entropy

MDPI

Article

Shielding Probabilistically Checkable Proofs: Zero-Knowledge PCPs from Leakage Resilience

Mor Weiss

The Alexander Kofkin Faculty of Engineering, Bar-Ilan University, Ramat-Gan 5290002, Israel; mor.weiss@biu.ac.il

Abstract: Probabilistically Checkable Proofs (PCPs) allows a randomized verifier, with oracle access to a purported proof, to probabilistically verify an input statement of the form "$x \in \mathcal{L}$" by querying only a few proof bits. *Zero-Knowledge* PCPs (ZK-PCPs) enhance standard PCPs to additionally guarantee that the view of any (possibly malicious) verifier querying a bounded number of proof bits can be efficiently simulated up to a small statistical distance. The first ZK-PCP construction of Kilian, Petrank and Tardos (STOC 1997), and following constructions employing similar techniques, necessitate that the *honest* verifier makes several rounds of queries to the proof. This undesirable property, which is inherent to their technique, translates into increased round complexity in cryptographic applications of ZK-PCPs. We survey two recent ZK-PCP constructions—due to Ishai, Yang and Weiss (TCC 2016-A), and Hazay, Venkitasubramaniam and Weiss (ITC 2021)—in which the honest verifier makes a *single* round of queries to the proof. Both constructions use entirely different techniques compared to previous ZK-PCP constructions, by showing connections to the seemingly-unrelated notion of *leakage resilience*. These constructions are incomparable to previous ZK-PCP constructions: while on the one hand the honest verifier only makes a single round of queries to the proof, these ZK-PCPs either obtain a smaller (polynomial) ratio between the query complexity of the honest and malicious verifiers or obtain a weaker ZK guarantee in which the ZK simulator is not necessarily efficient.

Keywords: Probabilistically Checkable Proofs; zero knowledge; leakage resilience

Citation: Weiss, M. Shielding Probabilistically Checkable Proofs: Zero-Knowledge PCPs from Leakage Resilience. *Entropy* **2022**, *24*, 970. https://doi.org/10.3390/e24070970

Academic Editor: Hemanta K. Maji

Received: 13 March 2022
Accepted: 5 July 2022
Published: 13 July 2022

1. Introduction

Proofs are a cornerstone of cryptography. They are an essential component of many cryptographic systems, guaranteeing correct execution in the presence of mutually-distrusting parties. Their applications range from mundane tasks such as proving one's identity when signing into an email account, to general tasks such as proving honest behaviour in distributed systems, i.e., attesting that one had followed its prescribed algorithm.

Due to their centrality, in the past decades a long line of works have studied the notion of proofs, extending it far beyond the traditional notion to also allow for interaction between a *prover* $\mathcal{P}$ and a *verifier* $\mathcal{V}$, as well as randomization. Various variants have arisen, depending on different properties of the system, such as the amount and type of communication allowed between the prover and verifier, their computational powers, and the type of randomness (for example, whether the verifier's random coin tosses are private or public). These advances have reshaped theoretical computer science.

Our focus is on *Probabilistically Checkable Proofs (PCPs)* [1,2] augmented with a cryptographic *Zero-Knowledge (ZK)* property that is very useful when such proofs are used in cryptographic applications. These zero-knowledge PCPs combine aspects of (interactive) zero-knowledge proofs and PCPs, concepts which we now discuss.

Probabilistically Checkable Proofs (PCPs). A *Probabilistically Checkable Proof (PCP)* for a language $\mathcal{L}$ allows a randomized verifier $\mathcal{V}$ to verify a statement of the form "$x \in \mathcal{L}$", while querying only few bits of an oracle proof π that was generated by a prover $\mathcal{P}$. (We note that the prover entity is not traditionally included as part of a PCP system. However, explicitly introducing this entity will be useful when PCPs are used for cryptographic applications, which necessitate efficient proof generation.) More specifically, $\mathcal{V}$ is given x as input, and $\mathcal{P}$—who is usually required to be efficient—may be given additional information needed to efficiently generate the proof, e.g., a witness in the case of an NP language. More generally,

for an NP-language $\mathcal{L}$ the PCP will usually be designed for some specific NP-relation $\mathcal{R}(x,w)$ associated with $\mathcal{L}$. In standard PCPs the prover is deterministic, whereas the verifier is randomized. (This is necessary since the verifier only queries few proof bits.) The verifier accepts true claims with probability 1, whereas the *soundness error*—namely the probability that a false claim is accepted—is small, regardless of the purported proof π^* given to the verifier. The celebrated PCP theorem [1–3] asserts that any NP language has a PCP system with soundness error $1/2$ in which the verifier $\mathcal{V}$ reads only a *constant* number of proof bits. Moreover, $\mathcal{V}$ is *non-adaptive*, namely its queries are determined solely by its randomness.

Interactive and Zero-Knowledge Proofs. *Interactive Proofs (IPs)* [4] are a different kind of proof system, in which the efficient probabilistic verifier $\mathcal{V}$ interacts with a prover $\mathcal{P}$, the goal of which is to convince $\mathcal{V}$ that $x \in \mathcal{L}$ for a joint input x. Such proofs are extremely powerful, compared to classical proofs: any PSPACE language has an interactive proof with a polynomial-time verifier [5], whereas classic proofs with a polynomial-time verifier only exist for NP languages. *Zero-Knowledge (ZK) Proofs* [4] are an important and useful generalization of IPs for NP, enhancing them to also guarantee privacy of the NP witness. These proofs carry no extra knowledge other than being convincing, in the sense that any information $\mathcal{V}$ can infer from its interaction with $\mathcal{P}$, it could have (efficiently) computed given only the input x. This holds even in the presence of a *malicious* verifier $\mathcal{V}^*$, namely a verifier who arbitrarily deviates from the protocol. It is important to note that while proofs are prominently used to protect *the verifier* (guaranteeing it would reject false claims), ZK is designed to protect the *prover*, guaranteeing that its private information (the NP witness, in this case) remains entirely hidden.

Zero-Knowledge PCPs. *Zero-Knowledge PCPs (ZK-PCPs)* [6] are proofs systems that combine the advantages of PCPs and ZK proofs. These are PCPs with the additional guarantee that the view—encompassing all the knowledge which the verifier posses of the interaction—of any (possibly malicious) verifier $\mathcal{V}^*$ who queries an *a-priori bounded* number of proof bits can be efficiently simulated given only the input, up to a small statistical distance.

ZK-PCPs differ from traditional PCPs in several respects. First, whereas the PCP prover is traditionally deterministic, in ZK-PCPs the proof is randomized, and this is inherent to obtaining ZK. Second, ZK-PCPs are used to protect the *prover's* private information (e.g., an NP witness) against *malicious verifiers*. More specifically, the system is associated with an a-priori query bound q^*, where ZK holds only against verifiers $\mathcal{V}^*$ who query at most q^* proof bits. We stress that this is the *only* restriction on $\mathcal{V}^*$, and no further assumptions or limitations are made on its computational power or the manner in which it operates. It is important to note that bounding the query complexity is inherent in systems with efficient provers—an essential requirement for such systems to be useful for cryptographic applications. Indeed, the proof has polynomial length len, so any (efficient) verifier running in time len could read the *entire* proof, thus necessarily learning some information about the witness. Finally, we note that ZK against query-bounded verifiers is a stronger guarantee than *Honest-Verifier ZK (HVZK)*, namely ZK only against the *honest* verifier. This is because the honest verifier is always query bounded.

The models also differ in the main parameters of interest. Specifically, for PCPs these consist of the randomness and query complexities of the verifier (i.e., the number of coins it tosses and the number of queries it makes to the proof), which also determine the proof length. Moreover, it is standard to consider a constant soundness error—with a verifier that queries a constant number of proof bits—since this setting has strong connections to proving hardness of approximation results. We currently have PCPs, the length of which is a quasi-linear length in the witness length, with a constant soundness error (which can be amplified through repetition), and a non-adaptive honest verifier that queries a constant number of proof bits [3,7].

On the other hand, for ZK-PCPs we desire the soundness error to be negligible (in the input length, or some security parameter) as is standard in cryptographic systems, and thus the query complexity is necessarily polylogarithmic. The query bound q^* on malicious verifiers is another important parameter of the system, and would ideally be much larger than the query complexity of the honest verifier, e.g., polynomial in the input length. Thus, the *query gap* between the query complexity needed to verify the proof, and the number of

queries a malicious verifier can make without violating ZK, would be exponential. Finally, we would like the *honest* verifier to be non-adaptive, namely to make a single round of queries to the proof. This should be contrasted with adaptive verifiers the queries of which might depend on the oracle answers to previous queries, and who therefore necessarily make several rounds of queries to the proof. As we will shortly explain, whereas the verifier in traditional PCPs is non-adaptive, the honest verifier in certain ZK-PCP constructions is (inherently) adaptive. We note that similar to traditional PCPs, the proof length is also of interest.

The focus on these parameters in ZK-PCP constructions stems from their effect on the properties and parameters of cryptographic systems using ZK-PCPs. Specifically, the query complexity and adaptivity of the honest verifier translates into communication and round complexities; the query-bound on malicious verifiers corresponds to the privacy guarantee of the resultant system; and in distributed proof systems (as in, e.g., [8]) the proof length and query bound translate into the total number of parties and the number of corrupted parties, respectively.

ZK-PCP Constructions. The first ZK-PCP for NP, due to Kilian, Petrank and Tardos [6], obtained a negligible soundness error with an honest verifier that queries $q = \mathsf{polylog}(|x|)$ proof bits, and ZK against verifiers making $q^* = p(|x|)$ queries to the proof for a *fixed* polynomial p that is much smaller than the proof length, but is much larger than q. (Earlier constructions, e.g., [9], obtained only limited ZK guarantees such as HVZK). Later works [10,11] simplified the system, making it more modular, and also generalized it to other proof models (specifically, PCPs of proximity with zero knowledge [11]). While obtaining a desirable exponential query gap, the honest verifier in all these constructions is *adaptive*, namely it makes several rounds of queries to the proof, a severe limitation when the system is used in cryptographic applications. Unfortunately, the honest verifier's adaptivity is inherent to these constructions, as we now explain. (We note that another line of works obtain non-adaptive verification by "pushing" adaptivity to the prover side; see Section 1.3. Having adaptive proof generation has similar disadvantages to having adaptive verification.)

These ZK-PCP constructions follow the blueprint of [6], who show a 2-step compiler from a standard non-ZK PCP into a ZK-PCP. In the first step, the PCP is transformed into a PCP with HVZK—a weak ZK guarantee that holds only for the *honest verifier*. In the second step, HVZK is "boosted" into full-fledged ZK, against any (possibly malicious) query-bounded verifier $\mathcal{V}^*$. This is obtained by forcing—through modifications made to the proof—$\mathcal{V}^*$'s queries to be distributed similarly to the queries of the honest verifier $\mathcal{V}$. This restriction on the verifier's queries is imposed by combining an information–theoretic analogue of a standard cryptographic commitment, called a "locking scheme" [6], with a modified version of the PCP obtained in the first step. The proof generated in this second step requires adaptive verification, due to the structure of the modified version of the proof, as well as the use of locking schemes. We note that this description of $\mathcal{V}^*$'s queries as being distributed "similarly" to the queries of $\mathcal{V}$ is in fact a gross over-simplification—for example, $\mathcal{V}^*$ can query many more proof bits compared to $\mathcal{V}$. Somewhat more accurately, these works effectively force $\mathcal{V}^*$'s queries to be distributed similarly to a "repeated" version of $\mathcal{V}$ (obtained by emulating $\mathcal{V}$ multiple times with independent random coins). We refer the interested reader to [11,12] for more details. We stress that these works do *not* make any *assumptions* on the query pattern of $\mathcal{V}^*$, but rather by appropriately constructing the proof they guarantee that *any* query pattern will be "harmless" in the sense that it reveals no information about the NP witness.

The second step can be very roughly (and somewhat inaccurately) illustrated through the following example: Alice is trying to locate a particular CD cd in Bob's CD collection, which Bob has mixed in the following way: (1) the CDs were taken out of their cases and randomly placed back into the cases (where Bob knows which case contains which CD), and then (2) each CD case was locked in a transparent box. To get cd, Alice must first ask Bob in which case he put cd. Once this is known, she can locate the box in which this CD case is, but still needs to ask Bob for the key which unlocks the box. Since Alice cannot predict in which case cd is, she must first wait for Bob's answer to her first query, before making her second query. (The PCP-version of this example will have Bob somehow write

down the list of pairs (CD, CD case), as well as the keys, in the oracle proof.) In the ZK-PCP, step (1) of randomly mixing the CDs corresponds to the modifications performed to the proof to guarantee that $\mathcal{V}^*$'s queries are "harmless", specifically that $\mathcal{V}^*$ cannot "cherry pick" specific locations in the HVZK PCP generated in the first step of the compiler. Moreover, step (2) of locking CDs in boxes corresponds to locking proof symbols in locking schemes, and even the process of unlocking (given the key) in itself is adaptive.

The cost of ZK in these ZK-PCPs is high: it incurs adaptive verification, even if the underlying PCP can be verified non-adaptively, which is indeed the case for traditional (non-ZK) PCPs. This naturally gives rise to the following research goal:

Design ZK-PCPs with a non-adaptive honest verifier, and ZK against malicious query-bounded verifiers.

Obtaining non-adaptive verification is motivated by the goal of matching the parameters of non-ZK PCPs, as well as by cryptographic applications of ZK-PCPs, in which adaptive verification translates into increased complexity of the resultant system.

The question of designing non-adaptive ZK-PCPs had remained open for nearly 20 years, until Ishai, Weiss and Yang [8] gave the first construction of a non-adaptive ZK-PCP, which was followed by the non-adaptive ZK-PCP of Hazay, Venkitasubramaniam and Weiss [13]. The focus of this survey is on describing and comparing these constructions.

1.1. Non-Adaptive ZK-PCPs

We survey two recent works [8,13] that construct ZK-PCPs for NP with a *non-adaptive honest verifier*, obtained through a novel connection to the seemingly unrelated field of *leakage-resilient cryptography*. These constructions differ drastically from the ZK-PCP constructions described above. This is not surprising, since adaptive verification is inherent to the latter, so obtaining non-adaptive verification necessitates an entirely new approach.

Malicious Verifiers Through the Leakage-Resilience Lens. Recall that the ZK-PCPs of [6,10,11] are obtained from a PCP with weak ZK guarantees (specifically, HVZK) by effectively *restricting the malicious verifier* , i.e., forcing its queries to be distributed similarly to the queries of (multiple independent copies of) the honest verifier. The ZK-PCPs of [8,13] take a different approach: instead of forcing a certain structure on $\mathcal{V}^*$'s queries, they classify the *type of information* which an arbitrary query-bounded $\mathcal{V}^*$ obtains by querying the proof, and modify the proof to guarantee this type of information reveals nothing about the underlying NP witness. Their insight is that the partial information obtained by deviating from the honest verifier's query pattern constitutes *leakage* on the proof and consequently, on the underlying witness. (In a broader cryptographic context, *leakage* roughly refers to the information an adversary obtains by deviating from the assumed adversarial model for which the system was designed. For example, so-called "side channel" attacks—such as measuring the power consumption of an object—exploit adversarial capabilities which were not taken into account when designing the system (adversarial and attack models traditionally disregard such information, that is obtained from the *physical implementation*, and is not part of the more abstract model description). Similarly, a malicious verifier querying a PCP is capable of querying more—and different sets of—proof bits compared to the honest verifier for which the system was designed.) Accordingly, they employ tools from the leakage-resilience literature to protect the witness and proof.

The works of [8,13] differ in the method they use to protect against leakage. Hazay et al. [13] chose to protect *the proof* itself, whereas Ishai et al. [8] protect the *process of proof generation* from the witness. Put differently, the latter protect *computation* against leakage, for which they employ leakage-resilient *circuits*, whereas the former protect *information* (the proof, once it has been generated), by using an appropriate leakage-resilient *encoding*. Consequently, these works differ in their requirements from the underlying PCP, and in the parameters and properties of the resultant ZK-PCP. We now elaborate on these differences (see also Table 1).

Table 1. Comparison of Existing ZK-PCP constructions. Here, "Underlying LR primitive" refers to the type of building block used in the leakage-resilience based constructions; "WI" stands for witness-indistinguishable; "ZK quality" refers to the efficiency of the ZK simulator (which is efficient in ZK systems and inefficient in WI systems); "Query ratio" is q^*/q, where q^* is the bound on the query complexity of a malicious verifier (ZK holds against any verifier querying at most q^* proof bits), and q is the query complexity of the honest verifier (needed to achieve soundness); "Underlying PCP" describes the properties which the transformations needs the underlying PCP system to have; and "Honest verification" refers to the adaptivity of the honest verifier, where "NA" stands for non-adaptive.

	Underlying LR Primitive	ZK Quality	Query Ratio	Underlying PCP	Honest Verification
[8]	Circuits	WI	Exponential	Any standard PCP	Nonadaptive
[13]	Encodings	ZK	Square-root	ZK-PCP, large alphabet	Nonadaptive
[6,11]	—	ZK	Exponential	Any standard PCP	Adaptive

1.1.1. The ZK-PCPs of Hazay et al.

Hazay et al. [13] construct a non-adaptive ZK-PCP in which the ratio between the bound q^* on the query complexity of a malicious verifier, and the query complexity q of the honest verifier, is polynomial.

The leakage-resilient primitive they employ is a *Leakage-Resilient Encoding (LRE)*. Roughly, an LRE consists of a randomized efficient encoding procedure Enc mapping a bit string m to an encoding c, and an efficient deterministic decoder algorithm that given an encoding c of m, outputs m. The leakage-resilience guarantee is that for any pair m, m', if $c \leftarrow \mathsf{Enc}(m), c' \leftarrow \mathsf{Enc}(m')$, then any small subset of bits in c, c' are identically distributed. (They actually need a stronger leakage-resilience guarantee, see Section 3.1.) Non-explicit constructions of such encodings easily follow from the existence of linear error-correcting codes with sufficiently "good" parameters, such as random linear codes (see, e.g., [14]). In Section 3, we also describe an explicit construction of such encodings.

The leakage-resilience guarantee of an LRE is restricted to protecting the codeword *once it has been encoded*, and does not protect the encoding procedure itself (as opposed to leakage-resilient circuits, see below). Therefore to obtain PCPs with full-fledged ZK against arbitrary query-bounded verifiers, the underlying PCP should possess a zero-knowledge property, which is weaker than full-fledged ZK, but stronger than the HVZK property used in [6,10,11]. Specifically, they use a ZK-PCP variant *over a large alphabet*, which is much easier to obtain compared to full-fledged ZK for standard PCPs, in which the proof is binary.

A ZK-PCP variant over a large alphabet can be thought of as a (standard) PCP which is divided into "regions", each corresponding to a single symbol in the large alphabet, where ZK is guaranteed only as long as the verifier queries "full" regions (i.e., all bits in the region). Thus, a malicious verifier does not have to follow the honest verifier's query pattern, but ZK holds only against verifiers querying at most q^* full regions, for some a-priori bound q^*. Such ZK-PCPs are constructed from general secure multi-party computation protocols in [15].

Given a ZK-PCP variant over a large alphabet, [13] design an alphabet reduction that preserves ZK. (Naïve alphabet reduction techniques do not preserve ZK; see Section 3.) This reduction uses the underlying PCP $(\mathcal{P}', \mathcal{V}')$ as a black box, transforming it into a ZK-PCP $(\mathcal{P}, \mathcal{V})$ in which the proof is over bits. The high-level idea is to interpret each symbol of a proof π' generated by $\mathcal{P}'$ as a bit string, and encode it using the LRE. $\mathcal{V}$ then emulates $\mathcal{V}'$, answering an oracle query i by reading the entire encoding of the i'th symbol from its proof, decoding it, and then providing $\mathcal{V}'$ with the resultant symbol as the oracle answer.

Hazay et al. [13] then apply their alphabet reduction to the ZK-PCP variant of [15] to obtain the following (see Theorem 3 in Section 3 for the formal statement).

Informal Theorem 1. *There exists a constant $\epsilon \in (0,1)$ such that for any ZK parameter $q^* \in \mathbb{N}$ and any NP-language $\mathcal{L}$ there exists a ZK-PCP for $\mathcal{L}$ with ZK against q^*-query bounded verifiers, and a negligible soundness error with a non-adaptive honest verifier that queries $(q^*)^\epsilon$ proof bits.*

We note that the ZK-PCP system described in Informal Theorem 1 requires a tighter analysis of the ZK-PCP variant of [15], which [13] provide. See Section 3 and [13] for further details.

1.1.2. The ZK-PCPs of Ishai et al.

Ishai et al. [8] construct a non-adaptive ZK-PCP in which the ratio between the bound q^* on the query complexity of a malicious verifier, and the query complexity q of the honest verifier, is *exponential*, but ZK holds with an inefficient simulator.

Their starting point is a *standard* PCP with no zero-knowledge guarantees. Consequently, to obtain full-fledged ZK, they rely on a stronger leakage-resilient tool. To describe their construction, it would be easier to consider the NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ associated with the NP-language $\mathcal{L}$, and its corresponding verification circuit C. We will assume without loss of generality that a witness w for x has canonical form, namely it consists of the entire wire values of C given input x, w', for some w' such that $C(x, w') = 1$. (In particular, w' is the information that we wish to keep secret.) The PCP prover generates the proof from this canonical NP witness w. Therefore, each proof bit is an information bit on w, i.e., on the wire values of C when evaluated on x, w'. This can be thought of as *leakage* on the *computation* in C, the purpose of which is to reveal information about the secret input w' of C. While in general, leakage on the wire values of C might reveal information on the secret input w', there are tools to compile C into a *leakage-resilient* circuit $\widehat{C}$ that resists such leakage.

However, it is well known that one cannot protect circuits against *general* polynomial-time leakage [16]. Consequently, leakage-resilient circuits are associated with a restricted leakage class from which leakage functions can be chosen, and the circuit only resists leakage computed by a function from the class. The main observation of [8] is that every PCP system naturally has such a restricted class $\mathcal{LEAK}$ of leakage functions associated with it. Indeed, while $\mathcal{V}^*$ can choose which proof bits to query, it has no control over the *type of functions* applied to w to generate the proof bits—these functions are determined solely by the prover algorithm. Specifically, for every subset $\mathcal{I}$ of at most q^* indices in the proof, the corresponding leakage function $\ell_{\mathcal{I}} \in \mathcal{LEAK}$ applies the PCP prover function to w, then outputs the restriction of the proof to the indices in $\mathcal{I}$.

The main building block is therefore a *Leakage-Resilient Circuit compiler (LRCC)*—a compiler that transforms a given circuit C into a leakage-resilient circuit $\widehat{C}$—that resists leakage from the class $\mathcal{LEAK}$ associated with the underlying PCP system. Informally, an LRCC is associated with a function class $\mathcal{LEAK}$ (the *leakage class*) and a (randomized) input encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$, and compiles a deterministic circuit C into a deterministic circuit $\widehat{C}$ that emulates C's operation over encoded inputs. $\widehat{C}$ is guaranteed to emulate C on *properly encoded inputs*, and is leakage resilient in the sense that for any pair of inputs z, z' for C such that $C(z) = C(z')$, and any $\ell \in \mathcal{LEAK}$, the output of ℓ on the wire values of $\widehat{C}$ when evaluated on a random encoding $\widehat{z} \leftarrow \mathsf{Enc}(z)$ is statistically close to its output when $\widehat{C}$ is evaluated on a random encoding $\widehat{z'} \leftarrow \mathsf{Enc}(z')$. (We stress that we will only need to consider *stateless* circuits C, in which we wish to hide the circuit's input. We additionally do not allow the leakage-resilient circuit $\widehat{C}$ to have *randomized, leak-free* components, and instead the needed randomness will be provided as part of its input encoding.) Notice that if the prover generated the proof from the wire values $\widehat{w}$ of $\widehat{C}$ (instead of the wire values w of C), then $\mathcal{V}^*$'s queries to the proof—which constitute leakage from $\mathcal{LEAK}$ on $\widehat{w}$—would reveal no information about w'.

This observation gives a general blueprint for compiling traditional PCPs into ZK-PCPs: given a PCP system $(\mathcal{P}', \mathcal{V}')$ with an associated leakage class $\mathcal{LEAK}$ as described above, and a compiler that transforms a given circuit into one that resists leakage from $\mathcal{LEAK}$, the ZK-PCP system $(\mathcal{P}, \mathcal{V})$ operates as follows. $\mathcal{P}$ on input x, w generates the leakage-resilient version $\widehat{C_x}$ of the circuit $C_x = C(x, \cdot)$ (i.e., C with x hard-wired into it), uses w to generate the entire wire values $\widehat{w}$ of $\widehat{C_x}$, then emulates $\mathcal{P}'$ on $\widehat{w}$ to generate a PCP π. $\mathcal{V}$ given input x and oracle access to π generates $\widehat{C_x}$ similarly to $\mathcal{P}$, and emulates $\mathcal{V}'$ on π to check that $\widehat{C_x}$ is satisfiable. In particular, whereas the claim which $\mathcal{V}$ set out to verify

can be phrased as "C_x is satisfiable", the underlying PCP system $(\mathcal{P}', \mathcal{V}')$ is used to verify a different claim, namely that *the leakage-resilient circuit* $\widehat{C_x}$ is satisfiable.

Unfortunately, this blueprint does not actually work. The reason is that the internal system $(\mathcal{P}', \mathcal{V}')$ is used to prove a *different* statement, namely that $\widehat{C_x}$ is satisfiable. For $(\mathcal{P}, \mathcal{V})$ to be sound, it should be the case that $\widehat{C_x}$ is satisfiable only if C_x is. This, however, is not generally guaranteed by LRCCs, as we now explain. Recall that the LRCC is correct for properly encoded inputs, in the sense that on such inputs, $\widehat{C}$ emulates C. However, LRCCs in general have no guarantee for inputs which are *not* properly encoded. This is not just an artifact of the definition, but is rather essential for leakage resilience to hold in existing constructions. A main technical contribution of [8] is in defining and constructing an LRCC that also guarantees soundness, in the sense that the leakage-resilient circuit $\widehat{C}$ is satisfiable (even by using invalid encodings as inputs) only if the original circuit C is satisfiable.

To turn this into an actual construction, one needs to design a sound LRCC for a leakage class $\mathcal{LEAK}$ associated with some standard PCP system. The most common leakage classes considered in the literature are either the "Only Computation Leaks" (OCL) model that assumes leakage is "local" in the sense that different "regions" of the circuit leak independently [17], or classes of functions that are "computationally simple" [18], i.e., from a low complexity class such as $\mathcal{AC}^0$. It is known that the leakage classes associated with a PCP system cannot be of the former type [19], so Ishai et al. [8] focus on the latter. Specifically, they show that the PCP system of Arora and Safra [2] has the property that "small" subsets of proof bits can be generated using the class $\mathcal{LEAK}$ of $\mathcal{AC}^0$ circuits (i.e., constant-depth polynomial-sized circuits with $\wedge, \vee, \neg$ gates of unbounded fan-in and fan-out) augmented with a small number of $\oplus$ gates. (See section 4.3 for a formal definition of this leakage class.) Then, they use correlation bounds of Lovett and Srivinasan [20] to show that their sound LRCC resists leakage from $\mathcal{LEAK}$. This yields the following result, where a witness-indistinguishable PCP is a ZK-PCP in which the ZK property holds with an *inefficient* simulator (see Theorem 7 in Section 4 for the formal statement).

Informal Theorem 2. *For any ZK parameter $q^* \in \mathbb{N}$ and any* NP*-language $\mathcal{L}$ there exists a witness-indistinguishable PCP for $\mathcal{L}$ with witness-indistinguishability against q^*-query bounded verifiers, and a negligible soundness error with a non-adaptive honest verifier that queries* $\mathsf{polylog}(q^*)$ *proof bits.*

Assuming the existence of one-way functions (a minimal assumption in cryptography), as well as a Common Random String (CRS) that is available to both parties, and using a standard cryptographic technique—the so-called "FLS technique" [21]—the witness-indistinguishable PCP of Informal Theorem 2 can be transformed into a *ZK*-PCP system where ZK holds against *computationally-bounded* query-bounded verifiers in the CRS model. (We refer the interested reader to [8] for further details, including a formal definition of the model).

