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

Yuval Ishai[1], Mor Weiss[2], and Guang Yang[3]

[1] Department of Computer Science, Technion, Haifa, Israel, and
Department of Computer Science, UCLA, LA, California, USA.
yuvali@cs.technion.ac.il

[2] Department of Computer Science, Technion, Haifa, Israel.
morw@cs.technion.ac.il

[3] Department of Computer Science, Tsinghua University, Beijing, China.
guang.research@gmail.com

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.

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 [14], 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 [21, 17, 19] 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" [21, 17]. 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 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 [18, 11, 25, 10, 15, 23].

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., [3–5] 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.[4] 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

[4] 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 [19].

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., [17] 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 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., [18, 11, 23]), 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 a malicious prover can 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 [18, 23] operate essentially in the same way.) The input of $\hat{C}$ includes sufficiently many encodings of 0 to be used for the entire computation.[5] 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 $\hat{C}'$ 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 can 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.[6] However, holding two copies of the original circuit still does not guarantee that the 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

[5] 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 [18, 23], and as a result we can incorporate the necessary encodings (used for refreshing) into the encoded input.

[6] This technique is reminiscent of the "2-key trick" of [24], used to convert a CPA-secure encryption scheme into a CCA-secure one.

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, and not *ZK*PCPs. If we had a SAT-respecting LRCC, the transformation described above would give a ZKPCP. However, we show that known LRCCs withstanding global leakage [18, 11, 23] 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 [22], 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 polylogarithmically many proof bits, and guarantee WI against (adaptive) malicious verifiers querying a fixed polynomial number of proof bits.

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

In Section 4 we describe a simple alternative approach for constructing CZKPCPs by applying a PCP on top of a standard non-interactive zero-knowledge (NIZK) proof. This should be contrasted with our main construction that 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 1 (Leakage-indistinguishability of distributions). *Let* D, D' *be finite sets,* $\mathcal{L} = \{\ell : D \rightarrow D'\}$ *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 1. In case $\mathcal{L}$ consists of functions over different 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 $\mathbf{y} = \left(\mathbf{y}^1, ..., \mathbf{y}^n\right)$. Second, $\mathsf{Dec}(\mathbf{y}) = \left(\langle \mathbf{r}^{\hat{n}(n)}, \mathbf{y}^1\rangle, ..., \langle \mathbf{r}^{\hat{n}(n)}, \mathbf{y}^n\rangle\right)$ (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 (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 is referred to as an *input gate*; or is labeled by a constant $\alpha \in \mathbb{F}$ and is 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 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. 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 3. *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 4 (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 5 (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 *For a Circuit C, a leakage function $\ell : \mathbb{F}^{|C|} \rightarrow \mathbb{F}^m$ 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 6 (Relaxed LRCC). *For a function class $\mathcal{L}$, $\epsilon (n) : \mathbb{N} \rightarrow \mathbb{R}^+$, and a size function $\mathsf{S} (n) : \mathbb{N} \rightarrow \mathbb{N}$, 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 boolean circuit C 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 $x \in \{0, 1\}^n$, we have $\mathsf{SD} \left(\ell [\mathsf{Sim} (C, C (x))], \ell \left[\hat{C}, \hat{x}\right]\right) \leq \epsilon (|x|)$, where $\hat{x} \leftarrow \mathsf{Enc} \left(x, 1^{|C|}\right)$.*

Definition 6 is relaxed in the sense that (unlike [18, 11, 23]) Sim is not required to be efficient.

The error in Definitions 5 and 6 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].

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 ×, 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 to the gadgets); 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 $C : \mathbb{F}^n \rightarrow \mathbb{F}$ 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 the compiler of [11]. 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 the circuit-compiler of [11]; 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 the circuit-compiler of [11]; 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 values.

