to get an estimate of the number of servers that have been contacted by the verifier.

[14] As each server receives a single bit, $m=\ell\cdot l$, where ℓ denotes the length of the WIPCP π.

b such that at least $\frac{tz}{8}$ of the bits $\pi_{i_j}^{s_1^j},...,\pi_{i_j}^{s_z^j}$ are equal to b, then V sets $\pi_{i_j}=b$. Otherwise, V rejects. At the end of this process, if V has not rejected then the bits $\pi_{i_1},...,\pi_{i_q}$ have been determined, and they are given to V_{PCP} as the oracle answers. V then either accepts or rejects x, according to the output of V_{PCP}.

Next, we analyze the properties of Protocol 5.5. For an odd l, and every set $s^1,...,s^l$ of strings (of equal length m), we define the length-m string s^* that is *consistent* with $s^1,..,s^l$, to be the string such that every bit s_i^* is the majority vote over $s_i^1,...,s_i^l$. The following lemma states that (for an appropriate choice of parameters) an execution of Protocol 5.5 with (possibly corrupted) proofs $\pi^1,...,\pi^l$ is essentially the same as an execution of the verification procedure of the underlying WIPCP system, with (possibly corrupted) proof π^*.

Lemma 5.6. *Let $l,t,z\in\mathbb{N}$ such that $l\geq 2^6\cdot t$ and $z\geq 16$, and let $s=s^1\circ...\circ s^l$ over $\{0,1\}$ be a string such that $|s^1|=...=|s^l|=m$. Define the string s^* as follows. $|s^*|=m$, and every bit s_i^* is equal to the majority vote over $s_i^1,...,s_i^l$. Let $\mathcal{A}$ be an adversary who chooses in advance a subset $\mathcal{B}$ of at most t bits of s which he can adaptively control. Then for every bit $1\leq j\leq m$, $\mathcal{A}$ wins the following game with probability at most $2^{1-\frac{z}{8}}$.*

- *z indices $i_1,...,i_z$ are picked at random, and $s_j^{i_1},...,s_j^{i_z}$ are queried. For every bit $b\in\{s_j^{i_1},...,s_j^{i_z}\}$ such that $b\in\mathcal{B}$, $\mathcal{A}$ determines the answer to the query (for every bit $b\notin\mathcal{B}$, the answer to the query is the corresponding bit in s).*
- *If there exists a bit b such that at least $\frac{7z}{8}$ of the answers are equal to b, then the output of the game is b. Otherwise, the output is $\perp$.*
- *The adversary wins if the output of the game is $b\in\{0,1\}$ such that $b\neq s_j^*$.*

The lemma guarantees that if P is honest, then (except with probability at most $2^{1-\frac{z}{8}}$) every single bit that V reconstructs in Protocol 5.5 is consistent with the corresponding bit of π. This holds even if $t'\leq t$ servers are corrupted. Moreover, even if a corrupted P colludes with a subset of $t'\leq t$ corrupted servers, then except with probability at most $2^{1-\frac{z}{8}}$ the bit that V reconstructs is consistent with π^*, where π^* is the (possibly corrupted) "majority vote" proof defined by the proof copies that P distributed between the servers.

Proof. Let s_j^* be the bit that should be reconstructed in the game (determined according to the majority vote over s). The bit $b=\bar{s_j^*}$ is reconstructed when z random and independent indices $i_1,...,i_z$ are picked, only if at least $\frac{7z}{8}$ of the bits in $\left\{s_j^{i_1},...,s_j^{i_z}\right\}$ were equal to b. Let $\mathbb{E}_{\text{maj}}$ denote the event that at most $\frac{3z}{4}$ of the bits in $\left\{s_j^{i_1},...,s_j^{i_s}\right\}$ were equal to b. Conditioned on $\mathbb{E}_{\text{maj}}$, $\mathcal{A}$ wins only if he was able to flip (by flipping bits in $\mathcal{B}$) at least $\frac{7z}{8}-\frac{3z}{4}=\frac{z}{8}$ of the bits in $\left\{s_j^{i_1},...,s_j^{i_z}\right\}$. Let $\mathcal{I}:=\left\{i\in[l]:s_j^i\in\mathcal{B}\right\}$, then $|\mathcal{I}|\leq|\mathcal{B}|\leq t$. Consequently, conditioned on $\mathbb{E}_{\text{maj}}$, $\mathcal{A}$ wins with probability at most $2^{-\frac{z}{8}}$, since

$$\Pr_{i_1,...,i_z\in_R[l]}\left[\{i_1,...,i_z\}\cap\mathcal{I}\geq\frac{z}{8}\right]\leq\binom{z}{\frac{z}{8}}\cdot\left(\frac{t}{l}\right)^{\frac{z}{8}}\leq\left(\frac{z\cdot e}{\frac{z}{8}}\right)^{\frac{z}{8}}\left(\frac{t}{l}\right)^{\frac{z}{8}}=\left(\frac{8et}{l}\right)^{\frac{z}{8}}\leq 2^{-\frac{z}{8}}.$$

By the definition of s^*, less than half the bits $\left\{s_j^1,...,s_j^l\right\}$ are equal to b. As $i_1,...,i_z$ are random and independent, then (using Hoeffding's bound) $\Pr\left[\bar{\mathbb{E}}_{\text{maj}}\right]\leq 2^{-\frac{z}{8}}$. Consequently, $\mathcal{A}$ wins the game with probability at most $2\cdot 2^{-\frac{z}{8}}=2^{1-\frac{z}{8}}$. □

Lemma 5.7. *Let $\mathcal{R} = \mathcal{R}(x,w)$ be an* NP*-relation, and $t(n)$ be a polynomial. Then there exists a polynomial $m(n) > t(n)$ such that Protocol 5.5 is a (t,m)-distributed WI proof system for $\mathcal{R}$ with soundness error $\frac{1}{4} + \mathsf{negl}(n)$, where n is the input length.*

Proof. Let $t = t(n)$ by a polynomial, and let $(P_{\text{PCP}}, V_{\text{PCP}})$ be the proof system described in Remark 5.2. We set the parameters of Protocol 5.5 as follows: $z = \log^{c_q} n$, $l = 2^6(t+z)+1$ (we choose l to be odd so that the majority vote over the l copies would be uniquely defined), and $q^* = 2c_0(t+n)$, where $c_0 \in \mathbb{N}$ is a constant such that for every natural $c \geq c_0$, $\log^{c_q+2}(c^2) \leq \frac{c}{2}$.[15] (Notice that for this choice of q^*, $q^* \geq c_0$ so $\log^{c_q}(nq^*) + t \leq \log^{c_q}\left((q^*)^2\right) + \frac{q^*}{2} \leq q^*$. Therefore, the combined number of proof bits seen by the verifier V, and at most t servers, is at most q^*.) Notice that in this case the combined lengths of all copies is $l \cdot \mathsf{poly}(nq^*) = (\mathsf{poly}\log n + O(t)) \cdot \mathsf{poly}(nt) = \mathsf{poly}(n)$, and we take the number of servers, m, to be the corresponding polynomial. We analyze the properties of the protocol.

Complexity. The communication during the first round is $m = \mathsf{poly}(n)$, and during the second and third rounds the communication is $\mathsf{poly}\log(n)$.

Completeness. Follows directly from the protocol definition, and the completeness of the underlying WIPCP.

Soundness. Let $\pi^1, ..., \pi^l$ be the proofs that P distributed between the servers, and let π^* be the (possibly ill-formed) proof that is consistent with $\pi^1, ..., \pi^l$. Lemma 5.6 guarantees that the reconstructed answer to any single proof query is consistent with π^*, except with at most $2^{1-\frac{z}{8}}$ probability. Using the union bound, the emulation of V_{PCP} is executed with π^*, except with probability at most $q(n,q^*) \cdot 2^{1-\frac{\log^{c_q}(n)}{8}} = \log^{c_q}(nq^*) \cdot 2^{1-\frac{\log^{c_q}(n)}{8}} = \mathsf{negl}(n)$. Conditioned on the event that all reconstructed bits are consistent with π^*, the soundness of the underlying WIPCP system guarantees that V_{PCP} (and consequently also V) accepts with probability at most $\frac{1}{4}$, so the total soundness error is $\frac{1}{4} + \mathsf{negl}(n)$.

Witness-indistinguishability. Let $\mathcal{A}$ be a t-adversary corrupting V and a subset $\mathcal{I}$ of $t' \leq t$ servers. Let $x \in L_{\mathcal{R}}$ with witnesses w_1, w_2. Then $\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}(x,w_i)$ consists of the proof bits held by the servers $\{S_i : i \in \mathcal{I}\}$, and the proof bits sent to V in the third round. In total, the view of $\mathcal{A}$ consists of at most $t' + \log^{c_q}(nq^*) \leq q^*$ bits (this is because if V sends more queries, then the honest servers would not answer in the third round). Therefore, the WI of the underlying WIPCP system against *adaptive, possibly malicious* verifiers guarantees that $\mathsf{SD}\left(\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}(x,w_1), \mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}(x,w_2)\right)$ is $\mathsf{negl}(nq^*) = \mathsf{negl}(n)$. (We note that WI against *adaptive* verifiers is required because $\mathcal{A}$ receives the proof bits held by the corrupted servers *before* determining the queries of V. WI against *malicious* verifiers is required because the adversary sees the proof bits held by corrupted servers, and these correspond to (possibly malicious) proof queries. Notice that due to this reason, WI against malicious verifiers is needed even if the protocol *enforces* that the queries made by V correspond to the queries of an execution of V_{PCP}, e.g., by having V broadcast the randomness used to emulated V_{PCP}.) □

The proof system of Protocol 5.5 satisfies all the properties required in Theorem 5.4, except for the soundness error, which can be amplified in a standard way, as we now show.