1.2. Comparison between Different ZK-PCP Constructions

The ZK-PCPs of [13] (Informal Theorem 1) obtain ZK with efficient simulation as in the ZK-PCPs of [6,11], with a query gap of $q^*/(q^*)^\epsilon$. The query gap is inherited directly from the underlying ZK-PCP variant of [15], which requires the honest verifier to query $\Omega((q^*)^\epsilon)$ proof symbols to obtain a negligible soundness error. Therefore, the query gap could potentially be improved by replacing the underlying building block with a ZK-PCP variant with a larger query gap.

On the other hand, the witness-indistinguishable PCPs of [8] (Informal Theorem 2) obtain an exponential query gap, similar to the ZK-PCPs of [6,11]. This is possible because the underlying (non-ZK) PCP has a negligible soundness error with an honest verifier that queries a poly-logarithmic number of proof bits. However, the construction is only witness-indistinguishable, which is weaker than ZK (see Section 2.1.1). Ref. [8] show that unless $\mathsf{NP} = \mathsf{BPP}$, obtaining *ZK*-PCPs (i.e., with efficient simulation) using their technique would require a new and entirely different approach towards designing and analyzing security of leakage-resilient circuits. We further note that while the techniques of [6,13] extend also to PCPs *of Proximity* (see Section 2.1.1 for a description of this model, and [11,13]

for the constructions), the technique of [8] does not seem to readily lend itself to designing zero-knowledge (or even witness-indistinguishable) PCPs of proximity.

Finally, all the aforementioned ZK-PCP constructions [6,8,11,13] have polynomial-length proofs (whereas traditional PCPs can have quasi-linear length). However, while a polynomial blowup in proof length is inherent to the constructions of [6,11] due to their use of locking schemes, this is not the case for the leakage resilience based constructions [8,13], which could potentially have shorter proofs. This is an additional advantage of taking the leakage resilience based approach. The discussion is summarized in Table 1.

1.3. Related Notations, Extensions, and Cryptographic Applications

Many different variants of proof systems and ZK proof systems have been considered in the literature (see, e.g., Thaler's survey [22], and references therein). We briefly mention two notable notions that are closely related to ZK-PCPs; see also Table 2. The first is *Interactive Oracle Proofs (IOPs)* [23,24] (a special case of IOPs appeared earlier in [25]) which combine aspects of IPs and PCPs. Specifically, in an IOP the verifier and prover interact as in an IP, but the verifier has oracle access to prover messages as in PCPs. These proof systems have received increasing attention, partly due to their uses in blockchain applications. Unlike ZK-PCPs, which can be obtained from standard PCPs through generic compilers, existing ZK-IOPs (see, e.g., [25,26] and references therein) are constructed in an ad-hoc manner, in which ZK is "tailored" to a specific non-ZK IOP. (It is of course preferable to have a generic compiler, since it can be used to enhance any new IOP construction to also guarantee ZK.) The second notion is PCPs *of Proximity (PCPPs)* [3,7,27]—a generalization of PCPs in which the verifier does not read its entire input. Instead, $\mathcal{V}$ has oracle access to x, π, and wishes to check whether x is close to $\mathcal{L}$ in relative Hamming distance. *Zero-Knowledge* PCPPs (ZK-PCPPs) [11] extend ZK-PCPs to the PCPP realm. They guarantee that the view of any verifier $\mathcal{V}^*$ making q^* queries *to the input and the proof* can be efficiently simulated, up to a small statistical distance, *by making only* q^* *queries to the input*. Ishai and Weiss [11] construct ZK-PCPPs for NP with comparable parameters to the ZK-PCPs of [6,10], where soundness holds for inputs which are δ-far from the language, for δ which is constant or inverse polylogarithmic. The honest verifier in their construction is adaptive. Hazay et al. [13] show how to extend their techniques to the setting of PCPPs, constructing ZK-PCPPs for NP with a polynomial query gap with a *non-adaptive* honest verifier.

Table 2. Comparison Between Different Probabilistic Proof Systems. IOPPs are IOPs *of Proximity*, considered in, e.g., [28], in which the verifier has full access to the input, and oracle access to the witness. Here, "$\mathcal{P} \leftrightarrow \mathcal{V}$ Communication" refers to whether there is direct communication between the prover and verifier (in particular, whether the verifier can send messages to the prover); "Access to Prover Messages" states whether the verifier reads prover messages in full, or has oracle access to them; similarly, "Access to Input" indicates whether the verifier reads the input in full, or only has oracle access to it; "Soundness Guarantee" refers to the type of inputs which are guaranteed to be rejected, where "Full" means that all inputs not in the languages are rejected, whereas "Promise (input)" means that only inputs that are *far* (in relative Hamming distance) from the language are guaranteed to be rejected, and "Promise (witness)" only guarantees that $\mathcal{V}$ rejects when given oracle access to a witness which is far (in relative Hamming distance) from all valid witnesses for the input; "ZK Variant Hides" of a system $\mathcal{X}$ states, for the ZK variant ZK-$\mathcal{X}$ of system $\mathcal{X}$, which input of the prover remains hidden from the verifier, where "Witness" means verification reveals no information about the underlying NP witness, and "Witness, input (partial)" (resp. "Witness (partial)") roughly means that a verifier making q queries to the input and proof(s) (resp. witness) learns only q physical bits of the input (resp., witness).

	$\mathcal{P} \leftrightarrow \mathcal{V}$ **Communication**	**Access to Prover Messages**	**# Prover Messages**	**Access to Input**	**Soundness Guarantee**	**ZK Variant Hides**
IP	Yes	Full	Multiple	Full	Full	Witness
PCP	No	Oracle	Single	Full	Full	Witness
IOP	Yes	Oracle	Multiple	Full	Full	Witness
PCPP	No	Oracle	Single	Oracle	Promise (input)	Witness, input (partial)
IOPP	Yes	Oracle	Multiple	Full	Promise (witness)	Witness (partial)

ZK-PCPs (and ZK-PCPPs) are motivated not only from a purely theoretical perspective as a natural model of a proof system, but also from their usefulness for cryptographic applications. Specifically, they enable modular design of cryptographic proofs systems, by separating a "clean" information–theoretic proof system component, from the cryptographic assumptions (such as hardness assumptions or an augmented model of computation) which can then be used to transform—i.e., "compile" through a *cryptographic compiler*—the information–theoretic system into a computationally-secure system than can be implemented. Thus, one can design, analyze and optimize the information–theoretic proof system, then apply different cryptographic compilers to obtain different properties of the resultant system. This paradigm has been extremely successful, and is widely used. For example, ZK-PCPs are the underlying combinatorial building blocks in constructions of succinct zero-knowledge arguments [29].

ZK-PCPs and ZK-PCPPs also have more direct cryptographic applications both to two-party and multiparty scenarios which require highly efficient verification methods on secret data. In the two-party setting, these include constructions of constant-round (or non-interactive in the random-oracle model) black box arguments for NP with statistical ZK [10] (whereas constructions based on *non*-ZK PCPs are not black box [30,31]). ZK-PCPPs can additionally be used to design two-party and multiparty black box commit-and-prove protocols for NP [11]. While these applications only require a weaker ZK guarantee (specifically, ZK against the *honest* verifier), applications in multiparty settings require full-fledged ZK against *malicious* verifiers. These include constructions of certifiable versions of verifiable secret sharing from ZK-PCPPs, as well as sublinear ZK proofs in a distributed setting from ZK-PCPs. (Various other notions of ZK proofs in a distributed setting have been considered recently.) In the latter application, the prover and verifier are aided by multiple (potentially corrupted) servers. The motivation for this distributed setting is to minimize the round complexity, and underlying assumptions, of sublinear ZK proofs. Specifically, using PCPs with ZK against *malicious* verifiers, [8] construct distributed 3-round witness-indistinguishable proofs (respectively, ZK proofs in the computational setting with

a common random string) for NP, which are unconditionally secure (respectively, based on the existence of one-way functions) in which the total communication involving the verifier is sublinear in the input length. This should be contrasted with standard sublinear ZK arguments, that require at least four rounds of interaction, and require the existence of collision resistant hash functions [10,30].

2. Preliminaries

Basic notations. We denote the security parameter by κ. A function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$, and $\mathsf{negl}(\kappa)$ denotes the set of all negligible functions. We use the asymptotic notations $O(\cdot)$ and $\Omega(\cdot)$, where $\widetilde{O}(n)$ and $\widetilde{\Omega}(n)$ denote $n \cdot \mathsf{poly}(\log)n$ and $n/\mathsf{poly}(\log)n$,respectively. We use the abbreviation PPT to denote Probabilistic Polynomial-Time, and denote by $[n]$ the set of elements $\{1, \ldots, n\}$. For a string s of length n, and a subset $I \subseteq [n]$, we denote by $s|_I$ the restriction of s to the coordinates in I.

We usually denote vectors using boldface letters (e.g., $\mathbf{a}$), or as $\vec{a}$. For a pair $\mathbf{x}, \mathbf{y}$ of vectors, $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes their inner product, and $\mathsf{Ham}(\mathbf{x}, \mathbf{y})$ to denote their Hamming distance, i.e., $\mathsf{Ham}(\mathbf{x}, \mathbf{y}) = |\{i : \mathbf{x}_i \neq \mathbf{y}_i\}|$. For functions f, g, we denote their composition as $(f \circ g)(x) := f(g(x))$. The composition of families F, G of functions is defined as $F \circ G := \{f \circ g : f \in F, g \in G\}$.

For a distribution $\mathcal{D}$, sampling according to $\mathcal{D}$ is denote by $X \leftarrow \mathcal{D}$, or $X \in_R \mathcal{D}$. For a pair of random variables X, Y, we use $X \equiv Y$ to denote that X, Y are identically distributed. For random variables X_κ and Y_κ over a finite domain Ω, the *statistical distance* between them is defined as

$$\mathsf{SD}(X_\kappa, Y_\kappa) = \frac{1}{2} \sum_{w \in \Omega} \left| \Pr[X_\kappa = w] - \Pr[Y_\kappa = w] \right|.$$

X_κ and Y_κ are *ϵ-statistically close* if their statistical distance is at most $\epsilon(\kappa)$. Ensembles $\{X_\kappa\}_\kappa, \{Y_\kappa\}_\kappa$ are *statistically close*, denoted $X_\kappa \approx Y_\kappa$, if there exists an $\epsilon(\kappa) = \mathsf{negl}(\kappa)$ such that X_κ, Y_κ are $\epsilon(\kappa)$-close for every κ. We say that $\{X_\kappa\}_\kappa, \{Y_\kappa\}_\kappa$ are *computationally indistinguishable* if they have a $\mathsf{negl}(\kappa)$ *computational distance*, i.e., for every PPT distinguisher $\mathcal{D}$ there exists an $\epsilon(\kappa) = \mathsf{negl}(\kappa)$ such that for every κ:

$$|\Pr[\mathcal{D}(X_\kappa) = 1] - \Pr[\mathcal{D}(Y_\kappa) = 1]| \leq \epsilon(\kappa).$$

Languages and Relations. We will consider NP-relations $\mathcal{R} = \mathcal{R}(x, w)$, and the corresponding NP-languages $\mathcal{L} = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$. We sometimes write $\mathcal{R}_\mathcal{L}$ to refer to the NP-relation, the corresponding language of which is $\mathcal{L}$.

Encoding Schemes and Leakage-Resilient Encoding Schemes. Our constructions will rely on encodings schemes with leakage-resilience properties. We now provide a simple definition of an encoding scheme, and refer the reader to Section 3.1 for a more detailed discussion of the notion.

Definition 1. *Let $k, n \in \mathbb{N}$. An* Encoding Scheme *over an alphabet Σ is a pair* $(\mathsf{Enc}, \mathsf{Dec})$ *where* Enc *is a PPT algorithm, and* Dec *is a (deterministic) polynomial-time algorithm, that satisfy the following.*

- **Syntax.** Enc *on input a secret $x \in \Sigma^k$ outputs a codeword $c \in \Sigma^n$.* Dec *on input $c \in \Sigma^n$ outputs $x \in \Sigma^k$ or a special error symbol $\perp$.*
- **Correctness.** *There exists a $t \geq 0$ such that the following holds for every $x \in \Sigma^k$, and every $c \in \Sigma^n$: if there exists $c_x \in \mathsf{Supp}(\mathsf{Enc}(x))$ such that $\mathsf{Ham}(c, c_x) \leq t$ then $\mathsf{Dec}(c) = x$, otherwise* Dec *outputs $\perp$.*

A note on terminology. In this section and in Section 3, we use k, n respectively to denote the input and output lengths of Enc, as is customary in the context of error-correcting codes. In Section 4, the input and output lengths are denoted by $n, \widehat{n}$, respectively, which is sometimes used in the context of leakage resilience.

We now define two useful properties of encoding schemes. The first is that the scheme is *linear*. The second is that the scheme is *onto*, meaning any $c \in \Sigma^n$ can be interpreted as the encodings of *some* $x \in \Sigma^k$, in the sense that c would be decoded to x.

Definition 2. *An encoding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *over a field* $\mathbb{F}$ *is* linear *if for every k: (1) k divides n; and (2) there exists a decoding vector* $\mathbf{d}$ *such that for every* $x \in \mathbb{F}^k$*: (1) every* $\mathbf{x} \in \mathsf{Supp}(\mathsf{Enc}(x))$ *can be partitioned into k sub-vectors* $\mathbf{x} = \left(\mathbf{x}^1, \ldots, \mathbf{x}^k\right)$*, such that* $\mathsf{Dec}(\mathbf{x}) = \left(\langle \mathbf{d}, \mathbf{x}^1 \rangle, \ldots, \langle \mathbf{d}, \mathbf{x}^k \rangle\right)$.

$(\mathsf{Enc}, \mathsf{Dec})$ *is* onto, *if* Dec *is defined (i.e., does not output* $\perp$*) for every* $c \in \Sigma^n$.

Section 4 will use a *parameterized* notion of encoding schemes. In such encoding schemes, the encoding and decoding algorithms are given an additional input 1^σ, which is used as a security parameter. The encoding length can then depend also on σ (and not only on k), and we require that for every σ the resultant scheme is an encoding scheme. A parameterized encoding scheme is onto (linear, respectively) if it is onto (linear, respectively) for every σ.

We now define leakage resilience of distributions and encodings.

Definition 3 (Leakage Resilience—Distributions and Encoding Schemes)**.** *Let* $\mathcal{LEAK}$ *be a family of functions, and* $\epsilon > 0$*. For a finite set D, a pair of distributions X, Y over D are* $(\mathcal{LEAK}, \epsilon)$-leakage resilient *if for any function* $\ell \in \mathcal{LEAK}$ *with domain D it holds that* $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$.

A randomized function $f : \Sigma^n \to \Sigma^m$ *is* $(\mathcal{LEAK}, \epsilon)$-leakage resilient *if for every* $x, y \in \Sigma^n$*, the distributions* $f(x), f(y)$ *are* $(\mathcal{LEAK}, \epsilon)$*-leakage resilient.*

An encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ *is* $(\mathcal{LEAK}, \epsilon)$-leakage resilient *if for every large enough* $\sigma \in \mathbb{N}$, $\mathsf{Enc}(\cdot, 1^\sigma)$ *is* $(\mathcal{LEAK}, \epsilon)$*-leakage resilient.*

For the construction of Section 3 we will be particularly interested in *probing-resilient* encoding schemes, namely ones that are leakage resilient against leakage functions that probe bits of the codeword.

For a function family $\mathcal{LEAK}$, we sometimes use the term "leakage family $\mathcal{LEAK}$", or "leakage class $\mathcal{LEAK}$", and refer to functions in $\mathcal{LEAK}$ as "leakage functions".

2.1. PCPs and ZK-PCPs

The probabilistic proof system we focus on in this work is *Probabilistically Checkable Proofs (PCPs)* with zero-knowledge guarantees. We first describe the (standard, non-zero-knowledge) notion of PCPs.

At a high level, PCPs allow a randomized verifier to probabilistically verify the validity of some input statement by querying few bits of a purported proof to which it has oracle access. PCPs can be defined (and have been studied) in relation to various complexity classes (e.g., $DTIME(n)$). In this work, we focus on PCPs for NP since cryptographic applications usually require proof systems for NP (and this also simplified the presentation). In the context of complexity theory, the system consists solely of the verifier and the proof oracle, where the proof generation process is implicit. However, cryptographic applications necessitate that proof generation be efficient, given the NP-witness. Thus, we define PCPs as a system consisting of efficient prover and verifier. (This is by now standard in the literature of PCPs for cryptographic applications, e.g., [6], as well as other proof systems such as interactive oracle proofs [23,24].)

More specifically, a PCP system for a language $\mathcal{L} \in \mathsf{NP}$ consists of a polynomial-time prover $\mathcal{P}$ that given $x \in \mathcal{L}$ and a corresponding witness generates a proof π for x, and a PPT verifier $\mathcal{V}$ having direct access ("oracle access") to individual symbols of π. $\mathcal{V}$ will read only part of its proof string π (called oracle), where the queries to π are determined by $\mathcal{V}$'s input and coin tosses. Formally,

Definition 4 (PCP)**.** *A* Probabilistically Checkable Proof (PCP) *for a language* $\mathcal{L} \in \mathsf{NP}$ *consists of a polynomial-time prover* $\mathcal{P}$ *and a* PPT *verifier* $\mathcal{V}$ *such that there exists a negligible function* $\mathsf{ngl} = \mathsf{negl}(\kappa)$ *for which the following holds:*

- **Syntax:** *The prover $\mathcal{P}$ has input $1^\kappa, x, w$ (κ is the security parameter), and outputs a proof $\pi \in \{0,1\}^*$ for x. The verifier $\mathcal{V}$ has input $1^\kappa, x$, and oracle access to π. It makes q queries to π, and outputs either 0 or 1 (representing reject or accept, respectively). q is called the* query complexity *of the system, and the system is called a q-*query PCP*.*
- **Semantics:** *The system satisfies the following semantic properties:*
  - **Completeness:** *For every $(x, w) \in \mathcal{R}_\mathcal{L}$, and every proof $\pi \in \mathcal{P}(1^\kappa, x, w)$,*

    $$\Pr[\mathcal{V}^\pi(1^\kappa, x) = 1] = 1,$$

    where the probability is over the randomness of $\mathcal{V}$.
  - **Soundness:** *For every $x \notin \mathcal{L}$ and every oracle π^*,*

    $$\Pr[\mathcal{V}^{\pi^*}(1^\kappa, x) = 1] \leq \mathsf{negl}(\kappa),$$

    where the probability is over the coin tosses of the verifier. $\mathsf{negl}(\kappa)$ is called the soundness error *of the system.*

Zero-Knowledge PCPs (ZK-PCPs). Intuitively, ZK-PCPs are PCPs in which the witness remains entirely hidden throughout the verification procedure, in the sense that even a *malicious* verifier which deviates from the specified verification procedure learns nothing but the validity of the claim. Achieving this property requires some modifications to the standard notion of a PCP. First, the prover must now be *probabilistic*. Indeed, completeness of the PCP requires (intuitively, at least) that the proof encode the witness. That is, certain proof bits will carry *some* information about the witness, and by querying them a malicious verifier $\mathcal{V}^*$ may learn information about the witness. Allowing randomized proof generation is akin to having a probabilistic prover in zero-knowledge *interactive* proofs (whereas in standard interactive proofs the prover may be deterministic).

Second, while we do not impose any restrictions on the query *pattern* (i.e., the strategy) of a malicious verifier $\mathcal{V}^*$, we impose a bound on its query *complexity*, namely the number of queries it makes to the proof. To see why imposing *some* restriction is needed, recall that cryptographic applications necessitate an efficient honest prover, meaning the proof will have polynomial length. Therefore, a polynomial-time verifier $\mathcal{V}^*$ that is unrestricted in its access to the proof could potentially read the entire proof and thus necessarily learn information about the witness (since, as noted above, the proof carries information about the witness). There are several possible methods of restricting $\mathcal{V}^*$ to reading only *part* of the proof (e.g., requiring its runtime to be smaller than the proof length), where the one used in the literature is to restrict its *query complexity* to some a-priori bound q^* which is known prior to proof generation (in particular, the proof length may depend on q^*). One advantage of restricting the query complexity is that it allows us to obtain *information theoretic* ZK, namely that ZK holds against malicious verifiers with *unbounded computational power*, so long as the verifier makes at most q^* queries to the proof. It is also more inline with the main efficiency measures of standard PCPs, which focus on the query (and randomness) complexity.

To define ZK-PCPs, we first formalize the restriction on the query complexity of the verifier.

Definition 5 (Query-bounded verifier)**.** *We say that a (possibly malicious) verifier $\mathcal{V}^*$ with oracle access to a proof π is q^*-*query-bounded *if it makes at most q^* queries to π.*

As noted above, we will allow a malicious verifier to be computationally unbounded. Moreover, we will allow its query bound q^* to be much larger than that of the honest verifier. Ideally, the honest verifier will only make $\mathsf{polylog}(q^*)$ queries, and the proof will have length $\mathsf{poly}(q^*)$. We note that traditional PCPs can be verified with a constant number of queries, achieving a constant soundness error. However, since cryptographic applications usually necessitate a *negligible* soundness error, the query complexity of the honest verifier is necessarily polylogarithmic in the security parameter κ, or in q^*.

Another aspect in which we allow a malicious verifier to be more powerful than the honest verifier is *adaptivity*: whereas we would like the honest verifier to be non-adaptive—namely, make a single round of queries to the proof (as in standard PCPs), we allow a

malicious verifier to be adaptive—i.e., make several rounds of queries to the proof. This is formalized in the following definition.

Definition 6 (Adaptive and-non adaptive verifiers)**.** *We say that a (possibly malicious) verifier $\mathcal{V}^*$ is* non adaptive *if its queries are determined solely by its input x and randomness (in particular, $\mathcal{V}^*$ can make all its queries to the proof in a single round). Otherwise, we say that $\mathcal{V}^*$ is* adaptive *(in particular, the queries of an adaptive verifier may depend on the answers to previous queries).*

We are now ready to define ZK-PCPs. Similar to zero-knowledge *interactive* proofs (and, more generally, cryptographic protocols), we formalize zero knowledge by requiring the existence of an efficient *simulator* algorithm that, given the input $1^\kappa, x$, can simulate the *view* of the verifier, namely the verifier's input, random coins, and the oracle answers (the queries of the verifier can be computed from these values). Intuitively, this guarantees zero knowledge because the view of the verifier captures the entire "knowledge" it obtained through verification, and in particular by querying the proof oracle. Since the view can be simulated from the input alone, without any access to the NP-witness, this implies that the view contains no information about the witness. We now formalize this intuition.

Notation 1. *For a PCP system $(\mathcal{P}, \mathcal{V})$ and a (possibly malicious) verifier $\mathcal{V}^*$, we use $\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(\kappa, x, w)$ to denote the view of $\mathcal{V}^*$ when it has input $1^\kappa, x$ and oracle access to a proof that was randomly generated by $\mathcal{P}$ on input $(1^\kappa, x, w)$.*

Definition 7 (ZK-PCP)**.** *We say that a PCP system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}$ is a (q^*, ε)-*Zero-Knowledge PCP (ZK-PCP) *if for every (possibly malicious and adaptive) q^*-query-bounded verifier $\mathcal{V}^*$ there exists a* PPT *simulator* Sim*, such that for every $(x, w) \in \mathcal{R}$, $\mathsf{Sim}(1^\kappa, x)$ is distributed ε-statistically close to $\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x, w)$.*

One prevalent approach for designing simulators (which we will use) is to have the simulator emulate the verifier $\mathcal{V}^*$, simulating the answers to $\mathcal{V}^*$'s oracle queries.

2.1.1. Restrictions, Extensions and Generalizations

Definition 7 can be restricted, or alternatively, generalized, in several ways, as we now discuss.

A taxonomy based on ZK quality. While we define a statistical notion of ZK as the default for ZK-PCPs, one can also consider stronger or weaker forms of ZK. Specifically, we can require a stronger *perfect* ZK guarantee in which the simulated view is distributed identically to the real view of the verifier.

Notation 2. *We say that a ZK-PCP has q^*-ZK if it has perfect ZK against q^*-query bounded verifiers.*

Alternatively, we can settle for a weaker *computational* ZK guarantee which only holds against *computationally-bounded* verifiers $\mathcal{V}^*$, where the simulated view is computationally indistinguishable from the real view. One particular relaxation of ZK which we consider in this work is *Witness-Indistinguishability* which does not require that the ZK simulator be efficient. This can be obtained by removing the requirement that Sim be PPT in Definition 7, but the following alternative formulation would be more useful.

Definition 8 (WI-PCP)**.** *We say that a PCP system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}$ is a (q^*, ε)-*Witness-Indistinguishable PCP (WI-PCP) *if for every (possibly malicious and adaptive) q^*-query-bounded verifier $\mathcal{V}^*$, for any $x \in \mathcal{L}$, and any pair of corresponding witnesses w_1, w_2 such that (x, w_1), $(x, w_2) \in \mathcal{R}_\mathcal{L}$:*

$$\mathsf{SD}(\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x, w_1), \mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x, w_2)) \leq \varepsilon.$$

There are also various flavors of ZK based on different qualities of the ZK simulator. These include, for example, whether the simulator is straight line—i.e., it emulates the verifier without having to rewind it, and whether the simulator interacts with the verifier

as a black box. The ZK-PCP constructions described in this work have straight line, black box simulators.

Honest-Verifier ZK. Another natural restriction of the definition is by considering zero knowledge only against the *honest verifier* $\mathcal{V}$. Such systems are called *Honest-Verifier ZK (HVZK)*. As we explain below, it is fairly simple to obtain an HVZK PCP system from a standard PCP system (e.g., such a system was presented already in the paper of Kilian, Petrank and Tardos on ZK-PCPs [6], and a weaker construction with a large soundness error was presented in [9]). However, such a restriction is generally too weak to be used in cryptographic applications, for example, to prove honest behaviour in cryptographic protocols, in which case the verifying party might be maliciously corrupted. To see why settling for HVZK simplifies the problem considerably, notice that HVZK is preserved under standard soundness amplification techniques. More specifically, assume we have an HVZK PCP system with a large soundness error (obtaining such systems is relatively easy given known techniques such as "MPC-in-the-head" [15]). Soundness can then be amplified by having the prover generate many fresh, independent copies of the proof, and having the honest verifier repeat the verification procedure several times, each time using a fresh proof copy. HVZK of this verification procedure easily reduces to the HVZK of the original system. However, this amplification does not preserve *full-fledged* ZK (i.e., against malicious verifiers) *even if* the original system is ZK against malicious verifiers. Indeed, the reason is that the query complexity—even of the honest verifier—increases through this transformation, and in particular would exceed the ZK query bound of the original system. Thus, a malicious verifier that "concentrates" all its queries to a *single* proof copy might be able to violate the ZK of the underlying system.

Verifier Adaptivity. In *non-ZK* PCP constructions, the honest verifier is non-adaptive. In contrast, the classic ZK-PCP of [6], and all consequent ZK-PCPs—*except the ones based on leakage resilience* [8,13]—have an *adaptive* honest verifier. As discussed in Section 1, this is because, very roughly, they enhance an HVZK PCP to have full-fledged ZK, by modifying the proof such that even honest verification requires multiple rounds of queries to the proof. Moreover, this is inherent to their technique of using "locking schemes" [6]. Having a non-adaptive honest verifier is a major advantage of leakage resilience based ZK-PCPs (see Table 1 for a comparison of existing ZK-PCP systems), since having non-adaptive *honest* verification is a desirable feature of the system. Indeed, an adaptive honest verifier translates into multiple interaction rounds in cryptographic applications of ZK-PCPs.

Notation 3. *We say that a ZK-PCP system* $(\mathcal{P}, \mathcal{V})$ *is a* non-adaptive ZK-PCP *if the honest verifier* $\mathcal{V}$ *is non-adaptive.*

We note that an orthogonal measure of adaptivity is whether a *malicious* verifier is restricted to being non-adaptive. Unlike having a non-adaptive honest verifier, guaranteeing ZK only against non-adaptive malicious verifiers is an *undesirable restriction* of the system, since it means there are no guarantees against adaptive verifiers.