Construction 3 ("Binarization" sub-circuit $\mathcal{T}$) *$\mathcal{T} : \mathbb{F} \rightarrow \mathbb{F}$ is defined as $\mathcal{T}(z) = -\prod_{0 \neq a \in \mathbb{F}} (z - a)$, computed using $O(|\mathbb{F}|)$ × and constant gates arranged in $O(\log |\mathbb{F}|)$ layers.*

Observation 4 *$\mathcal{T}(0) = 1$, 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 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 5 (Oblivious mask-checking sub-circuit $\mathcal{T}_0$) *$\mathcal{T}_0 : \mathbb{F}^{\mathsf{r}} \times \mathbb{F}^{\mathsf{r}} \rightarrow \mathbb{F}$ 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 6 *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 4 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 guarantee 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 7 (Non-oblivious mask-checking sub-circuit $\mathcal{T}_V$) *Let $n, \sigma, \kappa \in \mathbb{N}$, $\hat{n} = \hat{n}(n + \kappa, \sigma)$, and $\left\{\boldsymbol{d}^{\hat{n}}\right\}$ be the decoding vectors of E^{in}. We define the* decoding sub-circuit *$\mathcal{D}_V : \mathbb{F}^{\hat{n}} \rightarrow \mathbb{F}$ corresponding to $\boldsymbol{d}^{\hat{n}}$ as follows: $\mathcal{D}_V(\boldsymbol{v}) = \langle \boldsymbol{d}^{\hat{n}}, \boldsymbol{v} \rangle$, 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 $\mathcal{T}_V : \left(\mathbb{F}^{\hat{n}}\right)^{\mathsf{r}_0} \rightarrow \mathbb{F}$ as follows: for $\boldsymbol{R} = (\boldsymbol{r}_1, ..., \boldsymbol{r}_{\mathsf{r}_0})$ where $\boldsymbol{r}_i \in \mathbb{F}^{\hat{n}}$ for every $1 \leq i \leq \mathsf{r}_0$, $\mathcal{T}_V(\boldsymbol{R}) = \prod_{i \in [\mathsf{r}_0]} \mathcal{T}(\mathcal{D}_V(\boldsymbol{r}_i))$. $\mathcal{T}_V$ is computed using $O(\mathsf{r}_0)$ × 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 8 *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 9) uses the ingredients described above. Comp first compiles 2 copies C_1, C_2 of C, and $\mathcal{T}_0$, into $\hat{C}_1, \hat{C}_2, \hat{\mathcal{T}}_0$ (respectively), using the compiler of [11]. 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 9 ($(\mathcal{L}, \epsilon(n), \mathsf{S}(n))$-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 $\mathsf{E}^{\mathrm{in}} = \left(\mathsf{Enc}^{\mathrm{in}}, \mathsf{Dec}^{\mathrm{in}}\right)$ *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* $\{\boldsymbol{d}^{\hat{n}_{\mathrm{in}}}\}$*. Then* $\mathsf{Enc}(x, 1^\sigma) = (\hat{x}_1, \hat{x}_2)$*, where* $\hat{x}_i \leftarrow \mathsf{Enc}^{\mathrm{in}}((x, 0^{\mathsf{r}+\mathsf{r}_0}), 1^\sigma)$*; and* $\mathsf{Dec}((\hat{x}_1, \hat{x}_2), 1^\sigma)$ *computes* $\mathsf{Dec}^{\mathrm{in}}(\hat{x}_1, 1^\sigma)$*, 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* $\hat{x}_i = \left(\hat{x}_i^{\mathrm{in}}, \boldsymbol{R}_i, \boldsymbol{R}_i^0\right)$*, where* $\hat{x}_i^{\mathrm{in}}$ *is the encoding of* x*, and* $\boldsymbol{R}_i, \boldsymbol{R}_i^0$ *are encodings of* $0^{\mathsf{r}}, 0^{\mathsf{r}_0}$*, respectively. (*$\boldsymbol{R}_2^0$ *is not used in the construction, but it is part of* $\hat{x}_2$ *because the same internal encoding scheme* $\mathsf{Enc}^{\mathrm{in}}$ *is used to generate* $\hat{x}_1, \hat{x}_2$*.)*

Let $\left(\mathsf{Comp}^{\mathrm{FRRTV}}, \mathsf{E}^{\mathrm{FRRTV}}\right)$ *be 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* $\mathsf{f}\left(\left(\hat{x}_1^{\mathrm{in}}, \boldsymbol{R}_1, \boldsymbol{R}_1^0\right), \left(\hat{x}_2^{\mathrm{in}}, \boldsymbol{R}_2, \boldsymbol{R}_2^0\right)\right) := \mathcal{T}\left(\hat{C}_1\left(\hat{x}_1^{\mathrm{in}}, \boldsymbol{R}_1\right) - \hat{C}_2\left(\hat{x}_2^{\mathrm{in}}, \boldsymbol{R}_2\right)\right) \times \hat{\mathcal{T}}_0\left(\left(\boldsymbol{R}_1, \boldsymbol{R}_2\right), \boldsymbol{R}_1^0\right) \times \mathcal{T}_V\left(\boldsymbol{R}_1^0\right)$*. (*$\mathsf{f} = 1$ *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:*

