Succinct Non-Interactive Arguments via Linear Interactive Proofs

Nir Bitansky* Tel Aviv University
Alessandro Chiesa MIT
Yuval Ishai† Technion
Rafail Ostrovsky‡ UCLA
Omer Paneth§ Boston University

September 15, 2013

*This research was done while visiting Boston University and IBM T. J. Watson Research Center. Supported by the Check Point Institute for Information Security, an ISF grant 20006317, and the Fulbright program.

†Supported by the European Research Council as part of the ERC project CaC (grant 259426), ISF grant 1361/10, and BSF grant 2008411. Research done in part while visiting UCLA and IBM T. J. Watson Research Center.

‡Department of Computer Science and Department of Mathematics, UCLA. Email: rafail@cs.ucla.edu. Research supported in part by NSF grants CNS-0830803; CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

§Supported by an NSF grant 1218461.

Abstract

Succinct non-interactive arguments (SNARGs) enable verifying NP statements with lower complexity than required for classical NP verification. Traditionally, the focus has been on minimizing the length of such arguments; nowadays researchers have focused also on minimizing verification time, by drawing motivation from the problem of delegating computation.

A common relaxation is a *preprocessing* SNARG, which allows the verifier to conduct an expensive offline phase that is independent of the statement to be proven later. Recent constructions of preprocessing SNARGs have achieved attractive features: they are publicly-verifiable, proofs consist of only $O(1)$ encrypted (or encoded) field elements, and verification is via arithmetic circuits of size linear in the NP statement. Additionally, these constructions seem to have "escaped the hegemony" of probabilistically-checkable proofs (PCPs) as a basic building block of succinct arguments.

We present a general methodology for the construction of preprocessing SNARGs, as well as resulting concrete efficiency improvements. Our contribution is three-fold:

(1) We introduce and study a natural extension of the interactive proof model that considers *algebraically-bounded* provers; this new setting is analogous to the common study of algebraically-bounded "adversaries" in other fields, such as pseudorandomness and randomness extraction. More concretely, in this work we focus on linear (or affine) provers, and provide several constructions of (succinct two-message) *linear-interactive proofs* (LIPs) for NP. Our constructions are based on general transformations applied to both *linear* PCPs (LPCPs) and traditional "unstructured" PCPs.

(2) We give conceptually simple cryptographic transformations from LIPs to preprocessing SNARGs, whose security can be based on different forms of *linear targeted malleability* (implied by previous knowledge assumptions). Our transformations convert arbitrary (two-message) LIPs into designated-verifier SNARGs, and LIPs with degree-bounded verifiers into publicly-verifiable SNARGs. We also extend our methodology to obtain *zero-knowledge* LIPs and SNARGs. Our techniques yield SNARGs *of knowledge* and thus can benefit from known recursive composition and bootstrapping techniques.

(3) Following this methodology, we exhibit several constructions achieving new efficiency features, such as "single-ciphertext preprocessing SNARGs" and improved succinctness-soundness tradeoffs. We also offer a new perspective on existing constructions of preprocessing SNARGs, revealing a direct connection of these to LPCPs and LIPs.

Keywords: interactive proofs, probabilistically-checkable proofs, succinct arguments, homomorphic encryption, zero-knowledge

Contents

1 Introduction

Interactive proofs [GMR89] are central to modern cryptography and complexity theory. One extensively studied aspect of interactive proofs is their expressibility, culminating with the result $\mathsf{IP} = \mathsf{PSPACE}$ [Sha92]. Another aspect, which is the focus of this work, is that proofs for NP statements can potentially be *much shorter* than an NP witness and be *verified much faster* than the time required for checking the NP witness.

1.1 Background

Succinct interactive arguments. In interactive proofs for NP with statistical soundness, significant savings in communication (let alone verification time) are unlikely [BHZ87, GH98, GVW02, Wee05]. If we settle for proof systems with *computational* soundness, known as *argument systems* [BCC88], then significant savings can be made. Using collision-resistant hashes (CRHs) and probabilistically-checkable proofs (PCPs) [BFLS91], Kilian [Kil92] showed a four-message interactive argument for NP where, to prove membership of an instance x in a given NP language L with NP machine M_L, communication and verification time are bounded by $\mathrm{poly}(\lambda + |M_L| + |x| + \log t)$, and the prover's running time is $\mathrm{poly}(\lambda + |M_L| + |x| + t)$. Here, t is the classical NP verification time of M_L for the instance x, λ is a security parameter, and poly is a *universal* polynomial (independent of λ, M_L, x, and t). We call such argument systems *succinct*.

Proof of knowledge. A natural strengthening of computational soundness is (computational) *proof of knowledge*: it requires that, whenever the verifier is convinced by an efficient prover, not only can we conclude that a valid witness for the theorem *exists*, but also that such a witness can be *extracted efficiently* from the prover. This property is satisfied by most proof system constructions, including the aforementioned one of Kilian [BG08], and is useful in many applications of succinct arguments.

Removing interaction. Kilian's protocol requires four messages. A challenge, which is of both theoretical and practical interest, is the construction of *non-interactive* succinct arguments. As a first step in this direction, Micali [Mic00] showed how to construct publicly-verifiable *one-message* succinct non-interactive arguments for NP, in the random oracle model, by applying the Fiat-Shamir heuristic [FS87] to Kilian's protocol. In the plain model, one-message solutions are impossible for hard-enough languages (against non-uniform provers), so one usually considers the weaker goal of two-message succinct arguments where the verifier message is generated *independently* of the statement to be proven. Following [GW11], we call such arguments *SNARGs*. More precisely, a SNARG for a language L is a triple of algorithms (G, P, V) where: the generator G, given the security parameter λ, samples a *reference string* σ and a corresponding *verification state* τ (G can be thought to be run during an offline phase, by the verifier, or by someone the verifier trusts); the (honest) prover $P(\sigma, x, w)$ produces a proof π for the statement "$x \in L$" given a witness w; then, $V(\tau, x, \pi)$ verifies the validity of π. Soundness should hold even if x is chosen depending on σ.

Gentry and Wichs [GW11] showed that no SNARG can be proven secure via a black-box reduction to a falsifiable assumption [Nao03]; this may justify using non-standard assumptions to construct SNARGs. (Note that [GW11] rule out SNARGs only for (hard-enough) NP languages. For the weaker goal of verifying deterministic polynomial-time computations in various models, there are beautiful constructions relying on standard assumptions, such as [GKR08, KR09, AIK10, CKV10, GGP10, BGV11, CRR11, CTY11, CMT12, FG12]. We focus on verifying *nondeterministic* polynomial-time computations.)

Extending earlier works [ABOR00, DLN+04, Mie08, DCL08], several works showed how to remove interaction in Kilian's PCP-based protocol and obtain SNARGs of knowledge (*SNARKs*) using *extractable collision-resistant hashing* [BCCT12, DFH12, GLR11], or construct MIP-based SNARKs using fully-homomorphic encryption *with an extractable homomorphism property* [BC12].

The preprocessing model. A notion that is weaker than a SNARK is that of a *preprocessing* SNARK: here, the verifier is allowed to conduct an *expensive* offline phase. More precisely, the generator G takes as an additional input a time bound T, may run in time $\mathrm{poly}(\lambda + T)$ (rather than $\mathrm{poly}(\lambda + \log T)$), and generates σ and τ that can be used, respectively, to prove and verify correctness of computations of length at most T. Bitansky et al. [BCCT13] showed that SNARKs can always be "algorithmically improved"; in particular, preprocessing SNARKs imply ones *without* preprocessing. (The result of [BCCT13] crucially relies on the fast verification time and the adaptive proof-of-knowledge property of the SNARK.) Thus, "preprocessing can always be removed" at the expense of only a $\mathrm{poly}(\lambda)$-loss in verification efficiency.

1.2 Motivation

The typical approach to construct succinct arguments (or, more generally, other forms of proof systems with nontrivial efficiency properties) conforms with the following methodology: first, give an information-theoretic construction, using some form of probabilistic checking to verify computations, in a model that enforces certain restrictions on provers (e.g., the PCP model [Kil92, Mic00, BG08, DCL08, BCCT12, DFH12, GLR11] or other models of probabilistic checking [IKO07, KR08, SBW11, SMBW12, SVP+12, BC12, SBV+12]); next, use cryptographic tools to compile the information-theoretic construction into an argument system (where there are no restrictions on the prover other than it being an efficient algorithm).

Existing constructions of preprocessing SNARKs seem to diverge from this methodology, while at the same time offering several attractive features: such as public verification, proofs consisting of only $O(1)$ encrypted (or encoded) field elements, and verification via arithmetic circuits that are linear in the statement.

Groth [Gro10] and Lipmaa [Lip12] (who builds on Groth's approach) introduced clever techniques for constructing preprocessing SNARKs by leveraging knowledge-of-exponent assumptions [Dam92, HT98, BP04a] in bilinear groups. At high level, Groth considered a simple reduction from circuit satisfaction problems to an algebraic satisfaction problem of quadratic equations, and then constructed a set of specific cryptographic tools to succinctly check satisfiability of this problem. Gennaro et al. [GGPR13] made a first step to better separate the "information-theoretic ingredient" from the "cryptographic ingredient" in preprocessing SNARKs. They formulated a new type of algebraic satisfaction problems, called *Quadratic Span Programs* (QSPs), which are expressive enough to allow for much simpler, and more efficient, cryptographic checking, essentially under the same assumptions used by Groth. In particular, they invested significant effort in obtaining an efficient reduction from circuit satisfiability to QSPs.

Comparing the latter to the probabilistic-checking-based approach described above, we note that a reduction to an algebraic satisfaction problem is a typical first step, because such satisfaction problems tend to be more amenable to probabilistic checking. As explained above, cryptographic tools are then usually invoked to enforce the relevant probabilistic-checking model (e.g., the PCP one). The aforementioned works [Gro10, Lip12, GGPR13], on the other hand, seem to somehow skip the probabilistic-checking step, and directly construct specific cryptographic tools for checking satisfiability of the algebraic problem itself. While this discrepancy may not be a problem per se, we believe that understanding it and formulating a clear methodology for the construction of preprocessing SNARKs are problems of great interest. Furthermore, a clear methodology may lead not only to a deeper conceptual understanding, but also to concrete improvements to different features of SNARKs (e.g., communication complexity, verifier complexity, prover complexity, and so on). Thus, we ask:

Is there a general methodology for the construction of preprocessing SNARKs?
Which improvements can it lead to?

1.3 Our Results

We present a general methodology for the construction of preprocessing SNARKs, as well as resulting concrete improvements. Our contribution is three-fold:

- We introduce a natural extension of the interactive proof model that considers *algebraically-bounded* provers. Concretely, we focus on *linear interactive proofs* (LIPs), where both honest and malicious provers are restricted to computing linear (or affine) functions of messages they receive over some finite field or ring. We then provide several (unconditional) constructions of succinct two-message LIPs for NP, obtained by applying simple and general transformations to two variants of PCPs.
- We give cryptographic transformations from (succinct two-message) LIPs to preprocessing SNARKs, based on different forms of *linear targeted malleability*, which can be instantiated based on existing knowledge assumptions. Our transformation is very intuitive: to force a prover to "act linearly" on the verifier message, simply encrypt (or encode) each field or ring element in the verifier message with an encryption scheme that only allows linear homomorphism.
- Following this methodology, we obtain several constructions that exhibit new efficiency features. These include "single-ciphertext preprocessing SNARKs" and improved succinctness-soundness tradeoffs. We also offer a new perspective on existing constructions of preprocessing SNARKs: namely, although existing constructions do not explicitly invoke PCPs, they can be reinterpreted as using *linear PCPs*, i.e., PCPs in which proof oracles (even malicious ones) are restricted to be a linear functions.[1]

We now discuss our results further, starting in Section 1.3.1 with the information-theoretic constructions of LIPs, followed in Section 1.3.2 by the cryptographic transformations to preprocessing SNARKs, and concluding in Section 1.3.3 with the new features we are able to obtain.

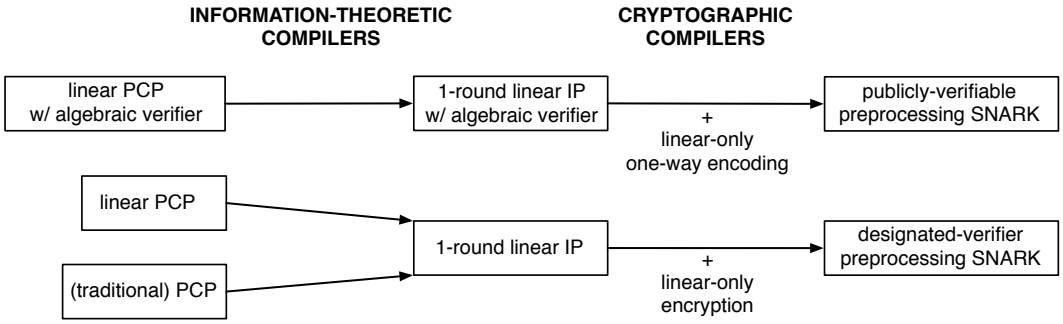

Figure 1: High-level summary of our transformations.

1.3.1 Linear interactive proofs.

The LIP model modifies the traditional interactive proofs model in a way analogous to the way the common study of algebraically-bounded "adversaries" modifies other settings, such as pseudorandomness [NN90, BV07] and randomness extraction [GR05, DGW09]. In the LIP model both honest and malicious provers are restricted to apply linear (or affine) functions over a finite field $\mathbb{F}$ to messages they receive from the

[1] A stronger notion of linear PCP has been used in other works [IKO07, SBW11, SMBW12, SVP+12, SBV+12] to obtain arguments for NP with nontrivial efficiency properties. See Section 1.4 for a comparison.

verifier. (The notion can be naturally generalized to apply over rings.) The choice of these linear functions can depend on auxiliary input to the prover (e.g., a witness), but not on the verifier's messages.

With the goal of non-interactive succinct verification in mind, we restrict our attention to (input-oblivious) *two-message* LIPs for boolean circuit satisfiability problems with the following template. To verify the relation $\mathcal{R}_C = \{(x, w) : C(x, w) = 1\}$ where C is a boolean circuit, the LIP verifier V_{LIP} sends to the LIP prover P_{LIP} a message $\mathbf{q}$ that is a vector of field elements, depending on C but not on x; V_{LIP} may also output a verification state $\mathbf{u}$. The LIP prover $P_{\mathsf{LIP}}(x, w)$ applies to $\mathbf{q}$ an affine transformation $\Pi = (\Pi', \boldsymbol{b})$, resulting in *only a constant number* of field elements. The prover's message $\mathbf{a} = \Pi' \cdot \mathbf{q} + \boldsymbol{b}$ can then be quickly verified (e.g., with $O(|x|)$ field operations) by V_{LIP}, and the soundness error is at most $O(1/|\mathbb{F}|)$. From here on, we shall use the term LIP to refer to LIPs that adhere to the above template.