Universal vs. Non-Universal Simulation. Definition 7 requires ZK to hold with a non-universal simulator, requiring, for every malicious verifier $\mathcal{V}^*$, the existence of a simulator $\mathsf{Sim}_{\mathcal{V}^*}$. A stronger possible definition would require the existence of a *universal* simulator Sim that can simulate the view of *any* query-bounded verifier $\mathcal{V}^*$. We note that all the ZK-PCPs described in this work (in fact, to the best of our knowledge, all existing ZK-PCP constructions) satisfy this stronger definition.

The Alphabet. Similar to PCPs, ZK-PCPs are defined as bit-strings. One could also consider a relaxed notion in which the proof is over some larger alphabet Σ (and this indeed has been done in the PCP context). We note, however, that while for standard PCPs the choice of alphabet affects only the parameters of the scheme (but not its its semantic properties), this is not the case for *zero-knowledge* PCPs. Indeed, any PCP over Σ can be transformed to a PCP over $\{0,1\}$ by replacing each symbol with a bit-string representation of it, without violating completeness or soundness. (We note that while more elaborate alphabet reduction techniques have been employed in the context of traditional PCP, e.g., in [3], their goal was to improve the system's parameters.) However, as we explain in Section 3, doing so for a *ZK*-PCP does not preserve ZK against *malicious* verifiers. ZK-PCPs

over a large alphabet Σ can still be useful as a building block for obtaining ZK-PCPs (with proofs over $\{0,1\}$), see Section 3.

PCPs of Proximity. A useful generalization of PCPs are PCPs of *Proximity* (PCPPs), that allow verification of an input claim while reading only a small portion of it. This is formalized by giving the verifier oracle access to the input, similar to how it accesses the proof. Of course, in this case the verifier cannot be expected to distinguish a true claim from a claim that is false, but very close to being true (e.g., a 3-CNF for which there exists an assignment that satisfies all but a tiny fraction of the clauses). Instead, soundness is defined similarly to correctness of promise problems: any input which is sufficiently far from the corresponding NP-language will be rejected with high probability. PCPPs are an important building block in PCP constructions, and a useful notion in its own right. There are PCPP constructions matching the properties of the best-known standard PCPs [32,33].

Zero-Knowledge PCPPs (ZK-PCPPs) [11] have a stronger ZK guarantee than ZK-PCPs: while ZK-PCPs guarantee that the *witness* remains entirely hidden through verification, ZK-PCPPs additionally guarantee that the *input* itself remains mostly hidden, in the sense that the verifier (even a malicious one) learns only few *physical* input bits. This is formalized using the simulation paradigm as in ZK-PCPs, where instead of giving the entire input to the simulator, it has oracle access to it, and is restricted to making q^* queries (where q^* is the query complexity of the verifier).

Certain techniques for constructing ZK-PCPs extend also to PCPPs, while others do not (or, at least, it is not clear how to extend them). In particular, the original ZK-PCP construction of [6] can be extended to also apply to ZK-PCPPs [11], and the ZK-PCPs based on leakage-resilient encodings described in Section 3 also applies to PCPPs (see [13] for a full description of the construction). On the other hand, the construction of ZK-PCPs from LR circuits (Section 4) does not seem to easily extend to the PCPP realm.

3. The ZK-PCPs of Hazay et al.: ZK from LR Encodings

The main result of this section is a construction of ZK-PCPs for NP with a non-adaptive honest verifier and a polynomial query gap (between the query complexity of the honest and malicious verifiers) due to [13]:

Theorem 3 (ZK-PCPs for NP, Formal statement of Informal Theorem 1)**.** *There exists a constant $\epsilon \in (0,1)$ such that for any ZK parameter $q^* \in \mathbb{N}$ there exists a non-adaptive $(q^*)^\epsilon$-query $\Omega(q^*)$-ZK-PCP for NP.*

We describe a simplified version of the construction of [13] which nonetheless suffices for designing ZK-PCPs. The interested reader is referred to [13] for a description of the more general paradigm which employs an equivocal notion of secret sharing instead of the weaker leakage-resilient encodings used here.

The construction employs Leakage-Resilient (LR) Encodings. The starting point is a ZK-PCP variant $(\mathcal{P}', \mathcal{V}')$ over a *large alphabet* Σ, namely where the proof π' is over Σ. To obtain a standard ZK-PCP $(\mathcal{P}, \mathcal{V})$—i.e., one in which the proof π is over *bits*—we need an *alphabet reduction*. That is, we are looking for a transformation that replaces each symbol $\pi'_i \in \Sigma$ with a bit-string "segment" $\mathsf{segm}_i \in \{0,1\}^*$. Then, given a proof $\pi' = (\pi'_1, \ldots, \pi'_N)$, the resultant proof would be $\pi = (\mathsf{segm}_1, \ldots, \mathsf{segm}_N)$.

As mentioned in Section 2.1.1, the naive alphabet reduction which replaces each symbol of Σ with a bit string representing it does not preserve ZK. Indeed, this alphabet reduction necessarily increases the query complexity of the honest verifier $\mathcal{V}$, who will need to query $q \cdot |\mathsf{segm}|$ proof bits (where q is the query complexity of $\mathcal{V}'$). Thus, a malicious verifier—the query complexity of which is at least as that of the honest verifier—with oracle access to the resultant proof π may query subsets of bits in *many* segments segm_i, effectively learning partial information about many proof symbols (in particular, more than the ZK guarantee of π'), and violating ZK. Therefore, we need an alphabet reduction which *preserves ZK*.

Viewed through the leakage-resilience lens, the information which a malicious verifier obtains on a symbol $\pi'_i \in \Sigma$ by querying bits of segm_i constitutes *probing leakage* on segm_i, and consequently also on π'_i. Thus, intuitively, ZK can be guaranteed by protecting the

segments segm_i from probing leakage. This gives a general blueprint for a ZK alphabet reduction: replace each symbol σ of Σ with its binary representation s_σ, then encode s_σ using a probing-resilient encoding. While this roughly describes the alphabet reduction of [13], there are a few subtleties, as we now describe.

Simulation Strategy for Malicious Verifiers. A probing-resilient encoding can only protect against probing of a sufficiently small subset of bits of the encoding, namely against probing of some a-priori fixed fraction τ of bits. (Indeed, since the message can be decoded from the encoding, an adversary that probes the entire encoding necessarily learns the underlying message.) However, a malicious verifier $\mathcal{V}^*$ may query an entire segment segm_i. To see why, notice that the query bound q^* imposed on the malicious verifier is expected to be much larger than the length of the encoding. Indeed, q^* should be at least as large as the query complexity of the *honest* verifier, which would need to read at least a few symbols of the original PCP, i.e., a few *full* encodings of symbols of the original PCP. More generally, $\mathcal{V}^*$ may read more than a τ-fraction of a segment, in which case the probing-resilience of the encoding cannot be used. We solve this issue in the simulation by dividing the segments segm_i into two types: "heavy" and "light" segments. Intuitively, heavy segments are ones from which $\mathcal{V}^*$ queried many bits, in particular, more than a τ-fraction. Light segments are segments that are not heavy. We use the probing-resilience of the underlying encoding to claim that $\mathcal{V}^*$ learns no information about the symbols encoded in the light segments, and use the ZK guarantee of the underlying ZK-PCP system $(\mathcal{P}', \mathcal{V}')$ to simulate the symbols encoded in heavy segments. This gives us a simulation strategy for $(\mathcal{P}, \mathcal{V})$: simulate heavy symbols using the simulator of the underlying system $(\mathcal{P}', \mathcal{V}')$, and simulate light symbols using random and independent encoding of an arbitrary value (e.g., the all-zeros string).

Simulating Partially-Leaked Symbols. The simulation strategy defined in the previous paragraph necessitates that the simulator Sim knows in advance which segments are heavy and which are light, since this determines how to generate the answer to a query. Whether or not a segment is heavy is a function of *the entire query pattern* of $\mathcal{V}^*$. Thus, this proof strategy only works against *non-adaptive* malicious verifiers, namely ones which make a single round of queries to the proof. (Indeed, in this case Sim learns all of $\mathcal{V}^*$'s queries before it needs to simulate the oracle answers.) ZK against *adaptive* malicious verifiers, namely ones which make several rounds of queries to the proof, where each query may depend on the answers to the previous queries, requires a somewhat different simulation strategy, and a stronger LR guarantee from the probing-resilient encoding, as we now explain.

The high-level idea is as follows. At the onset of the simulation, Sim treats all segments as light, answering queries using random and independent encodings of $\vec{0}$. At certain points in the simulation, a certain segment i may become heavy—namely, the number of queries $\mathcal{V}^*$ made to it exceeds the probing threshold τ. At this point, Sim uses the simulator Sim' of the underlying ZK-PCP system $(\mathcal{P}', \mathcal{V}')$ to simulate the symbol π'_i. To continue with the simulation, Sim must now generate an encoding of π'_i which is (1) consistent with the bits already probed from the i'th segment; and (2) is distributed as a random encoding of π'_i subject to (1). For this, we need the underlying probing-resilient encoding to be *equivocal*—allowing one to efficiently sample from this distribution. Such encodings are called *Reconstructable Probabilistic Encodings (RPEs)* [34,35].

Putting It Together. We are now ready to describe the full alphabet reduction (see Section 3.2, and Figure 1 in particular) that transforms a ZK-PCP variant $(\mathcal{P}', \mathcal{V}')$ over alphabet Σ into a ZK-PCP $(\mathcal{P}, \mathcal{V})$ (over bits). The reduction employs an RPE (see Section 3.1). For proof generation, the prover $\mathcal{P}$ first runs $\mathcal{P}'$ to generate a proof $\pi' = (\pi'_1, \ldots, \pi'_N)$ over Σ. Then, it replaces each proof symbol π'_i with its binary representation s_i, and uses the RPE to encode s_i into a segment segm_i. $\mathcal{P}$ outputs the proof $\pi = (\mathsf{segm}_1, \ldots, \mathsf{segm}_N)$. The verifier $\mathcal{V}$, given oracle access to π, verifies the proof by emulating $\mathcal{V}'$. Whenever $\mathcal{V}'$ queries a symbol π'_q of π', $\mathcal{V}$ queries segm_q from π, RPE-decodes it to obtain the binary representation s_q of a symbol $\sigma_q \in \Sigma$, and provides σ_q to $\mathcal{V}'$ as the answer of the oracle. When the emulation ends, $\mathcal{V}$ outputs whatever $\mathcal{V}'$ outputs. In the following sections, we describe the RPE building block (Section 3.1), and analyze the resultant ZK-PCP scheme (Section 3.2).

Construction 4 (Alphabet reduction for ZK-PCPs). *Let κ be a security parameter.*
Building blocks:

- *A PCP system $(\mathcal{P}', \mathcal{V}')$ over alphabet Σ of size $|\Sigma| = 2^k$;*
- *An RPE $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$ for secrets in $\{0,1\}^k$.*

Prover algorithm. *$\mathcal{P}$ has input $1^\kappa, x, w$. It runs $\mathcal{P}'$ with input $1^\kappa, x, w$ to obtain a proof $\pi' = (\pi'_1, \ldots, \pi'_N)$ over Σ. For every proof symbol π'_i, let s_i denote its bit-string representation, then $\mathcal{P}$ encodes $\mathsf{segm}_i \leftarrow \mathsf{Enc}(\mathsf{s}_i)$. Finally, $\mathcal{P}$ outputs the proof $\pi = (\mathsf{segm}_1, \ldots, \mathsf{segm}_N)$.*

Verifier algorithm. *$\mathcal{V}$ is given input $1^\kappa, x$ and oracle access to π. It runs $\mathcal{V}'$ with input $1^\kappa, x$, and emulates the oracle π' for $\mathcal{V}'$ as follows. Whenever $\mathcal{V}'$ reads a symbol σ from π', $\mathcal{V}$ reads the entire encoding of σ from π. Then, it uses Dec to recover the symbol σ, which it gives to $\mathcal{V}'$ as the answer of the oracle. When the emulation terminates, $\mathcal{V}$ outputs whatever $\mathcal{V}'$ outputs.*

Figure 1. Alphabet Reduction For ZK-PCPs [13].

3.1. Main Building Block: Reconstructable Probabilistic Encdoings (RPEs)

The main building block of the ZK-preserving alphabet reduction is an encoding scheme with equivocation properties called *Reconstructable Probabilistic Encoding (RPE)* [34–37]. In this section, we formally define these objects.

Codes, or encoding schemes, are extensively used in computer science, the most notable example being Error-Correcting Codes (ECCs), which are used to guarantee that the data can still be recovered even if faults occur (i.e., some of the symbols of the data are erased or corrupted). A code consists of an encoding procedure Enc which maps a message to a codeword, and a decoding procedure Dec which decodes the message from a (possibly corrupted) codeword. In the context of error-correction, $\mathsf{Enc}, \mathsf{Dec}$ are usually deterministic. Most ECCs are linear codes, where the code is defined by a *generator matrix*, and encoding simply multiplies the generator matrix with the message. The main parameters of interest for such codes are: (1) the *rate* of the code—the ratio between the length of the encoding (also known as a *codeword*) and the length of the original message; (2) its distance—namely the minimal distance between a pair of codewords, which is the number of coordinates in which they differ; and (3) the alphabet size (where most ECCs are binary). There are numerous extensions and generalizations of ECCs that guarantee additional properties beyond error correction.

We will be interested in a generalized notion of an ECC which also guarantees *probing-resilience* in the sense that few codeword symbols reveal no information (in an information–theoretic sense) on the encoded message. Of course, such a guarantee cannot be satisfied if encoding is deterministic. Probing-resilient encodings therefore allow for a *randomized* encoding procedure, where each message has a subset of codewords to which it can be mapped, and encoding chooses one of them at random. It is fairly simple to obtain such a leakage-resilient encoding from a linear code, as long as its generator matrix has a "good" structure. (See, e.g., [38] for a description of the needed properties and how encoding works.).

As explained above, probing-resilience alone is insufficient to guarantee ZK against *adaptive* malicious verifiers. Instead, we rely on a stronger *equivocation* property which guarantees that as long as the probing threshold τ had not been violated, the probed bits can be efficiently "explained" as the bits in an encoding of *any* arbitrary message msg. Intuitively, an RPE is an encoding scheme (se defined in Section 2) which is probing-resilient, and is additionally associated with a *resampling/reconstruction* algorithm Rec that can "explain" the probed bits. Formally:

Definition 9 (Reconstructable Probabilistic Encoding (RPE)). *Let $k, n, \ell \in \mathbb{N}$. A (k, n, ℓ)-*Reconstructable Probabilistic Encoding (RPE) *is a triple $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$ where $\mathsf{Enc}, \mathsf{Rec}$ are PPT algorithms, and Dec is a (deterministic) polynomial-time algorithm, that satisfy the following.*

- **Syntax.** Enc *on input a secret $x \in \{0,1\}^k$ outputs a codeword $c \in \{0,1\}^n$.* Dec *on input $c \in \{0,1\}^n$ outputs $x \in \{0,1\}^k$ or a special error symbol $\perp$.* Rec *on input a secret x, a set $\mathcal{I} \subset [n]$ of size $|\mathcal{I}| \leq \ell$, and ℓ bits $(c_i)_{i \in \mathcal{I}}$, outputs $c' \in \{0,1\}^n$;*

- **Correctness.** *There exists a* $t \geq 0$ *such that the following holds for every* $x \in \{0,1\}^k$*, and every* $c \in \{0,1\}^n$*: if there exists* $c_x \in \mathsf{Supp}(\mathsf{Enc}(x))$ *such that* $\mathsf{Ham}(c, c_x) \leq t$ *then* $\mathsf{Dec}(c) = x$*, otherwise* Dec *outputs* $\perp$*;*
- ℓ**-Secrecy (of partial views).** *For every pair of secrets* x, x'*, and any subset* $\mathcal{I} \subseteq [n]$ *such that* $|\mathcal{I}| \leq \ell$*,* $\mathsf{Enc}(x)|_{\mathcal{I}} \equiv \mathsf{Enc}(x')|_{\mathcal{I}}$*;*
- ℓ**-Reconstruction (from partial views).** *For any secret* x*, any subset* $\mathcal{I} \subseteq [n]$ *of size* $|\mathcal{I}| \leq \ell$*, and any set* $\left(c'_i\right)_{i \in \mathcal{I}}$ *of bits,* $\mathsf{Rec}\left(x, \mathcal{I}, \left(c'_i\right)_{i \in \mathcal{I}}\right)$ *is distributed identically to an encoding* $c \in \mathsf{Supp}(\mathsf{Enc}(x))$ *that is random subject to being consistent with* $\left(c'_i\right)_{i \in \mathcal{I}}$*.*

A few remarks are in order. First, our constructions can make do with a relaxed RPE notion in which the secrecy and reconstruction properties hold statistically with statistical distance ϵ. (In this case, the resultant ZK-PCP will have statistical ZK with a statistical error of roughly $N \cdot \epsilon$, where N denotes the proof length, see [13] for details.) Second, while we define RPEs for a single message length k, the notion naturally generalizes to *families* of codes such that for every $k \in \mathbb{N}$ there exists a code the codewords of which have length $n = n(k)$; and there exists a uniform algorithm that given 1^k as input, generates the encoding, decoding and resampling procedures for message length k. We will only consider (efficiently encodable and decodable) *families* of codes in this work. For simplicity and clarity of the definitions, we do not explicitly refer to a family of codes, but k should be understood as a general input length parameter. (This is standard in the literature.) Finally, we note that non-explicit constructions of RPEs follow easily from the existence of linear error-correcting codes with sufficiently "good" parameters, which are satisfied by random linear codes (see, e.g., [14]). Indeed, such codes posses the secrecy property of RPEs, which guarantees that for every subset of ℓ codeword symbols, the resultant system of linear equations has a solution for *any* possible secret. This gives an efficient reconstructor Rec. We note that the final ZK-PCP construction will use an *explicit* RPE construction due to [37,39].

3.2. The ZK-PCP Construction

We now describe the alphabet reduction for ZK-PCPs, which uses RPEs to transform a ZK-PCP variant over a large alphabet to a ZK-PCP (over bits).

Construction 4 transforms a PCP system over a large alphabet into a PCP (over bits). The following theorem of ([40], Theorem 9) states that if the underlying PCP system is ZK, and the RPE is secure, then the resultant scheme is also ZK.

Theorem 5 (ZK-PCPs from LR encodings [13])**.** *Assume Construction 4 is instantiated with:*

- *A* (q^*, ϵ)*-ZK-PCP* $(\mathcal{P}', \mathcal{V}')$ *over alphabet* Σ *for a language* $\mathcal{L}$*;*
- *A* (k, n, ℓ)*-RPE* $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$*.*

Then, Construction 4 is a $((q^* + 1) \cdot (\ell + 1) - 1, \epsilon)$*-ZK-PCP for* $\mathcal{L}$*.*

Moreover, the transformation preserves the soundness and completeness of $(\mathcal{P}', \mathcal{V}')$*. Furthermore, if* $(\mathcal{P}', \mathcal{V}')$ *has proofs of length* N *that can be verified non-adaptively with* q' *queries, then Construction 4 has proofs of length* $N \cdot n$ *that can be verified non-adaptively with* $q = q' \cdot n$ *queries.*

Proof. Completeness follows directly from a combination of the completeness of $(\mathcal{P}', \mathcal{V}')$ and the correctness of the RPE (which guarantees that $\mathcal{V}$ perfectly emulates the proof oracle for $\mathcal{V}'$). The claim regarding q follows directly from the construction.

Soundness. Let $x^* \notin \mathcal{L}$, and let π^* be a purported proof oracle for $\mathcal{V}$. We show that $\mathcal{V}$ rejects x^* with the same probability as $\mathcal{V}'$. We partition π^* into N length-n segments $\pi^{*,1} \cdots \pi^{*,N}$, where the i'th segment $\pi^{*,i}$ contains the bits in locations $(i-1)n + 1, \ldots, i \cdot n$. (Notice that the i'th segment in an *honestly-generated proof* would contain the RPE-encoding of the i'th symbol in a proof over Σ.) Then the correctness of the RPE implies that for every $1 \leq i \leq N$ there exists a $\sigma_i \in \Sigma$ such that $\mathsf{Dec}\left(\pi^{*,i}\right) = \sigma_i$ (indeed, if $\mathsf{Dec}\left(\pi^{*,i}\right) = \perp$ then we set σ_i to some arbitrary symbol in Σ). Let $\pi^{*\prime} = (\sigma_1, \ldots, \sigma_N) \in \Sigma^N$, and notice that when $\mathcal{V}$ has oracle access to π^*, it emulates $\mathcal{V}'$ with oracle access to $\pi^{*\prime}$. The soundness of $(\mathcal{P}', \mathcal{V}')$ therefore guarantees that $\mathcal{V}'$ (and consequently also $\mathcal{V}$) accepts with probability ϵ.

Zero Knowledge. Let $q^{**} = (q^* + 1) \cdot (\ell + 1) - 1$, and let $\mathcal{V}^*$ be a (possibly malicious and adaptive) q^{**}-query bounded verifier. For simplicity of the description, we assume $\mathcal{V}^*$

makes its queries one at a time, and never repeats queries (this is without loss of generality). We describe a simulator Sim for $\mathcal{V}^*$, which relies on the simulator Sim' of the underlying ZK-PCP system $(\mathcal{P}', \mathcal{V}')$, and emulates a proof oracle π^* for $\mathcal{V}^*$ as follows:

1. $\pi^* \in \{0,1\}^{N \cdot n}$ is the concatenation of N segments $\mathsf{segm}_1, \ldots, \mathsf{segm}_N \in \{0,1\}^n$, which Sim initializes as random and independent RPE-encoding of 0^k, by computing $\mathsf{segm}_i \leftarrow \mathsf{Enc}\left(0^k\right)$.
   Sim additionally maintains N counters $\mathsf{Cnt}_1, \ldots, \mathsf{Cnt}_N$, initialize to 0, and N sets $\mathcal{I}_1, \ldots, \mathcal{I}_N$, initialized to $\emptyset$;
2. Sim answers each oracle query Q of $\mathcal{V}^*$ as follows. Assume that Q is a query to the j'th bit of the i'th segment (meaning Q queries the $(i-1)n + j$'th bit of the proof);
   (a) If $\mathsf{Cnt}_i < \ell$, or $\mathsf{Cnt}_i \geq \ell + 1$, then Sim answers with the j'th bit of segm_i, increases Cnt_i by 1, and adds j to $\mathcal{I}_i$;
   (b) Otherwise, $\mathsf{Cnt}_i = \ell$, meaning $\mathcal{V}^*$ has already queried ℓ bits from the encoding in the i'th segment. In this case, Sim uses Sim' to simulate the i'th symbol σ_i of a proof of the underlying ZK-PCP system. Then, Sim resamples an encoding of σ_i by computing $\mathsf{segm}'_i \leftarrow \mathsf{Rec}\left(\sigma_i, \mathcal{I}_i, (\mathsf{segm}_i)|_{\mathcal{I}_i}\right)$ (this resamples a fresh encoding of σ_i consistently with the oracle answers already simulated), sets $\mathsf{segm}_i := \mathsf{segm}'_i$, and provides the j'th bit of segm_i as the answer of the oracle. Finally, it increases Cnt_i by 1.

We now prove that the simulated and real views of $\mathcal{V}^*$ are ϵ-statistically close, using a hybrid argument.

$\mathcal{H}_0$: This is the view of $\mathcal{V}^*$ in the simulation described above;

$\mathcal{H}_1^0$: $\mathcal{H}_1^0$ is obtained from $\mathcal{H}_0$ by replacing the simulated answers of Sim' with the actual proof symbols of a proof π' honestly generated by $\mathcal{P}'$.

Then $\mathcal{H}_0$ and $\mathcal{H}_1^0$ are ϵ-statistically close by the (q^*, ϵ)-ZK of $(\mathcal{P}', \mathcal{V}')$.

Indeed, since Sim' is used to simulate the i'th proof symbol only when $\mathsf{Cnt}_i = \ell$, i.e., only on the $\ell + 1$ query to the i'th segment, and since $q^{**} = (q^* + 1)(\ell + 1) - 1$, Sim' is only used to simulate at most q^* symbols, and so the (adaptive) ZK of $(\mathcal{P}, \mathcal{V})$ implies that the simulated answers are ϵ-statistically close to the corresponding symbols in a real proof π'.

$\mathcal{H}_1^i, 1 \leq i \leq N$: $\mathcal{H}_1^i$ is obtained from $\mathcal{H}_1^{i-1}$ by replacing the simulated answers of Sim with the actual bits in the i'th section of the proof π. (In particular, these are bits in random RPE-encodings of $\pi'_1, \ldots, \pi'_N$.)

Then $\mathcal{H}_1^{i-1}$ and $\mathcal{H}_1^i$ are identically distributed by the ℓ-secrecy or ℓ-reconstruction of the RPE.

To see why this holds, we consider two cases depending on whether or not $\mathcal{V}^*$ made more than ℓ queries to the i'th segment. If $\mathcal{V}^*$ made at most ℓ queries to the i'th segment, then all queries in $\mathcal{H}_1^{i-1}$ were answered according to an RPE-encoding of 0^k, whereas all queries in $\mathcal{H}_1^i$ were answered according to an RPE-encoding of (the binary representation of) π'_i. The bits queried in the i'th segment (and consequently, also the entire hybrids) are therefore identically distributed by the ℓ-secrecy of the RPE.

If, on the other hand, $\mathcal{V}^*$ made *more than* ℓ queries to the i'th segment, then the first ℓ queries in $\mathcal{H}_1^{i-1}$ were answered according to an RPE-encoding of 0^k, and the remaining queries were answered using a resampled encoding of (the binary representation of) π'_i, resampled consistently with the answers to the first ℓ queries; whereas in $\mathcal{H}_1^i$ all queries to the i'th segment were answered according to a random encoding of (the binary representation of) π'_i. In this case, the distributions are identically distributed by the ℓ-reconstruction of the RPE.

We conclude the proof by noting that $\mathcal{H}_1^N$ is distributed identically to the real view of $\mathcal{V}^*$. □

Remark 5 (ZK-PCPs from weaker primitives)**.** *We note that if one only requires that ZK hold against malicious* non-adaptive *verifiers, then it suffices for the underlying ZK-PCP system over*

Σ *to have ZK against non-adaptive verifiers. Additionally, the reduction can be instantiated with an RPE in which secrecy and reconstruction hold* statistically *with some error* ϵ'*, in which case the overall simulation error will be* $\epsilon + \epsilon'(N - q^*)$*. We refer the interested reader to [40] for a proof of these claims.*

3.3. ZK-PCPs with Square-Root Gap

In this section we describe the $\sqrt{q^*}$-query q^*-ZK-PCPs of [13] (i.e., the system has q^*-ZK with an honest verifier that makes $\sqrt{q^*}$ queries), which are obtained by appropriately instantiating the building blocks of Construction 4. These are the only known PCPs to date that have full-fledged ZK with a non-adaptive honest verifier. We first describe how we instantiate the building blocks.

The Building Blocks. Hazay et al. [13] instantiate Construction 4 with a ZK-PCP of [15] (using an improved soundness analysis given in [13]), and an RPE based on linear codes.

More specifically, the ZK-PCP over Σ is obtained using the "MPC-in-the-head" technique. It has perfect ZK against malicious verifiers querying at most a constant fraction of proof symbols (for an a-priori bounded constant), where the honest verifier obtains a negligible soundness error by non-adaptively querying only a square-root of the proof symbols. We note that the original soundness analysis of [15] required the honest verifier to make as many queries as a malicious verifier, but this analysis was recently improved by [13].