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

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

Let $\mathsf{r}^{\mathrm{FRRTV}}$ *denote the maximal number of masking inputs used in a gadget used by the compiler of [11], and* $\mathsf{S}_0(\mathsf{r})$ *denote the size of* $\mathcal{T}_0$*. Then* $\mathsf{r}(\sigma) = \sigma \cdot \mathsf{r}^{\mathrm{FRRTV}}$ *and* $\mathsf{r}_0(\sigma) = \sigma \cdot \mathsf{S}_0\left(\mathsf{r}^{\mathrm{FRRTV}}\right)$*.*

Next, we briefly analyze the properties of the construction. (The full analysis appears in the full version.)

SAT-RESPECTING. 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 the compiler of [11] (applied to $\hat{\mathcal{T}}_0$)), guarantees that the flag is reset if the masking inputs of *both* $\hat{C}_1, \hat{C}_2$ are ill-formed. 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).

Relaxed leakage-resilience. At a high level, on input $C : \mathbb{F}^n \rightarrow \mathbb{F}$, 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 $\hat{C}$. If E is leakage-indistinguishable for a leakage class which is "somewhat stronger" than $\mathcal{L}$, then for every $\ell \in \mathcal{L}$, $\mathsf{SD}\left(\ell\left[\hat{C}, \hat{x}\right], \ell\left[\hat{C}, \hat{y}\right]\right) \leq \epsilon(n)$, 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 replacing 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 $\mathcal{L}^{\text{in}}$, 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 $\mathcal{L} \circ \mathcal{L}_R \subseteq \mathcal{L}^{\text{in}}$. In particular, if $\mathcal{L}^{\text{in}}$ 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. Thus, we get the following result.

Proposition 1 (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 $\mathsf{E}^{\text{in}} = \left(\mathsf{Enc}^{\text{in}}, \mathsf{Dec}^{\text{in}}\right)$ 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 $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{Shallow}\left(7, O\left(\hat{n}^4 (\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$. 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 $\left|\hat{C}\right| = O\left(|\mathbb{F}| \cdot \hat{n}^5 (\mathsf{S}(n)) \cdot |C|^2\right)$.*

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 9 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 9 (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).

The field elements of $\mathbb{F}$, and the arithmetic operations over $\mathbb{F}$ that are used by the arithmetic relaxed LRCC (Construction 9) will be represented using bit strings and boolean operations, respectively.

Representing field elements as bit strings. We can use any 1:1 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}.

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 Construction 11), 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 that will guarantee that the inputs to the compiled circuit encode binary strings.

Definition 7 (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 10 *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 $z \in \{0,1\}$.*

Construction 11 (SAT-respecting relaxed LRCC) *Let $E_b : \mathbb{F}_3 \to \{0,1\}^2$ such that $E_b(0) = 00$, $E_b(1) = 01$, and $E_b(2) = 11$, and let $E_b^{-1} : \{0,1\}^2 \to \mathbb{F}_3$ such that $E_b^{-1}(00) = 0$, $E_b^{-1}(01) = E_b^{-1}(10) = 1$, and $E_b^{-1}(11) = 2$. Let T' be an algorithm transforming boolean circuits into arithmetic circuits over $\mathbb{F}_3$, and $(\mathsf{Comp}, \mathsf{E} = (\mathsf{Enc}, \mathsf{Dec}))$ be the circuit compiler over $\mathbb{F}_3$ of Construction 9. The circuit compiler over $\mathbb{F}_2$ is $\left(\mathsf{Comp}^b, \mathsf{E}^b = \left(\mathsf{Enc}^b, \mathsf{Dec}^b\right)\right)$, 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 the 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 a $\vee$ 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 step 2.)*

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 12 *$\hat{C}_b(\hat{x}) \in \{0,1\}$ 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 $C'(x) = C(x) = 1$.*

In the full version, we prove that if Construction 9 is a SAT-respecting relaxed-LRCC over $\mathbb{F}_3$, then so is Construction 11 (over $\mathbb{F}_2$), against a somewhat-weaker leakage family. The leakage family is weaker because relaxed leakage-resilience is proved by reduction to the relaxed leakage-resilience of Construction 9 (the leakage function in the reduction, given the wire values of the arithmetic compiled circuit, generate the internal wires emulating these operations using boolean operations). Formally, we obtained the following.