LIP complexity measures. Our constructions provide different tradeoffs among several complexity measures of an LIP, which ultimately affect the features of the resulting preprocessing SNARKs. The two most basic complexity measures are the number of field elements sent by the verifier and the number of those sent by the prover. An additional measure that we consider in this work is the *algebraic complexity* of the verifier (when viewed as an $\mathbb{F}$-arithmetic circuit). Specifically, splitting the verifier into a query algorithm Q_{LIP} and a decision algorithm D_{LIP}, we say that it has *degree* (d_Q, d_D) if Q_{LIP} can be computed by a vector of multivariate polynomials of total degree d_Q each in the verifier's randomness, and D_{LIP} by a vector of multivariate polynomials of total degree d_D each in the LIP answers $\mathbf{a}$ and the verification state $\mathbf{u}$. Finally, of course, the running times of the query algorithm, decision algorithm, and prover algorithm are all complexity measures of interest. See Section 2.3 for a definition of LIPs and their complexity measures.

As mentioned above, our LIP constructions are obtained by applying general transformations to two types of PCPs. We now describe each of these transformations and the features they achieve. Some of the parameters of the resulting constructions are summarized in Table 1.

LIPs from linear PCPs. A *linear PCP* (LPCP) of length m is an oracle computing a linear function $\boldsymbol{\pi} \colon \mathbb{F}^m \to \mathbb{F}$; namely, the answer to each oracle query $\boldsymbol{q}_i \in \mathbb{F}^m$ is $a_i = \langle \boldsymbol{\pi}, \boldsymbol{q}_i \rangle$. Note that, unlike in an LIP where different affine functions, given by a matrix Π and shift $\boldsymbol{b}$, are applied to a message $\mathbf{q}$, in an LPCP there is one linear function $\boldsymbol{\pi}$, which is applied to different queries. (An LPCP with a *single* query can be viewed as a special case of an LIP.) This difference prevents a direct use of an LPCP as an LIP.

Our first transformation converts any (multi-query) LPCP into an LIP with closely related parameters. Concretely, we transform any k-query LPCP of length m over $\mathbb{F}$ into an LIP with verifier message in $\mathbb{F}^{(k+1)m}$, prover message in $\mathbb{F}^{k+1}$, and the same soundness error up to an additive term of $1/|\mathbb{F}|$. The transformation preserves the key properties of the LPCP, including the algebraic complexity of the verifier. Our transformation is quite natural: the verifier sends $\mathbf{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k+1})$ where $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ are the LPCP queries and $\boldsymbol{q}_{k+1} = \alpha_1 \boldsymbol{q}_1 + \ldots + \alpha_k \boldsymbol{q}_k$ is a random linear combination of these. The (honest) prover responds with $a_i = \langle \boldsymbol{\pi}, \boldsymbol{q}_i \rangle$, for $i = 1, \ldots, k+1$. To prevent a malicious prover from using inconsistent choices for $\boldsymbol{\pi}$, the verifier checks that $a_{k+1} = \alpha_1 a_1 + \ldots + \alpha_k a_k$.

By relying on two different LPCP instantiations, we obtain two corresponding LIP constructions:

- A variant of the Hadamard-based PCP of Arora et al. [ALM+98a] (ALMSS), extended to work over an arbitrary finite field $\mathbb{F}$, yields a very simple LPCP with three queries. After applying our transformation, for a circuit C of size s and input length n, the resulting LIP for $\mathcal{R}_C$ has verifier message in $\mathbb{F}^{O(s^2)}$, prover message in $\mathbb{F}^4$, and soundness error $O(1/|\mathbb{F}|)$. When viewed as $\mathbb{F}$-arithmetic circuits, the prover P_{LIP} and query algorithm Q_{LIP} are both of size $O(s^2)$, and the decision algorithm is of size $O(n)$. Furthermore, the degree of $(Q_{\mathsf{LIP}}, D_{\mathsf{LIP}})$ is $(2, 2)$.

- A (strong) quadratic span program (QSP), as defined by Gennaro et al. [GGPR13], directly yields a

corresponding LPCP with three queries. For a circuit C of size s and input length n, the resulting LIP for $\mathcal{R}_C$ has verifier message in $\mathbb{F}^{O(s)}$, prover message in $\mathbb{F}^4$, and soundness error $O(s/|\mathbb{F}|)$. When viewed as $\mathbb{F}$-arithmetic circuits, the prover P_{LIP} is of size $\widetilde{O}(s)$, the query algorithm Q_{LIP} is of size $O(s)$, and the decision algorithm is of size $O(n)$. The degree of $(Q_{\mathsf{LIP}}, D_{\mathsf{LIP}})$ is $(O(s), 2)$.

A notable feature of the LIPs obtained above is the very low "online complexity" of verification: in both cases, the decision algorithm is an arithmetic circuit of size $O(n)$. Moreover, all the efficiency features mentioned above apply not only to satisfiability of boolean circuits C, but also to satisfiability of $\mathbb{F}$-arithmetic circuits.

In both the above constructions, the circuit to be verified is first represented as an appropriate algebraic satisfaction problem, and then probabilistic checking machinery is invoked. In the first case, the problem is a system of quadratic equations over $\mathbb{F}$, and, in the second case, it is a (strong) quadratic span program (QSP) over $\mathbb{F}$. These algebraic problems are the very same problems underlying [Gro10, Lip12] and [GGPR13].

As explained earlier, [GGPR13] invested much effort to show an efficient reduction from circuit satisfiability problems to QSPs. Our work does *not* subsume nor simplify the reduction to QSPs of [GGPR13], but instead reveals a simple LPCP to check a QSP, and this LPCP can be plugged into our general transformations. Reducing circuit satisfiability to a system of quadratic equations over $\mathbb{F}$ is much simpler, but generating proofs for the resulting problem is quadratically more expensive. (Concretely, both [Gro10] and [Lip12] require $O(s^2)$ computation already in the preprocessing phase). See Section 3.1 for more details.

LIPs from traditional PCPs. Our second transformation relies on traditional "unstructured" PCPs. These PCPs are typically more difficult to construct than LPCPs; however, our second transformation has the advantage of requiring the prover to send only a *single* field element. Concretely, our transformation converts a traditional k-query PCP into a 1-query LPCP, over a sufficiently large field. Here the PCP oracle is represented via its truth table, which is assumed to be a binary string of polynomial size (unlike the LPCPs mentioned above, whose truth tables have size that is exponential in the circuit size). The transformation converts any k-query PCP of proof length m and soundness error ε into an LIP, with soundness error $O(\varepsilon)$ over a field of size $2^{O(k)}/\varepsilon$, in which the verifier sends m field elements and receives only a single field element in return. The high-level idea is to use a sparse linear combination of the PCP entries to pack the k answer bits into a single field element. The choice of this linear combination uses additional random noise to ensure that the prover's coefficients are restricted to binary values, and uses easy instances of subset-sum to enable an efficient decoding of the k answer bits.

Taking time complexity to an extreme, we can apply this transformation to the PCPs of Ben-Sasson et al. [BSCGT13b] and get LIPs where the prover and verifier complexity are both optimal up to $\mathrm{polylog}(s)$ factors, but where the prover sends a single element in a field of size $|\mathbb{F}| = 2^{\lambda \cdot \mathrm{polylog}(s)}$. Taking succinctness to an extreme, we can apply our transformation to PCPs with soundness error $2^{-\lambda}$ and $O(\lambda)$ queries, obtaining an LIP with similar soundness error in which the prover sends a single element in a field of size $|\mathbb{F}| = 2^{\lambda \cdot O(1)}$. For instance, using the query-efficient PCPs of Håstad and Khot [HK05], the field size is only $|\mathbb{F}| = 2^{\lambda \cdot (3+o(1))}$.[2] (Jumping ahead, this means that a field element can be encrypted using a *single*, *normal-size* ciphertext of homomorphic encryption schemes such as Paillier or Elgamal even when $\lambda = 100$.) On the down side, the degrees of the LIP verifiers obtained via this transformation are high; we give evidence that this is inherent when starting from "unstructured" PCPs. See Section 3.2 for more details.

Honest-verifier zero-knowledge LIPs. We also show how to make the above LIPs zero-knowledge against honest verifiers (HVZK). Looking ahead, using HVZK LIPs in our cryptographic transformations results in

[2] In the case of [HK05], we do not obtain an input-oblivious LIP, because the queries in their PCP depend on the input. While it is plausible to conjecture that the queries can be made input-oblivious, we did not check that.

preprocessing SNARKs that are zero-knowledge (against malicious verifiers in the CRS model).

For the Hadamard-based LIP, an HVZK variant can be obtained directly with essentially no additional cost. More generally, we show how to transform *any* LPCP where the decision algorithm is of low degree to an HVZK LPCP with the same parameters up to constant factors (see Appendix B); this HVZK LPCP can then be plugged into our first transformation to obtain an HVZK LIP. Both of the LPCP constructions mentioned earlier satisfy the requisite degree constraints.

For the second transformation, which applies to traditional PCPs (whose verifiers, as discussed above, must have high degree and thus cannot benefit from our general HVZK transformation), we show that if the PCP is HVZK (see [DFK$^+$92] for efficient constructions), then so is the resulting LIP; in particular, the HVZK LIP answer still consists of a *single* field element.

Proof of knowledge. In each of the above transformations, we ensure not only soundness for the LIP, but also a proof of knowledge property. Namely, it is possible to efficiently extract from a convincing affine function Π a witness for the underlying statement. The proof of knowledge property is then preserved in the subsequent cryptographic compilations, ultimately allowing to establish the proof of knowledge property for the preprocessing SNARK. As discussed in Section 1.1, proof of knowledge is a very desirable property for preprocessing SNARKs; for instance, it enables to remove the preprocessing phase, as well as to improve the complexity of the prover and verifier, via the result of [BCCT13].

Thm. number	starting point of LIP construction	# field elements in verifier message	# field elements in prover message	algebraic properties of verifier	field size for $2^{-\lambda}$ knowledge error
3.3	Hadamard PCP	$O(s^2)$	4	$(d_Q, d_D) = (2, 2)$	$2^\lambda \cdot O(1)$
3.4	QSPs of [GGPR13]	$O(s)$	4	$(d_Q, d_D) = (O(s), 2)$	$2^\lambda \cdot O(s)$
3.9	PCPs of [BSCGT13b]	$\widetilde{O}(s)$	1	none	$2^{\lambda \cdot \mathrm{polylog}(s)}$
3.10	PCPs of [HK05]	$\mathrm{poly}(s)$	1	none	$2^{\lambda \cdot (3+o(1))}$

Table 1: Summary of our LIP constructions. See each theorem for more details, including the running times of the prover, query, and decision algorithms.

1.3.2 Preprocessing SNARKs from LIPs.

We explain how to use cryptographic tools to transform an LIP into a corresponding preprocessing SNARK. At high level, the challenge is to ensure that an arbitrary (yet computationally-bounded) prover behaves as if it was a linear (or affine) function. The idea, which also implicitly appears in previous constructions, is to use an encryption scheme with targeted malleability [BSW12] for the class of affine functions: namely, an encryption scheme that "only allows affine homomorphic operations" on an encrypted plaintext (and these operations are independent of the underlying plaintexts). Intuitively, the verifier would simply encrypt each field element in the LIP message $\mathbf{q}$, send the resulting ciphertexts to the prover, and have the prover homomorphically evaluate the LIP affine function on the ciphertexts; targeted malleability ensures that malicious provers can only invoke (malicious) affine strategies.

We concretize the above approach in several ways, depending on the properties of the LIP and the exact flavor of targeted malleability; different choices will induce different properties for the resulting preprocessing SNARK. In particular, we identify natural sufficient properties that enable an LIP to be compiled into a *publicly-verifiable* SNARK. We also discuss possible instantiations of the cryptographic tools, based on existing knowledge assumptions. (Recall that, in light of the negative result of [GW11], the use of nonstandard

cryptographic assumptions seems to be justified.)

Designated-verifier preprocessing SNARKs from arbitrary LIPs. First, we show that *any* LIP can be compiled into a corresponding *designated-verifier* preprocessing SNARK with similar parameters. (Recall that "designated verifier" means that the verifier needs to maintain a secret verification state.) To do so, we rely on what we call *linear-only* encryption: an additively homomorphic encryption that is (a) semantically-secure, and (b) linear-only. The linear-only property essentially says that, given a public key pk and ciphertexts $\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m)$, it is infeasible to compute a new ciphertext c' in the image of $\mathsf{Enc}_{\mathsf{pk}}$, except by "knowing" $\beta, \alpha_1, \ldots, \alpha_m$ such that $c' \in \mathsf{Enc}_{\mathsf{pk}}(\beta + \sum_{i=1}^{m} \alpha_i a_i)$. Formally, the property is captured by guaranteeing that, whenever $A(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m))$ produces valid ciphertexts $(c'_1, \ldots, c'_k)$, an efficient extractor E (non-uniformly depending on A) can extract a corresponding affine function Π "explaining" the ciphertexts. As a candidate for such an encryption scheme, we propose variants of Paillier encryption [Pai99] (as also considered in [GGPR13]) and of Elgamal encryption [EG85] (in those cases where the plaintext is guaranteed to belong to a polynomial-size set, so that decryption can be done efficiently). These variants are "sparsified" versions of their standard counterparts; concretely, a ciphertext does not only include $\mathsf{Enc}_{\mathsf{pk}}(a)$, but also $\mathsf{Enc}_{\mathsf{pk}}(\alpha \cdot a)$, for a secret field element α. (This "sparsification" follows a pattern found in many constructions conjectured to satisfy "knowledge-of-exponent" assumptions.) As for Paillier encryption, we have to consider LIPs over the ring $\mathbb{Z}_{\mathfrak{pq}}$ (instead of a finite field $\mathbb{F}$); essentially, the same results also hold in this setting (except that soundness is $O(1/\min\{\mathfrak{p}, \mathfrak{q}\})$ instead of $O(1/|\mathbb{F}|)$).

We also consider a notion of targeted malleability, weaker than linear-only encryption, that is closer to the definition template of Boneh et al. [BSW12]. In such a notion, the extractor is replaced by a simulator. Relying on this weaker variant, we are only able to prove the security of our preprocessing SNARKs against *non-adaptive* choices of statements (and still prove soundness, though not proof of knowledge, if the simulator is allowed to be inefficient). Nonetheless, for natural instantiations, even adaptive security seems likely to hold for our construction, but we do not know how to prove it. One advantage of working with this weaker variant is that it seems to allow for more efficient candidates constructions. Concretely, the linear-only property rules out any encryption scheme where ciphertexts can be sampled obliviously; instead, the weaker notion does not, and thus allows for shorter ciphertexts. For example, we can consider a standard ("non-sparsified") version of Paillier encryption. We will get back to this point in Section 1.3.3.

For further details on the above transformations, see Section 6.1.