Theorem 6 (Non-adaptive ZK-PCPs over large alphabets with $\sqrt{Q}$-gap, implicit in [15])**.** *For any* $\mathcal{L} \in \mathsf{NP}$*, any* $Q \geq 3$*, and any input length* n*, there exists an alphabet* Σ *of size* $|\Sigma| = 2^{\mathsf{poly}(n, \log Q)}$ *for which there exists a ZK-PCP for* $\mathcal{L}$ *over* Σ*, with* $\mathsf{negl}(Q)$ *soundness error with a non-adaptive honest verifier that makes* $\log Q \cdot \sqrt{Q}$ *queries, proofs of length* Q*, and perfect* $\Omega(Q)$*-ZK.*

The RPE that [13] uses is obtained by applying a general observation of [37,39]—that the existence of linear codes implies the existence of RPEs—to the linear codes of Decatur [41]. Specifically, Ball et al. ([37], Lemma 2) prove the following, where a code $\mathcal{C} \subseteq \{0,1\}^n$ is linear if its encoding procedure simply multiplies the input with a public generator matrix, and the distance of the code is $\min_{c \in \mathcal{C}} \mathsf{Ham}(c, 0^n)$ (i.e., the minimal weight of a non-zero codeword):

Lemma 1 (RPEs from linear error-correcting codes [37])**.** *If there exists a linear error-correcting code* $\mathcal{C} \subseteq \{0,1\}^n$ *with messages in* $\{0,1\}^k$ *and distance* d*, then there exists a* $(k, n, d-1)$*-RPE.*

To obtain an RPE with good parameters, we apply Lemma 1 to any explicit family of linear codes with constant rate and constant relative distance. For example, we can use the codes of ([41], Theorem 2.1), which already posses secrecy from partial views. (We note that the construction of [41] relies on Toeplitz matrices of logarithmic size which generate codes with good parameters. Such a matrix can be efficiently found by traversing all these matrices by some pre-defined order, and using the first matrix satisfying the desired properties.) In particular, we have

Corollary 1 (RPEs)**.** *For every message length* $k \in \mathbb{N}$*, there exists a* $(k, O(k), \Omega(k))$*-RPE.*

A ZK-PCP with Square-Root Query Gap. With these building blocks in place, we are ready to prove Theorem 3.

Proof of Theorem 3. We instantiate Theorem 5 with the ZK-PCP system of Theorem 6 and the RPE of Corollary 1. We assume without loss of generality that $q^* \geq n$ (where n is the input length). Since we set Q below to be polynomially-related to q^* (and consequently also to n), there exists a constant c such that the PCP of Theorem 6 is over an alphabet of size 2^{Q^c}. Let $\alpha = 1/(c+1)$, then we instantiate Theorem 6 with $Q := (q^*)^\alpha$, and set $k = Q^c$ in Corollary 1. Then Theorem 5 guarantees that the resultant ZK-PCP has proofs of length $Q \cdot O(Q^c) = O(Q^{c+1}) = (q^*)^{\alpha \cdot (c+1)} = O(q^*)$ with perfect ZK against (possibly malicious

and adaptive) verifiers making $\Omega(Q) \cdot \Omega(Q^c) = \Omega(q^*)$ queries, and a $\mathsf{negl}(Q) = \mathsf{negl}(q^*)$ soundness error with a non-adaptive honest verifier the query complexity of which is $\log Q \cdot \sqrt{Q} \cdot O(Q^c) = \tilde{O}\left(Q^{c+1/2}\right) = \tilde{O}\left((q^*)^{(c+1/2)/(c+1)}\right)$. The theorem now follows for any ϵ which is larger than $(c+1/2)/(c+1)$ (and for a sufficiently large q^*). □

Theorem 3 allows one to choose the ZK query bound q^*. Hazay et al. [13] also give an alternative formulation of Theorem 3, in which the square-root query gap obtained by the ZK-PCP system is more clearly manifested.

Corollary 2 (ZK-PCP with $\sqrt{n}$ query gap, Corollary 10 of [40])**.** *There exists a constant $c > 0$ such that there exists ZK-PCP with perfect $\Omega\left(n^{c+1}\right)$-ZK, and $\mathsf{negl}(n)$ soundness error with an honest verifier that non-adaptively queries $\tilde{O}\left(n^{c+1/2}\right)$ proof bits, where n denotes the input length.*

Proof. We instantiate Theorem 5 with the ZK-PCP system of Theorem 6 and the RPE of Corollary 1. Setting $Q = n$ in Theorem 6, let c be a constant such that the PCP of theorem 5 is over an alphabet of size n^c. We set $k = n^c$ in Corollary 1. Then Theorem 5 guarantees that the resultant ZK-PCP has proofs of length $n \cdot O(n^c) = O\left(n^{c+1}\right)$ with perfect ZK against (possibly malicious and adaptive) verifiers making $\Omega(n) \cdot \Omega(n^c) = \Omega\left(n^{c+1}\right)$ queries, and a $\mathsf{negl}(n)$ soundness error with a non-adaptive honest verifier the query complexity of which is $\log n \cdot \sqrt{n} \cdot O(n^c) = \tilde{O}\left(n^{c+1/2}\right)$. □

4. The ZK-PCPs of Ishai et al.: ZK from LR Circuits

The main result of this section is a construction of witness-indistinguishable PCPs for NP with a non-adaptive honest verifier and an exponential query gap (between the query complexity of the honest and malicious verifiers), which was given in [8]:

Theorem 7 (WI-PCPs for NP, formal statement of Informal Theorem 2)**.** *Let $n \in \mathbb{N}$ be an input length parameter. For any query bound $q^* = \mathsf{poly}(n)$ there exists a non-adaptive $\mathsf{poly}(\log q^*, \kappa)$-query $(q^*, \mathsf{negl}(q^*))$-WI-PCP for NP with $\mathsf{negl}(\kappa)$ soundness error, where κ is a statistical security parameter.*

The WI-PCP system is constructed from leakage-resilient *circuits*. Historically, this construction was presented before the ZK-PCPs of Section 3, and was the first to show a connection between ZK-PCPs and leakage resilience. Moreover, it was the first construction of PCPs with (relaxed) ZK against *malicious* verifiers with *non-adaptive honest* verification. Compared to the ZK-PCPs of Section 3, the scheme we describe in this section has the advantage of obtaining an *exponential* query gap: the scheme is ZK against q^*-query bounded verifiers, but the honest verifier only needs to query $\mathsf{polylog}(q^*)$ proof bits to verify the proof (setting $\kappa = \mathsf{polylog}(q^*)$). However, the PCP system described in this section obtains a weaker form of ZK called *Witness Indistinguishability (WI)* in which the ZK simulator is not guaranteed to be efficient. The construction of Section 3 has the added feature of being simpler.

The main building-block in the transformation of [8] are leakage-resilient *circuits*—a stronger primitive than the leakage-resilient *encodings* used by [13]. Indeed, leakage-resilient encodings only protect *information*, whereas leakage-resilient *circuits* protect *computations*. Thus, while Hazay et al. [13] could only use leakage-resilient encodings to protect the proof *once it was already generated*, Ishai et al. [8] employ leakage-resilient circuits to protect *proof generation itself*. When used in the context of proof generation, these leakage-resilient circuits in effect amplify leakage resilience: from a relatively low leakage bound on the witness, to a much larger leakage bound on the entire computation. This amplification results in an exponential query gap, which eluded the ZK-PCPs of Section 3.

High-Level Idea: Leakage-Resilient Proof Generation. The goal of the prover $\mathcal{P}$ is to convince the verifier $\mathcal{V}$ that $x \in \mathcal{L}$, where $\mathcal{P}$ has a corresponding witness w. Recall from Section 1.1.2 that one way of doing so is to emulate the verification circuit C of the corresponding NP-relation $\mathcal{R}_{\mathcal{L}}$ on (x, w), where $\mathcal{V}$ then checks that this computation was performed correctly. Indeed, without loss of generality we can assume that $\mathcal{R}_{\mathcal{L}}$ has a

canonical form in which the witness consists of the entire wire values of C. In particular, in a PCP system the prover would generate the PCP from the entire wire values of C, which we denote by $[C, x, w]$. While this might blatantly violate ZK (e.g., w itself is part of $[C, x, w]$, and the PCP might explicitly contain these wire values), the main observation of [8] is that by querying few proof bits, $\mathcal{V}$ is leaking on a *computation*—namely, the evaluation of C on (x, w). Therefore, one can use leakage-resilient circuits to guarantee that this leakage gives $\mathcal{V}$ no information on the witness w (i.e., C's input).

Concretely, one can first replace C with a *leakage-resilient version* $\widehat{C}$, then have $\mathcal{V}$ check the computation performed in $\widehat{C}$. If $\mathcal{V}$ were reading directly from the wire values of $\widehat{C}$, then we would need $\widehat{C}$ to resist probing leakage (similar to Section 3). However, $\mathcal{V}$ is actually probing bits in the *PCP* π, which was generated by applying the prover algorithm to the wire values of $\widehat{C}$. Therefore, $\mathcal{V}$ obtains more evolved forms of leakage on these wire values, and we thus need leakage resilience against a wider class of potential leakage functions. Still, $\mathcal{V}$ cannot obtain any leakage that it wants on the wire values of $\widehat{C}$, but rather it is restricted to whatever the prover computes on these wire values during proof generation. In particular, by restricting the types of functions the prover applies to the wire values of $\widehat{C}$, we can "control" the type of information which *even a malicious* verifier $\mathcal{V}^*$ obtains on the witness. (Indeed, $\mathcal{V}^*$ can deviate from an honest verification strategy by choosing to read different—and a larger number of—proof bits than the honest verifier, but cannot affect the proofs generation itself.) This gives a general blueprint for transforming standard PCPs into *ZK*-PCPs: the prover and verifier both hard-wire x into the verification circuit C, to obtain a circuit C_x. Then, they generate the LR-version of it $\widehat{C_x}$, which operates on encoded inputs (see Section 4.1 below). Then, $\mathcal{P}$ randomly encodes the witness w as $\widehat{w}$, and generates the PCP from the wire values $\left[\widehat{C}, \widehat{w}\right]$. Finally, the verifier $\mathcal{V}$ runs the PCP verifier to verify the proof.

Achieving Soundness. While (for an appropriate choice of the family of leakage functions) the blueprint described above would indeed yield ZK proofs, they would not be sound. The reason is that soundness of the original PCP relied on the fact that $C(x, \cdot)$ is satisfiable only if $x \in \mathcal{L}$, i.e., there exists a corresponding witness w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$. This, however, is not preserved in the *leakage-resilient* version $\widehat{C_x}$ of $C(x, \cdot)$. In fact, in many (if not all) existing constructions, $\widehat{C_x}$ is *necessarily* satisfiable—leakage resilience relies on this fact! To understand why this is the case, and how we can still achieve soundness, we need to first take a closer look at how these LR constructions work.

4.1. Leakage-Resilient Circuit Compilers (LRCCs)

In this section we describe the main building block used in the ZK-PCPs of this section: Leakage-Resilient Circuits Compilers (LRCCs). Compared to the LR encodings used in Section 3, LRCCs offer a stronger LR guarantee: they protect *computation* rather than *information*. More specifically, LRCCs are compilers which transform a given circuit C into a functionally-equivalent circuit $\widehat{C}$, which is leakage resilient in the sense that leakage on the wire values of $\widehat{C}$ reveals no information on its input. Of course, this cannot be obtained if the input of C is given to $\widehat{C}$ in the clear, since then leakage on the wire values reveals information about the input. Instead, the LRCC is associated with a (LR) encoding scheme, which is used to encode the inputs to (and intermediate values in the computation of) the LR circuit $\widehat{C}$.

We first define the circuit model which we use, then define circuit compilers, and finally define leakage-resilient circuit compilers.

Circuit Model. An (arithmetic) circuit C over the field $\mathbb{F}$ and the set $X = \{x_1, \ldots, x_n\}$ of variables is a directed acyclic graph the vertices of which are called *gates* and the edges of which are called *wires*, and are labeled with functions over X. Every gate in C of in-degree 0 has out-degree 1 and is either an *input* gate labeled by a variable from X; or a *constant* gate const_α labeled by a constant $\alpha \in \mathbb{F}$. Gates of in-degree 2 and out-degree 1 are labeled by one of the operations $+, -, \times$, i.e., addition, subtraction, and multiplication over $\mathbb{F}$. (Jumping ahead, we will use an LRCC of [42], which employs additional gates, e.g., to duplicate values in the circuit. Since we do not explicitly use these additional gates, we omit their

description to simplify the presentation.) The *size* $|C|$ of a circuit C is the sum of the number of wires, input gates, and output gates, in C. The *depth* of C is the number of gates on the longest path from inputs to outputs. We also consider *Boolean* circuits, with $\wedge, \vee$ gates (replacing the $+, -, \times$ gates of arithmetic circuits), const_0 and const_1 gates, and $\neg$ gates with in- and out-degree 1.

We define several circuit complexity classes, which restrict the size and depth of Boolean and arithmetic circuits. Specifically,

Notation 4. *$\mathcal{SHALLOW}(d, s)$ denotes the class of all depth-d, size-s arithmetic circuits over $\mathbb{F}$. Similarly, $\mathcal{BOOL}(d, s)$ denotes the class of all depth-d, size-s Boolean circuits. Somewhat abusing notation, we use the same notations to denote the* families of functions *computable by circuits in the respective class of circuits. $\mathcal{AC}^0$ denotes all constant-depth and polynomial-sized Boolean circuits over* unbounded fan-in and fan-out *(i.e., in-degree and out-degree) $\wedge, \vee, \neg, \mathsf{const}_0$ and const_1 gates.*

Definition 10 (Satisfiable Circuits)**.** *A circuit $C : \mathbb{F}^n \to \mathbb{F}$ is* satisfiable *if there exists an $x \in \mathbb{F}^n$ such that*

$$C(x) = \begin{cases} 1 & C \text{ is Boolean} \\ 0 & C \text{ is arithmetic} \end{cases}$$

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 C into a circuit $\widehat{C}$ that emulates the operation of C over encoded inputs. Formally,

Definition 11 (Circuit compiler over $\mathbb{F}$)**.** *A circuit compiler over $\mathbb{F}$ is a pair $(\mathsf{Comp}, (\mathsf{Enc}, \mathsf{Dec}))$ such that the following holds:*

- **Syntax:**
  - *$(\mathsf{Enc}, \mathsf{Dec})$ is an encoding scheme, where Enc is a PPT algorithm that on input a vector $x \in \mathbb{F}^n$, and an additional length parameter 1^{len} (which is used to determine the amount of random masks needed to protect the computation from leakage; see Definition 13), outputs a vector $\widehat{x}$, and Dec is a polynomial-time algorithm; We assume that $\widehat{x} \in \mathbb{F}^{\widehat{n}}$ for some $\widehat{n} = \widehat{n}(n, \mathsf{len})$.*
  - *Comp is a polynomial-time algorithm that on input an arithmetic circuit C over $\mathbb{F}$ outputs an arithmetic circuit $\widehat{C}$;*
- **correctness:** *For any arithmetic circuit C, and any input x for C, we have $\Pr\left[\widehat{C}(\widehat{x}) = C(x)\right] = 1$, where $\widehat{x} \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$.*

As discussed above, we will need circuit compilers that are also "sound" in the sense that the compiled circuit $\widehat{C}$ is satisfiable only if the original circuit C is satisfiable. We stress that this property should hold even when the inputs of $\widehat{C}$ are *not* valid encodings according to Enc.

Definition 12 (SAT-respecting circuit compiler)**.** *A circuit compiler $(\mathsf{Comp}, (\mathsf{Enc}, \mathsf{Dec}))$ is* SAT-respecting *if for every circuit $C : \mathbb{F}^n \to \mathbb{F}$, if $\widehat{C} = \mathsf{Comp}(C)$ is satisfiable then C is satisfiable. That is,*

- *For arithmetic C: if there exists an $\widehat{x}^* \in \mathbb{F}^{\widehat{n}}$ such that $\widehat{C}(\widehat{x}^*) = 0$, then there exists an $x \in \mathbb{F}^n$ such that $C(x) = 0$;*
- *For Boolean C: if there exists an $\widehat{x}^* \in \{0,1\}^{\widehat{n}}$ such that $\widehat{C}(\widehat{x}^*) = 1$, then there exists an $x \in \{0,1\}^n$ such that $C(x) = 1$.*

Leakage-Resilient Circuit Compilers. We now define *Leakage-Resilient* circuit compilers. An LRCC is associated with a class $\mathcal{LEAK}$ of leakage functions, and guarantees that when the input to $\widehat{C}$ is properly encoded, then leakage from $\mathcal{LEAK}$ on the wire values of $\widehat{C}$ reveals no information about the input and internal computations, except for the output of

$\widehat{C}$. This is formalized by requiring that the wire values are distributed statistically close for every pair of inputs x, x' such that $C(x) = C(x')$. We first set some notation.

Notation 5. *For a Circuit C, a leakage function $\ell : \mathbb{F}^{|C|} \to \mathbb{F}^m$ for some $m \in \mathbb{N}$, 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 13 (LRCC). *Let $\mathbb{F}$ be a finite field, $\mathcal{LEAK}$ be a function class, $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. A circuit compiler $(\mathsf{Comp}, (\mathsf{Enc}, \mathsf{Dec}))$ is $(\mathcal{LEAK}, \epsilon(n), \mathsf{S}(n))$-*leakage resilient *if for all sufficiently large n's, every arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^m$ (for some m) of size $|C| \leq \mathsf{S}(n)$, every $\ell \in \mathcal{LEAK}$ of input length $\left|\widehat{C}\right|$, and every $x, x' \in \mathbb{F}^n$ such that $C(x) = C(x')$, we have*

$$\mathsf{SD}\left(\ell\left[\widehat{C}, \widehat{x}\right], \ell\left[\widehat{C}, \widehat{x'}\right]\right) \leq \epsilon(n)$$

where $\widehat{x} \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$ and $\widehat{x'} \leftarrow \mathsf{Enc}\left(x', 1^{|C|}\right)$.

LRCCs for Boolean circuits are defined similarly.

A few remarks are in order. First, we note that the LR guarantee of Definition 13 is a relaxed version of the standard notion. Specifically, while Definition 13 only guarantees *indistinguishability*—namely, leakage functions cannot distinguish between the wire values of $\widehat{C}$ when evaluated on two different inputs (so long as C has the same output on both), the standard definition (e.g., in [18,42–44]) is *simulation-based*. That is, the standard definition requires the existence of a *PPT* simulator which, for every circuit C, every input x for C, and any leakage function $\ell \in \mathcal{LEAK}$, can simulate the leakage on the wire values of $\widehat{C}$ given only the description of C and its output $C(x)$. To see why this guarantee is stronger, notice that if for some input x there exists *no* x' such that $C(x) = C(x')$ then Definition 13 provides no secrecy guarantee for x. Moreover, Definition 13 is equivalent to a slightly modified version of the standard definition—specifically, in which there exists a simulator as specified above, but it is *not* guaranteed to be efficient. Indeed, given $C, C(x)$ the simulator could find on its own an x' such that $C(x) = C(x')$, then simulate the leakage by computing $\ell\left[\widehat{C}, \widehat{x'}\right]$. We focus on the relaxed version because it captures the security guarantee which we achieve (in fact, Ishai et al. [8] give strong indications that SAT-respecting LRCCs for "useful" leakage, with the stronger LR guarantee with *efficient* simulation, do not exist—see Section 1.2).

Second, we note that for simplicity, the error in Definitions 12 and 13 depends (only) on the input length n. This can be naturally extended such that the error depends also on a security parameter κ, which is given as input to Comp.

Gadget-Based LRCCs. A leading technique in constructing LRCCs — which is the one employed in all LRCCs described in this work — is *gadget-based*. Such constructions employ a double-layer of encoding, where the LRCC $(\mathsf{Comp}, \mathsf{E})$ is associated with an internal encoding scheme $\left(\mathsf{Enc}^{\mathsf{in}}, \mathsf{Dec}^{\mathsf{in}}\right)$, and Comp outputs a circuit $\widehat{C}$ in which the gates and wires of the circuit C over $\mathbb{F}$ are replaced with *gadgets* and *bundles*. A *bundle* $\mathbf{b}$ is an encoding (according to $\mathsf{Enc}^{\mathsf{in}}$) of some $b \in \mathbb{F}$, representing the value of a wire in C; and a *gadget* $\mathcal{G}_g$ is a (Boolean or arithmetic) circuit over $\mathbb{F}$ which operates over bundles and emulates the operation of the corresponding gate g. More specifically, in addition to taking as input bundle-encodings of g's inputs, $\mathcal{G}_g$ has additional *masking inputs*. These masking inputs are encodings (according to $\mathsf{Enc}^{\mathsf{in}}$) of some *masking values* in $\mathbb{F}^*$ with a *specific, pre-determined* structure (for example, in [42] a masking value is the all-0 string), which are used to obtain LR. We associate with $\mathcal{G}_g$ a set $\mathcal{WF}_g \subset \mathbb{F}^*$ (for *Well-Formed*) which consists of all encodings of masking values with the "correct" structure. We say that the masking inputs of $\mathcal{G}_g$ are *well formed* if they are in $\mathcal{WF}_g$ (otherwise we say they are *ill-formed*). A gadget $\mathcal{G}_g$ is guaranteed to emulate g when its inputs are valid encodings of inputs for g, and its masking inputs are well-formed. For example, if $g = \times$, then for every $x_1, x_2 \in \mathbb{F}$, for every bundle encodings $\mathbf{x}_i \leftarrow \mathsf{Enc}^{\mathsf{in}}(x_i), i = 1, 2$, and for every well-formed masking input bundles m, $\mathcal{G}_g(\mathbf{x}_1, \mathbf{x}_2, \mathsf{m})$ encodes $x_1 \times x_2$.

Remark 8. *We note that the double layer of encoding described above is implicit in most works on leakage resilience. The reason is that, as described in Section 1.1.2, standard LRCCs in the literature are described as* randomized *circuits that generate the needed randomness internally (sometimes, using "opaque" gates which are assumed to be leak-free), whereas we describe the LR circuit as deterministic, and provide the needed randomness as part of the input to the circuit. For this reason, we need to explicitly use a double layer of encoding.*

Leakage resilience of $\widehat{C}$ will follow from a combination of the leakage resilience of the internal encoding scheme $\mathsf{Enc}^{\mathsf{in}}$, and the following leakage-resilience guarantee of the gadgets. If the masking inputs are uniformly distributed over $\mathsf{Enc}^{\mathsf{in}}\left(\mathcal{WF}_g\right)$, then given any valid encodings as standard inputs to the gadget: (1) the outputs are random subject to encoding the correct output (as determined by the standard inputs and the gate operation); and (2) the internal wire values of $\mathcal{G}$ can be reconstructed (i.e., regenerated) in a low complexity class, where the reconstructed and actual wire values are statistically close.

The masking inputs for all gadgets of $\widehat{C}$ are provided as part of the inputs to $\widehat{C}$, using the double-layered encoding, as we now explain. To simplify the description, we focus on describing how this is performed for the LRCC of [42] (see Remark 9 below on how the double-layered encoding extends to other LRCCs as well). In the LRCC of [42], the masking values for *all gates* g are simply the all-0 string, and different gates only differ in the *length* of the string (i.e., the number of 0's). Let M denote the upper bound on the number of masking values used by any gate g. Then $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ where Enc takes as input an $x \in \mathbb{F}^*$, and a size bound S, and outputs an encoding $(\mathbf{x}, \mathbf{m}) = \mathsf{Enc}^{\mathsf{in}}\left(x, 0^{\mathsf{M}\cdot\mathsf{S}}\right)$. For a given a circuit C of size $|C| \leq \mathsf{S}$, its leakage-resilient version $\widehat{C} = \mathsf{Comp}(C)$ takes as input encodings of the form $(\mathbf{x}, \mathbf{m})$, where $\mathbf{x}$ are the input bundles of $\widehat{C}$, and $\mathbf{m}$ are used as the masking inputs of the gadgets of $\widehat{C}$. Since each gadget uses at most M encodings from $\mathbf{m}$, and $\widehat{C}$ has at most S gadgets, $\mathbf{m}$ contains sufficiently-many encodings (according to $\mathsf{Enc}^{\mathsf{in}}$) of 0 to enable the evaluation of all gadgets of $\hat{C}$.

The computation in $\widehat{C}$ is performed over encodings, so the outcome of this computation — at least as it was described above — results in an *encoding* of the output of C. Since $\widehat{C}$ should have the same output as C, the final step in $\widehat{C}$ consists of a decoder which decodes the output.

Remark 9. *We note that the double-layer encoding idea described above naturally extends to other LRCCs as well. Indeed, all that is needed is that all gadgets use the same masking values (differing only in the number of masking values they use). In fact, this idea can also be used if different gadgets use different masking values, in which case* $\mathbf{m}$ *can include* S *sets of masking inputs, each set containing masking inputs for every possible gate* g*, from which* $\widehat{C}$ *can choose the appropriate masking inputs for each gate. This will result in a correct and leakage resilient (albeit less efficient) construction.*

4.2. An SAT-Respecting LRCC for Arithmetic Circuits

In this section we describe a *SAT-respecting* LRCC of [8] for arithmetic circuits, which is based on the LRCC of [42]. We first explain why the LRCC of [42] (and similar LRCCs) are *not* SAT-respecting.

Why are standard LRCCs not SAT-Respecting? Recall that an LRCC is SAT-respecting if for every circuit C, the following holds: if its leakage-resilient variant $\widehat{C}$ is satisfiable, then so is C. This is not the case for many LRCCs, and specifically for gadget-based ones such as [42–44]. Indeed, in such LRCCs, the correctness of the computation relies on the assumption that the masking inputs of the gadgets are well-formed. However, $\widehat{C}$ does not necessarily emulate C if its masking inputs are *ill-formed*. In fact, in these LRCCs one can force the output of $\widehat{C}$ to be any desired value by appropriately choosing the masking values. This property is *crucial* for leakage resilience, since the security proof uses such ill-formed masking inputs to switch the input from x to a different x'. Therefore, the main challenge in obtaining the SAT-respecting property is to guarantee that the masking inputs are well-formed while still allowing the security reduction to use ill-formed masking inputs.

Before describing how this is done, we make two comments. First, recall that (as noted above) in the leakage-resilience literature the masking inputs are usually assumed to be generated by leak-free components, or opaque gates. In particular, the outputs of these components are guaranteed to be distributed according to the correct distribution (i.e., be well-formed), and their internal wires are unavailable to the leakage function. To eliminate this trust assumption, we instead have these encodings provided as part of the input, and check their validity to achieve the SAT-respecting property. Second, we note that the LRCC of [18] *does* posses the property that $\widehat{C}$ is satisfiable only if C is, but this LRCC only resists $\mathcal{AC}^0$ leakage [45], a leakage class which does not seem sufficiently strong to contain the function one needs to apply to an NP-witness to obtain a PCP.

Checking Validity of Encodings While Preserving Leakage Resilience. The main technical ingredient in the SAT-respecting LRCC of [8] is their component which checks well-formedness of encodings in a leakage-resilient manner. The technique is reminiscent of the "2-key trick" of [46] (used to convert a CPA-secure encryption scheme into a CCA-secure one) where they hold two copies of C, and in the security reduction one of the copies is used to achieve the SAT-respecting property, whereas the other is used to obtain leakage resilience. This component is tailored to the LRCC of [42], exploiting the fact that well-formed masking inputs of [42] are simply encodings of the all-0 string.

More specifically, the SAT-respecting LRCC of [8] consists of 3 parts. The first is two copies $\widehat{C}_1, \widehat{C}_2$ of the circuit $\widehat{C}$, obtained using the LRCC of [42]. The second is a mask-checking component $\mathcal{C}_{0-\mathsf{check}}$, which checks that at least one of the copies $\widehat{C}_1, \widehat{C}_2$ uses well-formed masking inputs. The important point is that, for the security proof to go through, $\mathcal{C}_{0-\mathsf{check}}$ must hide *which* of the copies uses well-formed masking inputs. (This is because in the security proof first $\widehat{C}_1$ and then $\widehat{C}_2$ use ill-formed masking inputs; for the hybrids to be indistinguishable it must be the case that one cannot distinguish between the two cases.) As it turns out, this can be achieved by replacing $\mathcal{C}_{0-\mathsf{check}}$ with its leakage-resilient version $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ (obtained using the LRCC of [42]). This now introduces a further complication since $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ has masking inputs of its own, the well-formedness of which must be verified. Indeed, if $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ is allowed to use any masking inputs, then it is no longer guaranteed to emulate $\mathcal{C}_{0-\mathsf{check}}$. In particular, by using ill-formed masking inputs in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ one can flip its output, thus causing it to accept even when the masking inputs of *both* $\widehat{C}_1, \widehat{C}_2$ are ill-formed, rendering the $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ component completely useless. The third component—a mask decoder $\mathcal{C}_{\mathsf{dec}}$—is used to guarantee that $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ uses well-formed masking inputs. Using a double-layer of mask checking (i.e., checking the masks of $\widehat{C}_1, \widehat{C}_2$, and then checking the masks of the mask-checker) is helpful because the computations in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ are *not directly related* to the inputs of $\widehat{C}_1, \widehat{C}_2$. As such, the masking inputs of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ can be *checked directly* by simply decoding them (and checking that all decoded values are 0) without violating LR. We now formally describe each of the components, and the resultant SAT-respecting LRCC.