Proof of Theorem 5.4. We amplify the soundness error of Protocol 5.5, and prove that the amplified system satisfies the requirements of Theorem 5.4. The amplification is performed as follows: the second round is repeated $k = \log^2 n$ independent times in parallel, and so is the third round. V accepts if and only if V_{PCP} accepted in all the iterations. The amplification increases the communication

[15] For the proof of this lemma, it would suffice that the power of the log would be c_q, but we will need the power to be $c_q + 2$ when we amplify the system to have the properties guaranteed in Theorem 5.4.

complexity in the second and third rounds by a multiplicative factor of $\mathsf{poly}\log(n)$, so the protocol is still sublinear. Completeness holds just as in the proof of Lemma 5.7. As for soundness, $x \notin L_{\mathcal{R}}$ is accepted if and only if V_{PCP} accepted x in $\log^2 n$ random and independent executions (and for each execution this happens with probability $\frac{1}{4} + \mathsf{negl}(n)$), which happens only with $\mathsf{negl}(n)$ probability. Finally, notice that the view of a t adversary $\mathcal{A}$ now consists of at most $\log^2 n \cdot \log^{c_q}(nq^*) + t$ proof bits, which (by the choice of parameters in Lemma 5.7) is at most q^*, so the same argument used in the proof of Lemma 5.7 shows that the system is WI. □

Remark 5.8. The distributed WI proof system *crucially* rely on the *non-adaptivity* of the honest WIPCP verifier. Indeed, if the honest verifier were adaptive then the protocol would have at least 4 rounds, since rounds cannot be compressed. Moreover, since the verifier may collude with a subset of servers, we needed a PCP system with WI against *malicious* verifiers.

5.2 Distributed Computational Zero-Knowledge Proof Systems

The Techniques of Section 5.1 can be used to construct a distributed proof system for NP that guarantees *zero-knowledge* against *computationally bounded* t-adversaries, when all parties have access to a shared random string. We first formally define the model.

Definition 5.9 (m-distributed proof system with a CRS)**.** An *m-distributed proof system with a common random string (CRS)* is an m-distributed proof system as in Definition 5.9, where all parties have access to a shared random string s.

Definition 5.10 ((t,m)-distributed CZK proof system in the CRS model)**.** Let $\mathcal{R} = \mathcal{R}(x,w)$ be an NP-relation, and $t = t(n), m = m(n)$. An m-distributed proof system as in Definition 5.9 is a *(t,m)-distributed CZK proof system for $\mathcal{R}$ in the CRS model* if all parties have access to a shared random string $s \in \{0,1\}^{\sigma}$ for some $\sigma = \mathsf{poly}(|x|)$, and the following holds:

- *Completeness.* For every $(x,w) \in \mathcal{R}$, and every CRS $s \in \{0,1\}^{\sigma}$, $V(x)$ outputs accept (with probability 1) in the execution $(P(x,w,s), S_1(s), ..., S_m(s), V(x,s))$ with the CRS s.
- *Soundness.* For every $x \notin L_{\mathcal{R}}$, a uniformly random $s \in_R \{0,1\}^{\sigma}$, and every (possibly malicious, possibly unbounded) t-adversary $\mathcal{A}$ corrupting the prover P, and a subset of $t' \leq t$ servers, $V(x)$, in an execution $(P(x,s), S_1(s), ..., S_m(s), V(x,s))_{\mathcal{A}}$ with the CRS s outputs reject except with $\mathsf{negl}(n)$ probability.
- *Computational zero-knowledge (CZK).* Let $\mathcal{A}$ be a PPT t-adversary corrupting V and a subset $\mathcal{I} \subseteq [m]$ of $t' \leq t$ servers. Then there exists a PPT simulator Sim such that for every $(x,w) \in \mathcal{R}$, $\mathsf{Sim}(x) \approx \left(s, \mathsf{View}_{\mathcal{A},P,\{S_i : i \in [m] \setminus \mathcal{I}\}}(x,w,s)\right)$, where $\approx$ denotes computational indistinguishability; $s \in_R \{0,1\}^{\sigma}$; and $\mathsf{View}_{\mathcal{A},P,\{S_i : i \in [m] \setminus \mathcal{I}\}}(x,w,s)$ denotes the view of $\mathcal{A}$ (consisting of the views of the verifier and the servers $\{S_i : i \in \mathcal{I}\}$), in the execution $(P(x,w,s), S_1(s), ..., S_m(s), V(x,s))_{\mathcal{A}}$.

We construct distributed CZK proof systems with similar properties to our distributed WI proof systems:

Theorem 5.11 (Sublinear distributed CZK proofs in the CRS model)**.** *Assume that OWFs exist. Then for every NP-relation $\mathcal{R}$, and polynomial $t(n)$, there exists a polynomial $m(n) > t(n)$ such that $\mathcal{R}$ has a 3-round sublinear (t,m)-distributed CZK proof system in the CRS model, where n is the input length.*

The construction is similar to Protocol 5.5, except that it uses a CZKPCP system (instead of a WIPCP system). (Also, we include the soundness amplification in the protocol description, whereas Protocol 5.5 did not amplify soundness.)

Protocol 5.12 (Distributed-CZK for relation $\mathcal{R}$)**.** The distributed CZK system consists of the prover P, m servers $S_1, ..., S_m$, and the verifier V, and uses a CZKPCP system $(P_{\text{PCP}}, V_{\text{PCP}})$ parameterized by the ZK parameter q^*; and the length σ of the common reference string s; where the honest verifier V_{PCP} is non-adaptive and makes $q(n, q^*)$ queries to its proof oracle (n denotes the input length). The system is parameterized by l, which determines the number of proof copies that P generates; and z, which determines how many copies V will use to answer a single query of V_{PCP}. The protocol is executed as follows.

- *Round 1.* P, on input x, w, and given the CRS s, generates a proof $\pi \leftarrow P_{\text{PCP}}\left(1^{q^*}, x, w, s\right)$ for $q^* = O(t + |x|)$ (see the proof of Theorem 5.11 below for the exact constant). P duplicates the proof l times: $\pi^1, ..., \pi^l$, and distributes $\pi^1, ..., \pi^l$ between the servers (each server is given a single bit).
- *Rounds 2 and 3.* V on input x, and the CRS s, emulates $V_{\text{PCP}}(q^*, x, s)$ $k = \mathsf{poly}\log(n)$ independent times (see the proof of Theorem 5.11 below for the exact constant). In each emulation, the answers to the oracle queries of V are determined according to the majority vote as described in Protocol 5.5.

We now use Protocol 5.12 to prove Theorem 5.11 (the proof is similar to the proof of Theorem 5.4, which uses Lemma 5.6).

Proof of Theorem 5.11. Let $t = t(n)$ by a polynomial, and let $(P_{\text{WIPCP}}, V_{\text{WIPCP}})$ be the WIPCP system described in Remark 5.2. Let $(P_{\text{PCP}}, V_{\text{PCP}})$ be the CZKPCP of Proposition 4.17, obtained from $(P_{\text{WIPCP}}, V_{\text{WIPCP}})$ using a PRG $G : \{0,1\}^\sigma \to \{0,1\}^{2\sigma}$, whose outputs are $\epsilon_G(\sigma)$-pseudorandom against non-uniform distinguishers, where $\epsilon_G(\sigma) = \mathsf{negl}(\sigma)$ (assuming OWFs exist, such a PRG exists by a standard reduction). We take $\sigma = n$, then Proposition 4.17 guarantees that there exist constants d_q, d_ℓ such that the honest verifier V_{PCP} makes $\log^{d_q}(nq^*)$ queries, and the proof has length $(nq^*)^{d_\ell}$.

We set the parameters of Protocol 5.12 as follows: $k = \log^2 n$, $z = \log^{d_q} n$, $l = 2^6(t + z) + 1$ (we choose l to be odd so that the majority vote over the l copies would be uniquely defined), and $q^* = 2c_0(t + 2n)$, where $c_0 \in \mathbb{N}$ is a constant such that for every natural $c \geq c_0$, $\log^{d_q+2}\left(c^2\right) \leq \frac{c}{2}$. (Notice that for this choice of q^*, $q^* \geq c_0$ so $\log^2(n) \cdot \log^{d_q}(nq^*) + t \leq \log^{d_q+2}\left((q^*)^2\right) + \frac{q^*}{2} \leq q^*$. Therefore, the combined number of proof bits seen by the verifier V, and at most t servers, is at most q^*.) Notice that in this case the combined lengths of all copies is $l \cdot \mathsf{poly}(2nq^*) = O(\mathsf{poly}\log(n) + t) \cdot \mathsf{poly}(nt) = \mathsf{poly}(n)$, and we take the number of servers, m, to be the corresponding polynomial.