Publicly-verifiable preprocessing SNARKs from LIPs with low-degree verifiers. Next, we identify properties of LIPs that are sufficient for a transformation to *publicly-verifiable* preprocessing SNARKs. Note that, if we aim for public verifiability, we cannot use semantically-secure encryption to encode the message of the LIP verifier, because we need to "publicly test" (without decryption) certain properties of the plaintext underlying the prover's response. The idea, implicit in previous publicly-verifiable preprocessing SNARK constructions, is to use linear-only encodings (rather than encryption) that do allow such public tests, while still providing certain one-wayness properties. When using such encodings with an LIP, however, it must be the case that the public tests support evaluating the decision algorithm of the LIP and, moreover, the LIP remains secure despite some "leakage" on the queries. We show that LIPs with *low-degree verifiers* (which we call *algebraic* LIPs), combined with appropriate one-way encodings, suffice for this purpose.

More concretely, like [Gro10, Lip12, GGPR13], we consider candidate encodings in bilinear groups under similar knowledge-of-exponent and computational Diffie-Hellman assumptions; for such encoding instantiations, we must start with an LIP where the degree d_D of the decision algorithm D_{LIP} is at most quadratic. (If we had multilinear maps supporting higher-degree polynomials, we could support higher values of d_D.) In addition to $d_D \leq 2$, to ensure security even in the presence of certain one-way leakage, we need the query algorithm Q_{LIP} to be of polynomial degree.

Both of the LIP constructions from LPCPs described in Section 1.3.1 satisfy these requirements. When combined with the above transformation, these LIP constructions imply new constructions of publicly-verifiable preprocessing SNARKs, one of which can be seen as a simplification of the construction of [Gro10] and the other as a reinterpretation (and slight simplification) of the construction of [GGPR13].

For more details, see Section 6.2.

Zero-knowledge. In all aforementioned transformations to preprocessing SNARKs, if we start with an HVZK LIP (such as those mentioned in Section 1.3.1) and additionally require a rerandomization property for the linear-only encryption/encoding (which is available in all of the candidate instantiations we consider), we obtain preprocessing SNARKs that are (perfect) zero-knowledge in the CRS model. In addition, for the case of *publicly-verifiable* (perfect) zero-knowledge preprocessing SNARKs, the CRS can be tested, so that (similarly to previous works [Gro10, Lip12, GGPR13]) we also obtain succinct ZAPs. See Section 6.3.

1.3.3 New efficiency features for SNARKs.

We obtain the following concrete improvements in communication complexity for preprocessing SNARKs.

"Single-ciphertext preprocessing SNARKs". If we combine the LIPs that we obtained from traditional PCPs (where the prover returns only a single field element) with "non-sparsified" Paillier encryption, we obtain (non-adaptive) preprocessing SNARKs that consist of *a single Paillier cipherext*. Moreover, when using the query-efficient PCP from [HK05] as the underlying PCP, even a standard-size Paillier ciphertext (with plaintext group $\mathbb{Z}_{\mathfrak{pq}}$ where $\mathfrak{p}, \mathfrak{q}$ are 512-bit primes) suffices for achieving soundness error $2^{-\lambda}$ with $\lambda = 100$. (For the case of [HK05], due to the queries' dependence on the input, the reference string of the SNARK also depends on the input.) Alternatively, using the sparsified version of Paillier encryption, we can also get security against adaptively-chosen statements with only *two Paillier ciphertexts*.

Towards optimal succinctness. A fundamental question about succinct arguments is how low can we push communication complexity. More accurately: what is the optimal tradeoff between communication complexity and soundness? Ideally, we would want succinct arguments that are *optimally succinct*: to achieve $2^{-\Omega(\lambda)}$ soundness against $2^{O(\lambda)}$-bounded provers, the proof length is $O(\lambda)$ bits long.

In existing constructions of succinct arguments, interactive or not, to provide $2^{-\Omega(\lambda)}$ soundness against $2^{O(\lambda)}$-bounded provers, the prover has to communicate $\omega(\lambda)$ bits to the verifier. Concretely, PCP-based (and MIP-based) solutions require $\Omega(\lambda^3)$ bits of communication. This also holds for known preprocessing SNARKs, because previous work and the constructions discussed above are based on bilinear groups or Paillier encryption, both of which suffer from subexponential-time attacks.

If we had a candidate for (linear-only) homomorphic encryption that did not suffer from subexponential-time attacks, our approach could perhaps yield preprocessing SNARKs that are optimally succinct. The only known such candidate is Elgamal encryption (say, in appropriate elliptic curve groups) [PQ12]. However, the problem with using Elgamal decryption in our approach is that it requires, in general, to compute discrete logarithms.

One way to overcome this problem is to ensure that honest proofs are always decrypted to a known polynomial-size set. This can be done by taking the LIP to be over a field $\mathbb{F}_{\mathfrak{p}}$ of only polynomial size, and ensuring that any honest proof $\boldsymbol{\pi}$ has small ℓ_1-norm $\|\boldsymbol{\pi}\|_1$, so that in particular, the prover's answer is taken from a set of size at most $\|\boldsymbol{\pi}\|_1 \cdot \mathfrak{p}$. For example, in the two LPCP-based constructions described in Section 1.3.1, this norm is $O(s^2)$ and $O(s)$ respectively for a circuit of size s. This approach, however, has two caveats: the soundness of the underlying LIP is only $1/\mathrm{poly}(\lambda)$ and moreover, the verifier's running time is proportional to s, and not independent of it, as we usually require.

A very interesting related question that may lead to a solution circumventing the aforementioned caveats

is whether there exist LIPs where the decision algorithm has *linear* degree. With such an LIP, we would be able to directly use Elgamal encryption because linear tests on the plaintexts can be carried out "in the exponent", without having to take discrete logarithms.

Finally, a rather generic approach for obtaining "almost-optimal succintness" is to use (linear-only) Elgamal encryption in conjunction with any linear homomorphic encryption scheme (perhaps not having the linear-only property) that is sufficiently secure. Concretely, the verifier sends his LIP message encrypted under both encryption schemes, and then the prover homomorphically evaluates the affine function on both. The additional ciphertext can be efficiently decrypted, and can assist in the decryption of the Elgamal ciphertext. For example, there are encryption schemes based on Ring-LWE [LPR10] that are conjectured to have quasiexponential security; by using these in the approach we just discussed, we can obtain $2^{-\Omega(\lambda)}$ soundness against $2^{O(\lambda)}$-bounded provers with $\widetilde{O}(\lambda)$ bits of communication.

Strong knowledge and reusability. Designated-verifier SNARKs typically suffer from a problem known as the *verifier rejection problem*: security is compromised if the prover can learn the verifier's responses to multiple adaptively-chosen statements and proofs. For example, the PCP-based (or MIP-based) SNARKs of [BCCT12, GLR11, DFH12, BC12] suffer from the verifier rejection problem because a prover can adaptively learn the encrypted PCP (or MIP) queries, by feeding different statements and proofs to the verifier and learning his responses, and since the secrecy of these queries is crucial, security is lost.

Of course, one way to avoid the verifier rejection problem is to generate a new reference string for each statement and proof. Indeed, this is an attractive solution for the aforementioned SNARKs because generating a new reference string is very cheap: it costs $\mathrm{poly}(\lambda)$. However, for a designated-verifier *preprocessing* SNARK, generating a new reference string is not cheap at all, and being able to *reuse* the same reference string across an unbounded number of adaptively-chosen statements and proofs is a very desirable property.

A property that is satisfied by all algebraic LIPs (including the LPCP-based LIPs discussed in Section 1.3.1), which we call *strong knowledge*, is that such attacks are impossible. Specifically, for such LIPs, every prover either makes the verifier accept with probability 1 or with probability less than $O(\mathrm{poly}(\lambda)/|\mathbb{F}|)$. (In Appendix C, we also show that traditional "unstructured" PCPs cannot satisfy this property.) Given LIPs with strong knowledge, it seems that designated-verifier SNARKs that have a reusable reference string can be constructed. Formalizing the connection between strong knowledge and reusable reference string actually requires notions of linear-only encryption that are somewhat more delicate than those we have considered so far. See details in Appendix C for additional discussions.

1.4 Structured PCPs In Other Works

Ishai et al. [IKO07] proposed the idea of constructing argument systems with nontrivial efficiency properties by using "structured" PCPs and cryptographic primitives with homomorphic properties, rather than (as in previous approaches) "unstructured" polynomial-size PCPs and collision-resistant hashing. We have shown how to apply this basic approach in order to obtain succinct non-interactive arguments with preprocessing. We now compare our work to other works that have also followed the basic approach of [IKO07].

Strong vs. weak linear PCPs. Both in our work and in [IKO07], the notion of a "structured" PCP is taken to be a *linear PCP*. However, the notion of a linear PCP used in our work does not coincide with the one used in [IKO07]. Indeed there are two ways in which one can formalize the intuitive notion of a linear PCP. Specifically:

- A *strong* linear PCP is a PCP in which the honest proof oracle is guaranteed to be a linear function, and soundness is required to hold for *all* (including non-linear) proof oracles.

- A *weak* linear PCP is a PCP in which the honest proof oracle is guaranteed to be a linear function, and soundness is required to hold *only* for linear proof oracles.

In particular, a weak linear PCP assumes an algebraically-bounded prover, while a strong linear PCP does not. While Ishai et al. [IKO07] considered strong linear PCPs, in our work we are interested in studying algebraically-bounded provers, and thus consider weak linear PCPs.

Arguments from strong linear PCPs. Ishai et al. [IKO07] constructed a four-message argument system for NP in which the prover-to-verifier communication is short (i.e., an argument with a *laconic* prover [GVW02]) by combining a strong linear PCP and (standard) linear homomorphic encryption; they also showed how to extend their approach to "balance" the communication between the prover and verifier and obtain a $O(1/\varepsilon)$-message argument system for NP with $O(n^{\varepsilon})$ communication complexity. Let us briefly compare their work with ours.

First, in this paper we focus on the non-interactive setting, while Ishai et al. focused on the interactive setting. In particular, in light of the negative result of Gentry and Wichs [GW11], this means that the use of non-standard assumptions in our setting (such as linear targeted malleability) may be justified; in contrast, Ishai et al. only relied on the standard semantic security of linear homomorphic encryption (and did not rely on linear targeted malleability properties). Second, we focus on constructing (non-interactive) succinct arguments, while Ishai et al. focus on constructing arguments with a laconic prover. Third, by relying on weak linear PCPs (instead of strong linear PCPs) we do not need to perform (explicitly or implicitly) linearity testing, while Ishai et al. do. Intuitively, this is because we rely on the assumption of linear targeted malleability, which ensures that a prover is algebraically bounded (in fact, in our case, linear); not having to perform proximity testing is crucial for preserving the algebraic properties of a linear PCP (and thus, e.g., obtain public verifiability) and obtaining $O(\mathrm{poly}(\lambda)/|\mathbb{F}|)$ soundness with only a constant number of encrypted/encoded group elements. (Recall that linearity testing only guarantees constant soundness with a constant number of queries.)

Turning to computational efficiency, while their basic protocol does not provide the verifier with any saving in computation, Ishai et al. noted that their protocol actually yields a *batching argument*: namely, an argument in which, in order to simultaneously verify the correct evaluation of ℓ circuits of size S, the verifier may run in time S (i.e., in time S/ℓ per circuit evaluation). In fact, a set of works [SBW11, SMBW12, SVP$^+$12, SBV$^+$12] has improved upon, optimized, and implemented the batching argument of Ishai et al. [IKO07] for the purpose of verifiable delegation of computation.

Finally, [SBV$^+$12] have also observed that QSPs can be used to construct weak linear PCPs; while we compile weak linear PCPs into LIPs, [SBV$^+$12] (as in previous work) compile weak linear PCPs into strong ones. Indeed, note that a weak linear PCP can always be compiled into a corresponding strong one, by letting the verifier additionally perform linearity testing and self-correction; this compilation does not affect proof length, increases query complexity by only a constant multiplicative factor, and guarantees constant soundness.

Remark 1.1. The notions of (strong or linear) PCP discussed above should not be confused with the (unrelated) notion of a *linear PCP of Proximity* (linear PCPP) [BSHLM09, Mei12], which we now recall for the purpose of comparison.

Given a field $\mathbb{F}$, an $\mathbb{F}$-*linear circuit* [Val77] is an $\mathbb{F}$-arithmetic circuit $C\colon \mathbb{F}^h \to \mathbb{F}^\ell$ in which every gate computes an $\mathbb{F}$-linear combination of its inputs; its *kernel*, denoted $\ker(C)$, is the set of all $w \in \mathbb{F}^h$ for which $C(w) = 0^\ell$. A *linear PCPP* for a field $\mathbb{F}$ is an oracle machine V with the following properties: (1) V takes as input an $\mathbb{F}$-linear circuit C and has oracle access to a vector $w \in \mathbb{F}^h$ and an auxiliary vector π of elements in $\mathbb{F}$, (2) if $w \in \ker(C)$ then there exists π so that $V^{w,\pi}(C)$ accepts with probability 1, and (3) if w is far from $\ker(C)$ then $V^{w,\pi}(C)$ rejects with high probability for every π.

Thus, a linear PCPP is a proximity tester for the kernels of linear circuits (which are not universal), while a (strong or weak) linear PCP is a PCP in which the proof oracle is a linear function.

1.5 Organization

In Section 2, we introduce the notions of LPCPs and LIPs. In Section 3, we present our transformations for constructing LIPs from several notions of PCPs. In Section 4, we give the basic definitions for preprocessing SNARKs. In Section 5, we define the relevant notions of linear targeted malleability, as well as candidate constructions for these. In Section 6, we present our transformations from LIPs to preprocessing SNARKs. In Appendix A, we discuss two constructions of algebraic LPCPs. In Appendix B, we present our general transformation to obtain HVZK for LPCPs with low-degree decision algorithms. In Appendix C, we discuss the notion of strong knowledge and its connection to designated-verifier SNARKs with a reusable reference string.

2 Definitions of LIPs and LPCPs

We begin with the information-theoretic part of the paper, by introducing the basic definitions of LPCPs, LIPs, and relevant conventions.

2.1 Polynomials, Degrees, and Schwartz–Zippel

Vectors are denoted in bold, while their coordinates are not; for example, we may write $\boldsymbol{a}$ to denote the ordered tuple $(a_1, \ldots, a_n)$ for some n. A field is denoted $\mathbb{F}$; we always work with fields that are finite. We say that a multivariate polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ has *degree* d if the total degree of f is at most d. A *multivalued multivariate polynomial* $\mathbf{f}\colon \mathbb{F}^m \to \mathbb{F}^\mu$ is a vector of polynomials $(f_1, \ldots, f_\mu)$ where each $f_i\colon \mathbb{F}^m \to \mathbb{F}$ is a (single-valued) multivariate polynomial.

A very useful fact about polynomials is the following:

Lemma 2.1 (Schwartz–Zippel)**.** *Let $\mathbb{F}$ be any field. For any nonzero polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ of total degree d and any finite subset S of $\mathbb{F}$,*

$$\Pr_{\boldsymbol{s}\leftarrow S^m}\Big[f(\boldsymbol{s}) = 0\Big] \leq \frac{d}{|S|} \ .$$

In particular, any two distinct polynomials $f, g\colon \mathbb{F}^m \to \mathbb{F}$ of total degree d can agree on at most a $d/|S|$ fraction of the points in S^m.