The Mask Checker $\widehat{\mathcal{C}}_{0-\mathsf{check}}$. The mask checker verifies that at least one of the copies $\widehat{C}_1, \widehat{C}_2$ uses well-formed masking inputs, while hiding which one. Recall that in the LRCC of [42] well-formed masking inputs are encodings of the all-0 string. Thus, if $\vec{\mathsf{m}}^1, \vec{\mathsf{m}}^2$ denote the masking values whose encodings are used in $\widehat{C}_1, \widehat{C}_2$, it suffices to check that $\mathsf{m}^1_j \times \mathsf{m}^2_l = 0$ for every j, l. (We note that this check assumes that any masking input is a valid encoding — according to $\mathsf{Enc}^{\mathsf{in}}$ — of *some* masking value, namely that the internal encoding scheme is onto. This is indeed the case for the internal encoding scheme used in [42].) We construct $\mathcal{C}_{0-\mathsf{check}}$ in two stages. We first describe a "binarization" component $\mathcal{T}$ which checks that a given field element is 0, then use it to construct $\mathcal{C}_{0-\mathsf{check}}$.

Notation 6 ("Binarization" component $\mathcal{T}$)**.** *$\mathcal{T} : \mathbb{F} \to \mathbb{F}$ is defined as $\mathcal{T}(z) = -\prod_{0 \neq a \in \mathbb{F}} (z - a)$, computed using $O(|\mathbb{F}|)$ constant and $\times$ gates arranged in $O(\log|\mathbb{F}|)$ layers. Notice that*

$$\mathcal{T}(z) = \begin{cases} 1 & \text{if } z = 0 \\ 0 & \text{if } z \neq 0. \end{cases}$$

Notation 7 (Mask Checker $\mathcal{C}_{0-\mathsf{check}}$)**.** *Let* $\mathsf{M} \in \mathbb{N}$*. The mask checker* $\mathcal{C}_{0-\mathsf{check}} : \mathbb{F}^{\mathsf{M}} \times \mathbb{F}^{\mathsf{M}} \to \mathbb{F}$ *is defined as follows.* $\mathcal{C}_{0-\mathsf{check}}(y,z) = \prod_{i,j\in[\mathsf{M}]} \mathcal{T}(y_i \times z_j)$*, computed using a multiplication tree of size* $O(\mathsf{M}^2)$ *and depth* $O(\log \mathsf{M})$ *(on top of the multiplication trees used to compute* $\mathcal{T}$*). Notice that*

$$\mathcal{C}_{0-\mathsf{check}}(y,z) = 1 \Leftrightarrow \mathcal{T}(y_i, z_j) = 1 \quad \forall i,j \in [\mathsf{M}] \Leftrightarrow y = 0^{\mathsf{M}} \vee z = 0^{\mathsf{M}}.$$

The mask decoder $\mathcal{C}_{\mathsf{dec}}$**.** As noted above, the mask decoder simply decodes the masking inputs used in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$, and checks that all decoded values are 0. Decoding is performed using the decoder $\mathsf{Dec}^{\mathsf{in}}$ of the internal encoding procedure of [42]. We note that this encoding procedure is linear, and so decoding is performed simply by computing the inner product of the input encoding with some fixed vector (e.g., the all-1 vector). In particular, for each encoding length $\widehat{n}$, the corresponding decoding circuit $\mathsf{Dec}^{\mathsf{in}}$ can be implemented using $O(\widehat{n})$ gates arranged in $O(\log \widehat{n})$ layers.

Notation 8 (Mask Decoder $\mathcal{C}_{\mathsf{dec}}$)**.** *Let* $\mathsf{M}_0 \in \mathbb{N}$*, and let* $\widehat{n}$ *denote the encoding length of* $\left(\mathsf{Enc}^{\mathsf{in}}, \mathsf{Dec}^{\mathsf{in}}\right)$*. The mask decoder* $\mathcal{C}_{\mathsf{dec}} : \left(\mathbb{F}^{\widehat{n}}\right)^{\mathsf{M}_0} \to \mathbb{F}$*, on input* $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_{\mathsf{M}_0})$ *(where* $\mathbf{r}_i \in \mathbb{F}^{\widehat{n}}$ *for every* $1 \le i \le \mathsf{M}_0$*) outputs* $\prod_{i\in[\mathsf{M}_0]} \mathcal{T}\left(\mathsf{Dec}^{\mathsf{in}}(\mathbf{r}_i)\right)$*. Notice that* $\mathcal{C}_{\mathsf{dec}}$ *outputs 1 if and only if all* $\mathbf{r}_i$*'s are well-formed, otherwise it outputs 0.* $\mathcal{C}_{\mathsf{dec}}$ *is computed using* $O(\mathsf{M}_0)$*-many* $\times$ *gates, arranged in a tree of depth* $O(\log \mathsf{M}_0)$ *(on top of the sub-circuits* $\mathcal{T} \circ \mathsf{Dec}^{\mathsf{in}}$*).*

The SAT-Respecting LRCC. Having defined the three components of the SAT-respecting LRCC, we are now ready to describe the compiler itself, which is given in Figure 2.

The following claim summarizes the properties of Construction 11. Roughly, it states that if the internal encoding scheme used in Construction 11 is leakage resilient against a leakage family $\mathcal{LEAK}_{\mathsf{E}}$, then Construction 11 is an SAT-respecting LRCC against a slightly weaker leakage family $\mathcal{LEAK}$. Formally,

Claim 10 (SAT-respecting LRCC over $\mathbb{F}$)**.** *Let* $\mathcal{LEAK}, \mathcal{LEAK}_{\mathsf{E}}$ *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* $\mathsf{E}^{\mathsf{in}} = \left(\mathsf{Enc}^{\mathsf{in}}, \mathsf{Dec}^{\mathsf{in}}\right)$ *be a linear, onto,* $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$*-leakage-resilient encoding scheme with parameters* n*,* σ *and* $\widehat{n} = \widehat{n}(n,\sigma)$*, such that* $\mathcal{LEAK}_{\mathsf{E}} = \mathcal{LEAK} \circ \mathcal{SHALLOW}\left(7, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$*. Then, there exists an SAT-respecting,* $(\mathcal{LEAK}, 4\epsilon(n) \cdot (\widehat{n}(1,\mathsf{S}(n)) + 1) \cdot \mathsf{S}(n), \mathsf{S}(n))$*-LRCC over* $\mathbb{F}$*. Moreover, for every* $C : \mathbb{F}^n \to \mathbb{F}$*, the compiled circuit* $\widehat{C}$ *has size* $\left|\widehat{C}\right| = O\left(|\mathbb{F}|^2 \cdot \widehat{n}^4(1,\mathsf{S}(n)) \cdot |C|^2\right)$*.*

Proof. The SAT-respecting property follows from Lemma 2. The leakage resilience property follows from Lemma 3. As for the size $\left|\widehat{C}\right|$ of the leakage-resilient version $\widehat{C}$ of C, in each of the copies $\widehat{C}_1, \widehat{C}_2$ each gate is replaced with a size-$O(\widehat{n}^2(1,\mathsf{S}(n)))$ gadget (this is the size of gadgets generated by $\mathsf{Comp}^\star$, see Fact 12 below), So $\left|\widehat{C}_1\right|, \left|\widehat{C}_2\right| \le O(\widehat{n}^2(1,\mathsf{S}(n)) \cdot |C|)$. Since each gadget uses at most $O(\widehat{n}(1,\mathsf{S}(n)))$ masking inputs (see Fact 12 below), then $\mathcal{C}_{0-\mathsf{check}}$ contains $O\left(\widehat{n}^2(1,\mathsf{S}(n)) \cdot |C|^2\right)$ "binarization" components $\mathcal{T}$, each of size at most $O(|\mathbb{F}|)$, arranged in a tree with $O\left(\widehat{n}^2(1,\mathsf{S}(n)) \cdot |C|^2\right)$ multiplications, so $|\mathcal{C}_{0-\mathsf{check}}| \le O\left(|\mathbb{F}| \cdot \widehat{n}^2(1,\mathsf{S}(n)) \cdot |C|^2\right)$, and consequently $\left|\widehat{\mathcal{C}}_{0-\mathsf{check}}\right| \le O\left(|\mathbb{F}| \cdot \widehat{n}^4(1,\mathsf{S}(n)) \cdot |C|^2\right)$. Finally, $\mathcal{C}_{\mathsf{dec}}$ contains a decoding sub-circuit for each of the $O(\widehat{n}(1,\mathsf{S}(n)))$ masking inputs used in the $O\left(|\mathbb{F}| \cdot \widehat{n}^2(1,\mathsf{S}(n)) \cdot |C|^2\right)$ gadgets of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$. Because E is linear, each of these decoding sub-circuits consists of $\widehat{n}(1,\mathsf{S}(n))$ $\times$ gates followed by $\widehat{n}(1,\mathsf{S}(n))$ $+$ gates. In addition, $\mathcal{C}_{\mathsf{dec}}$ contains a "binarization" component $\mathcal{T}$ of size $O(|\mathbb{F}|)$ for each decoding sub-circuit, followed by $O\left(|\mathbb{F}| \cdot \widehat{n}^3(1,\mathsf{S}(n)) \cdot |C|^2\right)$ $\times$ gates, so overall $|\mathcal{C}_{\mathsf{dec}}| \le O\left(|\mathbb{F}|^2 \cdot \widehat{n}^3(1,\mathsf{S}(n)) \cdot |C|^2\right)$. □

The proof of Claim 10 used the following fact regarding the LRCC of [42].

Construction 11 (SAT-Respecting LRCC). *Let σ be a security parameter, and $\mathsf{M} = \mathsf{M}(\sigma), \mathsf{M}_0 = \mathsf{M}_0(\sigma) : \mathbb{N} \to \mathbb{N}$ (the value of these parameters will be set in Remark 11 below). Let $(\mathsf{Comp}^\star, (\mathsf{Enc}^\star, \mathsf{Dec}^\star))$ denote the LRCC of [42], and let $\left(\mathsf{Enc}^{\mathsf{in}} : \mathbb{F}^n \times \{1\}^* \to \mathbb{F}^{\widehat{n}_{\mathsf{in}}(n,\sigma)}, \mathsf{Dec}^{\mathsf{in}}\right)$ denote the internal encoding scheme which it employs.*
The SAT-respecting LRCC $(\mathsf{Comp}, \mathsf{E} = (\mathsf{Enc}, \mathsf{Dec}))$ is defined as follows.

The Encoding Scheme. $\mathsf{Enc} : \mathbb{F}^n \times \{1\}^* \to \mathbb{F}^{2\widehat{n}_{\mathsf{in}}+2\widehat{\mathsf{M}}+2\widehat{\mathsf{M}_0}}$ *on input $x, 1^\sigma$ outputs an encoding $(\widehat{x}_1, \widehat{x}_2)$, where $\widehat{x}_i \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(\left(x, 0^{\mathsf{M}+\mathsf{M}_0}\right), 1^\sigma\right)$, and $\widehat{n}_{\mathsf{in}}, \widehat{\mathsf{M}}, \widehat{\mathsf{M}_0}$ denote the lengths of encodings of messages of length $m, \mathsf{M}, \mathsf{M}_0$, respectively. Dec on input $(\widehat{x}_1, \widehat{x}_2), 1^\sigma$ computes $\mathsf{Dec}^{\mathsf{in}}(\widehat{x}_1, 1^\sigma)$, and discards the last $\mathsf{M} + \mathsf{M}_0$ field elements. We use $\widehat{n} = \widehat{n}(n, \sigma)$ to denote the length of encodings output by Enc. For $(\widehat{x}_1, \widehat{x}_2) \leftarrow \mathsf{Enc}(x, 1^\sigma)$, we interpret $\widehat{x}_i = \left(\widehat{x}_i^{\mathsf{in}}, \mathbf{r}^i, \mathbf{r}^{i,0}\right)$, where $\widehat{x}_i^{\mathsf{in}}$ encodes x, and $\mathbf{r}^i, \mathbf{r}^{i,0}$ encode $0^{\mathsf{M}}, 0^{\mathsf{M}_0}$, respectively. (We note that $\mathbf{r}^{2,0}$ is not used by the leakage-resilient circuit, but is included in $\widehat{x}_2$ because the same internal encoding scheme $\mathsf{Enc}^{\mathsf{in}}$ is used to generate both $\widehat{x}_1$ and $\widehat{x}_2$.)*

The Compiler Comp. *On input a circuit $C : \mathbb{F}^n \to \mathbb{F}$, the compiler Comp outputs the circuit $\widehat{C} : \mathbb{F}^{\widehat{n}(n,|C|)} \to \mathbb{F}$ that on input $\left(\widehat{x}_1^{\mathsf{in}}, \mathbf{r}^1, \mathbf{r}^{1,0}\right), \left(\widehat{x}_2^{\mathsf{in}}, \mathbf{r}^2, \mathbf{r}^{2,0}\right)$ operates as follows.*

- *Let $C_1 = C_2 = C$ denote two copies of C, let $\widehat{C}_i = \mathsf{Comp}^\star(C_i)$ for $i = 1, 2$, and $\widehat{\mathcal{C}}_{0-\mathsf{check}} = \mathsf{Comp}^\star(\mathcal{C}_{0-\mathsf{check}})$.*
- *$\widehat{C}$ computes*

$$\mathsf{f} := \mathcal{T}\left(\widehat{C}_1\left(\widehat{x}_1^{\mathsf{in}}, \mathbf{r}^1\right) - \widehat{C}_2\left(\widehat{x}_2^{\mathsf{in}}, \mathbf{r}^2\right)\right) \times \widehat{\mathcal{C}}_{0-\mathsf{check}}\left(\left(\mathbf{r}^1, \mathbf{r}^2\right), \mathbf{r}^{1,0}\right) \times \mathcal{C}_{\mathsf{dec}}\left(\mathbf{r}^{1,0}\right).$$

  (Notice that $\mathsf{f} = 1$ if and only if (1) $\widehat{C}_1\left(\widehat{x}_1^{\mathsf{in}}, \mathbf{r}^1\right) = \widehat{C}_2\left(\widehat{x}_2^{\mathsf{in}}, \mathbf{r}^2\right)$, and (2) the masking inputs used in at least one of them, as well as in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$, are well-formed. Otherwise, $\mathsf{f} = 0$.)
- *$\widehat{C}$ outputs*

$$\left(1 - \mathsf{f}\left(\left(\widehat{x}_1^{\mathsf{in}}, \mathbf{r}^1, \mathbf{r}^{1,0}\right), \left(\widehat{x}_2^{\mathsf{in}}, \mathbf{r}^2, \mathbf{r}^{2,0}\right)\right)\right) + \mathsf{f}\left(\left(\widehat{x}_1^{\mathsf{in}}, \mathbf{r}^1, \mathbf{r}^{1,0}\right), \left(\widehat{x}_2^{\mathsf{in}}, \mathbf{r}^2, \mathbf{r}^{2,0}\right)\right) \cdot \widehat{C}_1\left(\widehat{x}_1^{\mathsf{in}}, \mathbf{r}^1\right)$$

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

Figure 2. SAT-Respecting LRCC [8].

Remark 11 (Setting the parameters). *Let $\mathsf{M}^\star = \mathsf{M}^\star(\sigma)$ denote the maximal number of masking inputs used in a gadget of $\mathsf{Comp}^\star$, and $\mathsf{S}_0(\mathsf{M}')$ denote the size of $\mathcal{C}_{0-\mathsf{check}}$ on inputs of length M'. Then $\mathsf{M}(\sigma) = \sigma \cdot \mathsf{M}^\star$ and $\mathsf{M}_0(\sigma) = \mathsf{M}^\star \cdot \mathsf{S}_0(\mathsf{M}) = \mathsf{M}^\star \cdot \mathsf{S}_0(\sigma \cdot \mathsf{M}^\star)$.*

Fact 12. *Each gadget generated by the LRCC $(\mathsf{Comp}^\star, \mathsf{E}^\star)$ of [42] has size at most $O\left(\widehat{n}^2(1, \mathsf{S}(n))\right)$, and uses at most $O(\widehat{n}(1, \mathsf{S}(n)))$ masking inputs.*

Lemma 2. *If E^{in} is linear and onto, then Construction 11 is SAT-respecting.*

Proof. Assume that $\widehat{C}(\widehat{x}) = 0$ for some $\widehat{x} \in \mathbb{F}^{\widehat{n}}$, and denote $\widehat{x} = \left(\left(\widehat{x}_1^*, \mathbf{r}^1, \mathbf{r}^{1,0}\right), \left(\widehat{x}_2^*, \mathbf{r}^2, \mathbf{r}^{2,0}\right)\right)$. We show that there exists an $x \in \mathbb{F}^n$ such that $C(x) = 0$. Since $\widehat{C}(\widehat{x}) = 0$ then by the definition of $\widehat{C}$, we have (1) $\mathsf{f} = 1$, and (2) $\widehat{C}_1\left(\widehat{x}_1^*, \mathbf{r}^1\right) = 0$. By (1), $\widehat{C}_1\left(\widehat{x}_1^*, \mathbf{r}^1\right) = \widehat{C}_2\left(\widehat{x}_2^*, \mathbf{r}^2\right)$, and $\mathcal{C}_{\mathsf{dec}}, \mathcal{C}_{0-\mathsf{check}}$ output 1. Notation 8 then guarantees that $\mathbf{r}^{1,0}$ is well-formed which—by the correctness of $\mathsf{Comp}^\star$—guarantees that $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ emulates $\mathcal{C}_{0-\mathsf{check}}$. (Here, we also use the fact that $\mathcal{C}_{\mathsf{dec}}$ is independent of all other components of, and inputs to, $\widehat{C}$.) Moreover, since the encoding scheme is onto then $\mathbf{r}^1, \mathbf{r}^2$ define inputs to $\mathcal{C}_{0-\mathsf{check}}$ which cause it to output 1(because $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ outputs 1, and its masking inputs are well-formed). Notation 7 then guarantees that at least one of $\mathbf{r}^1, \mathbf{r}^2$ is well-formed. Assuming (without loss of generality) that $\mathbf{r}^1$ is well-formed, then the correctness of $\mathsf{Comp}^\star$ guarantees that $\widehat{C}_1$ emulates C, so

$0 = \widehat{C}_1(\widehat{x}_1^*, \mathbf{r}^1) = C(x)$, where $x \in \mathbb{F}^n$ is obtained by computing $x = \mathsf{Dec}^{\mathsf{in}}(\widehat{x}_1^*)$ (x is well-defined because E^{in} is onto). □

Lemma 3. *If* E^{in} *is* $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$*-leakage resilient, then for every function class* $\mathcal{LEAK}$ *such that* $\mathcal{LEAK} \circ \mathcal{SHALLOW}(7, O(\widehat{n}^5(\mathsf{S}(n)) \cdot \mathsf{S}(n))) \subseteq \mathcal{LEAK}_{\mathsf{E}}$*, every circuit* $C : \mathbb{F}^n \to \mathbb{F}$ *of size* $|C| \leq \mathsf{S}(n)$*, and every* $x, y \in \mathbb{F}^n$ *such that* $C(x) = C(y)$*, it holds that* $\left[\widehat{C}, \widehat{x}\right]$ *and* $\left[\widehat{C}, \widehat{y}\right]$ *are* $(\mathcal{LEAK}, 4\epsilon(n) \cdot (\widehat{n}(1, \mathsf{S}(n)) + 1) \cdot \mathsf{S}(n))$*-leakage resilient, where* $\widehat{x} \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$ *and* $\widehat{y} \leftarrow \mathsf{Enc}\left(y, 1^{|C|}\right)$.

Proof of Lemma 3

In this section we prove the leakage-resilience property of Construction 11. The analysis follows a (by now standard) proof paradigm for gadget-based LRCCs, but is more complex (compared to, e.g., the analysis in [42]) because of the mask-checker and mask-decoder components.

High-Level Description of the Leakage-Resilience Argument. Recall that the goal is to prove that leakage functions in some leakage class $\mathcal{LEAK}$ cannot distinguish between the wire values of $\widehat{C}$ when evaluated on (encodings of) two different inputs x, y. This is achieved by reduction to the leakage resilience of the underlying encoding scheme E against leakage classes in a somewhat larger leakage class $\mathcal{LEAK}_{\mathsf{E}}$. The wire bundles of $\widehat{C}$ are divided into two sets: *internal* bundles that are part of the internal computations in a gadget, and *external* bundles that connect two gadgets. The proof uses a hybrid argument in which we first replace all the internal bundles from carrying the real encodings to "simulated" encodings, and then replace all external bundles from encodings of the real values to encodings of random values. We then prove leakage resilience by showing that each pair of adjacent hybrids are computationally indistinguishable.

More specifically, consider two hybrid distributions $\mathcal{H}, \mathcal{H}'$ in which we replace some external bundle i. Proving that $\mathcal{H} \approx \mathcal{H}'$ is by reduction to the leakage resilience of the underlying encoding scheme, where we show that if $\mathcal{H}$ and $\mathcal{H}'$ are distinguishable by a function $\ell \in \mathcal{LEAK}$, then there exists a function $\ell_{\mathsf{E}} \in \mathcal{LEAK}_{\mathsf{E}}$ that can distinguish between encodings of two different values v, v'. The idea is that given an encoding $\mathbf{e}$ of either v or v', ℓ_{E} would use $\mathbf{e}$ to generate *the entire hybrid distribution* (which would be either $\mathcal{H}$ or $\mathcal{H}'$, depending on whether $\mathbf{e}$ encodes v or v', respectively), then evaluate ℓ. It turns out that the only wires of $\mathcal{H}, \mathcal{H}'$ that depend on $\mathbf{e}$ are the internal wire bundles of the (at most two) gadgets that "touch" bundle i, in the sense that bundle i is either an input or output bundle of the gadget. Thus, while all other wire bundles can be hard-wired into ℓ_{E}, it would still need to generate the internal wire bundles of the gadget(s) that touch bundle i. In particular, the computational complexity of ℓ_{E} would be higher than that of ℓ, which causes a loss in leakage resilience. It is therefore imperative that the internal wire values of the gadgets could be generated from their inputs and outputs by a function in a low complexity class. These functions are called *local reconstructors*. The crucial point here is that the local reconstructor is given not only the gadget's inputs, but also its *outputs* (otherwise, it would be impossible to generate the internal wires in a low complexity class, because some of the gadgets perform complex computations). Before delving deeper into the details of the analysis, we first set the needed terminology regarding gadgets and their local reconstructors.

Properties of Gadgets. The analysis will use two properties of gadgets. The first is *local reconstructibility*: for every pair of "legal" input and output encodings, the internal wires of the gadget (as determined by the input-output pair, and its masking inputs), can be simulated in a low complexity class.

Definition 14 (Local reconstructibility)**.** *Let* $\mathcal{G}$ *be a gadget and* $\mathcal{WF}$ *denote its set of well-formed masking inputs. A pair* $(\mathbf{x}, \mathbf{y})$ *of encodings is* plausible *for* $\mathcal{G}$ *if* $\mathcal{G}(\mathbf{x}, \mathsf{m}) = \mathbf{y}$ *for some* $\mathsf{m} \in \mathcal{WF}$*. For* $\epsilon > 0$*, and families* $\mathcal{LEAK}, \mathcal{F}_{\mathcal{G}}$ *of functions,* $\mathcal{G}$ *is* $(\mathcal{LEAK}, \epsilon)$-reconstructible by $\mathcal{F}_{\mathsf{G}}$ *if the following holds. There exists a distribution* REC *over functions* rec *such that:*

- $\mathsf{Supp}(\mathsf{REC}) \subseteq \mathcal{F}_{\mathcal{G}}$.

- rec *takes as input* $\mathcal{G}$*'s inputs and output, and simulates the masking inputs, and internal wires, of* $\mathcal{G}$*.*
- *The following distributions are* $(\mathcal{LEAK}, \epsilon)$*-leakage resilient for any plausible pair* $(\mathbf{x}, \mathbf{y})$ *for* $\mathcal{G}$*: (1)* $\mathsf{rec}(\mathbf{x}, \mathbf{y})$ *for* $\mathsf{rec} \leftarrow \mathsf{REC}$*; and (2) the actual distribution of the wires of* $\mathcal{G}$ *(as determined by the distribution of the masking inputs), conditioned on* $\mathbf{x}, \mathbf{y}$*.*

We will need the following result of [42] which shows that all gadgets in their LRCC $(\mathsf{Comp}^\star, \mathsf{E}^\star)$ are locally reconstructible in a low complexity class. (We note that Faust et al. [42] actually prove a stronger result, where some of the gadgets have local reconstructors in lower complexity classes than the one stated here, and indistinguishability holds regardless of the leakage resilience of E. For clarity reasons, we chose to give a simplified and weaker version of their results which nonetheless suffices for our needs).

Lemma 4 (Gadgets are locally reconstructible [42])**.** *Let* $\sigma \in \mathbb{N}$ *be a security parameter, let* $n : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *be a length parameter, let* $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$*, and let* $\mathcal{LEAK}, \mathcal{LEAK}_{\mathsf{E}}$ *be families of functions such that* $\mathcal{LEAK}_{\mathsf{E}} = \mathcal{LEAK} \circ \mathcal{SHALLOW}(3, O(\widehat{n}(1,\sigma)))$*. Let* $(\mathsf{Comp}^\star, \mathsf{E}^\star)$ *denote the LRCC of [42], and let* E^{in} *denote the internal encoding scheme which it uses. If* E^{in} *is* $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$*-leakage resilient, then all gadgets used in* $(\mathsf{Comp}^\star, \mathsf{E}^\star)$ *are* $(\mathcal{LEAK}, \widehat{n}(1,\sigma) \cdot \epsilon(n))$*-reconstructible by* $\mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1,\sigma)\right)\right)$*.*

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

Definition 15 (Gadget Re-Randomization)**.** *A gadget* $\mathcal{G}$ *with set* $\mathcal{WF}$ *of well-formed masking inputs is* re-randomizing *if for every standard input* $\mathbf{x} = \mathsf{Enc}(x)$*, when the masking input is sampled* $\mathsf{m} \leftarrow \mathcal{WF}$ *then* $\mathcal{G}(\mathbf{x}, \mathsf{m})$ *is random subject to encoding the correct output (as determined by* x*, and the gate which* $\mathcal{G}$ *emulates).*

The Hybrid Argument. Let $C : \mathbb{F}^n \to \mathbb{F}$ be an arithmetic circuit of size $|C| \leq \mathsf{S}(n)$, and let $x, y \in \mathbb{F}^n$. We show that for every $\ell \in \mathcal{LEAK}$,

$$\mathsf{SD}\left(\ell\left[\widehat{C}, \widehat{x}\right], \ell\left[\widehat{C}, \widehat{y}\right]\right) \leq \epsilon'(n)$$

where $\epsilon'(n) := 4\epsilon(n) \cdot (\widehat{n}(1, \mathsf{S}(n)) + 1) \cdot \mathsf{S}(n)$. We bound the statistical distance using a hybrid argument. We define:

$$\mathcal{H}^x := \left(\left[\widehat{C_1}, \widehat{x}_1, \mathbf{r}^1\right], \left[\widehat{C_2}, \widehat{x}_2, \mathbf{r}^2\right], \left[\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathbf{r}^1, \mathbf{r}^2, \mathbf{r}^{1,0}\right], \left[\mathcal{C}_{\mathsf{dec}}, \mathbf{r}^{1,0}\right]\right)$$
$$\mathcal{H}^y := \left(\left[\widehat{C_1}, \widehat{y}_1, \mathbf{r}^1\right], \left[\widehat{C_2}, \widehat{y}_2, \mathbf{r}^2\right], \left[\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathbf{r}^1, \mathbf{r}^2, \mathbf{r}^{1,0}\right], \left[\mathcal{C}_{\mathsf{dec}}, \mathbf{r}^{1,0}\right]\right)$$
$$\mathcal{H}^{y,x} := \left(\left[\widehat{C_1}, \widehat{y}_1, \mathbf{r}^1\right], \left[\widehat{C_2}, \widehat{x}_2, \mathbf{r}^2\right], \left[\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathbf{r}^1, \mathbf{r}^2, \mathbf{r}^{1,0}\right], \left[\mathcal{C}_{\mathsf{dec}}, \mathbf{r}^{1,0}\right]\right)$$

and notice that $\mathcal{H}^x$ are the wire values of $\widehat{C}$ on input an encoding of x, $\mathcal{H}^y$ are the wire values of $\widehat{C}$ on input an encoding of y, and $\mathcal{H}^{y,x}$ is a hybrid distribution, consisting of the wire values of $\widehat{C}$ when the first copy $\widehat{C}_1$ has input (an encoding according to $\mathsf{Enc}^{\mathsf{in}}$) of x, whereas the second copy $\widehat{C}_2$ has input (an encoding according to $\mathsf{Enc}^{\mathsf{in}}$) of y. We note that $\mathcal{H}^{y,x}$ is only used in the proof—it is never obtained in an actual evaluation of $\widehat{C}$. The proof proceeds by showing that for every $\ell \in \mathcal{LEAK}$, $\mathsf{SD}(\ell(\mathcal{H}^x), \ell(\mathcal{H}^{y,x}))$ and $\mathsf{SD}(\ell(\mathcal{H}^{y,x}), \ell(\mathcal{H}^y))$ are upper bounded by $2\epsilon(n) \cdot (\widehat{n}(1, \mathsf{S}(n)) + 1) \cdot \mathsf{S}(n)$, through a sequence of hybrids.