Complexity. In the first round, P sends a single bit to each of the $m = \mathsf{poly}(n)$ servers. In the second and third rounds, V_{PCP} is emulated $\log^2(n)$ times, each emulation requires V to broadcast, and receive, $\mathsf{poly}\log(n)$ bits (there are $\mathsf{poly}\log(n)$ queries, each reconstructed using $z = \mathsf{poly}\log n$ bits).

Completeness. Follows directly from the protocol definition, and the completeness of the underlying WIPCP.

Soundness. Let $\pi^1, ..., \pi^l$ be the proofs that P distributed between the servers, and let π^* be the (possibly ill-formed) proof that is consistent with $\pi^1, ..., \pi^l$. Lemma 5.6 guarantees that the reconstructed answer to any single proof query is consistent with π^*, except with at most $2^{1-\frac{z}{8}}$ probability. Using the union bound, the emulation of V_{PCP} is executed with π^*, except with probability at most

$q(n,q^*)\cdot 2^{1-\frac{\log^{d_q}(n)}{8}} = \log^{d_q}(nq^*)\cdot 2^{1-\frac{\log^{d_q}(n)}{8}} = \mathsf{negl}(n)$. Conditioned on the event that all reconstructed bits are consistent with π^*, the soundness of the underlying CZKPCP system guarantees that V_{PCP} accepts *in a single iteration* with probability at most $\frac{1}{4}+\mathsf{negl}(n)$. Since V accepts only if V_{PCP} accepted in all $\log^2(n)$ emulations, V accepts x only with $\mathsf{negl}(n)$ probability.

CZK. By Proposition 4.17, there exists a simulator $\mathsf{Sim}_{\mathrm{PCP}}$ for $(P_{\mathrm{PCP}},V_{\mathrm{PCP}})$, that can *adaptively* answer the queries of *any* (possibly malicious) q^*-bounded PPT verifier. Let $\mathcal{A}$ be a PPT t-adversary corrupting V and a subset $\mathcal{I}$ of $t'\leq t$ servers. We construct a simulator Sim that simulates the view of $\mathcal{A}$. Let $x\in L_{\mathcal{R}}$ with witness w, then Sim on input x runs $\mathsf{Sim}_{\mathrm{PCP}}$ to obtain the simulated CRS s_{Sim}, emulates $\mathcal{A}$ (with input x,s_{Sim}) to obtain the set $\mathcal{I}$ of $t'\leq t$ servers that $\mathcal{A}$ corrupts, and uses $\mathsf{Sim}_{\mathrm{PCP}}$ to simulate the proof bits held by these servers. Sim provides $\mathcal{A}$ with these bits, and receives from $\mathcal{A}$ the queries that V makes in the second phase. Sim then uses $\mathsf{Sim}_{\mathrm{PCP}}$ to generate these proof bits as well. (Notice that the set $\mathcal{I}$ of bits held by the corrupted servers, and the set $\mathcal{Q}$ of bits queried by the corrupted verifier, are bits in a proof $\pi^1\circ\ldots\circ\pi^l$ which consists of l copies of an honestly-generated proof π for x. For every bit in $\mathcal{I}\cup\mathcal{Q}$, Sim determines the corresponding bit of π, and uses $\mathsf{Sim}_{\mathrm{PCP}}$ to simulate that bit. Bits in $\mathcal{I}\cup\mathcal{Q}$ that correspond to the same bit of π are answered consistently, and incur only a single query to $\mathsf{Sim}_{\mathrm{PCP}}$.) When the emulation ends, Sim outputs s_{Sim}, concatenated with the simulated proof bits seen by $\mathcal{A}$.

Let V^* denote the PPT verifier (in the underlying CZKPCP system) that first queries the proof oracle about the bits corresponding to those held by the servers in $\mathcal{I}$, and then queries its oracle about the proof bits in $\mathcal{Q}$. Then $\left(s,\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}(x,w)\right) = \left(s,\widetilde{\mathsf{Real}}_{V^*,P_{\mathrm{PCP}}}(x,w,s)\right)$, where $\widetilde{\mathsf{Real}}_{V^*,P_{\mathrm{PCP}}}(x,w,s)$ is obtained from $\mathsf{Real}_{V^*,P_{\mathrm{PCP}}}(x,w,s)$ by duplicating the bits of π that appear several times in $\mathcal{I}\cup\mathcal{Q}$. Notice that $|\mathcal{I}\cup\mathcal{Q}|\leq t+\log^2 n\cdot q(n,q)\leq t+\log^{d_q+2}(nq^*)$ which by our choice of parameters is at most q^*. Therefore, the computational distance between
$Sim_{\mathrm{PCP}}(x)$ and $(x,s,\mathsf{Real}_{V^*,P_{\mathrm{PCP}}}(x,w,s))$ is at most $\mathsf{negl}(n)+\epsilon_G(n)=\mathsf{negl}(n)$. Since the output of Sim is obtained from the output of $\mathsf{Sim}_{\mathrm{PCP}}$ by (possibly) duplicating the bits that appear several times in $\mathcal{I}\cup\mathcal{Q}$, we conclude that $\mathsf{Sim}(x)\approx\left(x,s,\mathsf{View}_{\mathcal{A},P,\{S_i:i\in[m]\setminus\mathcal{I}\}}(x,w)\right)$. □

6 LRCC-Based ZKPCPs Imply NP ⊆ BPP

In this section we give evidence that the techniques we use to construct WIPCPs cannot be used to construct *ZK*PCPs, unless $\mathsf{NP}\subseteq\mathsf{BPP}$. At a high level, this is because the leakage-resilience guarantee of (non-relaxed) LRCCs withstanding global leakage is (in some sense) universal. Concretely, the LRCC simulator generates, in polynomial time, wire values for the entire computation of the circuit; and these wire values *simultaneously* fool every leakage function in the family of leakage functions. When the LRCC is used to construct PCPs with ZK guarantees, this gives a ZKPCP simulator that generates a "fake witness" that can be used to construct a "fake PCP". This "fake PCP" is *simultaneously "good"* for every set of verifier queries, in the sense that with high probability, the verifier is convinced. In effect, the simulator "commits" to a fake proof in advance, and this can be used to probabilistically decide the language, thus proving that the language is in BPP. We emphasize that the simulator constructed here is stronger than the simulator required by ZK, since in the definition of ZK, the simulator is first given the queries, and is then required to generate a *partial* fake proof, which should convince the verifier *conditioned on the event that it queries the particular pre-determined set of queries*. (This is the case when the verifier is non-adaptive. For adaptive verifiers, for every set of queries of the verifier, the simulator adaptively constructs the simulated proof symbols queried by this set.) We first formally define this stronger simulation notion, which we call *oblivious zero-knowledge*.

Definition 6.1 (Oblivious zero-knowledge)**.** We say that a probabilistic proof system (P, V) is an *oblivious ZKPCP* system for a relation $\mathcal{R} = \mathcal{R}(x, w)$, if it has the syntax, completeness and soundness properties of a WIPCP system (as defined in Definition 4.1), and the following (ϵ, q^*)-*oblivious zero-knowledge* property: for every q^*-query-bounded verifier V^* there exists a PPT simulator Sim such that the following holds for every $(x, w) \in \mathcal{R}$. Sim is given input $x, 1^\sigma$, and outputs a (possibly ill-formed) proof π^*, such that $\mathsf{SD}\left(\mathsf{Real}_{V^*,P}(x, w), \mathsf{Simulated}_{V^*,\pi^*}(x)\right) \leq \epsilon$, where $\mathsf{Real}_{V^*,P}(x, w)$ ($\mathsf{Simulated}_{V^*,\pi^*}(x)$) denotes the view of V^* on input $\epsilon_S, q^*, x, 1^{|w|}$, and given oracle access to $\pi \in_R P\left(\epsilon_S, 1^{q^*}, x, w\right)$ (to π^*).

Definition 6.2. We say that a circuit compiler is a (non-relaxed) LRCC if Definition 2.8 holds with a *PPT* simulator Sim.

The next remark states that our transformation from relaxed-LRCC to WIPCP used a (non-relaxed) LRCC, then the resultant probabilistic proof system would be oblivious ZK.

Remark 6.3. We note that if the relaxed-LRCC of Proposition 4.4 (and Corollary 4.10) was replaced with a (*non-relaxed*) LRCC, then the resultant PCP system would be oblivious-ZK. Indeed, the PPT simulator of the LRCC would give a PPT simulator for the PCP system. Moreover, the LRCC simulator generates simulated wire values for the entire circuit, and these wire values constitute a complete "fake" witness, so the PCP simulator could generate an entire "fake" proof π^* (the simulator would first run the LRCC-simulator to obtain the fake witness, then run the honest PCP prover to obtain a fake proof from this fake witness). Moreover, every possible set of verifier queries to the proof correspond to applying a leakage function (from the family of leakage functions against which the LRCC is leakage-resilient) on the wire values of the compiled circuit, so π^* is simultaneously convincing (with high probability) for every possible set of verifier queries.