2.2 Linear PCPs

A *linear* probabilistically-checkable proof (LPCP) system for a relation $\mathcal{R}$ over a field $\mathbb{F}$ is one where the PCP oracle is restricted to compute a linear function $\boldsymbol{\pi}\colon \mathbb{F}^m \to \mathbb{F}$ of the verifier's queries. Viewed as a traditional PCP, $\boldsymbol{\pi}$ has length $|\mathbb{F}|^m$ (and alphabet $\mathbb{F}$). For simplicity, we ignore the computational complexity issues in the following definition, and refer to them later when they are needed.

Definition 2.2 (Linear PCP (LPCP))**.** *Let $\mathcal{R}$ be a binary relation, $\mathbb{F}$ a finite field, P_{LPCP} a deterministic prover algorithm, and V_{LPCP} a probabilistic oracle verifier algorithm. We say that the pair $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is a (input-oblivious) k-query* **linear PCP** *for $\mathcal{R}$ over $\mathbb{F}$ with knowledge error ε and query length m if it satisfies the following requirements:*

- *Syntax. On any input x and oracle $\boldsymbol{\pi}$, the verifier $V^{\boldsymbol{\pi}}_{\mathsf{LPCP}}(x)$ makes k input-oblivious queries to $\boldsymbol{\pi}$ and then decides whether to accept or reject. More precisely, V_{LPCP} consists of a probabilistic query algorithm Q_{LPCP} and a deterministic decision algorithm D_{LPCP} working as follows. Based on its internal randomness, and independently of x, Q_{LPCP} generates k queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k \in \mathbb{F}^m$ to $\boldsymbol{\pi}$ and state information $\mathbf{u}$; then, given x, $\mathbf{u}$, and the k oracle answers $a_1 = \langle \boldsymbol{\pi}, \boldsymbol{q}_1 \rangle, \ldots, a_k = \langle \boldsymbol{\pi}, \boldsymbol{q}_k \rangle$, D_{LPCP} accepts or rejects.*

- *Completeness. For every $(x, w) \in \mathcal{R}$, the output of $P_{\mathsf{LPCP}}(x, w)$ is a description of a linear function $\boldsymbol{\pi} \colon \mathbb{F}^m \to \mathbb{F}$ such that $V^{\boldsymbol{\pi}}_{\mathsf{LPCP}}(x)$ accepts with probability* 1.

- *Knowledge. There exists a knowledge extractor E_{LPCP} such that for every linear function $\boldsymbol{\pi}^* \colon \mathbb{F}^m \to \mathbb{F}$ if the probability that $V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x)$ accepts is greater than ε then $E^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x)$ outputs w such that $(x, w) \in \mathcal{R}$.*[3]

Furthermore, we say that $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has **degree** *(d_Q, d_D) if, additionally,*

1. *the query algorithm Q_{LPCP} is computed by a degree d_Q arithmetic circuit (i.e., there are k polynomials $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k \colon \mathbb{F}^\mu \to \mathbb{F}^m$ and state polynomial $\boldsymbol{p} \colon \mathbb{F}^\mu \to \mathbb{F}^{m'}$, all of degree d_Q, such that the LPCP queries are $\boldsymbol{q}_1 = \boldsymbol{p}_1(\boldsymbol{r}), \ldots, \boldsymbol{q}_k = \boldsymbol{p}_k(\boldsymbol{r})$ and the state is $\mathbf{u} = \boldsymbol{p}(\boldsymbol{r})$ for a random $\boldsymbol{r} \in \mathbb{F}^\mu$), and*

2. *the decision algorithm D_{LPCP} is computed by a degree d_D arithmetic circuit (i.e., for every input x there is a test polynomial $\boldsymbol{t}_x \colon \mathbb{F}^{m'+k} \to \mathbb{F}^\eta$ of degree d_D such that $\boldsymbol{t}_x(\mathbf{u}, a_1, \ldots, a_k) = 0^\eta$ if and only if $D_{\mathsf{LPCP}}(x, \mathbf{u}, a_1, \ldots, a_k)$ accepts);*

Finally, for a security parameter λ, we say that $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is an **algebraic LPCP** *(for λ) if it has degree* $(\mathrm{poly}(\lambda), \mathrm{poly}(\lambda))$.

Remark 2.3 (infinite relations $\mathcal{R}$)**.** When $\mathcal{R}$ is an infinite relation $\cup_{\ell \in \mathbb{N}} \mathcal{R}_\ell$, both $V_{\mathsf{LPCP}} = (Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}})$ and P_{LPCP} also get as input 1^ℓ. In this case, all parameters k, m, μ, m', η may also be a function of ℓ.

Some of the aforementioned properties only relate to the LPCP verifier V_{LPCP}, so we will also say things like "V_{LPCP} has degree...", i.e., using the verifier as the subject (rather than the LPCP).

Honest-verifier zero-knowledge LPCPs. We also consider *honest-verifier zero-knowledge* (HVZK) LPCPs. In an HVZK LPCP, soundness or knowledge is defined as in a usual LPCP, and HVZK is defined as in a usual HVZK PCP. For convenience, let us recall the definition of a HVZK PCP:

Definition 2.4 (honest-verifier zero-knowledge PCP (HVZK PCP))**.** *A PCP system $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ for a relation $\mathcal{R}$, where P_{PCP} is also probabilistic, is* **δ-statistical HVZK** *if there exists a simulator S_{PCP}, running in expected polynomial time, for which the following two ensembles are δ-close (δ can be a function of the field, input length, and so on):*

$$\left\{S_{\mathsf{PCP}}(x)\right\}_{(x,w)\in\mathcal{R}} \text{ and } \left\{\mathrm{View}\left(V^{\pi_{x,w}}_{\mathsf{PCP}}(x)\right) \mid \pi_{x,w} \leftarrow P_{\mathsf{PCP}}(x,w)\right\}_{(x,w)\in\mathcal{R}} \quad ,$$

where View *represents the view of the verifier, including its coins and the induced answers according to π.*

If the above two distributions are identically distributed then we say that $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ is **perfect** *HVZK.*

[3] In particular, $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has soundness error ε: for every x such that $(x, w) \notin \mathcal{R}$ for all w, and for every linear function $\boldsymbol{\pi}^* \colon \mathbb{F}^m \to \mathbb{F}$, the probability that $V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x)$ accepts is at most ε.

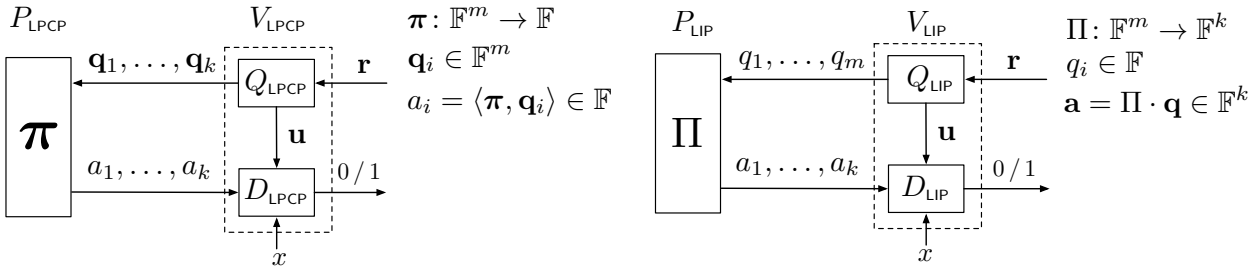

Figure 2: Diagram of an LPCP and an input-oblivious two-message LIP.

2.3 Linear Interactive Proofs

A linear interactive proof (LIP) is defined similarly to a standard interactive proof [GMR89], except that each message sent by a prover (either an honest or a malicious one) must be a linear function of the previous messages sent by the verifier. In fact, it will be convenient for our purposes to consider a slightly weaker notion that allows a malicious prover to compute an affine function of the messages. While we will only make use of two-message LIPs in which the verifier's message is independent of its input, below we define the more general notion.

Definition 2.5 (Linear Interactive Proof (LIP))**.** *A* **linear interactive proof** *over a finite field $\mathbb{F}$ is defined similarly to a standard interactive proof [GMR89], with the following differences.*

- *Each message exchanged between the prover P_{LIP} and the verifier V_{LIP} is a vector $\mathbf{q}_i \in \mathbb{F}^m$ over $\mathbb{F}$.*
- *The honest prover's strategy is linear in the sense that each of the prover's messages is computed by applying some linear function $\Pi_i \colon \mathbb{F}^m \to \mathbb{F}^k$ to the verifier's previous messages $(\mathbf{q}_1, \ldots, \mathbf{q}_i)$. This function is determined only by the input x, the witness w, and the round number i.*
- *Knowledge should only hold with respect to affine prover strategies $\Pi^* = (\Pi, \boldsymbol{b})$, where Π is a linear function, and $\boldsymbol{b}$ is some affine shift.*

Analogously to the case of LPCPs (Definition 2.2), we say that a two-message LIP is **input-oblivious** *if the verifier's messages do not depend on the input x. In such a case the verifier can be split into a query algorithm Q_{LIP} that outputs the query $\mathbf{q}$ and possibly a verification state $\mathbf{u}$, and a decision algorithm D_{LIP} that takes as input $\mathbf{u}$, x, and the LIP answer $\Pi \cdot \mathbf{q}$. We also consider notions of degree and algebraic LIPs, also defined analogously to the LPCP case.*

Remark 2.6 (LPCPs and LIPs over rings)**.** The notions of LPCP and an LIP can be easily extended to be over a *ring* rather than over a field. One case of particular interest is LIPs over $\mathbb{Z}_N$, where N is the product of two primes $\mathfrak{p}$ and $\mathfrak{q}$. (LIPs over $\mathbb{Z}_N$ are needed, e.g., when used in conjunction with Paillier encryption; see Section 5.3.) All of our results generalize, rather directly, to the case of $\mathbb{Z}_N$, where instead of achieving soundness-error $O(1/|\mathbb{F}|)$, we achieve soundness $O(1/\min\{\mathfrak{p}, \mathfrak{q}\})$. For simplicity, when presenting most results, we shall restrict attention to fields.

Remark 2.7 (honest-verifier zero knowledge)**.** We also consider an honest-verifier zero-knowledge variant of LIPs (HVZK LIPs), which is defined analogously to Definition 2.4. In this case, the honest prover is probabilistic.

Remark 2.8 (LIP vs. LPCP)**.** Note that a one-query LPCP is an LIP where the prover returns a single field element; however, when the prover returns more than one field element, an LIP is not a one-query LPCP. In this paper we construct both LIPs where the prover returns more than a single field element (see Section 3.1) and LIPs where the prover returns a single field element (see Section 3.2).

3 Constructions of LIPs

We present two transformations for constructing LIPs, in Section 3.1, Section 3.2 respectively.

3.1 LIPs From LPCPs

We show how to transform any LPCP into a two-message LIP with similar parameters. Crucially, our transformation does not significantly affect strong knowledge or algebraic properties of the LPCP verifier. Note that a non-trivial transformation is indeed required in general because the LIP verifier cannot simply send to the LIP prover the queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ generated by the LPCP verifier. Unlike in the LPCP model, there is no guarantee that the LIP prover will apply the *same* linear function to each of these queries; instead, we only know that the LIP prover will apply some affine function Π to the concatenation of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$. Thus, we show how to transform any LPCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ with knowledge error ε into a two-message LIP $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ with knowledge error at most $\varepsilon + \frac{1}{|\mathbb{F}|}$. If the LPCP has k queries of length m and is over a field $\mathbb{F}$, then the LIP verifier V_{LIP} will send $(k+1)m$ field elements and receive $(k+1)$ field elements from the LIP prover P_{LIP}. The idea of the transformation is for V_{LIP} to run V_{LPCP} and then also perform a consistency test (consisting of also sending to P_{LIP} a random linear combination of the k queries of V_{LPCP} and then verifying the obvious condition on the received answers).

More precisely, we construct a two-message LIP $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ from an LPCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ as follows:

Construction 3.1. *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be a k-query LPCP over $\mathbb{F}$ with query length m. Define a two-message LIP $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ as follows.*

- *The LIP verifier V_{LIP} runs the LPCP verifier V_{LPCP} to obtain k queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k \in \mathbb{F}^m$, draws $\alpha_1, \ldots, \alpha_k$ in $\mathbb{F}$ uniformly at random, and sends to the LIP prover P_{LIP} the $(k+1)m$ field elements obtained by concatenating the k queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ together with the additional query $\boldsymbol{q}_{k+1} := \sum_{i=1}^{k} \alpha_i \boldsymbol{q}_i$.*
- *The LIP prover P_{LIP} runs the LPCP prover P_{LPCP} to obtain a linear function $\boldsymbol{\pi} \colon \mathbb{F}^m \to \mathbb{F}$, parses the $(k+1)m$ received field elements as $k+1$ queries of m field elements each, applies $\boldsymbol{\pi}$ to each of these queries to obtain $k+1$ corresponding field elements $a_1, \ldots, a_{k+1}$, and sends these answers to the LIP verifier V_{LIP}.*
- *The LIP verifier V_{LIP} checks that $a_{k+1} = \sum_{i=1}^{k} \alpha_i a_i$ (if this is not the case, it rejects) and decides whether to accept or reject by feeding the LPCP verifier V_{LPCP} with the answers $a_1, \ldots, a_k$.*

Lemma 3.2 (from LPCP to LIP)**.** *Suppose $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is a k-query LPCP for a relation $\mathcal{R}$ over $\mathbb{F}$ with query length m and knowledge error ε. Then, $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ from Construction 3.1 is a two-message LIP for $\mathcal{R}$ over $\mathbb{F}$ with verifier message in $\mathbb{F}^{(k+1)m}$, prover message in $\mathbb{F}^{k+1}$, and knowledge error $\varepsilon + \frac{1}{|\mathbb{F}|}$. Moreover,*

- *if $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has strong knowledge, then $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ also does, and*
- *if $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has an algebraic verifier of degree (d_Q, d_D), then $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ has one with degree $(d_Q, \max\{2, d_D\})$.*

Proof. Syntactic properties and completeness are easy to verify. Furthermore, since the construction of V_{LIP} from V_{LPCP} only involves an additional quadratic test, the degree of V_{LIP} is $(d_Q, \max\{2, d_D\})$. We are left to argue knowledge (and strong knowledge).

Let $\Pi\colon \mathbb{F}^{(k+1)m} \to \mathbb{F}^{k+1}$ be an affine strategy of a potentially malicious LIP prover P^*_{LIP}. We specify Π by $(k+1)^2$ linear functions $\boldsymbol{\pi}_{i,j}\colon \mathbb{F}^m \to \mathbb{F}$ for $i,j \in \{1,\ldots,k+1\}$ and a constant vector $\gamma = (\gamma_1,\ldots,\gamma_{k+1}) \in \mathbb{F}^{k+1}$ such that the i-th answer of P^*_{LIP} is given by $a_i := \sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle + \gamma_i$. It suffices to show that, for any choice of queries $\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k$, exactly one of the following conditions holds:

- $a_i = \langle \boldsymbol{\pi}_{k+1,k+1}, \boldsymbol{q}_i \rangle$ for all $i \in [k]$, or
- with probability greater than $1 - \frac{1}{|\mathbb{F}|}$ over $\alpha_1,\ldots,\alpha_k$, P^*_{LIP} does not pass the consistency check.