Bounding $\mathsf{SD}(\ell(\mathcal{H}^x), \ell(\mathcal{H}^{y,x}))$ **for a leakage function** ℓ**.** The difference between these distributions is that in $\mathcal{H}^x$ both copies $\widehat{C}_1, \widehat{C}_2$ of (the leakage-resilient version of) C are evaluated on input x, whereas in $\mathcal{H}^{y,x}$ the first is evaluated on y (and the second on x). We bound the statistical distance through a sequence of hybrids, defined as follows.

$\mathcal{H}^x_{\mathsf{in}}$**:** this hybrid distribution replaces the internal wires of gadgets, and is obtained by: (1) evaluating $\widehat{C}$ honestly on $\widehat{x} \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$; (2) picking local reconstructors for

all gadgets of $\widehat{C}_1$ (such reconstructors exist by Lemma 4), and re-computing their internal wires using these reconstructors; and (3) re-evaluating $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ on the new masking inputs generated for the gadgets of $\widehat{C}_1$, by re-computing the internal wires of all gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$, the inputs of which are masking inputs used in gadgets of $\widehat{C}_1$—but without changing the masking inputs these $\mathcal{G}'$ gadgets use, or their outputs. Lemma 5 below shows that this is indeed possible. (Crucially, since re-evaluating $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ does not influence its masking inputs, this does not influence the computation in $\mathcal{C}_{\mathsf{dec}}$, so there is no need to re-evaluate it.)

$\mathcal{H}^x_{\mathsf{ext}}$**:** this distribution replaces the external wires (i.e., wires connecting gadgets), and is obtained as follows:

- *Generating the wires of* $\widehat{C}_2$*:* encode $\widehat{x} = \left(\left(\widehat{x}_1, \mathbf{r}^1, \mathbf{r}^{1,0}\right), \left(\widehat{x}_2, \mathbf{r}^2, \mathbf{r}^{2,0}\right)\right) \leftarrow \mathsf{Enc}\left(x, 1^{|C|}\right)$, and honestly evaluate $\widehat{C}_2$ on $\widehat{x}_2$ with masking inputs $\mathbf{r}^2$.
- *Generating the wires of* $\widehat{C}_1$*:* pick a random encoding $\mathsf{out} \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(1, 1^{|C|}\right)$ for the output of $\widehat{C}_1$, and honestly compute the wires of the output decoder of $\widehat{C}_1$. (As discussed in Section 4.1, $\widehat{C}_1$ contains an output decoder, which is needed because the computations in $\widehat{C}_1$ are performed over encodings.) Then, pick a random input $z \in_R \mathbb{F}^n$ for $\widehat{C}_1$ and encode $\widehat{z}_1 \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(z, 1^{|C|}\right)$. Next, pick random encodings (according to $\mathsf{Enc}^{\mathsf{in}}$) for the outputs of all gadgets (except the gadgets the outputs of which are the inputs of the output decoder, since the outputs of these gadgets have already been fixed). This effectively determines the standard inputs, and outputs, of all gadgets of $\widehat{C}_1$. Next, pick local reconstructors for all gadgets of $\widehat{C}_1$, and use them to compute the internal wires of the gadgets. The reconstructors determine the (possibly ill-formed) masking inputs of the gadgets, which we denote by $\mathbf{r}^{1\prime}$. $\mathbf{r}^{1\prime}$ together with $\mathbf{r}^2$ form the standard inputs of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$.
- *Generating the wires of* $\widehat{\mathcal{C}}_{0-\mathsf{check}}$*:* Evaluates $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ on $\mathbf{r}^{1\prime}, \mathbf{r}^2$, with masking inputs $\mathbf{r}^{1,0}$.
- *Generating the wires of* $\mathcal{C}_{\mathsf{dec}}$*:* Evaluate $\mathcal{C}_{\mathsf{dec}}$ on $\mathbf{r}^{1,0}$.
- Use the outputs of $\widehat{C}_1, \widehat{C}_2, \widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathcal{C}_{\mathsf{dec}}$ to generate the flag f, and the output of $\widehat{C}$.
- $\mathcal{H}^x_{\mathsf{ext}}$ consists of the concatenation of all these wire values.

$\mathcal{H}^{y,x}_{\mathsf{ext}}$**:** this hybrid is generated similarly to $\mathcal{H}^x_{\mathsf{ext}}$, except that instead of evaluating $\widehat{C}$ on an encoding of x, we use the internal encoding scheme to generate encodings of $\left(\widehat{y}_1, \mathbf{r}^1, \mathbf{r}^{1,0}\right)$ and $\left(\widehat{x}_2, \mathbf{r}^2, \mathbf{r}^{2,0}\right)$ (where $\widehat{y}_1, \widehat{x}_2$ encode y, x, respectively), and use them as inputs to $\widehat{C}_1, \widehat{C}_2$, respectively.

$\mathcal{H}^{y,x}_{\mathsf{in}}$**:** this hybrid is generated similarly to $\mathcal{H}^x_{\mathsf{in}}$, except that instead of evaluating $\widehat{C}$ on an encoding of x, we use $\left(\widehat{y}_1, \mathbf{r}^1, \mathbf{r}^{1,0}\right)$ and $\left(\widehat{x}_2, \mathbf{r}^2, \mathbf{r}^{2,0}\right)$ as inputs to $\widehat{C}_1, \widehat{C}_2$, respectively.

We now bound the statistical distance between the outputs of leakage functions on each pair of adjacent hybrids. The following notation will be useful.

Notation 9. *We say that a gadget* $\mathcal{G}'$ *of* $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ *is* connected *to a gadget* $\mathcal{G}_1$ $\widehat{C}_1$ *(alternatively, a gadget* $\mathcal{G}_2$ *of* $\widehat{C}_2$*) if an input of* $\mathcal{G}'$ *is a masking input of* $\mathcal{G}_1$ *(alternatively,* $\mathcal{G}_2$*).*

Bounding $\mathsf{SD}\left(\ell(\mathcal{H}^x), \ell\left(\mathcal{H}^x_{\mathsf{in}}\right)\right)$**:** we show that $\mathsf{SD}\left(\ell(\mathcal{H}^x), \ell\left(\mathcal{H}^x_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\left(5, O\left(\widehat{n}^5(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. We use a hybrid argument, replacing the internal wires of the $M \leq \mathsf{S}(n)$ gadgets of $\widehat{C}_1$ one at a time. We define hybrids $\mathcal{H}_0, \ldots, \mathcal{H}_M$ where $\mathcal{H}_i$ is obtained by evaluating $\widehat{C}$ on (an encoding of) x, then recomputing the internal wires of *the first* i *gadgets of* $\widehat{C}_1$ using their local reconstructors. Additionally, we recompute the internal wires of gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to one of these i gadgets, but *without changing the masking inputs*

of $\mathcal{G}'$ (this is possible by Lemma 5 below). Then $\mathcal{H}_0 = \mathcal{H}^x, \mathcal{H}_M = \mathcal{H}^x_{\mathsf{in}}$. To show that $\mathsf{SD}\big(\ell(\mathcal{H}^x), \ell\big(\mathcal{H}^x_{\mathsf{in}}\big)\big) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$, we show that for every $m \in [M]$, and any $\ell \in \mathcal{LEAK}'$, it holds that $\mathsf{SD}(\ell(\mathcal{H}_m), \ell(\mathcal{H}_{m-1})) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n))$. Denote the m'th gadget by $\mathcal{G}$. We use Lemma 4 to show that this follows from the leakage resilience of the internal encoding scheme E_{in}. For this, we will need to generate—in a low complexity class—the entire hybrid distributions given only the (either real or reconstructed) internal wires of $\mathcal{G}$. The problem is that since the internal wires of $\mathcal{G}$ contain also its masking inputs, changing them affects also computations in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$. The following lemma from [8] states that the influence of modifying the masking inputs of $\mathcal{G}$ can be blocked, specifically: (1) that it only affects the internal wires of gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}$, and more importantly (2) these internal wire values can be reconstructed *without* changing the masking inputs used in $\mathcal{G}'$.

Lemma 5 (Restating of Lemma 3.13 of [47])**.** *Let $\mathcal{G}'$ be a gadget of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to gadgets of $\widehat{\mathcal{C}}_1, \widehat{\mathcal{C}}_2$. Then for any fixed well-formed inputs r^1, r^2 from gadgets of $\widehat{\mathcal{C}}_1, \widehat{\mathcal{C}}_2$ (respectively), and any fixed well-formed masking inputs m for $\mathcal{G}'$, the following holds. For any $r^{1\prime}$, the internal wires of $\mathcal{G}'$ on input $r^{1\prime}, r^2, \mathsf{m}$ can be computed in $\mathcal{SHALLOW}\big(2, O\big(\widehat{n}^2(1, \mathsf{S}(n))\big)\big)$ given $r^{1\prime}$ and the output out of $\mathcal{G}'$ when evaluated on input r^1, r^2 and masking inputs m.*

Using an averaging argument, we can fix all wires of $\widehat{\mathcal{C}}_2$, and all wires of $\widehat{\mathcal{C}}_1$ except the internal wires of $\mathcal{G}$ (its input and output wires *can* be fixed). Lemma 5 (which can be used because $\widehat{\mathcal{C}}_2$ and $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ use well-formed masking inputs) shows that we can further fix all wires of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ and $\mathcal{C}_{\mathsf{dec}}$, except for the internal wires of all gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}$ (crucially, the masking inputs of these gadgets *can* be fixed).

Consequently, given the wire values of $\mathcal{G}$ (either the real values $\mathcal{W}_R$, or the reconstructed wires $\mathcal{W}_S$), we can generate the entire hybrid distribution (either $\mathcal{H}_{m-1}$ or $\mathcal{H}_m$, respectively) in $\mathcal{SHALLOW}\big(2, O\big(\widehat{n}^5(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\big)\big)$ by recomputing the internal wires of each of the (at most) $O\big(\widehat{n}^3(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\big)$ gadgets of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}$. (Here, we use Fact 12—$\mathcal{G}$ uses at most $\widehat{n}(1, \mathsf{S}(n))$ masking inputs; $\widehat{\mathcal{C}}_2$ has at most $O(\widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n))$ gadgets, each using at most $\widehat{n}(1, \mathsf{S}(n))$ masking inputs; and each masking input of $\mathcal{G}$ is connected to every masking input used in $\widehat{\mathcal{C}}_2$.) Using Lemma 5, the internal wires of each $\mathcal{G}'$ can be computed in $\mathcal{SHALLOW}\big(2, O\big(\widehat{n}^2(1, \mathsf{S}(n))\big)\big)$, and all these computations can be performed in parallel. Therefore, the hybrid distributions can be generated in $\mathcal{SHALLOW}\big(2, O\big(\widehat{n}^5(1, \mathsf{S}(n))\big) \cdot \mathsf{S}(n)\big)$. By the assumption of Claim 10, E^{in} is $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$-leakage resilient, which by Lemma 4 implies that $\mathcal{W}_R, \mathcal{W}_S$ are $(\mathcal{LEAK}'', \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)))$-leakage resilient for any class $\mathcal{LEAK}''$ of leakage functions such that $\mathcal{LEAK}'' \circ \mathcal{SHALLOW}\big(3, O\big(\widehat{n}(1, \mathsf{S}(n))\big)\big) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. We now claim that $\mathcal{H}_{m-1}, \mathcal{H}_m$ are $(\mathcal{LEAK}', \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)))$-leakage resilient for every family $\mathcal{LEAK}'$ of leakage functions such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\big(2, O\big(\widehat{n}^5(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\big)\big) \subseteq \mathcal{LEAK}''$, i.e., for any $\mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\big(5, O\big(\widehat{n}^5(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\big)\big) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. Indeed, this follows from the following lemma of [42] (by choosing $\mathcal{F}$ to be a singleton containing the function described above which generates $\mathcal{H}_{m-1}, \mathcal{H}_m$ from $\mathcal{W}_R, \mathcal{W}_S$).

Lemma 6 ([42])**.** *Let $n \in \mathbb{N}$, let $\mathcal{W}_R, \mathcal{W}_S$ be distributions over $\mathbb{F}^n$, let $\mathcal{LEAK}, \mathcal{F}$ be families of functions, and let $\epsilon > 0$. Let $\mathcal{D}$ be a distribution over functions in $\mathcal{F}$ of input length n. For $f \leftarrow \mathcal{D}$, let $\mathcal{W}'_R := f(\mathcal{W}_R), \mathcal{W}'_S := f(\mathcal{W}_S)$. If $\mathcal{W}_R, \mathcal{W}_S$ are $(\mathcal{LEAK}, \epsilon)$-leakage resilient, then $\mathcal{W}'_R, \mathcal{W}'_S$ are $(\mathcal{LEAK}', \epsilon)$-leakage resilient for any family $\mathcal{LEAK}'$ of leakage functions such that $\mathcal{LEAK}' \circ \mathcal{F} \subseteq \mathcal{LEAK}$.*

Using Lemma 6, for any $\ell \in \mathcal{LEAK}'$ we have

$$\mathsf{SD}(\ell(\mathcal{H}_m), \ell(\mathcal{H}_{m-1})) = \mathsf{SD}(\ell(\mathcal{W}_S), \ell(\mathcal{W}_R)) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)).$$

Bounding $\mathsf{SD}\big(\ell(\mathcal{H}^x_{\mathsf{ext}}), \ell\big(\mathcal{H}^x_{\mathsf{in}}\big)\big)$: We show that $\mathsf{SD}\big(\ell(\mathcal{H}^x_{\mathsf{ext}}), \ell\big(\mathcal{H}^x_{\mathsf{in}}\big)\big) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\big(4, O\big(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\big)\big) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. We again use a hybrid argument, this time replacing the $M \leq \mathsf{S}(n)$ inputs bundles of $\widehat{\mathcal{C}}_1$ (i.e.,

bundles corresponding to input wires of C), and bundles at the output of gadgets of $\widehat{C}_1$ (except for bundles that are used as input of the output decoder of $\widehat{C}_1$). We define hybrids $\mathcal{H}_0, \ldots, \mathcal{H}_M$, where $\mathcal{H}_i$ is generated as follows. We evaluate $\widehat{C}$ on a random encoding of x. Then, we replace the first i bundles with random encodings of random values, except if one of these bundles corresponds to the output wire of C, in which case we replace it with a random encoding of 1. Finally, we recompute the internal wires of the gadgets of $\widehat{C}_1$ using their gadget reconstructors (Lemma 4), and recompute (using Lemma 5) the internal wires (except the masking inputs and outputs) of all gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to one of these gadgets. In particular, since the inputs of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ contain also masking inputs used in gadgets of $\widehat{C}_1$, this re-computation of wires of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ uses the masking inputs generated by the reconstructors (for any gadget of $\widehat{C}_1$ whose internal wires were re-constructed). Then $\mathcal{H}_0 = H^x_{\mathsf{in}}$ and $\mathcal{H}_M = \mathcal{H}^x_{\mathsf{ext}}$. Therefore, to prove that $\mathsf{SD}\left(\ell(\mathcal{H}^x_{\mathsf{ext}}), \ell\left(H^x_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$, it suffices to prove that $\mathsf{SD}(\ell(\mathcal{H}_m), \ell(\mathcal{H}_{m-1})) \leq \epsilon(n)$ for all $m \in [M]$. Let $\mathcal{G}_o$ ($\mathcal{G}_i$) denote the gadget the output (input) of which is the m'th bundle. (If the m'th bundle is an input bundle, then we consider only the gadget $\mathcal{G}_i$.) Using an averaging argument, we can fix all wires in $\mathcal{H}_m, \mathcal{H}_{m-1}$ except for: the m'th bundle; the masking inputs, outputs, and internal wires of $\mathcal{G}_o$; the masking inputs, and internal wires, of $\mathcal{G}_i$, as well as its input wire corresponding to the m'th bundle; and the internal wires of all gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}_o$ or $\mathcal{G}_i$ (except for the masking inputs and the output of $\mathcal{G}'$, which *can* be fixed, see Lemma 5).

Let b denote the value encoded by the m'th bundle in H^x_{in}. Let $\mathcal{W}_R \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(b, 1^{\mathsf{S}(n)}\right)$, and $\mathcal{W}_S \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(r, 1^{\mathsf{S}(n)}\right)$ for a random r (except when the m'th bundle corresponds to the output wire of C, in which case we set $\mathcal{W}_S \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(1, 1^{\mathsf{S}(n)}\right)$). Then $\mathcal{W}_R, \mathcal{W}_S$ are distributed identically to the m'th bundle in $\mathcal{H}_{m-1}, \mathcal{H}_m$, respectively. We define a distribution $\mathcal{F}$ over $\mathcal{SHALLOW}\left(4, O\left(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$ as follows. Sampling a function $f \leftarrow \mathcal{F}$ is performed by sampling $\mathsf{rec}_o, \mathsf{rec}_i$ from the distribution over reconstructors for $\mathcal{G}_o, \mathcal{G}_i$, respectively (see Definition 14). The function f has all the hard-wired values of $\mathcal{H}_{m-1}$ hard-wired into it. On input $\mathbf{e} \in \mathbb{F}^{\widehat{n}(1,\mathsf{S}(n))}$, f performs the following: (1) evaluates rec_o on the (hard-wired) inputs of $\mathcal{G}_o$, and the output $\mathbf{e}$ (this reconstructs the masking inputs, and internal wires, of $\mathcal{G}_o$); (2) evaluates rec_i on $\mathbf{e}$ as one of the inputs, and the other (hard-wired) input and output of $\mathcal{G}_i$; and finally (3) for every gadget $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}_o$ or $\mathcal{G}_i$, generates its internal wires using Lemma 5 (without changing the output or masking inputs of $\mathcal{G}'$). Then $f \in \mathcal{SHALLOW}\left(4, O\left(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$. Indeed, by Lemma 4, $\mathsf{rec}_o, \mathsf{rec}_i \in \mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1, \mathsf{S}(n))\right)\right)$ (and given $\mathbf{e}$, they can be evaluated in parallel). Moreover, given the internal wires of $\mathcal{G}_o, \mathcal{G}_i$, the wires that need to be computed in a gadget $\mathcal{G}'$ connected to them are computable in $\mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1, \mathsf{S}(n))\right)\right)$ (by Lemma 5, and since the masking inputs of $\widehat{C}_2$ and $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ are well-formed). We conclude by noting that there are at most $O\left(\widehat{n}^2(1, \mathsf{S}(n))\right) \cdot \mathsf{S}(n)$ such gadgets, and they can be evaluated in parallel.

Denote $\mathcal{W}'_R := f(\mathcal{W}_R), \mathcal{W}'_S := f(\mathcal{W}_S))$ for $f \leftarrow \mathcal{F}$, then $\mathcal{W}'_R \equiv \mathcal{H}_{m-1}$ and $\mathcal{W}'_S \equiv \mathcal{H}_m$ because the gadgets are re-randomizing (which, in particular, guarantees that these equivalences hold *despite* the fact some of the values in the computation were fixed in advance). By the assumption of Lemma 3, $\mathcal{W}_R, \mathcal{W}_S$—which are encodings according to E^{in}—are $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$-leakage resilient. Therefore, by Lemma 6, $\mathcal{W}'_R, \mathcal{W}'_S$ (and consequently also $\mathcal{H}_{m-1}, \mathcal{H}_m$) are $(\mathcal{LEAK}', \epsilon(n))$-leakage resilient for any $\mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\left(4, O\left(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$.

Bounding $\mathsf{SD}\left(\ell(\mathcal{H}^x_{\mathsf{ext}}), \ell\left(\mathcal{H}^{y,x}_{\mathsf{ext}}\right)\right)$: We show that $\mathsf{SD}\left(\ell(\mathcal{H}^x_{\mathsf{ext}}), \ell\left(\mathcal{H}^{y,x}_{\mathsf{ext}}\right)\right) = 0$ for every ℓ. Indeed, $\mathcal{H}^x_{\mathsf{ext}} \equiv \mathcal{H}^{y,x}_{\mathsf{ext}}$ because the hybrids are independent of the input for $\widehat{C}_1$ (since the input is re-sampled as a fresh $z \in_R \mathbb{F}^n$ in both hybrids).

Bounding $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^{y,x}_{\mathsf{in}}\right)\right)$: We show that $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^{y,x}_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ where $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\left(4, O\left(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. The proof is similar to the proof that $\mathsf{SD}\left(\ell(\mathcal{H}^x_{\mathsf{ext}}), \ell(\mathcal{H}^x_{\mathsf{in}})\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$, because the argument was independent of the actual inputs used in $\widehat{C}_1, \widehat{C}_2$ (as long as both hybrids use the same input in each copy).

Bounding $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{in}}\right), \ell(\mathcal{H}^{y,x})\right)$**:** We show that $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{in}}\right), \ell(\mathcal{H}^{y,x})\right) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$, where $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\left(5, O\left(\widehat{n}^5(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. The proof is similar to the proof that $\mathsf{SD}\left(\ell(\mathcal{H}^x), \ell\left(\mathcal{H}^x_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$, because the argument was independent of the actual inputs used in $\widehat{C}_1, \widehat{C}_2$.

Bounding $\mathsf{SD}(\ell(\mathcal{H}^y), \ell(\mathcal{H}^{y,x}))$ **for a leakage function** ℓ**.** The argument here follows the same blueprint as the one used to bound $\mathsf{SD}(\ell(\mathcal{H}^x), \ell(\mathcal{H}^{y,x}))$, but is more involved because we now need to switch the input of the *second copy* $\widehat{C}_2$. In particular, the wire values in this hybrid argument will no longer correspond to the values in an *actual* evaluation of $\widehat{C}_2$. While the computations in $\widehat{C}_1$ will still be performed honestly, we will no longer be able to claim that reconstructing the internal wires of gadgets of $\widehat{C}_2$ (or the external wires connecting gadgets of $\widehat{C}_2$) does not affect the computations in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ and $\mathcal{C}_{\mathsf{dec}}$. This is because the manner in which $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ uses the masking inputs from $\widehat{C}_1, \widehat{C}_2$ is *not symmetric*, and in particular, resampling the masking inputs used in $\widehat{C}_2$ (as is performed by the local reconstructors of gadgets of $\widehat{C}_2$) *will* affect the computations in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$, and will necessitate reevaluating it. This, in turn, will affect the masking inputs used in $\widehat{\mathcal{C}}_{0-\mathsf{check}}$, which will affect the computations in $\mathcal{C}_{\mathsf{dec}}$. However, we cannot simply re-evaluate $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ and $\mathcal{C}_{\mathsf{dec}}$ in the hybrid argument. Indeed, this would require evaluating circuits of large depth, and the leakage resilience guarantee will therefore deteriorate significantly. Instead, Ishai et al. [8] use an alternative method of locally reconstructing the needed wire values of $\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathcal{C}_{\mathsf{dec}}$. Specifically, they show a low-depth reconstructor for the gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to gadgets of $\widehat{C}_2$, which generates the "correct" distribution if the input of $\mathcal{G}'$ from $\widehat{C}_1$, and $\mathcal{G}'$'s output, are well formed. In particular, this implies that the internal wires of $\mathcal{G}'$ can be reconstructed without modifying its output. They also show a low-depth reconstructor for each decoding circuit of $\mathcal{C}_{\mathsf{dec}}$, that generates the "correct" distribution if its inputs are well formed (again, without changing the output of these decoding circuits). Specifically, we will use the following results from [8].

Lemma 7 (Local reconstructors for $\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathcal{C}_{\mathsf{dec}}$, restatement of Lemmas 3.17 and 3.18 of [47])**.** *There exists a distribution* REC *over* $\mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1, \mathsf{S}(n))\right)\right)$ *such that the following holds for any gadget* $\mathcal{G}'$ *of* $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ *connected to a gadget of* $\widehat{C}_1$ *or* $\widehat{C}_2$*. For every plausible pair* $\left(\left(\mathbf{r}^{1,0}, \mathbf{r}^{2,0}\right), \mathbf{c}\right)$ *for* $\mathcal{G}'$ *such that* $\mathbf{r}^{1,0}$ *is well formed, when* $\mathsf{rec} \leftarrow \mathsf{REC}$ *then* $\mathsf{rec}\left(\mathbf{r}^{1,0}, \mathbf{r}^{2,0}, \mathbf{c}\right)$ *is distributed identically to the wire values of* $\mathcal{G}'$ *in a real execution, conditioned on the input-output pair* $\left(\left(\mathbf{r}^{1,0}, \mathbf{r}^{2,0}\right), \mathbf{c}\right)$*. In particular,* $\mathbf{c}$ *and the masking inputs computed by* rec *are well-formed.*

Moreover, for a fixed $\mathsf{rec} \in \mathsf{Supp}(\mathsf{REC})$*, and any fixed well-formed* $\mathbf{r}^{1,0}, \mathbf{c}$*, there exists a function* $\mathsf{rec}^{\mathbf{r}^{1,0}, \mathbf{c}, \mathsf{rec}}_{\mathsf{Dec}} \in \mathcal{SHALLOW}\left(2, O(\widehat{n}(1, \mathsf{S}(n)))\right)$ *such that the following holds for any input* $\mathbf{r}^{2,0}$ *of* $\mathcal{G}'$*. If* $\left(\left(\mathbf{r}^{1,0}, \mathbf{r}^{2,0}\right), \mathbf{c}\right)$ *is a plausible pair for* $\mathcal{G}'$*, then* $\mathsf{rec}^{\mathbf{r}^{1,0}, \mathbf{c}, \mathsf{rec}}_{\mathsf{Dec}}\left(\mathbf{r}^{2,0}\right)$ *generates the wire values of the decoding sub-circuits of* $\mathcal{C}_{\mathsf{dec}}$*, the inputs of which are the masking inputs of* $\mathcal{G}'$ *which* rec *generated. In particular, because the masking inputs generated by* rec *are well-formed, the outputs of these decoding sub-circuits are 0.*

The proof now follows using a hybrid argument. The hybrids are similar to the ones used to bound $\mathsf{SD}(\ell(\mathcal{H}^x), \ell(\mathcal{H}^{y,x}))$, except that we will need to regenerate the internal wires of $\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathcal{C}_{\mathsf{dec}}$ using the reconstructors of Lemma 7. We now define the hybrids.