Proposition 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 $\mathcal{L}_{\mathsf{E}} = \mathcal{L} \circ \mathsf{ShallowB}\left(33, O\left(\hat{n}^5(\mathsf{S}(n)) \cdot \mathsf{S}(n)^2\right)\right)$. 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, $\left|\hat{C}_b\right| = O\left(\hat{n}^5(\mathsf{S}(n))\,|C|^2\right)$.*

Taking E^{in} to be the parity encoding in the previous proposition, and using a result of Håstad [16] that AC^0 circuits (i.e., constant-depth and polynomial-sized boolean circuits with unbounded fan-in $\wedge, \vee$ and $\neg$ gates) cannot distinguish parity encodings of 0 and 1, we obtain an LRCC against AC^0-leakage. (We note that the compiler can also be made to withstand leakage that outputs more than one bit, using a result of Dubrov and Ishai [9]. The details of this construction, and the proof of Corollary 2, are deferred to the full version.)

Corollary 2. *There exists a SAT-respecting $\left(\mathsf{AC}^0, \mathsf{negl}(n), \mathsf{poly}(n)\right)$-relaxed-LRCC over $\mathbb{F}_2$.*

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

Recall that AC^0 denotes the class of constant-depth, polynomial-sized boolean circuits over unbounded fan-in and fan-out $\wedge, \vee, \neg$ gates. In this section we describe a SAT-respecting circuit-compiler withstanding leakage computed by AC^0 circuits, augmented with a sublinear number of $\oplus$ gates of unbounded fan-in and fan-out. Concretely, we use Construction 11, 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 [22], 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}. (This reduction is non-trivial and appears in Appendix A.) Using the leakage-indistinguishability of E^{in}, we prove the existence of a circuit compiler withstanding leakage from AC^0 circuits that have several output bits and are augmented with a sublinear number of $\oplus$ gates. (The proof appears in the full version.)

Theorem 13. *For input length parameter n, leakage length bound $\hat{n} = \hat{n}(n)$, size bound $s = s(n)$, output length bound $m = m(n)$, parity gate bound $t = t(n)$, 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 $s' = s'(n)$, there exists a polynomial $l(n)$, such that there exists a SAT-respecting $\left(\mathcal{L}^{m}_{l,d,l^c,\oplus t}, 2^{-n^c}, s'(n)\right)$-relaxed LRCC over $\mathbb{F}_2$, which on input a circuit $C : \{0,1\}^n \rightarrow \{0,1\}$ of size $|C| \leq s'(n)$ outputs a circuit $\hat{C}$ of size $|\hat{C}| \leq l(n)$.*

4 WIPCPs and CZKPCPs

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.

WIPCPs. A probabilistic proof system is a WIPCP for an NP-relation $\mathcal{R} = \mathcal{R}(x,w)$ if it satisfies the following. First, when given $x \in L_{\mathcal{R}}$, and oracle access to an honestly generated proof, the verifier accepts with probability 1 (this is called *completeness*). Second, given $x \notin L_{\mathcal{R}}$, the verifier rejects except with some probability ϵ_S, *regardless* of its "proof" oracle (this is called *ϵ_S-soundness*). 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) (this is called *$(\epsilon_{\text{ZK}}, q^*)$-WI*). A WIPCP is a *non-adaptive WIPCP (NA-WIPCP)* system for a relation $\mathcal{R} = \mathcal{R}(x,w)$, if the *honest* verifier is non-adaptive. In the following, we denote by $\mathsf{NA-WIPCP}[r,q,q^*,\epsilon_S,\epsilon_{\text{ZK}},\ell]$ 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 $O(r)$ coins, queries $O(q)$ proof bits, and rejects false claims except with probability at most ϵ_S. We use $\mathsf{PCP}[r,q,\epsilon,\ell]$ to denote the class of NP-languages admitting a standard (i.e., non-WI) PCP system with the same properties, and write $\mathcal{R} \in \mathsf{PCP}[r,q,\epsilon,\ell]$ to denote that $L_{\mathcal{R}} \in \mathsf{PCP}[r,q,\epsilon,\ell]$. We denote $\mathsf{NA-WIPCP} := \mathsf{NA-WIPCP}[\mathsf{poly}\log n, \mathsf{poly}\log n, \mathsf{poly}(n), \mathsf{negl}(n), \mathsf{negl}(n), \mathsf{poly}(n)]$.