Indeed, the above tells us that if Π makes V_{LIP} accept with probability greater than $\varepsilon + \frac{1}{|\mathbb{F}|}$, then $\boldsymbol{\pi}_{k+1,k+1}$ makes V_{PCP} accept with probability greater than ε. Knowledge (and strong knowledge) thus follow as claimed.

To show the above, fix a tuple of queries, and assume that, for some $i^* \in [k]$, $a_{i^*} \neq \langle \boldsymbol{\pi}_{k+1,k+1}, \boldsymbol{q}_{i^*} \rangle$. For the consistency check to pass, it should hold that:

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle + \gamma_i \right) = \sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle + \gamma_{k+1} \ .$$

Equivalently,

$$\sum_{j=1}^{k+1} \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle + \sum_{i=1}^{k} \alpha_i \gamma_i = \sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle + \gamma_{k+1} \ .$$

Breaking the first summation using the equality $\boldsymbol{q}_{k+1} = \sum_{j=1}^{k} \alpha_j \boldsymbol{q}_j$, we get:

$$\sum_{j=1}^{k} \sum_{i=1}^{k} \alpha_i \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle + \sum_{i=1}^{k} \alpha_i \left\langle \boldsymbol{\pi}_{i,k+1}, \left(\sum_{j=1}^{k} \alpha_j \boldsymbol{q}_j \right) \right\rangle + \sum_{i=1}^{k} \alpha_i \gamma_i = \sum_{j=1}^{k} \langle \boldsymbol{\pi}_{k+1,j}, \boldsymbol{q}_j \rangle + \left\langle \boldsymbol{\pi}_{k+1,k+1}, \sum_{j=1}^{k} \alpha_j \boldsymbol{q}_j \right\rangle + \gamma_{k+1} \ .$$

Rearranging, we see that the consistency check reduces to verifying the following equation:

$$\sum_{i,j=1}^{k} \alpha_i \alpha_j \langle \boldsymbol{\pi}_{i,k+1}, \boldsymbol{q}_j \rangle + \sum_{i=1}^{k} \alpha_i \left(\sum_{j=1}^{k} \langle \boldsymbol{\pi}_{i,j}, \boldsymbol{q}_j \rangle - \langle \boldsymbol{\pi}_{k+1,k+1}, \boldsymbol{q}_i \rangle + \gamma_i \right) - \left(\sum_{i=1}^{k} \langle \boldsymbol{\pi}_{k+1,i}, \boldsymbol{q}_i \rangle + \gamma_{k+1} \right) = 0 \ .$$

Because $\sum_{j=1}^{k+1} \langle \boldsymbol{\pi}_{i^*,j}, \boldsymbol{q}_j \rangle + \gamma_{i^*} = a_{i^*} \neq \langle \boldsymbol{\pi}_{k+1,k+1}, \boldsymbol{q}_{i^*} \rangle$, the coefficient of α_{i^*} in the above polynomial is non-zero. Hence, by the Schwartz–Zippel Lemma (see Lemma 2.1), the identity holds with probability at most $\frac{1}{|\mathbb{F}|}$. □

In light of the two LPCP constructions described in Appendix A, we deduce the following two theorems.

Theorem 3.3. *Let $\mathbb{F}$ be a finite field and $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ a boolean circuit of size s. There is an input-oblivious two-message LIP for $\mathcal{R}_C$ with knowledge error $O(1/|\mathbb{F}|)$, verifier message in $\mathbb{F}^{O(s^2)}$, prover message in $\mathbb{F}^4$, and degree $(2,2)$. Furthermore:*

- *the LIP prover P_{LIP} is an arithmetic circuit of size $O(s^2)$;*
- *the LIP query algorithm Q_{LIP} is an arithmetic circuit of size $O(s^2)$;*
- *the LIP decision algorithm D_{LIP} is an arithmetic circuit of size $O(n)$.*

Theorem 3.4. *Let $\mathbb{F}$ be a finite field and $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ a boolean circuit of size s. There is an input-oblivious two-message LIP for $\mathcal{R}_C$ with knowledge error $O(s/|\mathbb{F}|)$, verifier message in $\mathbb{F}^{O(s)}$, prover message in $\mathbb{F}^4$, and degree $(O(s), 2)$. Furthermore:*

- *the LIP prover P_{LIP} is an arithmetic circuit of size $\widetilde{O}(s)$;*
- *the LIP query algorithm Q_{LIP} is an arithmetic circuit of size $O(s)$;*
- *the LIP decision algorithm D_{LIP} is an arithmetic circuit of size $O(n)$.*

3.1.1 Zero-knowledge.

The LIPs we obtain by via above transformation can all be made honest-verifier zero-knowledge (HVZK) by starting with an HVZK LPCP. For this purpose, we show in Appendix B a *general* transformation from *any* LPCP with $d_D = O(1)$ to a corresponding HVZK LPCP, with only small overhead in parameters.

3.2 LIPs From (Traditional) PCPs

We present a second general construction of LIPs. Instead of LPCPs, this time we rely on a traditional k-query PCP in which the proof π is a binary string of length $m = \mathrm{poly}(|x|)$. While any PCP can be viewed as an LPCP (by mapping each query location $q \in [m]$ to the unit vector $\boldsymbol{e}_q$ equal to 1 at the q-th position and 0 everywhere else), applying the transformation from Section 3.1 yields an LIP in which the prover's message consists of $k+1$ field elements. Here we rely on the *sparseness* of the queries of an LPCP that is obtained from a PCP in order to reduce the number of field elements returned by the prover to 1. (In particular, the transformation presented in this section does not apply to either of the LPCPs we construct in Appendix A, because they do not have sparse queries.) The construction relies on the easiness of solving instances of subset sum in which each integer is bigger than the sum of the previous integers (see [MH78]).

Fact 3.5. *There is a quasilinear-time algorithm for the following problem:*

- ***input:*** *Non-negative integers $w_1, \ldots, w_k, a$ such that each w_i is bigger than the sum of the previous w_j.*
- ***output:*** *A binary vector $(a_1, \ldots, a_k) \in \{0,1\}^k$ such that $a = \sum_{i=1}^k a_i w_i$ (if one exists).*

(All integers are given in binary representation.)

The following construction uses a parameter ℓ that will affect the soundness error. We assume that the field $\mathbb{F}$ is of a prime order $\mathfrak{p}$ where $\mathfrak{p} > 2^k \ell$ and identify its elements with the integers $0, \ldots, \mathfrak{p}-1$.

Construction 3.6. *Let $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ be a k-query PCP with proof length m. Define an LIP $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ over $\mathbb{F}$ as follows.*

- *The LIP verifier V_{LIP} runs the PCP verifier V_{PCP} to obtain k distinct query locations $q_1, \ldots, q_k \in [m]$, picks a sequence of k random field elements*

$$w_1 \leftarrow [0, \ell-1]\,,\ w_2 \leftarrow [\ell, 2\ell-1]\,,\ w_3 \leftarrow [3\ell, 4\ell-1]\,,\ \ldots\,,\ w_k \leftarrow [(2^{k-1}-1)\ell, 2^{k-1}\ell - 1]\ ,$$

  and sends to the LIP prover P_{LIP} the vector $\boldsymbol{q} = \sum_{i=1}^k w_i \boldsymbol{e}_{q_i}$, where $\boldsymbol{e}_j$ is the j-th unit vector in $\mathbb{F}^m$.

- *The LIP prover P_{LIP} responds by applying to $\boldsymbol{q}$ the linear function $\boldsymbol{\pi}\colon \mathbb{F}^m \to \mathbb{F}$ whose coefficients are specified by the m bits of the PCP generated by the PCP prover P_{PCP}. Let a denote the field element returned by P_{LIP}.*

- *The LIP verifier V_{LIP} applies the subset sum algorithm of Fact 3.5 to find $(a_1, \ldots, a_k) \in \{0,1\}^k$ such that $a = \sum_{i=1}^k a_i w_i$ (if none exists it rejects) and decides whether to accept by feeding the PCP verifier V_{PCP} with $a_1, \ldots, a_k$.*

Lemma 3.7 (from PCP to LIP)**.** *Suppose $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ is a k-query PCP for a relation $\mathcal{R}$ with proof length m and knowledge error ε, and $\mathbb{F}$ is a field of prime order $\mathfrak{p}$ with $\mathfrak{p} > 2^k \ell$. Then $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ from Construction 3.6 is a two-message LIP for $\mathcal{R}$ over $\mathbb{F}$ with verifier message in $\mathbb{F}^m$, prover message in $\mathbb{F}$, and knowledge error $\varepsilon + \frac{2^k}{\ell}$.*

Proof. Because the prover message is in $\mathbb{F}$ (i.e., the prover returns a single field element) the prover strategy is an affine function $\Pi^* \colon \mathbb{F}^m \to \mathbb{F}$ (i.e., as in an LPCP, see Remark (2.8)). Let $\boldsymbol{\pi}^* \colon \mathbb{F}^m \to \mathbb{F}$ be a linear function and $\gamma^* \in \mathbb{F}$ be a constant such that $\Pi^*(\boldsymbol{q}) = \langle \boldsymbol{\pi}^*, \boldsymbol{q} \rangle + \gamma^*$ for all $\boldsymbol{q} \in \mathbb{F}^m$.

We say that query positions $q_1, \ldots, q_k \in [m]$ are *invalid* with respect to Π^* if $\gamma^* \neq 0$ or there is $i \in \{1, \ldots, k\}$ such that $\Pi^*(\boldsymbol{e}_{q_i}) \notin \{0, 1\}$. It suffices to show that, for any strategy Π^* as above, conditioned on any choice of invalid query positions $q_1, \ldots, q_k$ by V_{LIP}, the probability of V_{LIP} accepting is bounded by $2^k/\ell$. Indeed, for queries for which Π^* is valid, it holds that $\Pi^*(q_i) = \langle \boldsymbol{\pi}^*, q_i \rangle \in \{0, 1\}$ corresponding a traditional PCP oracle $\boldsymbol{\pi}^*$, so that the knowledge guarantees of $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ would kick in.

The above follows from the sparseness of the answers a that correspond to valid strategies and the high entropy of the answer resulting from any invalid strategy. Concretely, fix any candidate solution $(a_1, \ldots, a_k) \in \{0, 1\}^k$ and pick $w_1, \ldots, w_k$ as in Construction 3.6. Since each w_i is picked uniformly from an interval of size ℓ,

$$\begin{aligned}
\Pr_{w_1,\ldots,w_k}\left[\Pi^* \cdot \left(\sum_{i=1}^{k} w_i \boldsymbol{e}_{q_i} \right)^{\top} = \sum_{i=1}^{k} a_i w_i \right] &= \Pr_{w_1,\ldots,w_k}\left[\left\langle \boldsymbol{\pi}^*, \sum_{i=1}^{k} w_i \boldsymbol{e}_{q_i} \right\rangle + \gamma^* = \sum_{i=1}^{k} a_i w_i \right] \\
&= \Pr_{w_1,\ldots,w_k}\left[\sum_{i=1}^{k} (\pi^*_{q_i} - a_i) w_i + \gamma^* = 0 \right] \\
&\leq \frac{1}{\ell} \ .
\end{aligned}$$

Indeed, noting that $\sum_{i=1}^{k} (\pi^*_{q_i} - a_i) w_i + \gamma^*$ is a degree-1 polynomial in the variables $w_1, \ldots, w_k$,

- if there is $i \in \{1, \ldots, k\}$ such that $\Pi^*(\boldsymbol{e}_{q_i}) \notin \{0, 1\}$ then the coefficient of w_i is non-zero (since $a_i \in \{0, 1\}$) and thus, by the Schwartz–Zippel Lemma (see Lemma 2.1), the probability that the polynomial vanishes is at most $1/\ell$; and
- if instead for all $i \in \{1, \ldots, k\}$ it holds that $\Pi^*(\boldsymbol{e}_{q_i}) \in \{0, 1\}$ then it must be that $\gamma^* \neq 0$; if there is $i \in \{1, \ldots, k\}$ such that $\pi^*_{q_i} \neq a_i$ then the same argument as in the previous bullet holds; otherwise, $\gamma^* = 0$ with probability 0 since we know that $\gamma^* \neq 0$.

By a union bound, the probability that there *exists* solution $(a_1, \ldots, a_k) \in \{0, 1\}^k$ such that $\Pi(\sum_{i=1}^{k} w_i \boldsymbol{e}_{q_i}) = \sum_{i=1}^{k} a_i w_i$ is at most $2^k/\ell$. Hence, the subset sum algorithm will fail to find a solution and V_{LIP} will reject except with at most $2^k/\ell$ probability. □

By setting $\ell := 2^k/\varepsilon$, we obtain the following corollary:

Corollary 3.8. *Suppose $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ is a k-query PCP for a relation $\mathcal{R}$ with proof length m and knowledge error ε, and $\mathbb{F}$ is a field of prime order $\mathfrak{p}$ with $\mathfrak{p} > 2^{2k}/\varepsilon$. Then $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ from Construction 3.6 is a two-message LIP for $\mathcal{R}$ over $\mathbb{F}$ with verifier message in $\mathbb{F}^m$, prover message in $\mathbb{F}$, and knowledge error 2ε.*

There are many PCPs in the literature (e.g., [BFLS91, FGL+96, AS98, ALM+98b, PS94, RS97, HS00, BSSVW03, BSGH+04, BSGH+05, GS06, Din07, BSS08, MR08, Mei12]), optimizing various parameters.

Focusing on asysmptotic time complexity, perhaps the most relevant PCPs for our purposes here are those of Ben-Sasson et al. [BSCGT13b]. They constructed PCPs where, to prove and verify that a random-access machine M accepts (x, w) within t steps for some w with $|w| \leq t$, the prover runs in time $(|M| + |x| + t) \cdot \mathrm{polylog}(t)$ and the verifier runs in time $(|M| + |x|) \cdot \mathrm{polylog}(t)$ (while asking $\mathrm{polylog}(t)$ queries, for constant soundness). Invoking Corollary 3.8 with these PCPs, one can deduce the following theorem.

Theorem 3.9. *Let $\mathbb{F}$ be a finite field and $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ a boolean circuit of size s. There is an input-oblivious two-message LIP for $\mathcal{R}_C$ with knowledge error $2^{-\lambda}$, verifier message in $\mathbb{F}^{\tilde{O}(s)}$, prover message in $\mathbb{F}$, and $|\mathbb{F}| > 2^{\lambda \cdot \mathrm{polylog}(s)}$. Furthermore:*

- *the LIP prover P_{LIP} runs in time $\tilde{O}(s)$;*
- *the LIP query algorithm Q_{LIP} runs in time $\tilde{O}(s) + \lambda \cdot n \cdot \mathrm{polylog}(s)$;*
- *the LIP decision algorithm D_{LIP} runs in time $\lambda \cdot n \cdot \mathrm{polylog}(s)$.*

(All the above running times are up to $\mathrm{polylog}(|\mathbb{F}|)$ factors.)