$\mathcal{H}^y_{\mathsf{in}}$**:** this hybrid distribution replaces the internal wires of gadgets (similar to $\mathcal{H}^x_{\mathsf{in}}$), and is obtained by: (1) evaluating $\widehat{C}$ honestly on $\widehat{y} \leftarrow \mathsf{Enc}\left(y, 1^{|C|}\right)$; (2) picking local reconstructors for all gadgets of $\widehat{C}_2$, and re-computing their internal wires using these reconstructors; (3) re-computing the internal wires of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ using $\mathsf{rec} \leftarrow \mathsf{REC}$, where REC is the distribution of Lemma 7; and (4) re-computing the internal wires of $\mathcal{C}_{\mathsf{dec}}$ using the functions $\mathsf{rec}^{\mathbf{r}^{1,0}, \mathbf{c}, \mathsf{rec}}_{\mathsf{Dec}}$ defined in Lemma 7 (here, $\mathbf{r}^{1,0}, \mathbf{c}$ are determined by the re-computed wires values of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$);

$\mathcal{H}^{y}_{\mathsf{ext}}$**:** this distribution replaces the external wires (similar to $\mathcal{H}^{x}_{\mathsf{ext}}$), and is obtained as follows:

- *Generating the wires of* $\widehat{C}_1$*:* encode $\widehat{y} = \left((\widehat{y}_1, \mathbf{r}^1, \mathbf{r}^{1,0}), (\widehat{y}_2, \mathbf{r}^2, \mathbf{r}^{2,0})\right) \leftarrow \mathsf{Enc}\left(y, 1^{|C|}\right)$, and honestly evaluate $\widehat{C}_1$ on $\widehat{y}_1$ with masking inputs $\mathbf{r}^1$;
- *Generating the wires of* $\widehat{C}_2$*:* pick a random input $z \in_R \mathbb{F}^n$ for $\widehat{C}_2$, and generate random encodings $\mathsf{out} \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(1, 1^{|C|}\right)$, $\widehat{z}_1 \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(z, 1^{|C|}\right)$ for the output and input of $\widehat{C}_2$. Next, pick random encodings (according to $\mathsf{Enc}^{\mathsf{in}}$) for the outputs of all gadgets (except the gadgets whose outputs are the inputs of the output decoder, since the outputs of these gadgets have already been fixed). Then, pick local reconstructors for all gadgets of $\widehat{C}_2$, and use them to compute the internal wires of the gadgets. The reconstructors determine the (possibly ill-formed) masking inputs $\mathbf{r}^{2\prime}$ of the gadgets, which (together with $\mathbf{r}^1$) form the standard inputs of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$;
- *Generating the wires of* $\widehat{\mathcal{C}}_{0-\mathsf{check}}$*:* for every gadget $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to a gadget of $\widehat{C}_2$, pick a reconstructor for it according to the distribution REC (Lemma 7) and use it to compute the internal wires of $\mathcal{G}'$. These reconstructors determine the inputs to the decoding sub-circuits of $\mathcal{C}_{\mathsf{dec}}$;
- *Generating the wires of* $\mathcal{C}_{\mathsf{dec}}$*:* Use the functions of Lemma 7 to compute the internal wires of the decoding sub-circuits of the $\mathcal{C}_{\mathsf{dec}}$;
- Use the outputs of $\widehat{C}_1, \widehat{C}_2, \widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathcal{C}_{\mathsf{dec}}$ to generate the flag f, and the output of $\widehat{C}$;
- $\mathcal{H}^{y}_{\mathsf{ext}}$ consists of the concatenation of all these wire values.

$\mathcal{H}^{y,x}_{\mathsf{ext},2}$**:** this hybrid is generated similarly to $\mathcal{H}^{y}_{\mathsf{ext}}$, except that instead of evaluating $\widehat{C}$ on an encoding of y, we use the internal encoding scheme to generate encodings of $(\widehat{y}_1, \mathbf{r}^1, \mathbf{r}^{1,0})$ and $(\widehat{x}_2, \mathbf{r}^2, \mathbf{r}^{2,0})$ (where $\widehat{y}_1, \widehat{x}_2$ encode y, x, respectively), and use them as inputs to $\widehat{C}_1, \widehat{C}_2$, respectively;

$\mathcal{H}^{y,x}_{\mathsf{in},2}$**:** this hybrid is generated similarly to $\mathcal{H}^{y}_{\mathsf{in}}$, except that instead of evaluating $\widehat{C}$ on an encoding of y, we use $(\widehat{y}_1, \mathbf{r}^1, \mathbf{r}^{1,0})$ and $(\widehat{x}_2, \mathbf{r}^2, \mathbf{r}^{2,0})$ as inputs to $\widehat{C}_1, \widehat{C}_2$, respectively.

The indistinguishability of the hybrids now follows similarly to the proof bounding $\mathsf{SD}(\mathcal{H}^x, \mathcal{H}^{y,x})$, and we only sketch the difference.

Bounding $\mathsf{SD}\left(\ell(\mathcal{H}^y), \ell\left(\mathcal{H}^y_{\mathsf{in}}\right)\right)$**:** We show that $\mathsf{SD}\left(\ell(\mathcal{H}^y), \ell\left(\mathcal{H}^y_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}(7, O(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n))) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. We define the hybrids $\mathcal{H}_0, \ldots, \mathcal{H}_M$, where $\mathcal{H}_i$ is obtained by: (1) evaluating $\widehat{C}$ on (an encoding of) y, then recomputing the internal wires of the first i gadgets *of* $\widehat{C}_2$ using their local reconstructors; (2) *re-computing the internal wires of* $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ that are influenced by this re-computation of the first i gadgets of $\widehat{C}_2$ (i.e., gadgets of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ that are connected to one of these i gadgets); and (3) (using the functions of Lemma 7) *recomputing the internal wires of* $\mathcal{C}_{\mathsf{dec}}$ that were influenced by re-computing $\widehat{\mathcal{C}}_{0-\mathsf{check}}$. Then $\mathcal{H}_0 = \mathcal{H}^y, \mathcal{H}_M = \mathcal{H}^y_{\mathsf{in}}$, and we show that $\mathsf{SD}(\ell(\mathcal{H}_m), \ell(\mathcal{H}_{m-1})) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n))$ for every $m \in [M]$, and any $\ell \in \mathcal{LEAK}'$. Denote the m'th gadget by $\mathcal{G}$. We can fix all wires of $\widehat{C}_1$, all wires of $\widehat{C}_2$ except the internal wires $\mathcal{W}$ of $\mathcal{G}$ (its input and output wires *can* be fixed), and all the internal wires of $\widehat{\mathcal{C}}_{0-\mathsf{check}}, \mathcal{C}_{\mathsf{dec}}$ that are influenced by $\mathcal{W}$. (These consist of the internal wires—but not the output—of any gadget $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}$, and the internal wires—but not the outputs—in the decoding sub-circuits of $\mathcal{C}_{\mathsf{dec}}$ that decode masking inputs of $\mathcal{G}'$.)

We now describe a distribution $\mathcal{F}$ over $\mathcal{SHALLOW}(4, O(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)))$, where given the wire values of $\mathcal{G}$ (either the real wire values $\mathcal{W}_R$ or the reconstructed wires $\mathcal{W}_S$), $f \in \mathcal{F}$ generates the entire hybrid distribution (either $\mathcal{H}_{m-1}$ or $\mathcal{H}_m$, respectively), as follows. For every gadget $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}$, $\mathcal{F}$ chooses a reconstructor $\mathsf{rec}_{\mathcal{G}'} \leftarrow \mathsf{REC}$ (see Lemma 7). The function f has all the hard-wired values of $\mathcal{H}_m$ hard-wired into it. On input $\mathcal{W}$ it evaluates the $\mathsf{rec}_{\mathcal{G}'}$'s on the masking inputs of $\mathcal{G}$ (as

reported in $\mathcal{W}$) and the hard-wired values, to generate the internal wires (including the masking inputs) of $\mathcal{G}'$, and then uses the functions defined in Lemma 7 to reconstruct the internal wires of the decoding sub-circuits of $\mathcal{C}_{\mathsf{dec}}$ that decode the masking inputs of $\mathcal{G}'$. Then $f \in \mathcal{SHALLOW}\left(4, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$ because by Lemma 7 the reconstructors for the $\mathcal{G}'$'s (the decoding sub-circuits, respectively) are computable in $\mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1,\mathsf{S}(n))\right)\right)$ ($\mathcal{SHALLOW}(2, O(\widehat{n}(1,\mathsf{S}(n))))$, respectively); the reconstructors of all the (at most) $O\left(\widehat{n}^2(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)$ gadgets $\mathcal{G}'$ can be computed in parallel); and the reconstructors for all of the (at most) $O\left(\widehat{n}^3(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)$ decoding sub-circuits (each $\mathcal{G}'$ is connected to $O(\widehat{n}(1,\mathsf{S}(n)))$ decoding sub-circuits) can be computed in parallel.

Since $\mathcal{W}_R, \mathcal{W}_S$ are $(\mathcal{LEAK}', \epsilon(n) \cdot \widehat{n}(1,\mathsf{S}(n)))$-leakage resilient for any class $\mathcal{LEAK}'$ of leakage functions such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}(3, O(\widehat{n}(1,\sigma))) \subseteq \mathcal{LEAK}_{\mathsf{E}}$ (this follows from Lemma 4 because E^{in} is $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$-leakage resilient), Lemma 6 guarantees that $\mathcal{H}_{m-1}, \mathcal{H}_m$ are $(\mathcal{LEAK}'', \epsilon(n) \cdot \widehat{n}(1,\mathsf{S}(n)))$-leakage resilient for every family $\mathcal{LEAK}''$ of leakage functions such that $\mathcal{LEAK}'' \circ \mathcal{SHALLOW}\left(4, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}'$, i.e., for any $\mathcal{LEAK}''$ such that $\mathcal{LEAK}'' \circ \mathcal{SHALLOW}\left(7, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$.

Bounding $\mathsf{SD}\left(\ell\left(\mathcal{H}^y_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^y_{\mathsf{in}}\right)\right)$**:** We show that $\mathsf{SD}\left(\ell\left(\mathcal{H}^y_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^y_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}\left(6, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. The proof is by a hybrid argument in which we replace the input bundles *of* $\widehat{C}_2$, and the bundles at the output of gadget *of* $\widehat{C}_2$, one at a time. More specifically, we define hybrids $\mathcal{H}_0, \ldots, \mathcal{H}_M$, where $\mathcal{H}_i$ is generated from $\mathcal{H}^y_{\mathsf{in}}$, by: (1) replacing the first i bundles with random encodings of random values (except for the bundle corresponding to the output of $\widehat{C}_2$, which is set to a random encoding of 1); (2) recomputing the internal wires of the first i gadgets *of* $\widehat{C}_2$ using the gadget reconstructors; (3) re-computing the internal wires of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ that are influenced by this re-computation of the first i gadgets *of* $\widehat{C}_2$; and (4) (using the functions of Lemma 7) recomputing the internal wires of $\mathcal{C}_{\mathsf{dec}}$ that were influenced by re-computing $\widehat{\mathcal{C}}_{0-\mathsf{check}}$. Then $\mathcal{H}_0 = H^y_{\mathsf{in}}$ and $\mathcal{H}_M = \mathcal{H}^y_{\mathsf{ext}}$, and we show that $\mathsf{SD}(\ell(\mathcal{H}_m), \ell(\mathcal{H}_{m-1})) \leq \epsilon(n)$ for all $m \in [M]$ and $\ell \in \mathcal{LEAK}'$. We denote by $\mathcal{G}_o$ ($\mathcal{G}_i$) the gadget whose output (input) is the m'th bundle, and fix all wires in $\mathcal{H}_m, \mathcal{H}_{m-1}$ except for: the m'th bundle; the masking inputs, outputs, and internal wires of $\mathcal{G}_o$; the masking inputs, and internal wires, of $\mathcal{G}_i$, as well as its input wire corresponding to the m'th bundle; the internal wires of all gadgets $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}_o$ or $\mathcal{G}_i$; and the internal wires of the decoding sub-circuits of $\mathcal{C}_{\mathsf{dec}}$ whose inputs are masking inputs of one of these $\mathcal{G}'$'s.

Let $\mathcal{W}_R \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(b, 1^{\mathsf{S}(n)}\right)$ (where b is the value encoded by the m'th bundle in H^y_{in}), and $\mathcal{W}_S \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(r, 1^{\mathsf{S}(n)}\right)$ for a random r (except if m corresponds to the output bundle, in which case $\mathcal{W}_S \leftarrow \mathsf{Enc}^{\mathsf{in}}\left(1, 1^{\mathsf{S}(n)}\right)$). We define a distribution $\mathcal{F}$ over $\mathcal{SHALLOW}\left(6, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$ as follows. Sampling a function $f \leftarrow \mathcal{F}$ is performed by sampling $\mathsf{rec}_o, \mathsf{rec}_i$ from the distribution over reconstructors for $\mathcal{G}_o, \mathcal{G}_i$, respectively (see Definition 14), and sampling reconstructors $\mathsf{rec}_{\mathcal{G}'}$ for the gadgets of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}_o$ or $\mathcal{G}_i$. The function f has all the hard-wired values of $\mathcal{H}_{m-1}$ hard-wired into it. On input $\mathbf{e} \in \mathbb{F}^{\widehat{n}(1,\mathsf{S}n)}$, f: (1) evaluates rec_o on the (hard-wired) inputs of $\mathcal{G}_o$, and the output $\mathbf{e}$ (this reconstructs the masking inputs, and internal wires, of $\mathcal{G}_o$); (2) evaluates rec_i on $\mathbf{e}$ as one of the inputs, and the other (hard-wired) input and output of $\mathcal{G}_i$; (3) for every gadget $\mathcal{G}'$ of $\widehat{\mathcal{C}}_{0-\mathsf{check}}$ connected to $\mathcal{G}_o$ or $\mathcal{G}_i$, uses $\mathsf{rec}_{\mathcal{G}'}$ to generate its internal wires; and finally (4) uses the functions of Lemma 7 to generate the internal wires of the decoding sub-circuits that decode the masking inputs used in the $\mathcal{G}'$'s. Then $f \in \mathcal{SHALLOW}\left(6, O\left(\widehat{n}^4(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right)$ because the reconstructors $\mathsf{rec}_o, \mathsf{rec}_i \in \mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1,\mathsf{S}(n))\right)\right)$ (by Lemma 4) and can be evaluated in parallel, the $O\left(\widehat{n}^2(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)$ reconstructors $\mathsf{rec}_{\mathcal{G}'} \in \mathcal{SHALLOW}\left(2, O\left(\widehat{n}^2(1,\mathsf{S}(n))\right)\right)$ (by Lemma 7), and can be evaluated in parallel, and the $O\left(\widehat{n}^3(1,\mathsf{S}(n)) \cdot \mathsf{S}(n)\right)$ reconstructors of the decoding sub-circuits are each computable in $\mathcal{SHALLOW}(2, O(\widehat{n}(1,\mathsf{S}(n))))$ (by Lemma 7) and can be evaluated in parallel.

Consequently, if $\mathcal{W}'_R := f(\mathcal{W}_R), \mathcal{W}'_S := f(\mathcal{W}_S)$ for $f \leftarrow \mathcal{F}$, then $\mathcal{W}'_R \equiv \mathcal{H}_{m-1}$, $\mathcal{W}'_S \equiv \mathcal{H}_m$ and by Lemma 6, $\mathcal{W}'_R, \mathcal{W}'_S$ are $(\mathcal{LEAK}', \epsilon(n))$-leakage resilient for any $\mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}(6, O(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n))) \subseteq \mathcal{LEAK}_{\mathsf{E}}$.

Bounding $\mathsf{SD}\left(\ell\left(\mathcal{H}^y_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^{y,x}_{\mathsf{ext},2}\right)\right)$**:** it holds that $\mathsf{SD}\left(\ell\left(\mathcal{H}^y_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^{y,x}_{\mathsf{ext},2}\right)\right) = 0$ for every ℓ because the hybrids are independent of the input for $\widehat{C}_2$ (since the input is re-sampled as a fresh $z \in_R \mathbb{F}^n$ in both).

Bounding $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{ext},2}\right), \ell\left(\mathcal{H}^{y,x}_{\mathsf{in},2}\right)\right)$**:** We show that $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{ext},2}\right), \ell\left(\mathcal{H}^{y,x}_{\mathsf{in},2}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}(6, O(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n))) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. The proof is similar to the proof that $\mathsf{SD}\left(\ell\left(\mathcal{H}^y_{\mathsf{ext}}\right), \ell\left(\mathcal{H}^y_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \mathsf{S}(n)$, because the argument was independent of the actual inputs used in $\widehat{C}_1, \widehat{C}_2$ (as long as both hybrids use the same input in each copy).

Bounding $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{in},2}\right), \ell(\mathcal{H}^{y,x})\right)$**:** We show that $\mathsf{SD}\left(\ell\left(\mathcal{H}^{y,x}_{\mathsf{in},2}\right), \ell(\mathcal{H}^{y,x})\right) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$ for all $\ell \in \mathcal{LEAK}'$ such that $\mathcal{LEAK}' \circ \mathcal{SHALLOW}(7, O(\widehat{n}^4(1, \mathsf{S}(n)) \cdot \mathsf{S}(n))) \subseteq \mathcal{LEAK}_{\mathsf{E}}$. The proof is similar to the proof that $\mathsf{SD}\left(\ell(\mathcal{H}^y), \ell\left(\mathcal{H}^y_{\mathsf{in}}\right)\right) \leq \epsilon(n) \cdot \widehat{n}(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)$, because the argument was independent of the actual inputs used in $\widehat{C}_1, \widehat{C}_2$.

Bounding $\mathsf{SD}(\ell(\mathcal{H}^x), \ell(\mathcal{H}^y))$ **for** $\ell \in \mathcal{LEAK}'$**.** From this analysis, we can now conclude using the union bound that for every $\ell \in \mathcal{LEAK}'$ such that

$$\mathcal{LEAK}' \circ \mathcal{SHALLOW}\left(7, O\left(\widehat{n}^5(1, \mathsf{S}(n)) \cdot \mathsf{S}(n)\right)\right) \subseteq \mathcal{LEAK}_{\mathsf{E}}$$

it holds that

$$\mathsf{SD}(\ell(\mathcal{H}^x), \ell(\mathcal{H}^y)) \leq 4\epsilon(n) \cdot \mathsf{S}(n) \cdot (\widehat{n}(1, \mathsf{S}(n)) + 1).$$

4.3. An SAT-Respecting LRCC Against "Useful" Leakage

Ishai et al. [8] use Construction 11 to devise an SAT-respecting LRCC which they later employ in their WI-PCP construction (described in Section 4.4). This is performed in two steps. First, since the LRCC will be used to compile verification circuits of NP relations, we need the compiler to be *Boolean*. Second, we need to instantiate the internal encoding scheme E such that it would resist leakage computable by functions that: (1) apply the PCP-prover algorithm, and then (2) restrict the output to a small subset of bits. Indeed, this is exactly the "leakage" on the witness which a query-bounded verifier (even a malicious one) obtains by querying the proof. We now provide more details on each of these steps.

Step (1): An SAT-Respecting LRCC for *Boolean* Circuits. The high-level idea is to transform the Boolean circuit $C : \{0,1\}^n \to \{0,1\}$ into a functionally-equivalent arithmetic circuit C_3 over $\mathbb{F}_3$ (i.e., the field with 3 elements), use Construction 11 over the field $\mathbb{F}_3$ to compile C_3 into its leakage-resilient version $\widehat{C_3}$, and then output the Boolean circuit $\widehat{C}$ that emulates $\widehat{C_3}$ using Boolean operations. This is an over-simplified description of the compiler, where the actual construction needs to address several subtleties. We now describe each of these steps, and the subtleties that arise, in more detail.

From Boolean to Arithmetic Operations. The circuit C_3 is obtained from C by representing each Boolean operation using an appropriate polynomial over $\mathbb{F}_3$ in the natural way. While C_3 is guaranteed to be functionally equivalent to C on binary strings, two issues arise concerning the SAT-respecting property. First, satisfiability over $\mathbb{F}_2$ means that $C(x) = 1$ for some x, whereas satisfiability over $\mathbb{F}_3$ means $C_3(y) = 0$ for some y. In particular, we want the leakage-resilient circuit to output 1 only if there exists an x such that $C(x)$ outputs 1, whereas the SAT-respecting property guarantees only that if $\widehat{C_3}(\widehat{y}) = 0$ for some $\widehat{y}$, then $C_3(y) = 0$ for some y. Therefore, we need to "translate" a 1-output of C into a 0-output of C_3, and a 0-output of C into a non-0-output of C_3. Thus, we will have that $\widehat{C_3} = (\widehat{y}) = 0$ for some $\widehat{y}$ only if $C_3(y) = 0$ for some y. This brings us to the second issue: while we would like to use the function-equivalence of C and C_3 to claim that if $C_3(y) = 0$ for some $y \in \mathbb{F}_3^n$ then $C(x) = 1$ for some $x \in \mathbb{F}_2^n$, this is not necessarily the case, because C_3's inputs are from $\mathbb{F}_3^n$ and might *not* correspond to an input in $\mathbb{F}_2^n$. To overcome this issue, we add to C_3 an

"input checker" component which checks that each of its n inputs is a bit. We denote this "enhanced" version of C_3—which flips the output and checks validity of the inputs—by C'_3.

From Arithmetic Back To Boolean. Once we generate the leakage-resilient version $\widehat{C'_3}$ of C'_3, we need to represent it using a *Boolean* circuit. (Indeed, the original circuit C was Boolean, and its leakage-resilient version should also be a Boolean circuit.) We do so by replacing each field element with a binary string representing it, and implementing each gate over $\mathbb{F}_3$ by a Boolean sub-circuit. We also flip the output of $\widehat{C'_3}$, so the resultant circuit would again be functionally-equivalent to C. There are two important points that we need to handle. First, while we can represent field elements using any (injective) encoding scheme, to preserve the SAT-respecting property it must also be onto. Otherwise, the Boolean circuit could potentially be satisfied using invalid encodings, namely ones that do not encode any field element, and thus the computation in the Boolean circuit would not correspond to a computation in $\widehat{C'_3}$. In particular, the Boolean sub-circuits implementing gates over $\mathbb{F}_3$ must be defined for all possible encodings—even ones that would not be used in an honest evaluation of the circuit.

Second, these Boolean sub-circuits should have small depth and size. The reason is the reduction from the leakage resilience of the final circuit $\widehat{C}$ to the leakage resilience of $\widehat{C'_3}$. More specifically, the reduction proceeds by assuming that a leakage function ℓ in some class $\mathcal{LEAK}$ can distinguish between the wire values $\left[\widehat{C}, \widehat{x}\right]$ of $\widehat{C}$ on an encoding $\widehat{x}$ of some input x, and its wire values $\left[\widehat{C}, \widehat{x'}\right]$ on the encoding $\widehat{x'}$ of some other input x' such that $C(x) = C(x')$. It then uses this to break the leakage resilience of $\widehat{C'_3}$ for some leakage function ℓ_3 in the leakage class $\mathcal{LEAK}_3$ against which the arithmetic LRCC is secure. This is performed as follows: given the wire values of $\widehat{C'_3}$ on some input (these values are elements of $\mathbb{F}_3$), the reduction first replaces the field elements with the corresponding encodings. The resultant values constitute only *part* of the wire values of $\widehat{C}$. Specifically, these are the values of the wires *between the sub-circuits emulating the gates over* $\mathbb{F}_3$, whereas $\widehat{C}$ contains also the internal wires of these sub-circuits, namely wires which do not appear in $\widehat{C'_3}$. Thus, the leakage function ℓ_3 must first generate these missing wires, and only then can it evaluate ℓ. In particular, $\mathcal{LEAK} \subset \mathcal{LEAK}_3$, where the difference between the two classes depends on the complexity of the Boolean sub-circuits implementing gates over $\mathbb{F}_3$. Fortunately, these sub-circuits are both shallow and small.

In summary, Ishai et al. show [47, Proposition 3.31] the following Boolean LRCC.

Claim 13 (Boolean SAT-respecting LRCC). *Let $\mathcal{LEAK}, \mathcal{LEAK}_{\mathsf{E}}$ 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 $\mathsf{E}^{\mathsf{in}} = \left(\mathsf{Enc}^{\mathsf{in}}, \mathsf{Dec}^{\mathsf{in}}\right)$ be a linear, onto, $(\mathcal{LEAK}_{\mathsf{E}}, \epsilon(n))$-leakage-resilient encoding scheme with parameters n, σ and $\widehat{n} = \widehat{n}(n, \sigma)$, such that $\mathcal{LEAK}_{\mathsf{E}} = \mathcal{LEAK} \circ \mathcal{BOOL}\big(33, O\big(\widehat{n}^6(1, \mathsf{S}(n)) \cdot \mathsf{S}^2(n)\big)\big)$. Then there exist constants $c, c' > 0$ for which there exists an SAT-respecting, $(\mathcal{LEAK}, c' \cdot \epsilon(n) \cdot (\widehat{n}(1, c \cdot \mathsf{S}(n)) + 1) \cdot c \cdot \mathsf{S}(n), \mathsf{S}(n))$-LRCC over $\{0,1\}$. Moreover, for every $C : \{0,1\}^n \to \{0,1\}$, the compiled circuit $\widehat{C}$ has size $\left|\widehat{C}\right| = O\left(\widehat{n}^6(1, c \cdot \mathsf{S}(n)) \cdot |C|^2\right)$.*

Step (2): Leakage Resilience Against "Useful" Leakage. The second component of the construction is an internal encoding scheme E_{in}—resisting leakage from a "useful" class of leakage functions—with which we instantiate Claim 13. More specifically, the leakage class consists of $\mathcal{AC}^0$ circuits (namely, constant-depth, polynomial-sized Boolean circuits over unbounded fan-in and fan-out $\wedge, \vee, \neg$ gates), *augmented with a sublinear number of $\oplus$ gates of unbounded fan-in and fan-out*. Formally,

Notation 10 ($\mathcal{L}^m_{n,d,s,\oplus t}$ leakage family). *Let $n, d, s \in \mathbb{N}$ be length, depth and size parameters (respectively), and let $t \in \mathbb{N}$ be a parity gate bound. The family $\mathcal{L}_{n,d,s,\oplus t}$ consists of all functions computable by a Boolean circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most s and depth d, with unbounded fan-in and fan-out $\wedge, \vee, \neg, \oplus$ gates, out of which at most t are $\oplus$ gates. We denote $\mathcal{L}_{d,s,\oplus t} = \cup_{n \in \mathbb{N}} \mathcal{L}_{n,d,s,\oplus t}$.*

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

The encodings scheme we use encodes elements $\gamma \in \mathbb{F}_3$ as binary strings whose sum mod 3 is γ. Formally:

Notation 11. *For $\gamma \in \{0,1,2\}$ and $n \in \mathbb{N}$, $\mathcal{U}^n_\gamma$ denotes the uniform distribution over $\left\{v \in \{0,1\}^{3n} : \#_1(v) \equiv \gamma \mod 3\right\}$, where $\#_1(v)$ denotes the number of 1's in v.*

Definition 16. *The encodings scheme $\mathsf{E}_3 = (\mathsf{Enc}_3, \mathsf{Dec}_3)$ is defined as follows. For every $\gamma \in \mathbb{F}_3$, $\mathsf{Enc}_3(\gamma, 1^n)$ samples from $\mathcal{U}^n_\gamma$, and $\mathsf{Dec}_3(v)$ returns $(\#_1(v) \mod 3)$. We note that E_3 is linear and onto.*

Remark 14. Enc_3 *can be computed efficiently by repeating the following procedure n^2 times. Pick $v \in \{0,1\}^{3n}$ uniformly at random, compute $t := \#_1(v)$, and if $t = \gamma$ then return v. If all iterations fail, return a fixed $v_\gamma \in \{0,1\}^{3n}$ such that $\#_1(v_\gamma) = \gamma$. Then the output of* Enc_3 *is thus statistically close to $\mathcal{U}^n_\gamma$.*

Ishai et al. [8] show that the encoding scheme of Definition 16 resists leakage from $\mathcal{AC}^0$ circuits augmented with few $\oplus$ gates:

Corollary 3 (Corollary 3.44 in [47])**.** *For every constant depth parameter $d \in \mathbb{N}$ there exist constants $c, \epsilon \in (0,1)$, such that for every constant $l \in \mathbb{N}$ there exists a minimal length parameter $n_0 \in \mathbb{N}$ such that for every $n \ge n_0$ the encoding scheme $\mathsf{Enc}_3(\cdot, 1^n)$ of Definition 16 is $\left(\mathcal{L}_{3n,d,n^l,\oplus n^\epsilon}, 2^{-n^c}\right)$-leakage resilient.*

Instantiating Claim 13 with the encoding scheme of Definition 16 as the internal encoding scheme, and using Corollary 3, ref. [8] show the existence of a Boolean LRCC resisting leakage from $\mathcal{AC}^0$ circuits with few $\oplus$ gates:

Theorem 15 (Boolean SAT-respecting LRCC for $\mathcal{AC}^0$ circuits with $\oplus$ gates, Theorem 3.37 in [47])**.** *Let $n \in \mathbb{N}$ be an input length parameter. For every positive constant d, c, polynomials $m = m(n), t = (n)$, and polynomial size bound $s = s(n)$, there exists a polynomial $l(n)$, such that the following holds. There exists an SAT-respecting $\left(\mathcal{L}^m_{l,d,l^c,\oplus t}, 2^{-n^c}, s(n)\right)$-LRCC over $\{0,1\}$, which on input a circuit $C : \{0,1\}^n \to \{0,1\}$ of size $|C| \le s(n)$ outputs a circuit $\widehat{C}$ of size $\left|\widehat{C}\right| \le l(n)$.*

4.4. The Witness-Indistinguishable PCP

In this section, we describe the WI-PCPs for NP of [8], which rely on the Boolean SAT-respecting LRCC of Theorem 15. They use also a PCP system $(\mathcal{P}', \mathcal{V}')$ for the language 3SAT of all satisfiable 3CNF formulas, in which the prover algorithm can be implemented in a low complexity class.