Definition 6.4 (BPP)**.** The complexity class BPP (bounded-error probabilistic polynomial time) consists of all languages L for which there exists a PPT algorithm $\mathcal{A}_L$ with the following properties (the probability is over the randomness of $\mathcal{A}_L$):

- $x \in L \Rightarrow \Pr\left[\mathcal{A}_L(x) = \mathsf{accept}\right] \geq \frac{2}{3}$.
- $x \notin L \Rightarrow \Pr\left[\mathcal{A}_L(x) = \mathsf{accept}\right] \leq \frac{1}{3}$.

The next theorem states that oblivious ZKPCP systems exist only for languages in BPP. As we show below (Corollary 6.7), the PCP systems obtained from Construction 4.8, when it uses an LRCC with a *PPT* simulator, are oblivious-ZKPCPs. Therefore, it seems implausible that we could use this transformation to obtain the stronger notion of a *ZK*PCP for NP.

Theorem 6.5. *Let $\mathcal{R} = \mathcal{R}(x, w)$ be a polynomially-bounded relation. If $L_\mathcal{R}$ has an $[r, q, q^*, \epsilon_S, \epsilon_{\mathrm{ZK}}, \ell]$-ZKPCP system with* oblivious ZK*, constants $\epsilon_S, \epsilon_{\mathrm{ZK}} \leq \frac{1}{6}$, and $r, q, \ell = \mathsf{poly}(n)$ such that $q \leq q^*$, then $L \in \mathsf{BPP}$.*

Remark 6.6. We have defined probabilistic proofs systems as having perfect completeness. For Theorem 6.5 to hold, the probabilistic proof system should only have a noticeable gap between the probability of accepting $x \in L$ and $x \notin L$. Specifically, is suffices that every $x \in L$ is accepted with probability at least $\frac{5}{6}$ (when the verifier is given an honestly-generated proof), and that every $x \notin L$ is accepted with probability at most $\frac{1}{6}$ (for *any* "proof" given to the verifier). Moreover, we only need *honest-verifier* ZK, i.e., that an efficient simulator exists for the honest verifier, and it suffices to have a *computational* distance of at most $\frac{1}{6}$ between $\mathsf{Real}_{V^*,P}(x, w), \mathsf{Simulated}_{V^*,\pi^*}(x)$.

Proof. Let (P, V) be the oblivious ZKPCP system for $L = L_{\mathcal{R}}$, and let Sim denote the corresponding simulator. We assume that every $x \in L$ is accepted with probability at least $\frac{5}{6}$ (when the verifier is given an honestly-generated proof), that every $x \notin L$ is accepted with probability at most $\frac{1}{6}$ (for *any* "proof" given to the verifier), and that the *computational* distance between $\mathsf{Real}_{V,P}(x, w), \mathsf{Simulated}_{V,\pi^*}(x)$ is at most $\frac{1}{6}$. (Notice that we only assume that the system has honest-verifier ZK, which follows from the ZK guarantee of the system because $q \leq q^*$.) Then the following probabilistic algorithm decides L:

- On input x, run Sim on x to obtain a simulated proof π^*.
- Run V on input x, with oracle access to π^*.
- If V accepts x in this run, output $x \in L$, otherwise output $x \notin L$.

We first show that if $x \in L$ then the algorithm outputs $x \in L$ with probability at least $\frac{2}{3}$. Otherwise, we consider the case that the output of V in the simulated view $\mathsf{Simulated}_{V^*,\pi^*}(x)$ is accept with probability less than $\frac{2}{3}$. Note that, his output in $\mathsf{Real}_{V^*,P}(x, w)$ (i.e., his output on x given oracle access to an honestly-generated proof) is accept with probability at least $\frac{5}{6}$ by the completeness property. Therefore, the computational (and consequently, also statistical) distance between $\mathsf{Simulated}_{V^*,\pi^*}(x), \mathsf{Real}_{V^*,P}(x, w)$ is more than $\frac{1}{6}$ (the corresponding distinguisher outputs 1 if V accepts), thus contradicting the $\left(\frac{1}{6}, q^*\right)$-oblivious ZK.

Next, we show that if $x \notin L$ then the algorithm outputs "$x \in L$" with probability at most $\frac{1}{3}$. Otherwise, using an averaging argument there exists a "proof" π^* for x, on which V accepts x with probability more than $\frac{1}{3}$. But this contradicts the soundness of the system. □

The combination of Theorem 6.5 with Remark 6.3 shows that if there exists a SAT-respecting LRCC withstanding leakage from a leakage class $\mathcal{L}$, and in addition, there exists a PCP system (P, V) for 3SAT such that the queries of V to the PCP π constitute a leakage function $\ell \in \mathcal{L}$ of the NP-witness, then $\mathsf{NP} \subseteq \mathsf{BPP}$. This is formalized in the next corollary.

Corollary 6.7. *Let n be a length parameter, and $\mathcal{L}_{O(1),\mathsf{poly}(n),\oplus 1}$ be the class of functions computable by constant depth, poly-sized boolean circuits with unbounded fan-in $\wedge, \vee, \neg$ gates, and a single $\oplus$ gate of unbounded fan-in and fan-out. If for every constant c there exists a SAT-respecting $\left(\mathcal{L}_{O(1),\mathsf{poly}(n),\oplus 1}, \mathsf{negl}(n), cn\right)$-LRCC, then $\mathsf{NP} \subseteq \mathsf{BPP}$.*

Proof. Notice that a 3CNF of size n can be verified by a boolean circuit C^{SAT} of size cn, for some constant c. We take the constant in the theorem statement to be this c. Let $(\mathsf{Comp}, \mathsf{E})$ denote the SAT-respecting $\left(\mathcal{L}_{O(1),\mathsf{poly}(n),\oplus 1}, \epsilon, cn\right)$-LRCC whose existence is guaranteed by the theorem statement, where $\epsilon = \mathsf{negl}(n)$. Then for every circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most cn, and every $z_1, z_2 \in \{0,1\}^n$ that satisfy C, the wire values $[\mathsf{Comp}(C), \hat{z}_1], [\mathsf{Comp}(C), \hat{z}_2]$ are $\left(\mathcal{L}_{O(1),\mathsf{poly}(n),\oplus 1}, \epsilon\right)$-leakage-indistinguishable, where $\hat{z}_i$ is a random encoding of z_i according to E. Using Lemma 3.55, $[\mathsf{Comp}(C), \hat{z}_1], [\mathsf{Comp}(C), \hat{z}_2]$ are also $\left(\mathcal{L}^{\mathsf{poly}\log n}_{O(1),\mathsf{poly}(n),\oplus 1}, \epsilon \cdot 2^{\mathsf{poly}\log n}\right)$-leakage-indistinguishable, so $(\mathsf{Comp}, \mathsf{E})$ is also an $\left(\mathcal{L}^{\mathsf{poly}\log n}_{O(1),\mathsf{poly}(n),\oplus 1}, \epsilon \cdot 2^{\mathsf{poly}\log n}, cn\right)$-LRCC.

Let $(P_{3\mathrm{SAT}}, V_{3\mathrm{SAT}})$ denote the PCP system of [2, Theorem 1] (used to prove that $3\mathrm{SAT} \in \mathsf{PCP}\left[\log n, \log^2 n, \frac{1}{2}, \mathsf{poly}(n)\right]$), amplified to have soundness error $\epsilon_S < \frac{1}{6}$. We apply Construction 4.8 to $(\mathsf{Comp}, \mathsf{E})$ and $(P_{3\mathrm{SAT}}, V_{3\mathrm{SAT}})$, and show that the resultant PCP system for 3SAT has *oblivious honest-verifier ZK*, and the view of the verifier, given oracle access to a simulated proof, is $\epsilon \cdot 2^{\mathsf{poly}\log n}$ statistically-close to his view when given oracle access to an honestly-generated proof. To that effect, we repeat the proof of Proposition 4.4, using the fact that $\left|C^{3\mathrm{SAT}}\right| = cn$. The only difference from

the proof of Proposition 4.4, is that we now need to prove ZK. We define a simulator Sim as follows: on input x, Sim runs the LRCC simulator Sim^{in} (whose existence is guaranteed by the leakage-resilience of the LRCC) to generate an assignment $\mathcal{W}^S$ to φ_x (which is also a wire assignment to the circuit $\hat{C}_x$, which is the output of Comp on the circuit "C^{SAT} with x hard-wired into it"); runs P_{3SAT} on $\varphi_x, \mathcal{W}^S$ to generate a PCP π^S for the claim "$\varphi_x \in$ 3SAT"; and outputs π^S. Notice that every subset $\mathcal{I}$ of at most $\mathsf{poly}\log(n)$ proof symbols constitutes a leakage function $\ell_{\mathcal{I}}$ on the proof oracle. If for every such set $\mathcal{I}$, $\ell_{\mathcal{I}} \in \mathcal{L}^{\mathsf{poly}\log(n)}_{O(1),\mathsf{poly}(n),\oplus 1}$, and the honest verifier makes at most $\mathsf{poly}\log(n)$ oracle queries, then the leakage-resilience of the LRCC guarantees that $\mathsf{Simulated}_{V,\pi^S}(x)$ and $\mathsf{Real}_{V,P}(x,w)$ are statistically close (where P, V are the prover and verifier of Construction 4.8). For our choice of the underlying PCP system $(P_{\text{3SAT}}, V_{\text{3SAT}})$, this condition on the subsets $\mathcal{I}$ of proof symbols is satisfied (due to the prover algorithm, and since the honest verifier makes only $\mathsf{poly}\log(n)$ oracle queries, tosses only $r = \mathsf{poly}\log(n)$ coins, and the proof has size $\mathsf{poly}(n)$), and so the resultant system is an oblivious-ZKPCP system for 3SAT with soundness error $\frac{1}{6}$. Since $\epsilon = \mathsf{negl}(n)$ then for a large enough n, $\epsilon \cdot 2^{\mathsf{poly}\log(n)} \leq \frac{1}{6}$, so Theorem 6.5 implies that 3SAT $\in \mathsf{BPP}$, so $\mathsf{NP} \subseteq \mathsf{BPP}$. □

Acknowledgements

We thank the anonymous TCC reviewers for helpful comments, and in particular for pointing out the simple construction of CZKPCP from PCP and NIZK. The first author was supported by ERC starting grant 259426, ISF grant 1709/14, and BSF grant 2012378. Research done in part while visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467. Research also supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. The second author was supported by ERC starting grant 259426 and an IBM PhD Fellowship. The third author was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, and the National Natural Science Foundation of China Grant 61033001, 61350110536, 61361136003.