Focusing on communication complexity instead, we can invoke Corollary 3.8 with the query-efficient PCPs of Håstad and Khot [HK05], which have $\lambda + o(\lambda)$ queries for soundness $2^{-\lambda}$. (Because their PCPs have a query algorithm that depends on the input, we only obtain an LIP where the verifier's message depends on the input; it is plausible that [HK05] can be modified to be input oblivious, but we did not check this.)

Theorem 3.10. *Let $\mathbb{F}$ be a finite field and $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ a boolean circuit of size s. There is a two-message LIP for $\mathcal{R}_C$ with knowledge error $2^{-\lambda}$, verifier message in $\mathbb{F}^{\mathrm{poly}(s)}$, prover message in $\mathbb{F}$, and $|\mathbb{F}| > 2^{\lambda \cdot (3 + o(1))}$. Furthermore:*

- *the LIP prover P_{LIP} runs in time $\mathrm{poly}(s)$;*
- *the LIP query algorithm Q_{LIP} runs in time $\mathrm{poly}(s) + \lambda \cdot n \cdot \mathrm{polylog}(s)$;*
- *the LIP decision algorithm D_{LIP} runs in time $\lambda \cdot n \cdot \mathrm{polylog}(s)$.*

(All the above running times are up to $\mathrm{polylog}(|\mathbb{F}|)$ factors.)

The verifiers of the PCPs of Ben-Sasson et al. [BSCGT13b] (used to derive Theorem 3.9) and of Håstad and Khot [HK05] (used to derive Theorem 3.10) do not have low degree, and thus the LIPs they induce via our transformation are not algebraic. In Appendix C, we give evidence that this is inherent, because we prove that *no* PCP (for a hard enough language) can have low-degree verifiers (or even satisfy a weaker, but still useful, property that we call "strong knowledge").

3.2.1 Zero-knowledge.

In Section 3.1.1 we discussed a generic transformation from any LPCP with $d_D = O(1)$ to a corresponding HVZK LPCP. A (traditional) PCP does not typically induce an LPCP with $d_D = O(1)$. Thus, if we want to obtain an HVZK LIP through Construction 3.6, we need a different approach.

We observe that if we plug into Construction 3.6 a PCP that is HVZK (see Definition 2.4), then the corresponding LIP is also HVZK.

Lemma 3.11. *In Lemma 3.7, if $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ is a HVZK PCP then $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ is a HVZK LIP.*

4 Definitions of SNARKs and Preprocessing SNARKs

We now turn to the cryptographic part of this work. We start by recalling the notions of a SNARK and a preprocessing SNARK.

In fact, before we do so, we first recall the *universal relation* [BG08], which provides us with a canonical form to represent verification-of-computation problems. Because such problems typically arise in the form of *algorithms* (e.g., "is there w that makes program $\boldsymbol{P}$ accept (x, w)?"), we adopt the universal relation relative to random-access machines [CR72, AV77].

Definition 4.1. *The* **universal relation** *is the set $\mathcal{R}_{\mathcal{U}}$ of instance-witness pairs $(y, w) = \big((M, x, t), w\big)$, where $|y|, |w| \leq t$ and M is a random-access machine, such that M accepts (x, w) after at most t steps.*[4] *We denote by $\mathcal{L}_{\mathcal{U}}$ the* **universal language** *corresponding to $\mathcal{R}_{\mathcal{U}}$.*

We now proceed to define SNARGs and preprocessing SNARGs. A *succinct non-interactive argument* (SNARG) is a triple of algorithms (G, P, V) that works as follows. The (probabilistic) generator G, on input the security parameter λ (presented in unary) and a time bound T, outputs a *reference string* σ and a corresponding *verification state* τ. The honest prover $P(\sigma, y, w)$ produces a proof π for the instance $y = (M, x, t)$ given a witness w, provided that $t \leq T$; then $V(\tau, y, \pi)$ verifies the validity of π.

The SNARG is *adaptive* if the prover may choose the statement *after* seeing σ, otherwise, it is *non-adaptive*; the SNARG is *fully-succinct* if G runs "fast", otherwise, it is of the *preprocessing* kind.

Definition 4.2. *A triple of algorithms (G, P, V) is a* **SNARG** *for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if the following conditions are satisfied:*

1. *Completeness*

   For every large enough security parameter $\lambda \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every instance-witness pair $(y, w) = \big((M, x, t), w\big) \in \mathcal{R}$ with $t \leq T$,

$$\Pr\left[V(\tau, y, \pi) = 1 \;\middle|\; \begin{array}{c}(\sigma, \tau) \leftarrow G(1^\lambda, T) \\ \pi \leftarrow P(\sigma, y, w)\end{array}\right] = 1 \ .$$

2. *Soundness (depending on which notion is considered)*

   - ***non-adaptive:*** *For every polynomial-size prover P^*, every large enough security parameter $\lambda \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every instance $y = (M, x, t)$ for which $\nexists\, w$ s.t. $(y, w) \in \mathcal{R}$,*

$$\Pr\left[V(\tau, y, \pi) = 1 \;\middle|\; \begin{array}{c}(\sigma, \tau) \leftarrow G(1^\lambda, T) \\ \pi \leftarrow P^*(\sigma, y)\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

   - ***adaptive:*** *For every polynomial-size prover P^*, every large enough security parameter $\lambda \in \mathbb{N}$, and every time bound $T \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c}V(\tau, y, \pi) = 1 \\ \nexists\, w \text{ s.t. } (y, w) \in \mathcal{R}\end{array} \;\middle|\; \begin{array}{c}(\sigma, \tau) \leftarrow G(1^\lambda, T) \\ (y, \pi) \leftarrow P^*(\sigma)\end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

3. *Efficiency*

   There exists a universal polynomial p (independent of $\mathcal{R}$) such that, for every large enough security parameter $\lambda \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every instance $y = (M, x, t)$ with $t \leq T$,

[4] While the witness w for an instance $y = (M, x, t)$ has size at most t, there is *no a-priori* polynomial bounding t in terms of $|x|$. Also, the restriction that $|y|, |w| \leq t$ simplifies notation but comes with essentially no loss of generality: see [BSCGT13a] for a discussion of how to deal with "large inputs" (i.e., x or w much larger than t, in the model where M has random access to them).

- *the generator G runs in time* $\begin{cases} p(\lambda + \log T) & \textit{for a fully-succinct SNARG} \\ p(\lambda + T) & \textit{for a preprocessing SNARG} \end{cases}$;
- *the prover P runs in time* $\begin{cases} p(\lambda + |M| + |x| + t + \log T) & \textit{for a fully-succinct SNARG} \\ p(\lambda + |M| + |x| + T) & \textit{for a preprocessing SNARG} \end{cases}$;
- *the verifier V runs in time $p(\lambda + |M| + |x| + \log T)$;*
- *an honestly generated proof has size $p(\lambda + \log T)$.*

Proof of knowledge. A SNARG *of knowledge* (SNARK) is a SNARG where soundness is strengthened as follows:

Definition 4.3. *A triple of algorithms (G, P, V) is a* **SNARK** *for the relation $\mathcal{R}$ if it is a SNARG for $\mathcal{R}$ where adaptive soundness is replaced by the following stronger requirement:*

- *Adaptive proof of knowledge*[5]

  For every polynomial-size prover P^* there exists a polynomial-size extractor E such that for every large enough security parameter $\lambda \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and every time bound $T \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c|r} V(\tau, y, \pi) = 1 & (\sigma, \tau) \leftarrow G(1^\lambda, T) \\ (y, w) \notin \mathcal{R} & (y, \pi) \leftarrow P^*(z, \sigma) \\ & w \leftarrow E(z, \sigma) \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

One may want to distinguish between the case where the verification state τ is allowed to be public or needs to remain private: a *publicly-verifiable SNARK* (pvSNARK) is one where security holds even if τ is public; in contrast, a *designated-verifier SNARK* (dvSNARK) is one where τ needs to remain secret.

Zero-knowledge. A *zero-knowledge* SNARK (or "succinct NIZK of knowledge") is a SNARK satisfying a zero-knowledge property. Namely, zero knowledge ensures that the honest prover can generate valid proofs for true theorems without leaking any information about the theorem beyond the fact that the theorem is true (in particular, without leaking any information about the witness that he used to generate the proof for the theorem). Of course, when considering zero-knowledge SNARKs, the reference string σ must be a common reference string that is trusted, not only by the verifier, but also by the prover.

Definition 4.4. *A triple of algorithms (G, P, V) is a* **(perfect) zero-knowledge SNARK** *for the relation $\mathcal{R}$ if it is a SNARK for $\mathcal{R}$ and, moreover, satisfies the following property:*

- *Zero Knowledge*

  There exists a stateful interactive polynomial-size simulator S such that for all stateful interactive polynomial-size distinguishers $\mathcal{D}$, large enough security parameter $\lambda \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and every time bound $T \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c|r} t \leq T & (\sigma, \tau) \leftarrow G(1^\lambda, T) \\ (y, w) \in \mathcal{R}_{\mathcal{U}} & (y, w) \leftarrow \mathcal{D}(z, \sigma) \\ \mathcal{D}(\pi) = 1 & \pi \leftarrow P(\sigma, y, w) \end{array}\right] = \Pr\left[\begin{array}{c|r} t \leq T & (\sigma, \tau, \mathsf{trap}) \leftarrow S(1^\lambda, T) \\ (y, w) \in \mathcal{R}_{\mathcal{U}} & (y, w) \leftarrow \mathcal{D}(z, \sigma) \\ \mathcal{D}(\pi) = 1 & \pi \leftarrow S(z, \sigma, y, \mathsf{trap}) \end{array}\right] \ .$$

[5] One can also formulate weaker proof of knowledge notions. In this work we focus on the above strong notion.

As usual, Definition 4.4 can be relaxed to consider the case in which the distributions are only *statistically* or *computationally* close.

As observed in [BCCT12], dvSNARKs (resp., pvSNARKs) can be combined with zero-knowledge (not-necessarily-succinct) non-interactive arguments (NIZKs) of knowledge to obtain zero-knowledge dvSNARKs (resp., pvSNARKs). This observation immediately extends to preprocessing SNARKs, thereby providing a generic method to construct zero-knowledge preprocessing SNARKs from preprocessing SNARKs.

In this work, we also consider more "direct", and potentially more efficient, ways to construct zero-knowledge preprocessing SNARKs by relying on various constructions of HVZK LIPs (and without relying on generic NIZKs). See Section 6.3.

(We note that when applying the transformations of [BCCT13], e.g. to remove preprocessing, zero knowledge is preserved.[6])

Multiple theorems. A desirable property (especially so when preprocessing is expensive) is the ability to generate σ once and for all and then reuse it in polynomially-many proofs (potentially by different provers). Doing so requires security also against provers that *have access to a proof-verification oracle*. While for pvSNARKs this *multi-theorem proof of knowledge* property is automatically guaranteed, this is not the case for dvSNARKs. In Appendix C, we formally define and discuss this stronger notion of security, and show that some of our dvSNARKs achieve this security notion because of the "strong knowledge" of the underlying LIPs. (We note that, like zero-knowledge above, the multi-theorem proof-of-knowledge property is preserved by the transformations of [BCCT13].)

OUR FOCUS. In this work we study *preprocessing* SNARKs, where (as stated in Definition 4.2) the generator G may run in time polynomial in the security parameter λ and time bound T.

4.1 Preprocessing SNARKs for Boolean Circuit Satisfaction Problems

In Section 4, we have defined SNARKs for the universal relation. In this work, at times it will be more convenient to discuss preprocessing SNARKs for boolean circuit satisfaction problems rather than for the universal relation.[7] We thus briefly sketch the relevant definitions, and also explain how preprocessing SNARKs for boolean circuit satisfaction problems suffice for obtaining preprocessing SNARKs, with similar efficiency, for the universal relation. (Indeed, because we are often interested in the correctness of algorithms, and not boolean circuits, it is important that this transformation be efficient!)

We begin by introducing boolean circuit satisfaction problems:

Definition 4.5. *The* **boolean circuit satisfaction problem** *of a boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ is the relation $\mathcal{R}_C = \{(x,w) \in \{0,1\}^n \times \{0,1\}^h : C(x,w) = 1\}$; its language is denoted $\mathcal{L}_C$. For a family of boolean circuits $\mathcal{C} = \left\{C_\ell\colon \{0,1\}^{n(\ell)} \times \{0,1\}^{h(\ell)} \to \{0,1\}\right\}_{\ell\in\mathbb{N}}$, we denote the corresponding infinite relation and language by $\mathcal{R}_\mathcal{C} = \bigcup_{\ell\in\mathbb{N}} \mathcal{R}_{C_\ell}$ and $\mathcal{L}_\mathcal{C} = \bigcup_{\ell\in\mathbb{N}} \mathcal{L}_{C_\ell}$.*

A preprocessing SNARK for a uniform family of boolean circuits $\mathcal{C}$ is defined analogously to a preprocessing SNARK for the universal relation, with only small syntactic modifications. The (probabilistic) generator G, on input the security parameter λ and an index ℓ for the circuit $C_\ell\colon \{0,1\}^{n(\ell)} \times \{0,1\}^{h(\ell)} \to \{0,1\}$,

[6] The definition of proof of knowledge of a SNARK (Definition 4.3) says that the extractor is given the same inputs that are given to the prover, but nothing else; in particular, the extractor does not receive any trapdoor. The transformation of Bitansky et al. [BCCT13] crucially relies on this fact. Thus, if one is interested in constructing a zero-knowledge SNARK via the result of [BCCT12] and then invoke the result of [BCCT13], one must be mindful to rely on (not-necessarily-succinct) non-interactive arguments of knowledge where the extractor does not require a trapdoor. (E.g., the extractor in [AF07] does not require a trapdoor.)

[7] This is because our information-theoretic constructions (see Section 3), which we use towards the construction of preprocessing SNARKs, are more conveniently stated for boolean circuit satisfaction problems.

outputs a *reference string* σ and a corresponding *verification state* τ. (Both τ and σ can be thought to include λ and ℓ.) Given w, the honest prover $P(\sigma, x, w)$ produces a proof π attesting that $x \in \mathcal{L}_{C_\ell}$; then, $V(\tau, x, \pi)$ verifies the validity of π. As for efficiency, we require that there exists a universal polynomial p (independent of the family $\mathcal{C}$) such that for every large enough security parameter $\lambda \in \mathbb{N}$, index $\ell \in \mathbb{N}$, and input $x \in \{0,1\}^{n(\ell)}$:

- the generator G runs in time $p(\lambda + |C_\ell|)$
- the prover P runs in time $p(\lambda + |C_\ell|)$;
- the verifier V runs in time $p(\lambda + |x| + \log |C_\ell|)$;
- an honestly generated proof has size $p(\lambda + \log |C_\ell|)$.