We describe a transformation from PCPs to NA-WIPCPs, which can be applied to any PCP system in which the proof is obtained from the witness through an "easy" function (we formalize this notion below). Recall that a standard PCP π can be generated from the wire values $[C_{\mathcal{R}}, (x, w)]$ of the verification circuit $C_{\mathcal{R}}$ of the relation, on input x and witness w. If the function f taking $[C_{\mathcal{R}}, (x, w)]$ to π is in a function class $\mathcal{L}$, then the system can be made WI as follows. The prover and verifier both compile $C_{\mathcal{R}}(x, \cdot)$ (i.e., $C_{\mathcal{R}}$ with x hard-wired into it) into a SAT-respecting circuit $\hat{C}_{\mathcal{R}}$ that is relaxed leakage-resilient against $\mathcal{L}$. The prover then samples a random encoding $\hat{w}$ of w, and generates the PCP $\pi = f\left[\hat{C}_{\mathcal{R}}, \hat{w}\right]$. The verifier probabilistically verifies that $\hat{C}_{\mathcal{R}}$ 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 $\hat{C}_{\mathcal{R}}$. This gives the following result. (The detailed construction, and the proof of Proposition 3, appear in the full version.)

Proposition 3. *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 $|\mathsf{Comp(C)}| \leq g(|C|)$;*
- *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 $\mathcal{R} \in \mathsf{NA-WIPCP}[r(t), q(t), q^*, \epsilon_S, 2\epsilon_{\mathrm{ZK}}, \ell(t)]$, where $t = O\left(g\left(\left|C^{\mathcal{R}}\right|\right)\right)$, and WI holds against non-adaptive *verifiers.*

In the full version we use techniques of [7] to generalize the WI property of Proposition 3 to *adaptive* verifiers, while increasing the statistical distance of the WI by a multiplicative factor of roughly ℓ^{q^*} (all other parameters remain unchanged). Then, we prove that the PCP system of [2] for 3SAT has the property that every proof bit is generated from the NP-witness by an AC^0 circuit, augmented with "few" $\oplus$ gates. Theorem 1 follows by combining these two results with Theorem 13.

CZKPCPs in the CRS model. A probabilistic proof system is a CZKPCP in the CRS model for an NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ 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 $x \in L_{\mathcal{R}}$, $\mathsf{Sim}(x)$ 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 (this is called *computational ZK (CZK)*). Similar to NA-WIPCPs, a CZKPCP system is *non-adaptive (NA-CZKPCP)* if the honest verifier is non-adaptive. Applying the techniques of [12] to Proposition 3, we obtain a general transformation from NA-WIPCPs to NA-CZKPCPs, and Corollary 1 follows by using the NA-WIPCP of Theorem 1 (see the full version for details).

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].

The (im)possibility of SAT-respecting non-relaxed LRCCs. Known constructions of LRCCs withstanding global leakage [18, 11, 23] guarantee a *universal* simulation property, in the sense that the simulator generates the simulated wire values *without* knowing the identity of the leakage function; and these values are guaranteed to be indistinguishable from the actual wire values, *for every leakage function in the leakage class*. Consequently, our construction (which is based on the LRCC of [11]), also guarantees this universal simulation property. Our general transformation from SAT-respecting relaxed LRCCs to WIPCPs can also be applied to a SAT-respecting *non-relaxed* LRCC, in which case we would get *ZK*PCP for all NP, with a *universal PPT* simulator that generates a simulated proof *without seeing the queries of the verifier*. This simulator can be used to decide the NP-language, so the existence of SAT-respecting LRCCs with a universal simulator would imply that $\mathsf{NP} \subseteq \mathsf{BPP}$. (See the full version for additional details.) We note that our transformation of Section 4 does *not* require the LRCC simulator to be universal. However, the construction of (SAT-respecting) non-relaxed LRCCs with a non-universal simulator would require developing new techniques for constructing LRCCs.