References

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

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

[3] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013, 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108, 2013.

[4] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *Advances in Cryptology - CRYPTO 2014, 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 276–294, 2014.

[5] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a Von Neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, pages 781–796, 2014.

[6] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[7] 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, 20th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceedings*, pages 262–279, 2001.

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

[9] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, STOC 2006, Seattle, WA, USA, May 21-23, 2006*, pages 711–720, 2006.

[10] 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.

[11] 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, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 135–156, 2010.

[12] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 308–317, 1990.

[13] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[14] Oded Goldreich. Three XOR-Lemmas - an exposition. In *Studies in Complexity and Cryptography*, pages 248–272. 2011.

[15] 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, STOC 1985, Providence, Rhode Island, USA, May 6-8, 1985*, pages 291–304, 1985.

[16] 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.

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

[18] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, STOC 1986, Berkeley, California, USA, May 28-30, 1986*, pages 6–20, 1986.

[19] Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge PCPs. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 151–168, 2012.

[20] Yuval Ishai, Amit Sahai, and David 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.

[21] 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.

[22] 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.

[23] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC 1992, Victoria, British Columbia, Canada, May 4-6, 1992*, pages 723–732, 1992.

[24] Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, STOC 1997, El Paso, Texas, USA, May 4-6, 1997*, pages 496–505, 1997.

[25] Adam Klivans. On the derandomization of constant depth circuits. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques - 4th International Workshop, APPROX 2001, and 5th International Workshop, RANDOM 2001, Berkeley, CA, USA, August 18-20, 2001, Proceedings*, pages 249–260, 2001.

[26] Shachar Lovett and Srikanth Srinivasan. Correlation bounds for poly-size AC^0 circuits with $n^{1-o(1)}$ symmetric gates. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, pages 640–651, 2011.

[27] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography - 1st Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, pages 278–296, 2004.

[28] Eric Miles and Emanuele Viola. Shielding circuits with groups. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC 2013, Palo Alto, CA, USA, June 1-4, 2013*, pages 251–260, 2013.

[29] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437, 1990.

[30] 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.

[31] U.V. Vazirani. Randomness, adversaries and computation. Ph.D. Thesis, EECS, UC Berkeley.

A The Circuit Compiler of Faust Et Al. [11]

In this section we present the circuit compiler of Faust et al. [11]. Their construction uses the gadgets described in Construction A.2.[16]

Construction A.1 (Circuit compiler, [11])**.** The circuit compiler $\left(\mathsf{Comp}^{\mathrm{FRRTV}}, \mathsf{E}^{\mathrm{FRRTV}}\right)$ is defined as follows. Let $\mathsf{E}^{\mathrm{in}} = \left(\mathsf{Enc}^{\mathrm{in}}, \mathsf{Dec}^{\mathrm{in}}\right)$ be a linear encoding scheme which outputs encodings of length $\hat{n}\left(n,\sigma\right)$, with decoding vectors $\mathbf{d}^{\hat{n}(n,\sigma)}$. We denote $\hat{n}_1 = \hat{n}\left(1,\sigma\right)$, and $n'\left(n,\sigma\right) = \sigma\left(\hat{n}_1^2+1\right)$. Then $\mathsf{E}^{\mathrm{FRRTV}} = \left(\mathsf{Enc}^{\mathrm{FRRTV}}, \mathsf{Dec}^{\mathrm{FRRTV}}\right)$, where $\mathsf{Enc}^{\mathrm{FRRTV}}\left(x, 1^\sigma\right) = \mathsf{Enc}^{\mathrm{in}}\left(\left(x, 0^{n'}\right), 1^\sigma\right)$, and $\mathsf{Dec}^{\mathrm{FRRTV}}\left(\left(\mathbf{x},\mathbf{m}\right), 1^\sigma\right) = \mathsf{Dec}^{\mathrm{in}}\left(\mathbf{x}, 1^\sigma\right)$, where $\mathbf{x} \in \mathbb{F}^{\hat{n}(n,\sigma)}$.

$\mathsf{Comp}^{\mathrm{FRRTV}}$ on input a circuit $C : \mathbb{F}^n \to \mathbb{F}^s$ containing $+, -, \times, \mathsf{id}, \mathsf{copy}$ and const_α gates, outputs a circuit C^{FRRTV}, in which every gate is replaced with the corresponding gadget, as described in Construction A.2. The input $(\mathbf{x},\mathbf{m}) \in \mathsf{Enc}^{\mathrm{FRRTV}}\left(x, 1^{|C|}\right)$ of C^{FRRTV} is interpreted as an encoding $\mathbf{x}$ of some input x for C, and a collection $\mathbf{m}$ of masking inputs for the gadgets of C^{FRRTV}. The masking inputs used in the gadgets are taken from $\mathbf{m}$ (every masking input in $\mathbf{m}$ is used at most once).

In the compiled circuit all gates (including the output gate) are replaced with gadgets that operate on encodings. To allow compiled circuits to have *decoded* outputs, [11] add a decoding sub-circuit following its output gate. For notational convenience, they assume that the original (i.e., un-compiled) circuit contains a special "decoder" gate following its output gate, which appears nowhere else in the circuit, and computes the identity function. We will make the same assumption.

Construction A.2. Let $\mathsf{E}^{\mathrm{in}} = \left(\mathsf{Enc}^{\mathrm{in}}, \mathsf{Dec}^{\mathrm{in}}\right)$ be a linear encoding scheme which outputs encodings of length $\hat{n}\left(n,\sigma\right)$, with decoding vectors $\mathbf{d}^{\hat{n}(n,\sigma)}$. We denote $\hat{n}_1 = \hat{n}\left(1,\sigma\right)$.

$\times$ **gadget:** inputs $\mathbf{a} \in \mathsf{Enc}^{\mathrm{in}}\left(a, 1^\sigma\right), \mathbf{b} \in \mathsf{Enc}^{\mathrm{in}}\left(b, 1^\sigma\right)$ for $a, b \in \mathbb{F}$, and masking inputs $\mathbf{r}^1, ..., \mathbf{r}^{\hat{n}_1+1} \in \mathsf{Enc}^{\mathrm{in}}\left(0, 1^\sigma\right)$; output $\mathbf{c} \in \mathsf{Enc}^{\mathrm{in}}\left(a \times b, 1^\sigma\right)$.

1. Computes an $\hat{n}_1 \times \hat{n}_1$ matrix $B = \mathbf{a}\mathbf{b}^T = \left(a_i \times b_j\right)_{i,j\in[m]}$ (using $\hat{n}_1^2$ $\times$ gates).

2. Computes an $\hat{n}_1 \times \hat{n}_1$ matrix S as the matrix whose columns are $\mathbf{r}^1, ..., \mathbf{r}^{\hat{n}_1}$.

[16] We note that in [11], the field operations were denoted $\oplus, \ominus, \odot$, while we use the notation $+, -, \times$. Moreover, they consider circuits that contain also fan-out 1 random gates (denoted by \$) that take no input and output a random field element. We do not explicitly use such gates in our constructions, because our applications use deterministic circuits (with no random gates), but our constructions naturally generalize to circuits containing random gates.

3. Computes an $\hat{n}_1 \times \hat{n}_1$ matrix $U = B + S$ (using $\hat{n}^2$ + gates).

4. Computes a vector $\mathbf{q} = U\mathbf{d}^{\hat{n}_1}$, i.e., q_i is the value that the i'th row of U encodes. This is computed using $\hat{n}_1$ $\times$, const_α, and + gates

5. Computes $\mathbf{c} = \mathbf{q} + \mathbf{r}^{\hat{n}_1+1}$.

$+, -$ **gadgets:** inputs $\mathbf{a} \in \mathsf{Enc}^{\text{in}}(a, 1^\sigma), \mathbf{b} \in \mathsf{Enc}^{\text{in}}(b, 1^\sigma)$ for $a, b \in \mathbb{F}$, and masking input $\mathbf{r} \in \mathsf{Enc}^{\text{in}}(0, 1^\sigma)$; output $\mathbf{c} \in \mathsf{Enc}^{\text{in}}(a + b, 1^\sigma)$ (or $\mathbf{c} \in \mathsf{Enc}^{\text{in}}(a - b, 1^\sigma)$ for the $-$ gadget).