We can also consider the case where $\mathcal{C}$ is a non-uniform family, in which case G and P will get as additional input the circuit C_ℓ.

Let us now explain how to obtain a preprocessing SNARK for $\mathcal{R}_\mathcal{U}$ from preprocessing SNARKs for uniform families of boolean circuits. To do so, we need to introduce the notion of a universal RAM simulator:

Definition 4.6. *Let $n \in \mathbb{N}$. We say that a boolean circuit family $\mathcal{C}_n = \{C_T \colon \{0,1\}^n \times \{0,1\}^{h(T)} \to \{0,1\}\}_T$ is a* **universal RAM simulator** *for n-bit instances if, for every $y = (M, x, t)$ with $|y| = n$, $C_T(y, \cdot)$ is satisfiable if and only if $y \in \mathcal{L}_\mathcal{U}$ and $t \leq T$. A* **witness map** *of $\mathcal{C}_n$, denoted w, is a function such that, for every $y = (M, x, t)$ with $|y| = n$ and $t \leq T$, if $(y, w) \in \mathcal{R}_\mathcal{U}$ then $C_T(y, \mathsf{w}(T, y, w)) = 1$. An* **inverse witness map** *of $\mathcal{C}_n$, denoted w^{-1}, is a function such that, for every $y = (M, x, t)$ with $|y| = n$ and $t \leq T$, if $C_T(y, w') = 1$ then $(y, \mathsf{w}^{-1}(T, y, w')) \in \mathcal{R}_\mathcal{U}$.*

For every $n \in \mathbb{N}$, given a preprocessing SNARK (G, P, V) for a universal RAM simulator $\mathcal{C}_n$ (for n-bit instances) with (efficient) witness map w and inverse witness map wit^{-1}, we can construct a preprocessing SNARK (G'_n, P'_n, V'_n) for those pairs (y, w) in the universal relation with $|y| = n$ as follows:

- $G'_n(1^\lambda, T) := G(1^\lambda, T)$;
- $P'_n(\sigma, y, w) := P(\sigma, y, \mathsf{w}(T, y, w))$;
- $V'_n(\tau, y, \pi) := V(\tau, y, \pi)$.

Note that wit^{-1} does not take part in the construction of (G'_n, P'_n, V'_n), but its existence ensures that proof of knowledge is preserved. (Concretely, a knowledge extractor E'_n for a prover convincing V'_n would first run a knowledge extractor for the same prover convincing V and then run wit^{-1} to obtain a suitable witness.)

The efficiency of $\mathcal{C}$, w, and w^{-1} has direct implications to the efficiency of (G'_n, P'_n, V'_n). Concretely:

- Let $f(T) := |C_T|$. The growth rate of $f(T)$ affects the efficiency of G'_n and P'_n, because the efficiency of G and P depends on $|C_T|$. So, for instance, if G and P run in time $|C_T|^2 \cdot \mathrm{poly}(\lambda)$ and $f(T) = \Omega(T^2)$, then G'_n and P'_n run in time $\Omega(T^4) \cdot \mathrm{poly}(\lambda)$.
- The running time of w affects the running time of P'_n. Indeed, P'_n must first transform the witness w for y into a witness w' for $C_T(y, \cdot)$, and only then he can invoke P. So, for instance, even if $f(T) = \tilde{O}(T)$ but w runs in time $\Omega(T^3)$, then P'_n will run in time $\Omega(T^3)$.

- The running time of w^{-1} sometimes affects the running time of G'_n, P'_n, and V'_n. Indeed, if the proof of knowledge property of (G'_n, P'_n, V'_n) is used in a security reduction (e.g., verifying the correctness of cryptographic computations) then the slower w^{-1} is the more expensive is the security reduction, and thus the larger the security parameter has to be chosen for (G, P, V). A larger security parameter affects the efficiency of all three algorithms.

We thus wish the growth rate of $f(T)$ to be as small as possible, and that w and w^{-1} be as fast as possible. The reduction from RAM computations to circuits of Ben-Sasson et al. [BSCGT13a] implies that there is a universal RAM simulator where $f(T) = \tilde{O}(T)$ and both w and w^{-1} run in sequential time $\tilde{O}(T)$ (or in parallel time $O((\log T)^2)$).

Next, we explain how to remove the restriction on the instance size, by using collision-resistant hashing. (Indeed, (G'_n, P'_n, V'_n) only handles instances y with $|y| = n$.) Let $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ be a collision-resistant hash-function family such that functions in $\mathcal{H}_\lambda$ map $\{0,1\}^*$ to $\{0,1\}^\lambda$. For any $h \in \mathcal{H}_\lambda$ and instance y, define y_h to be the instance $(U_h, h(x), \mathrm{poly}(\lambda) + O(t))$, where U_h is a universal random-access machine that, on input $(\tilde{x}, \tilde{w})$, parses $\tilde{w}$ as $((M, x, t), w)$, verifies that $\tilde{x} = h(M, x, t)$, and then runs $M(x, w)$ for at most t steps. Because we can assume a uniform super-polynomial upper bound on t, say $t \leq \lambda^{\log\lambda}$, there is a constant $c > 0$ for which we can assume that $|y_h| = \lambda^c$. Then, we can construct a preprocessing SNARK (G'', P'', V'') for the universal relation as follows:

- $G''(1^\lambda, T)$ outputs $(\tilde{\sigma}, \tilde{\tau}) := \big((\sigma, h), (\tau, h)\big)$ where $(\sigma, \tau) \leftarrow G'_{\lambda^c}(1^\lambda, T)$ and $h \leftarrow \mathcal{H}_\lambda$;
- $P''(\tilde{\sigma}, y, w) := P'_{\lambda^c}(\sigma, y_h, (y, w))$;
- $V''(\tilde{\tau}, y, \pi) := V'_{\lambda^c}(\tau, y_h, \pi)$.

The proof of knowledge property and the collision-resistant property ensure that the construction above is a preprocessing SNARK for the universal relation.

In sum, asymptotically, *we incur in essentially no overhead if we focus on constructing preprocessing SNARKs for uniform families of boolean circuits*.

5 Linear-Only Encryption and Encodings

We introduce and discuss the two cryptographic tools used in this paper. First, in Section 5.1, we present *linear-only encryption* and then, in Section 5.2, *linear-only one-way encodings*. In Section 5.3, we discuss candidate instantiations for both. Later, in Section 6, we describe how to use these tools in our transformations from LIPs to SNARKs (or SNARGs).

5.1 Linear-Only Encryption

At high-level, a *linear-only encryption scheme* is a semantically-secure encryption scheme that supports linear homomorphic operations, but does not allow any other form of homomorphism.

We first introduce the syntax and correctness properties of linear-only encryption; then its (standard) semantic-security property; and finally its *linear-only property*. In fact, we consider *two* formalizations of the linear-only property (a stronger one and a weaker one).

Syntax and correctness. A *linear-only encryption scheme* is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{ImVer})$ with the following syntax and correctness properties:

- Given a security parameter λ (presented in unary), Gen generates a secret key sk and a public key pk. The public key pk also includes a description of a field $\mathbb{F}$ representing the plaintext space.
- Enc and Dec are (randomized) encryption and (deterministic) decryption algorithms working in the usual way.
- $\mathsf{Add}(\mathsf{pk}, c_1, \ldots, c_m, \alpha_1, \ldots, \alpha_m)$ is a homomorphic evaluation algorithm for linear combinations. Namely, given a public key pk, ciphertexts $\{c_i \in \mathsf{Enc}_{\mathsf{pk}}(a_i)\}_{i\in[m]}$, and field elements $\{\alpha_i\}_{i\in[m]}$, Add computes an evaluated ciphertext $\hat{c} \in \mathsf{Enc}_{\mathsf{pk}}(\sum_{i\in[m]} \alpha_i a_i)$.
- $\mathsf{ImVer}_{\mathsf{sk}}(c')$ tests, using the secret key sk, whether a given candidate ciphertext c' is in the image of $\mathsf{Enc}_{\mathsf{pk}}$.

Remark 5.1. Because in most of this paper we restrict attention to LPCPs and LIPs over fields, we present linear-only encryption schemes for the case where plaintexts belong to a field. The definition naturally extends to the case where plaintexts belong to a ring. Typically, we are interested in the ring $\mathbb{Z}_N$ for either the case where N is a prime $\mathfrak{p}$ (in which case the ring $\mathbb{Z}_{\mathfrak{p}}$ is isomorphic to the field $\mathbb{F}_{\mathfrak{p}}$) or where N is the product of two primes. (See corresponding Remark (2.6) in the LIP definition.)

Remark 5.2. A symmetric-key variant of linear-only encryption can be easily defined. While ultimately a private-key linear homomorphic encryption implies a public-key one [Rot11], using a private-key encryption could, in principle, have efficiency benefits.

Remark 5.3. The linear homomorphism property can be relaxed to allow for cases where the evaluated ciphertext $\hat{c}$ is not necessarily in the image of $\mathsf{Enc}_{\mathsf{pk}}$, but only decrypts to the correct plaintext; in particular, it may not be possible to perform further homomorphic operations on such a cipher.

Semantic security. Semantic security of linear-only encryption is defined as usual. Namely, for any polynomial-size adversary A and large enough security parameter $\lambda \in \mathbb{N}$:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (a_0, a_1) \leftarrow A(\mathsf{pk}) \\ b \leftarrow \{0,1\} \\ b' \leftarrow A\big(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(a_b)\big) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ .$$

Linear-only homomorphism. The linear-only (homomorphism) property essentially says that, given a public key pk and ciphertexts $(c_1, \ldots, c_m)$, it is infeasible to compute a new ciphertext c' in the image of $\mathsf{Enc}_{\mathsf{pk}}$, except by evaluating an affine combination of the ciphertexts $(c_1, \ldots, c_m)$. (Affinity accounts for adversaries encrypting plaintexts from scratch and then adding them to linear combinations of the c_i.) Formally, the property is captured by guaranteeing that, whenever the adversary produces a valid ciphertext, it is possible to efficiently extract a corresponding affine function "explaining" the ciphertext.

Definition 5.4. *An encryption scheme has the* **linear-only (homomorphism)** *property if for any polynomial-size adversary A there is a polynomial-size extractor E such that, for any sufficiently large $\lambda \in \mathbb{N}$, any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any plaintext generator $\mathcal{M}$,*

$$\Pr\left[\begin{array}{c} \exists\, i \in [k] \textit{ s.t.} \\ \mathsf{ImVer}_{\mathsf{sk}}(c'_i) = 1 \\ \textit{and} \\ \mathsf{Dec}_{\mathsf{sk}}(c'_i) \neq a'_i \end{array} \;\middle|\; \begin{array}{r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (a_1, \ldots, a_m) \leftarrow \mathcal{M}(\mathsf{pk}) \\ (c_1, \ldots, c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m)) \\ (c'_1, \ldots, c'_k) \leftarrow A(\mathsf{pk}, c_1, \ldots, c_m; z) \\ (\Pi, \boldsymbol{b}) \leftarrow E(\mathsf{pk}, c_1, \ldots, c_m; z) \\ (a'_1, \ldots, a'_k)^\top \leftarrow \Pi \cdot (a_1, \ldots, a_m)^\top + \boldsymbol{b} \end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where $\Pi \in \mathbb{F}^{k \times m}$ *and* $\boldsymbol{b} \in \mathbb{F}^k$.

Remark 5.5 (on the auxiliary input z)**.** In Definition 5.4, the polynomial-size extractor is required to succeed for any (adversarial) auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$. This requirement seems rather strong considering the fact that z could potentially encode arbitrary circuits. For example, z could encode a circuit that, given as input public key pk, outputs $\mathsf{Enc}_{\mathsf{pk}}(x)$ where $x = f_s(\mathsf{pk})$ and f_s is some hardwired pseudorandom function. In this case, the extractor would be required to (efficiently) reverse engineer the circuit, which seems to be a rather strong requirement (or even an impossible one, under certain obfuscation assumptions).

While for presentational purposes Definition 5.4 is simple and convenient, it can be relaxed to only consider specific "benign" auxiliary-input distributions. Indeed, in our application, it will be sufficient to only consider a truly-random auxiliary input z. (Requiring less than that seems to be not expressive enough, because we would at least like to allow the adversary to toss random coins.)

An analogous remark holds for both Definitions 5.8 and 5.17.

Remark 5.6 (oblivious ciphertext sampling)**.** Definition 5.4 has a similar flavor to plaintext awareness. In fact, an encryption scheme cannot satisfy the definition if it allows for "oblivious sampling" of ciphertexts. (For instance, both standard Elgamal and Paillier encryption do.) Thus, the set of strings c that are valid (i.e., for which $\mathsf{ImVer}_{\mathsf{sk}}(c) = 1$) must be "sparse". Later on, we define a weaker notion of linear-only encryption that does not have this restriction.

Remark 5.7. In order for Definition 5.4 to be non-trivial, the extractor E has to be efficient (for otherwise it could run the adversary A, obtain A's outputs, decrypt them, and then output a zero linear function and hard-code the correct values in the constant term). As for the equivalent formulation in Remark (5.11), for similar reasons the simulator S has to be efficient; additionally, requiring statistical indistinguishability instead of computational indistinguishability does not strengthen the assumption.

Linear targeted malleability. We also consider a weaker variant of the linear-only property, which we call *linear targeted malleability*. (Indeed, the definition follows the lines of the notion of targeted malleability proposed by Boneh et al. [BSW12], when restricted to the class of linear, or affine, functions.)

Definition 5.8. *An encryption scheme has the* **linear targeted malleability** *property if for any polynomial-size adversary A and plaintext generator $\mathcal{M}$ there is a polynomial-size simulator S such that, for any sufficiently large $\lambda \in \mathbb{N}$, and any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, the following two distributions are computationally indistinguishable:*

$$\left\{ \begin{array}{l|r} \mathsf{pk}, & (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ a_1,\ldots,a_m, & (s,a_1,\ldots,a_m) \leftarrow \mathcal{M}(\mathsf{pk}) \\ s, & (c_1,\ldots,c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(a_1),\ldots,\mathsf{Enc}_{\mathsf{pk}}(a_m)) \\ \mathsf{Dec}_{\mathsf{sk}}(c'_1),\ldots,\mathsf{Dec}_{\mathsf{sk}}(c'_k) & (c'_1,\ldots,c'_k) \leftarrow A(\mathsf{pk},c_1,\ldots,c_m;z) \\ & \textit{where} \\ & \mathsf{ImVer}_{\mathsf{sk}}(c'_1)=1,\ldots,\mathsf{ImVer}_{\mathsf{sk}}(c'_k)=1 \end{array} \right\},$$

and

$$\left\{ \begin{array}{l|r} \mathsf{pk}, & (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ a_1,\ldots,a_m, & (s,a_1,\ldots,a_m) \leftarrow \mathcal{M}(\mathsf{pk}) \\ s, & (\Pi,\boldsymbol{b}) \leftarrow S(\mathsf{pk};z) \\ a'_1,\ldots,a'_k & (a'_1,\ldots,a'_k)^\top \leftarrow \Pi \cdot (a_1,\ldots,a_m)^\top + \boldsymbol{b} \end{array} \right\}$$

where $\Pi \in \mathbb{F}^{k \times m}$, $\boldsymbol{b} \in \mathbb{F}^k$, and s is some arbitrary string (possibly correlated with the plaintexts).