4.1 Distributed ZK and WI 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*. We note that P has input (x, w), V has input x, and the servers have no input. 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 [20, 17]. (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.

Distributed CZK\WI proof systems. An m-distributed proof system is a (t, m)-distributed ZK proof system for an NP-relation $\mathcal{R}$ if it satisfies the following properties. First, if all parties are honest and $(x, w) \in \mathcal{R}$ then V accepts

x with probability 1 (the *correctness* property). Second, if $x \notin L_{\mathcal{R}}$ then V rejects x except with negligible probability, even if the prover is corrupted and colludes with at most t corrupted servers (the *soundness* property). Thirdly, for every adversary $\mathcal{A}$ corrupting V and $t' \leq t$ servers there exists a PPT simulator Sim such that for every $x \in L_{\mathcal{R}}$, $\mathsf{Sim}(x)$ is computationally indistinguishable from the the view of $\mathcal{A}$ in the protocol execution, when it has input x. This notion can be naturally relaxed to WI, or CZK in the CRS model.

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 [19] (a more detailed description and analysis of these distributed proof systems appears in the full version). Thus, we obtain the following results.

Theorem 14 (Sublinear distributed WI proofs). *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 WI proof system, where* n *is the input length.*

Theorem 15 (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.*

These constructions *crucially* rely on the *non-adaptivity* of the honest WIPCP (respectively, CZKPCP) verifier (otherwise we would need at least 4 rounds, since rounds cannot be compressed). Moreover, the verifier may collude with a subset of servers, so the PCP should be WI (respectively, CZK) against *malicious* verifiers.

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. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.

23. 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.
24. 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.
25. 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.

A A Leakage-Indistinguishable Encoding Scheme

In this section we define the encoding scheme that is used to prove Theorem 13, and use correlation bounds of [22] to show that it is leakage-indistinguishable against leakage computable by AC^0 circuits, augmented with few $\oplus$ gates.

Notation 16 *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 8. *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,*[7] *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.*

The leakage class we consider is "AC^0, augmented with few $\oplus$ gates":

Definition 9 ($\mathcal{L}_{n,d,s,\oplus t}^m$ 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 $C : \{0,1\}^n \rightarrow \{0,1\}$ 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 $f : \{0,1\}^n \rightarrow \{0,1\}^m$, let $f_i(x_1, ..., x_n), i \in [m]$ denote the i'th output bit of f. We use the following notation: $\mathcal{L}_{n,d,s,\oplus t}^m = \{f : \{0,1\}^n \rightarrow \{0,1\}^m : \forall 1 \leq i \leq m, f_i \in \mathcal{L}_{n,d,s,\oplus t}\}$, 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 [22, 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

[7] Enc_3 can be computed efficiently by repeating the following procedure n^2 times. Pick $v \in \{0,1\}^n$ 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.

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 10 (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 17 (MOD_s function) *Let $s \in \mathbb{N}$. The function $\mathrm{MOD}_s^n : \{0,1\}^{3n} \to \{0,1\}$ is defined as $\mathrm{MOD}_s(x) = 0$ 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 18 ([22], 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 $n \geq n_0$, $\mathsf{Corr}_{\mathcal{D}_3^n}\left(\mathcal{L}_{3n,d,n^l,\oplus n^\epsilon}, \mathrm{MOD}_3^n\right) \leq 2^{-n^c}$, 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 18 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 8. Formally:

Corollary 3. *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 8 is $\left(\mathcal{L}_{3n,d,n^l,\oplus n^\epsilon}, 2^{-n^c}\right)$-leakage-indistinguishable.*

We proceed to prove Corollary 3 in two steps. First, we show that Theorem 18 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 1. *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 $\mathsf{Corr}_{\mathcal{D}_3^n}(\mathcal{G}, \mathrm{MOD}_3^n) \leq \epsilon$ 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 18.*

Proof. Let $g \in \mathcal{G}$. We first establish the connection between the probability $p_g := \Pr_{x\leftarrow\mathcal{D}_3^n}[g(x) = \mathrm{MOD}_3^n(x)]$ 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,

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

Therefore, we get:

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

□

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 2. *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 $n \ge n_0$, $U_0^n, U_{1,2}^n$ 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 $n \ge n_0'$, U_0^n, U_1^n are $\left(\mathcal{L}_{3n,d,s,\oplus t}, \sqrt{7\epsilon}\right)$-leakage-indistinguishable.*

In the following proofs, we use the following notation, and the following observation regarding the connection between U_1^n, U_2^n and $U_{1,2}^n$.