1. Computes $\mathbf{q} = \mathbf{a} + \mathbf{b}$ (using $\hat{n}_1$ + gates). (The $-$ gadget computes $\mathbf{q} = \mathbf{a} - \mathbf{b}$.)

2. Computes $\mathbf{c} = \mathbf{q} + \mathbf{r}$ (using $\hat{n}_1$ + gates).

const_f **gadget for** $f \in \mathbb{F}$**:** masking input $\mathbf{r} \in \mathsf{Enc}^{\text{in}}(0, 1^\sigma)$; output $\mathbf{c} \in \mathsf{Enc}^{\text{in}}(f, 1^\sigma)$.

1. Computes some fixed encoding $\mathbf{f} \in \mathsf{Enc}^{\text{in}}(f)$ (using $\hat{n}_1$ const_α gates).

2. Computes $\mathbf{c} = \mathbf{f} + \mathbf{r}$ (using $\hat{n}_1$ + gates).

copy **gadget:** input $\mathbf{a} \in \mathsf{Enc}^{\text{in}}(a, 1^\sigma)$ for $a \in \mathbb{F}$, masking inputs $\mathbf{r}^1, \mathbf{r}^2 \in \mathsf{Enc}^{\text{in}}(0, 1^\sigma)$; outputs $\mathbf{b}, \mathbf{c} \in \mathsf{Enc}^{\text{in}}(a, 1^\sigma)$.

1. Computes $\mathbf{b} = \mathbf{a} + \mathbf{r}^1$ (using $\hat{n}_1$ + gates).

2. Computes $\mathbf{c} = \mathbf{a} + \mathbf{r}^2$ (using $\hat{n}_1$ + gates).

id **gadget:** input $\mathbf{a} \in \mathsf{Enc}^{\text{in}}(a, 1^\sigma)$ for $a \in \mathbb{F}$, masking input $\mathbf{r} \in \mathsf{Enc}^{\text{in}}(0, 1^\sigma)$; outputs $\mathbf{c} \in \mathsf{Enc}^{\text{in}}(a, 1^\sigma)$.

1. Computes $\mathbf{c} = \mathbf{a} + \mathbf{r}$ (using $\hat{n}_1$ + gates).

The leakage resilience proof of [11] used two gadget properties. First, their gadgets are locally reconstructible, meaning that for every 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) could be simulated in a low complexity class. Formally,

Definition A.3 (Local reconstructibility)**.** Let G be a gadget. A pair $(\mathbf{x}, \mathbf{y})$ of encodings is *plausible* for G if for some well-formed masking input m, G on input $(\mathbf{x}, \mathsf{m})$ outputs $\mathbf{y}$. Given a gadget G, $\epsilon > 0$, and families $\mathcal{L}, \mathcal{L}_G$ of functions, G is $(\mathcal{L}, \epsilon)$*-reconstructible by* $\mathcal{L}_G$ if the following holds. There exists a distribution REC over functions rec that take as input the standard inputs of G, and its output, and output simulated values for the masking inputs, and internal wires of G, such that for every plausible pair $(\mathbf{x}, \mathbf{y})$: $\mathrm{supp}(\mathsf{REC}) \subseteq \mathcal{L}_G$; and if rec is chosen according to REC then $\mathsf{rec}(\mathbf{x}, \mathbf{y})$ is $(\mathcal{L}, \epsilon)$-leakage-indistinguishable from the actual distribution of the wires of G (as determined by the distribution of the masking inputs), conditioned on $\mathbf{x}, \mathbf{y}$.

Faust et al. [11] prove that their gadgets (Construction A.2) are locally reconstructible in a low complexity class. Specifically:

Lemma A.4 (Gadgets are locally reconstructible, [11])**.** *Let $n \in \mathbb{N}$, $\mathcal{L}, \mathcal{L}_{\mathsf{E}}$ be families of functions, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. Then the following holds for the gadgets described in Construction A.2.*

- *$+$ and $-$ gadgets are $(\mathcal{L}, 0)$-reconstructible by $\mathsf{Shallow}(3\hat{n}_1, 2, O(\hat{n}_1))$.*
- *copy gadgets are $(\mathcal{L}, 0)$-reconstructible by $\mathsf{Shallow}(3\hat{n}_1, 1, O(\hat{n}_1))$.*

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

The second property is that gadgets are re-randomizing in the sense that the encodings at the output of each gadget are uniform subject to encoding the "correct" value. Formally,

Definition A.5 (Re-randomization)**.** A gadget G is *re-randomizing* if for every standard input $\mathbf{x} = \mathsf{E}(x)$, and every masking input m chosen according to the distribution of masking inputs to G, $G(\mathbf{x}, \mathsf{m})$ is random subject to encoding the correct output (as determined by x, and the operation which G emulates).

Faust et al. [11] also prove that the gadgets are re-randomizing, which follows directly from the fact that the outputs of the gadgets are masked with random and independent well-formed vectors. Thus, they obtain the following result.

Proposition A.6 (LRCC, [11])**.** *Let* $\sigma \in \mathbb{N}$ *be a security parameter,* $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$, $\mathsf{S}(n)$ *be a size function,* $\mathcal{L}, \mathcal{L}_{\mathsf{E}}$ *be families of functions, and* E^{in} *be an encoding scheme with encodings of length* $\hat{n}(n, \sigma)$*. If* E^{in} *is linear and* $(\mathcal{L}_{\mathsf{E}}, \epsilon(n))$*-leakage-indistinguishable, and* $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(3, O\left(\hat{n}\left(1, \mathsf{S}(n)\right)\right)\right)$*, then Construction A.1 is* $(\mathcal{L}, \epsilon(n) \cdot \hat{n}\left(1, \mathsf{S}(n)\right) \cdot \mathsf{S}(n), \mathsf{S}(n))$*-strong-leakage-resilient.*

B The Complexity of the Arora-Safra PCP [2]

The NA-WIPCP system of Section 4 used the PCP system of Arora and Safra [2, Theorem 1] as a building block (more specifically, on the system used to prove that $3\mathrm{SAT} \in \mathsf{PCP}\left[\log n, \log^2 n, \frac{1}{2}, \mathsf{poly}(n)\right]$), and relied on the property that every proof bit in that system is computable by a low-depth, polynomial-sized circuit with "few" $\oplus$ gates (specifically, in $\mathcal{L}_{O(1),\mathsf{poly}(n),\oplus O(1)}$, see Definition 3.40). In this section we analyze the construction of [2] and show that it has the required property. The PCP system of [2] is constructed for the NP-complete language 3SAT, where the prover, on input a 3CNF φ with n clauses and n variables,[17] and a satisfying assignment A for φ, transforms A into a PCP π for the claim "$\varphi \in 3\mathrm{SAT}$". Arora and Safra show a construction over large fields, which we described in Section B.1, and we show (Section B.2) that it can be extended to extension fields of $\mathsf{GF}(2)$. The analysis in this section is very succinct, and is focused on the properties needed for the proof of Theorem 1.1. We refer the reader to [2] for a more detailed description of the PCP generation and verification.

B.1 The Construction over Large Fields

At a high level, the PCP system of [2] operates as follows. The prover P, given a 3CNF φ, and a satisfying assignment A for φ, generates a low-degree polynomial $\hat{A} : \mathbb{F}^m \to \mathbb{F}$, where $m \in \mathbb{N}$, and $\mathbb{F}$ is a large finite field, such that $\hat{A}$ "represents" A (specifically, $\hat{A}$ is the Reed-Muller encoding of A). P then uses $\hat{A}$, and the structure of φ, to construct another low-degree polynomial g_A over $\mathbb{F}$. This representation is useful because [2] show that there exist an $H \subseteq \mathbb{F}$, and a (not too large) set $\mathcal{P}$ of low-degree polynomials $P_1, P_2, ..$, such that if P_i is chosen uniformly at random from $\mathcal{P}$, then:

[17] The assumption that there are n clauses and variables is without loss of generality, since the number of clauses and variables can be made equal by introducing new clauses or variables, if necessary.

if A satisfies φ then $P_i \cdot g_A$ sums to 0 on H^m with probability 1; and otherwise this happens with probability at most $\frac{1}{8}$. Moreover, the set $\mathcal{P}$ is independent of A and can be efficiently constructed in polynomial time. Consequently, to verify that $\varphi \in 3SAT$, the verifier V can pick a random $P_i \in \mathcal{P}$ and check that $g_A \cdot P_i$ vanishes on H^m. Arora and Safra show that *if g_A is guaranteed to be a low-degree polynomial*, then given access to the truth-table of g_A, and to an auxiliary proof, one can verify that $g_A \cdot P_i$ sums to 0 on H^m (this is called the *sum-check test*). (Notice that V can compute P_i on its own, so V does not need access to the truth-table of P_i, or $g_A \cdot P_i$.) Moreover, they show that the value of g_A at any random point can be computed given only φ and the value of $\hat{A}$ at 3 (related) random points, so it suffices to give V the truth-table of $\hat{A}$. Moreover, they show how to efficiently verify, given the truth-table of *any* function f, and an auxiliary proof, that f is a low-degree polynomial (this is called the *low-degree test*). Thus, the PCP prover P outputs a proof consisting of the truth-table of $\hat{A}$, a proof for the low-degree test for $\hat{A}$, and for every $P_i \in \mathcal{P}$, a proof for the sum-check test of $g_A \cdot P_i$. The verifier V first verifies that $\hat{A}$ has low degree, then picks a random $P_i \in \mathcal{P}$ and performs the sum-check test on $g_A \cdot P_i$ (using the truth table of $\hat{A}$ to generate values of g_A, when needed). We now describe the ingredients of the proof, and how they are constructed, in more detail.