The high-level idea of the construction for an NP-relation $\mathcal{R}$ with verification circuit C is that given input x, instead of verifying that $C_x(\cdot) := C(x,\cdot)$ is satisfiable (which holds if and only if $x \in \mathcal{L}$ for the corresponding NP-language $\mathcal{L}$), the verifier will check that the *leakage-resilient version* $\widehat{C_x}$ is satisfiable. For this, the prover and verifier will first represent $\widehat{C_x}$ as a 3CNF formula φ in the natural way. That is, φ will have a variable for each wire of $\widehat{C_x}$. It will contain, for each gate g of $\widehat{C_x}$, a sub-formula verifying that the output wire of g is consistent with the input wires and the operation of g, and it will also check that the output wire of $\widehat{C_x}$ is 1. A satisfying assignment for φ is the wire values of $\widehat{C_x}$ when evaluated on (an encoding of) a witness w for x. Then, the prover and verifier will use the PCP system $(\mathcal{P}', \mathcal{V}')$ to verify that $\varphi \in$ 3SAT. The construction is described in Figure 3.

Construction 16 (A Witness-Indistinguishable PCP). *Let $\mathcal{R} = \mathcal{R}(x, w)$ be an* NP*-relation with verification circuit C. (We note that $\mathcal{R}$ is actually associated with a* family *$\{C_n\}$ of circuits, where C_n is applied to inputs $x \in \{0,1\}^n$. Somewhat abusing notation, we refer to all these circuits simply as "C".)*

Building blocks:

- *A PCP system $(\mathcal{P}', \mathcal{V}')$ for* 3SAT*;*
- *A Boolean LRCC* $(\mathsf{Comp}, \mathsf{E} = (\mathsf{Enc}, \mathsf{Dec}))$.

Prover algorithm. *$\mathcal{P}$, on input $(x, w) \in \mathcal{R}$, operates as follows:*

- *Computes $\widehat{C_x}(\cdot) = \mathsf{Comp}(C_x)$, where C_x denotes the circuit C with x hard-wired into it.;*
- *Samples a random encoding $\widehat{w} \leftarrow \mathsf{Enc}\left(w, 1^{|C_x|}\right)$, and evaluates $\widehat{C_x}$ on $\widehat{w}$ to generate the wire values $\mathcal{W}$ of $\widehat{C_x}$;*
- *Constructs the 3CNF φ_x representing $\widehat{C_x}$;*
- *Computes a PCP $\pi = \mathcal{P}'(\varphi_x, \mathcal{W})$ for the claim "$\varphi_x \in$* 3SAT*", and outputs π.*

Verifier algorithm. *$\mathcal{V}$ is given input x and oracle access to π. It computes $\widehat{C_x}(\cdot) = \mathsf{Comp}(C_x)$, and constructs the 3CNF formula φ_x. Then, $\mathcal{V}$ emulates $\mathcal{V}'$ with input φ_x and oracle access to π, and outputs whatever $\mathcal{V}'$ outputs.*

Figure 3. Witness-Indistinguishable PCPs from SAT-Respecting LRCCs [8].

The following theorem (which is a combination of ([47], Proposition 4.4) and ([47], Corollary 4.10)) asserts the connection between the properties of the LRCC and the resultant PCP system $(\mathcal{P}, \mathcal{V})$.

Theorem 17. *Let $n \in \mathbb{N}$ be a length parameter, $q^* = q^*(n), \mathsf{S} = \mathsf{S}(n)$ be query and size functions, $\epsilon, \epsilon' \in [0,1]$, and $\mathcal{LEAK}$ be a family of leakage functions. Assume that Construction 16 is instantiated with:*

- *A Boolean SAT-respecting $(\mathcal{LEAK}, \epsilon, \mathsf{S})$-LRCC such that there exists a polynomial $g(\cdot)$ for which $|\mathsf{Comp}(C)| \leq g(|C|)$ for every circuit C; and*
- *A PCP system $(\mathcal{P}', \mathcal{V}')$ for* 3SAT *with proofs of length $\mathsf{len}(n)$, such that for every $(\varphi, \mathcal{W}) \in$* 3SAT*, every subset $\mathcal{Q}$ of q^* bits of an honestly-generated proof $\pi = \pi(\varphi, \mathcal{W})$ is computable from $\mathcal{W}$ by a function $f_{\varphi,\mathcal{Q}} \in \mathcal{LEAK}$.*

Then for every NP*-relation $\mathcal{R}$ with verification circuit C of size $|C| \leq \mathsf{S}$, the PCP system $(\mathcal{P}, \mathcal{V})$ is a (q^*, ϵ^*)-WI-PCP for $\mathcal{R}$, where $\epsilon^* = O\left(\epsilon \cdot q^* \cdot \mathsf{len}^{2q^*}(t)\right) + e^{-\Omega\left(q^* \cdot \mathsf{len}^{q^*}(t)\right)}$ and $t = O(g(|C|))$. Moreover, if $\mathcal{V}'$ is non-adaptive, the so is $\mathcal{V}$.*

Furthermore, the system is (q^*, ϵ)-WI against non-adaptive *(possibly malicious) verifiers. Moreover, proofs generated by $\mathcal{P}$ have length $\mathsf{len}(t)$, and if $\mathcal{V}'$ has query complexity $q(n)$ and tosses $r(n)$ coins, then $\mathcal{V}$ has query complexity $q(t)$, tosses $r(t)$ coins.*

Proof. We first analyze the parameters of the system. The wire assignment $\mathcal{W}$ to $\widehat{C_x}$ has size $|\mathcal{W}| = \left|\widehat{C_x}\right| \leq g(|C_x|) \leq g(|C|)$, and $|\varphi_x| = O\left(\left|\widehat{C_x}\right|\right) \leq O(g(|C|))$. Therefore, the internal PCP system $(\mathcal{P}', \mathcal{V}')$ is emulated using inputs of size $t = g(|C|)$.

Completeness follows directly from the completeness of the building blocks.

Soundness follows from a combination of the soundness of $(\mathcal{P}', \mathcal{V}')$ and the SAT-respecting property of $(\mathsf{Comp}, \mathsf{E})$, Indeed, if $x \notin \mathcal{L}$, then C_x is not satisfiable, and so (by the SAT-respecting property) $\widehat{C_x}$ is not satisfiable, i.e., $\varphi_x \notin$ 3SAT. Therefore, by the soundness of $(\mathcal{P}', \mathcal{V}')$ we have that $\Pr\left[\mathcal{V}^{\pi^*}(x) = 1\right] = \Pr\left[\mathcal{V}'^{\pi^*}(\varphi_x) = 1\right]$ is negligible.

Witness-indistinguishability. Let $x \in \mathcal{L}$, φ_x be the 3CNF formula representing $\widehat{C_x}$, and w_1, w_2 be two witnesses for x. We first show witness indistinguishability against *non-adaptive* verifiers. Let $\mathcal{V}^*$ be a non-adaptive q^*-query-bounded verifier, and let $\pi_i \leftarrow \mathcal{P}(x, w_i)$ for $i = 1, 2$. Since $\mathcal{V}^*$'s entire view can be generated from the oracle answers to its queries (and this cannot increase the statistical distance), it suffices to show that $\mathsf{SD}(\pi_1|_{\mathcal{Q}}, \pi_2|_{\mathcal{Q}}) \leq \epsilon$ for every set $\mathcal{Q}$ of queries of $\mathcal{V}^*$ such that size $|\mathcal{Q}| \leq q^*$, where $\pi_i|_{\mathcal{Q}}$

denotes the restriction of π_i to the entries in $\mathcal{Q}$. Since $|C| = |C_x| \leq \mathsf{S}$, the leakage resilience of the LRCC guarantees that $\mathsf{SD}\left(\ell\left[\widehat{C_x}, \widehat{w}_1\right], \ell\left[\widehat{C_x}, \widehat{w}_2\right]\right) \leq \epsilon$ for every $\ell \in \mathcal{LEAK}$. We conclude the proof by noting that $f_{\varphi_x,\mathcal{Q}} \in \mathcal{LEAK}$, and $\pi_i|_{\mathcal{Q}} = f_{\varphi_x,\mathcal{Q}}\left[\widehat{C_x}, \widehat{w}_i\right]$. We have shown that $(\mathcal{P}, \mathcal{V})$ is (q^*, ϵ)-witness indistinguishable against *non-adaptive* verifiers $\mathcal{V}^*$. Using Theorem 18 below, this implies that $(\mathcal{P}, \mathcal{V})$ is (q^*, ϵ^*)-WI (even against *adaptive* verifiers), for $\epsilon^* = O\left(\epsilon \cdot q^* \cdot \mathsf{len}^{2q^*}(t)\right) + e^{-\Omega\left(q^* \cdot \mathsf{len}^{q^*}(t)\right)}$. □

The proof of Theorem 17 used the following theorem, which is implicit in [48] (see also [47], Theorem 4.11).

Theorem 18 (Implicit in [48])**.** *Let $(\mathcal{P}, \mathcal{V})$ be a PCP system that is (q^*, ϵ)-WI against* non-adaptive *verifiers, with proofs of length* len*. Then $(\mathcal{P}, \mathcal{V})$ is (q^*, ϵ^*)-WI against* adaptive *verifiers, where $\epsilon^* = O\left(\epsilon \cdot q^* \cdot \mathsf{len}^{2q^*}\right) + e^{-\Omega\left(q^* \cdot \mathsf{len}^{q^*}\right)}$.*

Theorem 7 now follows as a corollary of Theorem 17, using the SAT-respecting LRCC of Theorem 15, and a PCP system of [2], whose prover algorithm can be implemented in a low complexity class (the analysis of the prover complexity is due to ([47], Appendix B)):

Theorem 19 (PCPs for NP, [2])**.** $\mathsf{3SAT}$ *has a PCP system $(\mathcal{P}, \mathcal{V})$ with soundness error* $1/2$ *with an honest verifier that queries $O\left(\log^2 n\right)$ proof bits. The proofs have length* $\mathsf{poly}(n)$*, where every proof bit can be generated by an $\mathcal{AC}^0$ circuit with a single $\oplus$ gate of unbounded fan-in.*

We are now ready to prove Theorem 7.

Proof of Theorem 7. We instantiate Construction 16 with the PCP system $(\mathcal{P}', \mathcal{V}')$ of Theorem 19 and the LRCC of Theorem 15. By Theorem 19, there exist constants $d, c \in \mathbb{N}$ such that every bit in a proof generated by $\mathcal{P}'$ is computable from the NP-witness in $\mathcal{L}_{d,n^c,\oplus 1}$, where n is the witness length, and the proofs have length $n^{c''}$, for some constant c''.

Let $\mathcal{R}$ be an NP-relation with verification circuit C, then $|C| = n^{c'}$ for some constant c'. We instantiate Theorem 15 with parameters $d^* = d, s^* = |C|, n^* = n, t^* = 1, m^* = q^*$, and $c^* \geq c$ which is a sufficiently large constant whose value is set below. Here, the superscript * is used to denote the parameters of Theorem 15, and s^*, t^*, m^* are in $\mathsf{poly}(n)$. Let $(\mathsf{Comp}, \mathsf{E})$ denote the LRCC obtained from Theorem 15. We compute $\widehat{C} = \mathsf{Comp}(C)$, where $\left|\widehat{C}\right| \leq l(n)$ which, because $|C| \leq s^*$, is $\left(\mathcal{L}^{q^*}_{d,l^c(n),\oplus 1}, 2^{-n^{c^*}}\right)$-LR (where $l(n)$ is the polynomial whose existence is guaranteed by Theorem 15).

Let φ denote the 3CNF representing $\widehat{C}$. Then by Theorem 19 (and using the fact that $|\mathcal{W}| = \left|\widehat{C}\right| \leq l(n)$), every bit of a proof generated by $\mathcal{P}'$ for φ can be generated from a wire assignment $\mathcal{W}$ of $\widehat{C}$ in $\mathcal{L}_{d,l^c(n),\oplus 1}$, so every q^* proof bits are computable from $\mathcal{W}$ in $\mathcal{L}^{q^*}_{d,l^c(n),\oplus 1}$. Therefore, Theorem 17 guarantees that the system $(\mathcal{P}, \mathcal{V})$ of Construction 16 is a non-adaptive WI-PCP system for $\mathcal{R}$, with $\left(q^*, O\left(2^{-n^{c^*}} \cdot q^* \cdot \mathsf{len}^{2q^*}(l(n))\right) + e^{-\Omega\left(q^* \cdot \mathsf{len}^{q^*}(l(n))\right)}\right)$-WI (where $\mathsf{len}(l(n))$ denotes the proof length), and soundness error $1/2$ with an honest verifier that queries $O\left(\log^2(l(n))\right) = \mathsf{polylog}(n) \leq \mathsf{polylog}(q^*)$ proof bits. We set c^* to be sufficiently large, such that the statistical LR error satisfies

$$O\left(2^{-n^{c^*}} \cdot q^* \cdot \mathsf{len}^{2q^*}(l(n))\right) + e^{-\Omega\left(q^* \cdot \mathsf{len}^{q^*}(l(n))\right)} = \mathsf{negl}(q^*) \leq \mathsf{negl}(\kappa).$$

(We note that such a constant exists since we assume that $q^* = \mathsf{poly}(n)$.) We conclude the proof by noting that soundness can be amplified to $\mathsf{negl}(\kappa)$ with only a $\mathsf{poly}(\kappa)$ blowup in the query complexity of the honest verifier. □

5. Discussion

The works of [8,13] show a connection between ZK-PCPs and the seemingly-unrelated field of leakage-resilient cryptography, and use it to circumvent an inherent limitation of previous constructions—that the *honest* verifier is adaptive. Specifically, using tools from the leakage-resilience literature, [8,13] put forth two new paradigms of constructing ZK-PCPs, yielding PCPs with ZK against malicious verifiers, in which the honest verifier is *non-adaptive*. In the context of cryptographic applications of ZK-PCPs, non-adaptive verification translates into fewer communication rounds. The paradigm of [13] also extends to ZK-PCPs *of proximity*.

Despite this recent progress, several interesting questions remain open. The obvious open problem is to obtain ZK-PCPs and ZK-PCPPs with an exponential query gap as in [6,11] but which can be verified non-adaptively. One possible approach is to design a ZK-PCP variant over a large alphabet with negligible soundness error and an honest verifier that makes fewer queries than [15] (hopefully, polylogarithmic). Another interesting research direction is to extend the techniques of [8,13] to other related proof systems, such as interactive oracle proofs. Finding further applications of ZK-PCPs and ZK-PCPPs is also an interesting question. Finally, though in this survey we have focused on other parameters of ZK-PCPs, reducing the proof length is a fascinating open problem worthy of study. Whereas the locking-scheme-based ZK-PCPs of [6,11] inherently incur a polynomial blowup in proof length, another advantage of the leakage-resilience-based approach is that it opens up the possibility of reducing the proof length of ZK-PCPs, potentially even matching the proof length of non-ZK PCPs.

Funding: The author is supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: I am extremely grateful to Yuval Ishai for introducing me to the world of ZK-PCPs. I am also grateful to all my collaborators on these works—Yuval Ishai, Carmit Hazay, Guang Yang, and Muthu Venkitasubramaniam—for many enjoyable discussions, and the fruitful collaborations whose results are surveyed here.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Arora, S.; Lund, C.; Motwani, R.; Sudan, M.; Szegedy, M. Proof Verification and Hardness of Approximation Problems. In Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, PA, USA, 24–27 October 1992; IEEE Computer Society: Washington, DC, USA, 1992; pp. 14–23. [CrossRef]
2. Arora, S.; Safra, S. Probabilistic Checking of Proofs: A New Characterization of NP. In Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, PA, USA, 24–27 October 1992; IEEE Computer Society: Washington, DC, USA, 1992; pp. 2–13. [CrossRef]
3. Dinur, I. The PCP theorem by gap amplification. In Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, 21–23 May 2006; ACM: New York, NY, USA, 2006; pp. 241–250. [CrossRef]
4. Goldwasser, S.; Micali, S.; Rackoff, C. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In Proceedings of the 17th Annual ACM Symposium on Theory of Computing, Providence, RI, USA, 6–8 May 1985; ACM: New York, NY, USA, 1985; pp. 291–304. [CrossRef]
5. Shamir, A. IP = PSPACE. In Proceedings of the 31st Annual Symposium on Foundations of Computer Science, St. Louis, MO, USA, 22–24 October 1990; IEEE Computer Society: Washington, DC, USA, 1990; Volume 1, pp. 11–15. [CrossRef]
6. Kilian, J.; Petrank, E.; Tardos, G. Probabilistically Checkable Proofs with Zero Knowledge. In Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, El Paso, TX, USA, 4–6 May 1997; ACM: New York, NY, USA, 1997; pp. 496–505. [CrossRef]
7. Ben-Sasson, E.; Goldreich, O.; Harsha, P.; Sudan, M.; Vadhan, S.P. Robust PCPs of proximity, shorter PCPs and applications to coding. In Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, 13–16 June 2004; ACM: New York, NY, USA, 2004; pp. 1–10. [CrossRef]
8. Ishai, Y.; Weiss, M.; Yang, G. Making the Best of a Leaky Situation: Zero-Knowledge PCPs from Leakage-Resilient Circuits. In *Theory of Cryptography, Proceedings of the 13th International Conference, TCC 2016-A, Part II*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; Volume 9563, pp. 3–32. [CrossRef]

9. Dwork, C.; Feige, U.; Kilian, J.; Naor, M.; Safra, S. Low Communication 2-Prover Zero-Knowledge Proofs for NP. In *Advances in Cryptology—CRYPTO*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 1992; Volume 740, pp. 215–227. [CrossRef]
10. Ishai, Y.; Mahmoody, M.; Sahai, A. On Efficient Zero-Knowledge PCPs. In *Theory of Cryptography, Proceedings of the 9th Theory of Cryptography Conference, TCC 2012*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2012; Volume 7194, pp. 151–168. [CrossRef]
11. Ishai, Y.; Weiss, M. Probabilistically Checkable Proofs of Proximity with Zero-Knowledge. In *Theory of Cryptography, Proceedings of the 11th Theory of Cryptography Conference, TCC*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8349, pp. 121–145. [CrossRef]
12. Weiss, M. Secure Computation and Probabilistic Checking. Ph.D. Thesis, Computer Science Department, Technion, Haifa, Israel, 2016.
13. Hazay, C.; Venkitasubramaniam, M.; Weiss, M. ZK-PCPs from Leakage-Resilient Secret Sharing. In Proceedings of the 2nd Conference on Information-Theoretic Cryptography, ITC 2021, LIPIcs, Virtual, 23–26 July 2021; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Wadern, Germany, 2021; Volume 199, pp. 6:1–6:21. [CrossRef]
14. Chen, H.; Cramer, R.; Goldwasser, S.; de Haan, R.; Vaikuntanathan, V. Secure Computation from Random Error Correcting Codes. In *Advances in Cryptology, Proceedings of the EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Lecture Notes in Computer Science; Naor, M., Ed.; Springer: Cham, Switzerland, 2007; Volume 4515, pp. 291–310. [CrossRef]
15. Ishai, Y.; Kushilevitz, E.; Ostrovsky, R.; Sahai, A. Zero-knowledge from secure multiparty computation. In Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), San Diego, CA, USA, 11–13 June 2007; ACM: New York, NY, USA, 2007; pp. 21–30. [CrossRef]
16. Barak, B.; Goldreich, O.; Impagliazzo, R.; Rudich, S.; Sahai, A.; Vadhan, S.P.; Yang, K. On the (Im)possibility of Obfuscating Programs. In *Advances in Cryptology—CRYPTO 2001*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2001; Volume 2139, pp. 1–18. [CrossRef]
17. Micali, S.; Reyzin, L. Physically Observable Cryptography (Extended Abstract). In *Theory of Cryptography, Proceedings of the First Theory of Cryptography Conference, TCC 2004*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2004; Volume 2951, pp. 278–296. [CrossRef]
18. Ishai, Y.; Sahai, A.; Wagner, D.A. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology—CRYPTO 2003, Proceedings of the 23rd Annual International Cryptology Conference*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2003; Volume 2729, pp. 463–481. [CrossRef]
19. Abboud, A.; Rubinstein, A.; Williams, R.R. Distributed PCP Theorems for Hardness of Approximation in P. In Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, 15–17 October 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 25–36. [CrossRef]
20. Lovett, S.; Srinivasan, S. Correlation Bounds for Poly-size $\mathcal{AC}^0$ Circuits with $n^{1-o(1)}$ Symmetric Gates. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Proceedings of the 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2011; Volume 6845, pp. 640–651. [CrossRef]
21. Feige, U.; Lapidot, D.; Shamir, A. Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract). In Proceedings of the 31st Annual Symposium on Foundations of Computer Science, St. Louis, MO, USA, 22–24 October 1990; IEEE Computer Society: Washington, DC, USA, 1990; Volume 1, pp. 308–317. [CrossRef]
22. Thaler, J. Proofs, Arguments, and Zero-Knowledge. 2022, *Manuscript*. Available online: https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf (accessed on 11 July 2022).
23. Ben-Sasson, E.; Chiesa, A.; Spooner, N. Interactive Oracle Proofs. In *Theory of Cryptography, Proceedings of the 14th International Conference, TCC 2016-B, Part II, Beijing, China, October 31–November 3, 2016*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, Volume 9986, pp. 31–60. [CrossRef]
24. Reingold, O.; Rothblum, G.N.; Rothblum, R.D. Constant-round interactive proofs for delegating computation. In Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, 19–21 June 2016; ACM: New York, NY, USA, 2016; pp. 49–62. [CrossRef]
25. Ben-Sasson, E.; Chiesa, A.; Gabizon, A.; Virza, M. Quasi-Linear Size Zero Knowledge from Linear-Algebraic PCPs. In *Theory of Cryptography, Proceedings of the 13th International Conference, TCC 2016-A, Part II*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; Volume 9563, pp. 33–64. [CrossRef]
26. Bootle, J.; Chiesa, A.; Liu, S. Zero-Knowledge IOPs with Linear-Time Prover and Polylogarithmic-Time Verifier. In *Advances in Cryptology, Proceedings of the EUROCRYPT 2022—41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*; Lecture Notes in Computer Science; Dunkelman, O., Dziembowski, S., Eds.; Springer: Cham, Switzerland, 2022; Volume 13276, pp. 275–304. [CrossRef]
27. Dinur, I.; Reingold, O. Assignment Testers: Towards a Combinatorial Proof of the PCP-Theorem. In Proceedings of the 45th Symposium on Foundations of Computer Science FOCS, Rome, Italy, 17–19 October 2004; IEEE Computer Society: Washington, DC, USA, 2004; pp. 155–164. [CrossRef]
28. Ben-Sasson, E.; Chiesa, A.; Forbes, M.A.; Gabizon, A.; Riabzev, M.; Spooner, N. On Probabilistic Checking in Perfect Zero Knowledge. *arXiv* **2016**, arXiv:1610.03798.

29. Ben-Sasson, E.; Chiesa, A.; Genkin, D.; Tromer, E.; Virza, M. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In *Proceedings of the Annual Cryptology Conference*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2013; Volume 8043, pp. 90–108. [CrossRef]
30. Kilian, J. A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*; Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A., Eds.; ACM: New York, NY, USA, 1992; pp. 723–732. [CrossRef]
31. Micali, S. CS Proofs (Extended Abstract). In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–24 November 1994; IEEE Computer Society: Washington, DC, USA, 1994; pp. 436–453. [CrossRef]
32. Ben-Sasson, E.; Sudan, M. Short PCPs with Polylog Query Complexity. *SIAM J. Comput.* **2008**, *38*, 551–607. [CrossRef]
33. Mie, T. Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries. *Ann. Math. Artif. Intell.* **2009**, *56*, 313–338. [CrossRef]
34. Choi, S.G.; Dachman-Soled, D.; Malkin, T.; Wee, H. Black-Box Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One. In *Theory of Cryptography, Proceedings of the Fifth Theory of Cryptography Conference, TCC 2008*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2008; Volume 4948, pp. 427–444. [CrossRef]
35. Choi, S.G.; Dachman-Soled, D.; Malkin, T.; Wee, H. A Black-Box Construction of Non-malleable Encryption from Semantically Secure Encryption. *J. Cryptol.* **2018**, *31*, 172–201. [CrossRef]
36. Ball, M.; Dachman-Soled, D.; Kulkarni, M.; Malkin, T. Non-malleable Codes for Bounded Depth, Bounded Fan-In Circuits. In *Advances in Cryptology—EUROCRYPT 2016, Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2016; Volume 9666, pp. 881–908. [CrossRef]
37. Ball, M.; Dachman-Soled, D.; Guo, S.; Malkin, T.; Tan, L. Non-Malleable Codes for Small-Depth Circuits. In Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, 7–9 October 2018; IEEE Computer Society: Washington, DC, USA, 2018; pp. 826–837. [CrossRef]
38. Ishai, Y.; Sahai, A.; Viderman, M.; Weiss, M. Zero Knowledge LTCs and Their Applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Proceedings of the 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2013; Volume 8096, pp. 607–622. [CrossRef]
39. Gammel, B.M.; Mangard, S. On the Duality of Probing and Fault Attacks. *J. Electron. Test.* **2010**, *26*, 483–493. [CrossRef]
40. Hazay, C.; Venkitasubramaniam, M.; Weiss, M. ZK-PCPs from Leakage-Resilient Secret Sharing (full version). In *IACR Cryptology ePrint Archive*; 2021. Available online: https://eprint.iacr.org/2021/606 (accessed on 11 July 2022).
41. Decatur, S.E.; Goldreich, O.; Ron, D. Computational Sample Complexity. *SIAM J. Comput.* **1999**, *29*, 854–879. [CrossRef]
42. Faust, S.; Rabin, T.; Reyzin, L.; Tromer, E.; Vaikuntanathan, V. Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In *Advances in Cryptology—EUROCRYPT 2010, Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2010*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2010; Volume 6110, pp. 135–156. [CrossRef]
43. Dziembowski, S.; Faust, S. Leakage-Resilient Circuits without Computational Assumptions. In *Theory of Cryptography, Proceedings of the 9th Theory of Cryptography Conference, TCC 2012*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2012; Volume 7194, pp. 230–247. [CrossRef]
44. Miles, E.; Viola, E. Shielding circuits with groups. In Proceedings of the Symposium on Theory of Computing Conference, STOC, Palo Alto, CA, USA, 1–4 June 2013; ACM: New York, NY, USA, 2013; pp. 251–260. [CrossRef]
45. Bogdanov, A.; Ishai, Y.; Srinivasan, A. Unconditionally Secure Computation Against Low-Complexity Leakage. In *Proceedings of the Advances in Cryptology—CRYPTO, Part II, Lecture Notes in Computer Science, 2019*; Springer: Cham, Switzerland, 2019; Volume 11693, pp. 387–416. [CrossRef]
46. Naor, M.; Yung, M. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 14–16 May 1990; ACM: New York, NY, USA, 1990; pp. 427–437. [CrossRef]
47. Ishai, Y.; Weiss, M.; Yang, G. Making the Best of a Leaky Situation: Zero-Knowledge PCPs from Leakage-Resilient Circuits (full version). In *IACR Cryptology ePrint Archive*; 2015. Available online: http://eprint.iacr.org/2015/1055 (accessed on 11 July 2022).
48. Canetti, R.; Damgård, I.; Dziembowski, S.; Ishai, Y.; Malkin, T. On Adaptive vs. Non-adaptive Security of Multiparty Protocols. In *Advances in Cryptology—EUROCRYPT 2001, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, 2001*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2001; Volume 2045, pp. 262–279. [CrossRef]