Notation 19 *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 20 *For every $n \in \mathbb{N}$, and every function $g : \{0,1\}^{3n} \to \{0,1\}$, by the law of total probability, and since $\Pr_{x \leftarrow U_{1,2}^n}\left[x \in \mathcal{S}_1^n\right] = \Pr_{x \leftarrow U_{1,2}^n}\left[x \in \mathcal{S}_2^n\right] = \frac{1}{2}$,*

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

Proof (of Lemma 2). 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 $\epsilon' = \epsilon'(n) > \sqrt{7\epsilon}$ 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 $p_\gamma^n := \Pr_{x \leftarrow U_\gamma^n}\left[\hat{D}_n(x) = 1\right]$. 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 $\mathcal{N}$, $p_2^n \leq p_0^n$; or $p_2^n > p_0^n$ for infinitely many n's in $\mathcal{N}$. In the first case, $\hat{D}$ has advantage at least $\frac{\epsilon'}{2} > \frac{\sqrt{7\epsilon}}{2} > \frac{\sqrt{4\epsilon}}{2} \geq^{\epsilon \leq 1} \epsilon$ 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 20,

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

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

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

Therefore, only the second case remains, and Lemma 3 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 in $\mathcal{L}_{6n,d+1,2s+1,\oplus 2t}$ with advantage at least $\frac{(\epsilon')^2}{6} + E(n) > \frac{\left(\sqrt{7\epsilon}\right)^2}{6} + E(n) = \epsilon + \frac{\epsilon + E(n)}{6}$, where $E(n) = O\left(2^{-3n}\right)$. Recall that $\epsilon = 2^{-n^c}$, so $E(n) = o(\epsilon)$, and let $n' \in \mathbb{N}$ such that for every $n \geq n'$, $|E(n)| \leq \epsilon$ (notice that $E(n)$ may be negative). Then for every $n \geq \max\{n', \hat{n}_0\}$ in $\mathcal{N}$ such that $p_2^n > p_0^2 \geq p_1^n + \epsilon'$ (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, if $p_0^n \geq p_1^n + \epsilon'$ for infinitely many n's in $\mathcal{N}$, then U_0^n, U_1^n are $\left(\mathcal{L}_{3n,d,s,\oplus t}, \sqrt{7\epsilon}\right)$-distinguishable only for finitely many n's.

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 in the same manner as before. □

We now prove the lemma used in the proof of Lemma 2, 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 uses the distributions $U_0^{2n}, U_{1,2}^{2n}$ over $6n$*-bit vectors*, and distinguishers over $3n$*-bit vectors.*

Lemma 3. *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 5, $\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y) = 1\right] = \frac{(p_0^n)^2 + 2p_1^n p_2^n}{3} + E_0(n) + E'_0(n) \cdot p_2^n$, where $E_0(n), E'_0(n)$ are error terms, and $|E_0(n)|, |E'_0(n)| = O\left(2^{-3n}\right)$. Using Lemma 6, $\Pr_{(x,y)\leftarrow U_{1,2}^{2n}}\left[D'_n(x,y) = 1\right] = \frac{2p_0^n p_1^n + (p_1^n)^2 + 2p_0^n p_2^n + (p_2^n)^2}{6} + E_{1,2}(n) + E'_{1,2}(n) \cdot p_2^n + E''_{1,2}(n) \cdot (p_2^n)^2$, where $E_{1,2}(n), E'_{1,2}(n), E''_{1,2}(n)$ are error terms, and $|E_{1,2}(n)|, \left|E'_{1,2}(n)\right|, \left|E''_{1,2}(n)\right| = O\left(2^{-3n}\right)$. Therefore,

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

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

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

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 $\mathcal{E}_{D'_n}|_{p_2^n = p_0^n} = \frac{(p_0^n - p_1^n)^2}{6} + E(n) + E'(n) \cdot p_0^n + E''(n) \cdot (p_0^n)^2 \geq \frac{\epsilon^2}{6} + E(n) + E'(n) \cdot p_0^n + E''(n) \cdot (p_0^n)^2 =^{p_0^n \in (0,1)} \frac{\epsilon^2}{6} + E^+(n)$, 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. We will need the following result about the values of k_0^n, k_1^n, k_2^n. (The proof, which is by induction and uses Observation 20, appears in the full version.)