Let φ be a 3CNF, and assume without loss of generality that φ has n variables and n clauses. Let $h = O(\log n)$ and $m = O\left(\frac{\log n}{\log \log n}\right)$ such that $n = (h+1)^m$. Denote $H = \{0, 1, ..., h\}$, and let $\mathbb{F} = \mathsf{GF}\left(2^t\right)$ for $t = O(\log h)$ (the constant in the definition of $O(\cdot)$ is taken to be large enough, such that $H \subseteq \mathbb{F}$). Identify every variable x_i and every clause c_i of φ with a vector $v \in H^m$ (e.g., the representation of i in basis h). Interpret the assignment A to φ as a function $A : H^m \to \mathbb{F}$ where $A(x_i) \in \{0, 1\}$ is the value that A assigns to x_i.

Constructing an extension-polynomial for the assignment A. Let $\mathbb{F}_h[x_1, ..., x_m]$ denote the class of m-variate polynomials of individual degree at most h over variables $x_1, ..., x_m$ with coefficients in $\mathbb{F}$. (A polynomial p has individual degree at most h if for every x_i, and every monomial M of p, the degree of x_i in M is at most h.) Let $\hat{A} \in \mathbb{F}_h[x_1, ..., x_m]$ be a polynomial such that $A(y) = \hat{A}(y)$ for every $y \in H^m$. $\hat{A}$ is called the *polynomial extension of A over $\mathbb{F}$* (because the degree is h, it is also unique), and can be constructed using the Lagrange polynomials L_y as follows:

$$\hat{A}(x_1, ..., x_m) = \Sigma_{y=(y_1,...,y_m) \in H^m} A(y) \cdot \Pi_{i=1}^{m} L_{y_i}(x_i)$$

where for every $y_i \in H$, $L_{y_i}(x_i) := \frac{\Pi_{y_i \neq z \in H}(x_i - z)}{\Pi_{y_i \neq z \in H}(y_i - z)}$. Notice that for every $y_i \in H$, L_{y_i} depends only on H, and is independent of both A and φ.

Representing the formula φ as a polynomial. Arora and Safra show that there exists a family $\mathcal{P}$ of $l := O(mh)$ polynomials $P_1, ..., P_l \in \mathbb{F}_{7h}[x_1, ..., x_{4m}]$ (that depend both on $\hat{A}$ and on φ), constructible in polynomial time, possessing the following 3 properties.

- If A satisfies φ then $\Sigma_{x_1,...,x_{4m} \in H} P_i(x_1, ..., x_{4m}) = 0$ for every $1 \leq i \leq l$.
- If A does not satisfy φ then $\Pr_{P \in_R \{P_1,...,P_l\}}\left[\Sigma_{x_1,...,x_{4m} \in H} P(x_1, ..., x_{4m}) = 0\right] \leq \frac{1}{8}$.
- For every $1 \leq i \leq l$ and every point $\alpha \in \mathbb{F}^{4m}$, the value of P_i at α is determined by the value of $\hat{A}$ at 3 points $a_1, a_2, a_3 \in \mathbb{F}^m$. Moreover, if α is uniformly distributed in $\mathbb{F}^{4m}$ then each of a_1, a_2, a_3 is uniformly distributed in $\mathbb{F}^m$.

Concretely, $P_1, P_2, ...$ are obtained through the following procedure:

- *(Arithmetization of φ.)* For $j = 1, 2, 3$, define $\chi_j : H^{2m} \to \{0, 1\}$ such that $\chi_j(c, v) = 1$ if and only if v is the j'th variable in clause c, and define $s_j : H^m \to \{0, 1\}$ such that $s_j(c) = 1$

if and only if the j'th variable in clause c is not negated (both polynomials can be defined using the Lagrange polynomials as explained above). Let $\hat{\chi}_j, \hat{s}_j$ be the polynomial extensions of χ_j, s_j over $\mathbb{F}$, respectively. Notice that $\hat{\chi}_j \hat{s}_j$ can be computed locally given φ, i.e., they are independent of A.

- *(Arithmetzation of A.)* Define $g_A : \mathbb{F}^{4m} \to \mathbb{F}$ as follows: $g_A(z, w_1, w_2, w_3) = \Pi_{j=1}^{3} \hat{\chi}_j(z, w_j) \left(\hat{s}_j(z) - \hat{A}(w_j) \right)$. Then g_A has individual degree at most $6h$, and its evaluation at any point in $\mathbb{F}^{4m}$ can be locally computed given φ and the value of $\hat{A}$ at 3 corresponding points in $\mathbb{F}^m$. Notice that A satisfies φ if and only if g_A vanishes on H^{4m}.

- Let $R_1, ..., R_l \in \mathbb{F}_h[x_1, ..., x_{4m}]$ be such that for every $f : H^{4m} \to \mathbb{F}$ that is not identically 0,

$$\Pr_{R \in_R \{R_1, ..., R_l\}} \left[\Sigma_{x_1, ..., x_{4m} \in H} R(x_1, ..., x_{4m}) \cdot f(x_1, ..., x_{4m}) = 0 \right] \leq \frac{1}{8}.$$

  Such a family exists and can be constructed in polynomial time (see [2, Lemma 10] for details). As will become evident later, the manner in which this family is constructed is of no interest to us, because every R_i has $\mathsf{poly}(n)$ monomials, and is independent of φ, A (and can therefore be locally constructed by the verifier).

  For every $1 \leq i \leq l$, set $P_i(x_1, ..., x_{4m}) := R_i(x_1, ..., x_{4m}) g_A(x_1, ..., x_{4m})$. Notice that given the evaluations of $\hat{A}$ on $\mathbb{F}^m$, the verifier can locally compute the values of $P_i(a_1, ..., a_{4m})$, for every $(a_1, ..., a_{4m}) \in \mathbb{F}^{4m}$, which requires the value of $\hat{A}$ at 3 points.

THE PROOF. The proof consists of a low-degree proof for the low-degree test, and a sum-check proof for the sum-check test, and is generated as follows:

- For both tests, the proof contains the evaluation of $\hat{A}$ on all points in $\mathbb{F}^m$.

- For the low-degree test, for every $j \in [m]$ and every $a_{-j} := (a_1, ..., a_{j-1}, a_{j+1}, ..., a_m) \in \mathbb{F}^{m-1}$, let

$$\hat{A}_{j, a_{-j}}(x) := \hat{A}(a_1, ..., a_{j-1}, x, a_{j+1}, ..., a_m),$$

  then the proof contains the evaluation of $\hat{A}_{j, a_{-j}}$ on every point in $\mathbb{F}$ (i.e., the proof contains the evaluations of restrictions of $\hat{A}$ to all lines in $\mathbb{F}^{m-1}$).

- For the sum-check test, for every polynomial P_i, every $j \in [4m]$ and every $a_1, ..., a_{j-1} \in \mathbb{F}$, let

$$P_{i, a_1, ..., a_{j-1}}(x) := \Sigma_{z_{j+1}, ..., z_{4m} \in H} P_i(a_1, ..., a_{j-1}, x, z_{j+1}, ..., z_{4m}),$$

  then the proof contains evaluations of $P_{i, a_1, ..., a_{j-1}}$ on all points in $\mathbb{F}$.

VERIFICATION. To verify that $\varphi \in$ 3SAT, the verifier performs the following checks.

- Checks that $\hat{A}$ has individual degree at most h, by running the low degree test on $\hat{A}$.

- Picks a random $i \in [l]$, and checks that P_i sums to 0 on H^m by running the sum-check test. Whenever the sum-check test requires the value of P_i at some (random) point, the verifier reads the value of $\hat{A}$ at (the) 3 (corresponding) points, and locally computes the value of P_i at that point.

B.2 The Construction over $\mathsf{GF}(2)$

In this section we show how to transform the PCP of [2] (which consists of field elements) into a proof over bits. (We are interested in representing the PCP over bits, because our NA-WIPCP construction needs the function generating the PCP from the NP witness to be in the leakage class that the SAT-respecting relaxed LRCC withstands. In our case, this class consists of functions computable by low-depth *boolean* circuits.)