Lemma 4. *Let $n \in \mathbb{N}$. Then $k_1^n = k_2^n = \frac{2^{3n} + (-1)^{n-1}}{3}$, and $k_0^n = \frac{2^{3n} + 2\cdot(-1)^n}{3}$.*

Lemma 5. *Let $D'_n, p_0^n, p_1^n, p_2^n$ be as defined in the proof of Lemma 3. Then $\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y) = 1\right] = \frac{(p_0^n)^2 + 2p_1^n p_2^n}{3} + E_0(n) + E'_0(n) \cdot p_2^n$, where $E_0(n), E'_0(n)$ are error terms, and $|E_0(n)|, |E'_0(n)| = O\left(2^{-3n}\right)$.*

Proof. Since

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

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

$$\Pr_{(x,y)\leftarrow U_0^{2n}}\left[D'_n(x,y)=1 | x,y \in \mathcal{S}_0^n\right] \cdot \Pr_{(x,y)\leftarrow U_0^{2n}}\left[x,y \in \mathcal{S}_0^n\right] +$$

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

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

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

If n is even, then by Lemma 4: $k_0^n = |\mathcal{S}_0^n| = \frac{2^{3n}+2}{3}$; $k_0^{2n} = \left|\mathcal{S}_0^{2n}\right| = \frac{2^{6n}+2}{3}$; and $k_1^n = |\mathcal{S}_1^n| = \frac{2^{3n}-1}{3}$. Therefore,

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

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

Otherwise, n is odd, and by Lemma 4: $k_0^n = |\mathcal{S}_0^n| = \frac{2^{3n}-2}{3}$; $k_0^{2n} = \left|\mathcal{S}_0^{2n}\right| = \frac{2^{6n}+2}{3}$; and $k_1^n = |\mathcal{S}_1^n| = \frac{2^{3n}+1}{3}$. Similar calculations give:

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

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

The proof of the following lemma is similar to the proof of Lemma 5, and appears in the full version.

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

Next, we prove that if U_0^n, U_1^n are leakage-indistinguishable against some family of leakage functions, then E_3 is leakage indistinguishable against a slightly weaker family of leakage functions.

Lemma 7. *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 exist infinitely many n's and for each a distinguisher $D_n \in \mathcal{L}_{3n,d-1,s-3n,\oplus t}$ that achieves advantage $\epsilon' > \epsilon$ in distinguishing between the distributions $\mathsf{Enc}_3(0, 1^n)$, $\mathsf{Enc}_3(2, 1^n)$. For every such 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, for every such n, $|\Pr[D_n'(\mathsf{Enc}(0, 1^n)) = 1] - \Pr[D_n'(\mathsf{Enc}(1, 1^n)) = 1]| = |\Pr[D_n(\mathsf{Enc}(0, 1^n)) = 1] - \Pr[D_n(\mathsf{Enc}(2, 1^n)) = 1]| = \epsilon' > \epsilon$, 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.

Proof (of Corollary 3). Let $d' = d + 2$, let ϵ, c be the constants for which Theorem 18 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 18 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 2 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 18 guarantees that $\mathsf{Corr}_{\mathcal{D}_3^n}\left(\mathcal{L}_{3n,d+2,2(n^l+3n)+1,\oplus 2n^{\epsilon'}}, \mathrm{MOD}_3^n\right) \leq 2^{-n^c}$ (since $n \geq n_0$ and $n \geq n_0'$). By Lemma 1, 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 $\left(\mathcal{L}_{6n,d+2,2(n^l+3n)+1,\oplus 2n^{\epsilon'}}, 2^{-n^c}\right)$-leakage-indistinguishable. By Lemma 2, 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 7, $\mathsf{E}_3(\cdot, 1^n)$ 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 □