We note first that since $[\mathbb{F} : \mathsf{GF}(2)] = t$, then it is isomorphic to the field $\mathsf{GF}(2)[X]/P_t(X)$, where $P_t(X)$ is an irreducible polynomial of degree t, so we can assume that $\mathbb{F}$ *is* the field $\mathsf{GF}(2)[X]/P_t(X)$. Therefore, there exists a bijection B between $\mathbb{F}$ and $(\mathsf{GF}(2))^t$, which maps a polynomial $\sum_{i=0}^{t-1} a_i X^i$ to the vector $(a_0, ..., a_{t-1})$ of its coefficients. ($\mathsf{B}, \mathsf{B}^{-1}$ are computable in time $O(t)$.) We will need the following fact connecting polynomials over extension fields to polynomials over the base field.

Fact B.1. *Let $\mathbb{F} = \mathsf{GF}(2^t)$, and $d, m \in \mathbb{N}$. Then every polynomial $p \in \mathbb{F}_d[x_1, ..., x_m]$ corresponds to a collection of t multi-linear polynomials $p_0, ..., p_{t-1} \in \mathsf{GF}(2)[y_1, ..., y_{m \cdot t}]$ of total degree at most $\min\{md, mt\}$, each with at most 2^{mt} monomials, such that for every $\alpha_1, ..., \alpha_m \in \mathbb{F}$, $\mathsf{B}(p(\alpha_1, ..., \alpha_m)) = (p_0(\mathsf{B}(\alpha_1), ..., \mathsf{B}(\alpha_m)), ..., p_{t-1}(\mathsf{B}(\alpha_1), ..., \mathsf{B}(\alpha_m)))$. Moreover, given the truth-table of p, the value of every p_i at any point in $\beta \in \mathsf{GF}(2)^{mt}$ is computable in time $O(tm)$, and requires the value of p at a single point $\alpha \in \mathbb{F}^m$; and given the truth-tables of $p_0, ..., p_{t-1}$, the value of p at any point in $\alpha \in \mathbb{F}^m$ is computable in time $O(tm)$, and requires the values of $p_0, ..., p_{t-1}$ at a single point $\beta \in \mathsf{GF}(2)^{mt}$.*

The PCP described in the previous section consists of the evaluations of a collection of polynomials over $\mathbb{F}$. The general idea of the bit-PCP is to replace the evaluations of these polynomials with the evaluations of the corresponding t polynomials over $\mathsf{GF}(2)$ whose existence is guaranteed by Fact B.1.

THE PROOF. Let $\hat{A}, \left\{\hat{A}_{j,a_{-j}}\right\}_{j,a_{-j}}, \left\{P_{i,a_1,...,a_{j-1}}\right\}_{i,a_1,...,a_{j-1}}$ be the polynomials whose evaluations appear in the PCP of Section B.1. Let $\{p_{\hat{A},0}, ..., p_{\hat{A},t-1}\}$, $\{p_{\hat{A}_{j,a_{-j}},0}, ..., p_{\hat{A}_{j,a_{-j}},t-1}\}_{j,a_{-j}}$, and $\{p_{P_{i,a_1,...,a_{j-1}},0}, ..., p_{P_{i,a_1,...,a_{j-1}},t-1}\}_{i,a_1,...,a_{j-1}}$ be the corresponding collections of t multivariate polynomials over $\mathsf{GF}(2)$, whose existence is guaranteed by Fact B.1. Then the proof consists of the evaluations of all these polynomials.

THE VERIFICATION PROCEDURE. The verification procedure is carried out essentially in the same way, except that the values of every polynomial $\hat{A}$ and $\hat{A}_{j,a_{-j}}, P_{i,a_1,...,a_{j-1}}$ at points $(\alpha_1, ..., \alpha_m) \in \mathbb{F}^m$ and $\alpha \in \mathbb{F}$, respectively, are reconstructed using the values of the corresponding multilinear polynomials over $\mathsf{GF}(2)$ at $(\mathsf{B}(\alpha_1), ..., \mathsf{B}(\alpha_m))$ and $\mathsf{B}(\alpha)$, respectively.

B.3 Setting the Parameters

We set the parameters of the construction to be $h = O(\log n)$, $m = \frac{\log n}{\log h} = O\left(\frac{\log n}{\log \log n}\right)$ and $\log |\mathbb{F}| = t := c_h \log h = c \log \log n$ (for large enough constants c, c_h). For this choice of parameters, the construction (used in [2, Theorem 1] to prove that $3\mathrm{SAT} \in \mathsf{PCP}\left[\log n, \log^2 n, \frac{1}{2}, \mathsf{poly}(n)\right]$) yields a PCP system with the following properties:

- *Query complexity.* The verifier makes $O(m)$ queries to symbols of size $h \log h$, reading a total of $q := m \cdot h \cdot \log h = O\left(\frac{\log n}{\log \log n} \log n \cdot \log \log n\right) = O\left(\log^2 n\right)$ bits of the proof.
- *Soundness.* The system has constant soundness error $\frac{1}{2}$.

- *Proof length.* The proof has length $\mathsf{poly}(n)$. Indeed, it consists of the evaluations of the following polynomials.
  - $\hat{A} \in \mathbb{F}_h[x_1, ..., x_m]$, which requires the evaluations on $\mathbb{F}^m$ (the evaluation at every point is a field element). This requires $|\mathbb{F}|^m$ field elements, i.e., $(2^c \log n)^{O\left(\frac{\log n}{\log\log n}\right)} = 2^{O\left(\log\log n \cdot \frac{\log n}{\log\log n}\right)} = \mathsf{poly}(n)$ bits.
  - $\hat{A}_{j,a_{-j}} \in \mathbb{F}_h[x]$, for $j \in [m]$ and $a_{-j} \in \mathbb{F}^{m-1}$. There are $m \cdot |\mathbb{F}|^{m-1}$ such polynomials, and the evaluation of each requires the value (in $\mathbb{F}$) at $\mathbb{F}$ points. Therefore, these evaluations require $m|\mathbb{F}|^{m-1} \cdot |\mathbb{F}| = m|\mathbb{F}|^m$ field elements, i.e., $|\mathbb{F}|^m \log|\mathbb{F}| = \mathsf{poly}(n)$ bits.
  - $P_{i,j,a_1,...,a_{j-1}} \in \mathbb{F}_h[x]$, for every $i \in [l], j \in [m]$, and $(a_1, ..., a_{j-1}) \in \mathbb{F}^{j-1}$. There are at most $l \cdot m \cdot |\mathbb{F}|^{m-1}$ such polynomials, where their value at each point of $\mathbb{F}$ is a field element in $\mathbb{F}$. Therefore, these evaluations require $l \cdot m \cdot |\mathbb{F}|^{m-1} \cdot |\mathbb{F}| = O(hm) \cdot m \cdot |\mathbb{F}|^m$ field elements, and $\frac{\log^3 n}{\log^2 \log n} \cdot \mathsf{poly}(n) = \mathsf{poly}(n)$ bits.
- *Proof structure.* Every proof symbol (in the construction over large fields, Section B.1) is computable by a multivariate polynomial over $\mathbb{F}$ with at most $4m$ variables, of individual degree at most $O(h) = O(\log n)$, with at most $\mathsf{poly}(n)$ monomials. Therefore, by Fact B.1 every proof bit in the bit-implementation of the PCP of [2] is computable by a multilinear polynomial over $\mathsf{GF}(2)$ with at most $4mt = O\left(\frac{\log n}{\log\log n} \cdot \log\log n\right) = O(\log n)$ variables, of total degree at most $\min\{4m \cdot O(h), 4m \cdot c\log\log n\} = O\left(\frac{\log^2 n}{\log\log n}\right)$, with at most $2^{4m \cdot t} = 2^{O\left(\frac{\log n}{\log\log n}\right) \cdot c\log\log n} = 2^{O(\log n)} = \mathsf{poly}(n)$ monomials. In particular, every proof bit can be computed by an AC^0 circuit of depth 2 with a single $\oplus$ gate of unbounded fan-in. (The first layer will compute all the monomials, using (for every monomial) a single AND gate of unbounded fan-in; and the second would sum all monomials, using a single $\oplus$ gate of unbounded fan-in.)
- *Computation time.* The proof over $\mathbb{F}$ can be constructed and verified in polynomial time. Generating the proof over $\mathsf{GF}(2)$ requires the prover to generate the truth tables of the polynomials $\{p_{\hat{A},0}, ..., p_{\hat{A},t-1}\}$, $\{p_{\hat{A}_{j,a_{-j}},0}, ..., p_{\hat{A}_{j,a_{-j}},t-1}\}_{j,a_{-j}}$, and $\{p_{P_{i,a_1,...,a_{j-1}},0}, ..., p_{P_{i,a_1,...,a_{j-1}},t-1}\}_{i,a_1,...,a_{j-1}}$. The generation of every such point from the truth-tables of $\hat{A}, \hat{A}_{j,a_{-j}}$ and $P_{i,a_1,...,a_{j-1}}$ requires time $O(4mt) = O(\log n)$, and so the proof generation time increases by a multiplicative $O(\log n)$ factor. The verification of the proof requires the verifier to generate, given the queries to the proof over $\mathbb{F}$, the corresponding queries over $\mathsf{GF}(2)$ (for every query, this requires at most $O(mt) = O(\log n)$ time), and then map the query answers in $\mathsf{GF}(2)^t$ back to field elements in $\mathbb{F}$, which takes time $O(mt)$ for every query. Therefore, the verification time also increases by a multiplicative factor of $O(\log n)$.