Remark 5.9. Definition 5.8 can be further relaxed to allow the simulator to be *inefficient*. Doing so does not let us prove knowledge but still enables us to prove soundness (i.e., obtain a SNARG instead of a SNARK). See Remark (6.4) in Section 6.1.

As mentioned above, Definition 5.8 is weaker than Definition 5.4, as shown by the following lemma.

Lemma 5.10. *If a semantically-secure encryption scheme has the linear-only property (Definition 5.4), then it has the linear targeted malleability property (Definition 5.8).*

Proof sketch. Let E be the (polynomial-size) extractor of a given polynomial-size adversary A. We use E to construct a (polynomial-size) simulator S for A. The simulator S simply runs E on fake ciphertexts:

$S(\mathsf{pk}; z) \equiv$

1. $(c_1, \ldots, c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(0), \ldots, \mathsf{Enc}_{\mathsf{pk}}(0))$;
2. $(y, c'_1, \ldots, c'_k) \leftarrow A(\mathsf{pk}, c_1, \ldots, c_m; z)$;
3. $(\Pi, \boldsymbol{b}) \leftarrow E(\mathsf{pk}, c_1, \ldots, c_m; z)$;
4. output $(y, \Pi, \boldsymbol{b})$.

By invoking semantic security and the extraction guarantee of E, we can show that S works. The proof follows by a standard hybrid argument. First we consider an experiment where S gives A and E an encryption of $\mathbf{a} \leftarrow \mathcal{M}(\mathsf{pk})$, rather than an encryption of zeros, and argue computational indistinguishability by semantic security. Then we can show that the output in this hybrid experiment is statistically close to that in the real experiment by invoking the extraction guarantee. □

A converse to Lemma 5.10 appears unlikely, because Definition 5.8 seems to allow for encryption schemes where ciphertexts can be obliviously sampled while Definition 5.4 does not.

Remark 5.11 (alternative formulation)**.** To better compare Definition 5.8 with Definition 5.4, we now give an equivalent formulation of Definition 5.4. For any polynomial-size adversary A there is a polynomial-size simulator S such that, for any sufficiently large $\lambda \in \mathbb{N}$, any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any plaintext generator $\mathcal{M}$, the following two distributions are computationally indistinguishable:

$$\left\{ \begin{array}{l|r} \mathsf{pk}, & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ a_1, \ldots, a_m, & (a_1, \ldots, a_m) \leftarrow \mathcal{M}(\mathsf{pk}) \\ c_1, \ldots, c_m, & (c_1, \ldots, c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m)) \\ \mathsf{out}_{\mathsf{sk}}(c'_1), \ldots, \mathsf{out}_{\mathsf{sk}}(c'_k), & (c'_1, \ldots, c'_k) \leftarrow A(\mathsf{pk}, c_1, \ldots, c_m; z) \\ z & \end{array} \right\}$$

where $\mathsf{out}_{\mathsf{sk}}(c') := \begin{cases} \mathsf{Dec}_{\mathsf{sk}}(c') & \text{if } \mathsf{ImVer}_{\mathsf{sk}}(c') = 1 \\ \perp & \text{if } \mathsf{ImVer}_{\mathsf{sk}}(c') = 0 \end{cases}$, and

$$\left\{ \begin{array}{l|r} \mathsf{pk}, & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ a_1, \ldots, a_m, & (a_1, \ldots, a_m) \leftarrow \mathcal{M}(\mathsf{pk}) \\ c_1, \ldots, c_m, & (c_1, \ldots, c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m)) \\ a'_1, \ldots, a'_k, & (\Pi, \boldsymbol{b}) \leftarrow S(\mathsf{pk}, c_1, \ldots, c_m; z) \\ z & (a'_1, \ldots, a'_k)^\top = \Pi \cdot (a_1, \ldots, a_m)^\top + \boldsymbol{b} \end{array} \right\}$$

where $\Pi \in \mathbb{F}^{k \times m}$ and $\boldsymbol{b} \in \mathbb{F}^k$, with the convention that if the i-th row of Π is left empty then $a'_i := \perp$.

5.2 Linear-Only One-Way Encoding

Unlike linear-only encryption schemes, *linear-only encoding schemes* allow to publicly test for certain properties of the underlying plaintexts without decryption (which is now allowed to be inefficient). In particular, linear-only encoding schemes cannot satisfy semantic security. Instead, we require that they only satisfy a certain one-wayness property.

We now define the syntax and correctness properties of linear-only encoding schemes, their one-wayness property, and their *linear-only property*.

Syntax and correctness. A *linear-only encoding scheme* is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{SEnc}, \mathsf{Test}, \mathsf{Add}, \mathsf{ImVer})$ with the following syntax and correctness properties:

- Given a security parameter λ (presented in unary), Gen generates a public key pk. The public key pk also includes a description of a field $\mathbb{F}$ representing the plaintext space.
- Encoding can be performed in two *modes*: $\mathsf{Enc}_{\mathsf{pk}}$ is an encoding algorithm that works in *linear-only mode*, and $\mathsf{SEnc}_{\mathsf{pk}}$ is a deterministic encoding algorithm that works in *standard mode*.
- As in linear-only encryption, $\mathsf{Add}(\mathsf{pk}, c_1, \ldots, c_m, \alpha_1, \ldots, \alpha_m)$ is a homomorphic evaluation algorithm for linear combinations. Namely, given a public key pk, encodings $\{c_i \in \mathsf{Enc}_{\mathsf{pk}}(a_i)\}_{i\in[m]}$, and field elements $\{\alpha_i\}_{i\in[m]}$, Add computes an evaluated encoding $\hat{c} \in \mathsf{Enc}_{\mathsf{pk}}(\sum_{i\in[m]} \alpha_i a_i)$. Also, Add works in the same way for any vector of standard-mode encodings $\{c_i \in \mathsf{SEnc}_{\mathsf{pk}}(a_i)\}_{i\in[m]}$.
- $\mathsf{ImVer}_{\mathsf{pk}}(c')$ tests whether a given candidate encoding c' is in the image of $\mathsf{Enc}_{\mathsf{pk}}$ (i.e., in the image of the encoding in linear-only mode).
- $\mathsf{Test}\big(\mathsf{pk}, \boldsymbol{t}, \mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m), \mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_1), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_{\tilde{m}})\big)$ is a public test for zeros of $\boldsymbol{t}$. Namely, given a public key pk, a test polynomial $\boldsymbol{t}\colon \mathbb{F}^m \to \mathbb{F}^\eta$, encodings $\mathsf{Enc}_{\mathsf{pk}}(a_i)$, and standard-mode encodings $\mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_i)$, Test tests whether $\boldsymbol{t}(a_1, \ldots, a_m, \tilde{a}_1, \ldots, \tilde{a}_{\tilde{m}}) = 0^\eta$.

Remark 5.12 (degrees supported by Test)**.** In this work, we restrict our attention to the case in which Test only takes as input test polynomials $\boldsymbol{t}$ of at most quadratic degree. This restriction comes from the fact that, at present, the only candidates for linear-only one-way encoding schemes that we know of are based on bilinear maps, which only let us support testing of quadratic degrees. (See Section 5.3.) This restriction propagates to the transformation from algebraic LIPs discussed in Section 6.2, where we must require that the degree of the LIP query algorithm is at most quadratic. Nonetheless our transformation holds more generally (for query algorithms of $\mathrm{poly}(\lambda)$ degree), when given linear-only one-way encoding schemes that support tests of the appropriate degree.

Δ-power one-wayness. In our main application of transforming algebraic LIPs into public-verifiable preprocessing SNARKs (see Section 6.2), linear-only encoding schemes are used to (linearly) manipulate polynomial evaluations over $\mathbb{F}$. The notion of one-wayness that we require is that, given polynomially-many encodings of low-degree polynomials evaluated at a random point s, it is hard to find s.

Definition 5.13. *A linear-only encoding scheme satisfies* **Δ-power one-wayness** *if for every polynomial-size A and all large enough security parameter $\lambda \in \mathbb{N}$,*

$$\Pr\left[\, s^* = s \;\middle|\; \begin{array}{r} \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda) \\ s \leftarrow \mathbb{F} \\ (c_1, \ldots, c_\Delta) \leftarrow \big(\mathsf{Enc}_{\mathsf{pk}}(s), \ldots, \mathsf{Enc}_{\mathsf{pk}}(s^\Delta)\big) \\ (\tilde{c}_1, \ldots, \tilde{c}_\Delta) \leftarrow \big(\mathsf{SEnc}_{\mathsf{pk}}(s), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(s^\Delta)\big) \\ s^* \leftarrow A\big(\mathsf{pk}, c_1, \ldots, c_\Delta, \tilde{c}_1, \cdots, \tilde{c}_\Delta\big) \end{array} \right] \leq \mathsf{negl}(\lambda) \ .$$

Our constructions of preprocessing SNARKs from LIPs also involve manipulations of *multivariate* polynomials. Thus, we are in fact interested in requiring a more general property of multivariate Δ-power one-wayness.

Definition 5.14. *A linear-only encoding scheme satisfies* **multivariate Δ-power one-wayness** *if, for every polynomial-size A, large enough security parameter $\lambda \in \mathbb{N}$, and μ-variate polynomials $(p_1, \ldots, p_\ell)$ of total degree at most Δ:*

$$\Pr\left[\begin{array}{c|r} & \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda) \\ p^* \not\equiv 0 & \boldsymbol{s} \leftarrow \mathbb{F}^\mu \\ \text{and} & (c_1, \ldots, c_\ell) \leftarrow \big(\mathsf{Enc}_{\mathsf{pk}}(p_1(\boldsymbol{s})), \ldots, \mathsf{Enc}_{\mathsf{pk}}(p_\ell(\boldsymbol{s}))\big) \\ p^*(\boldsymbol{s}) = 0 & (\tilde{c}_1, \ldots, \tilde{c}_\ell) \leftarrow \big(\mathsf{SEnc}_{\mathsf{pk}}(p_1(\boldsymbol{s})), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(p_\ell(\boldsymbol{s}))\big) \\ & p^* \leftarrow A\big(\mathsf{pk}, c_1, \ldots, c_\ell, \tilde{c}_1, \cdots, \tilde{c}_\ell\big) \end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where Δ, ℓ, μ are all $\mathrm{poly}(\lambda)$, and p^* is a μ-variate polynomial.

For the case of univariate polynomials (i.e., $\mu = 1$), it is immediate to see that Definition 5.14 is equivalent to Definition 5.13; this follows directly from the fact that univariate polynomials over finite fields can be efficiently factored into their roots [Ber70, BO81, CZ81, VZGP01]. We show that the two definitions are equivalent also for any $\mu = \mathrm{poly}(\lambda)$, provided that the encoding scheme is rerandomizable; indeed, in the instantiation discussed in this paper the encoding is deterministic and, in particular rerandomizable.

Proposition 5.15. *If $\mathsf{Enc}, \mathsf{SEnc}$ are rerandomizable (in particular, if deterministic), then (univariate) Δ-power one-wayness implies (multivariate) Δ-power one-wayness for any $\mu = \mathrm{poly}(\lambda)$.*

Proof. Assume that A violates the μ-variate Δ-power one-wayness with probability ε for a vector of polynomials $(p_1, \ldots, p_\ell)$. We use A to construct a new adversary A' that breaks (univariate) Δ-power one-wayness with probability at least $\varepsilon/\mu\Delta$.

Given input $\big(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(s), \ldots, \mathsf{Enc}_{\mathsf{pk}}(s^\Delta), \mathsf{SEnc}_{\mathsf{pk}}(s), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(s^\Delta)\big)$, A' first samples $i \in [\mu]$ and $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_\mu \in \mathbb{F}$ at random. Then, thinking of s as s_i and $\boldsymbol{s}$ as $(s_1, \ldots, s_\mu)$, A' uses the linear homomorphism and rerandomization to sample $\big(\mathsf{Enc}_{\mathsf{pk}}(p_1(\boldsymbol{s})), \ldots, \mathsf{Enc}_{\mathsf{pk}}(p_\ell(\boldsymbol{s})), \mathsf{SEnc}_{\mathsf{pk}}(p_1(\boldsymbol{s})), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(p_\ell(\boldsymbol{s}))\big)$ and feeds these to A, who in turn outputs a polynomial p^*. Next, A' does the following:

1. Let $p_1^* = p^*$, and set $j := 1$.
2. While $j < i$ and $p_j^*(x_j, x_{j+1}, \ldots, x_\mu) \not\equiv 0$:
   (a) Decompose p_j^* according to the x_j-monomials: $p_j^*(x_j, \ldots, x_\mu) = \sum_{k=0}^{\Delta} x_j^k p_{j+1,k}^*(x_{j+1}, \ldots, x_\mu)$.
   (b) Set p_{j+1}^* to be the non-zero polynomial $p_{j+1,k}^*$ with minimal k.
   (c) Set $j := j + 1$.
3. After computing p_i^*, restrict the $\mu - i$ last variables to $\boldsymbol{s}$, i.e. compute the x_i-univariate polynomial $p_i^*(x_i, s_{i+1}, \ldots, s_\mu)$, and factor it to find at most Δ roots; finally, output one of the roots at random as a guess for $s = s_i$.

To analyze the success probability of A', we rely on the following claim:

Claim 5.16. *If $p^* \not\equiv 0$ and $p^*(s_1, \ldots, s_\mu) = 0$, then there exists $i \in [\mu]$ such that:*

$$p_i^*(x, s_{i+1}, \ldots, s_\mu) \not\equiv 0$$
$$p_i^*(s_i, s_{i+1}, \ldots, s_\mu) = 0 \ .$$

Proof of Claim 5.16. The proof is by induction on i. The base case is when $i = 1$, for which it holds that:

$$p_1^*(s_1, \ldots, s_\mu) = 0$$
$$p_1^*(x_1, \ldots, x_\mu) \not\equiv 0 \ .$$

For any i with $2 < i < \mu$, suppose that:

$$p_i^*(s_i, \ldots, s_\mu) = 0$$
$$p_i^*(x_i, \ldots, x_\mu) \not\equiv 0$$
$$p_i^*(x_i, s_{i+1}, \ldots, s_\mu) \equiv 0 \ ;$$

then, by the construction of p_{i+1}^* from p_i^*,

$$p_{i+1}^*(s_{i+1}, \ldots, s_\mu) = 0$$
$$p_{i+1}^*(x_{i+1}, \ldots, x_\mu) \not\equiv 0 \ .$$

If this inductive process reaches p_μ^*, then it holds that:

$$p_\mu^*(s_\mu) = 0$$
$$p_\mu^*(x_\mu) \not\equiv 0 \ ,$$

which already satisfies the claim. □

Note that A' guesses the i guaranteed by Claim 5.16 with probability $1/\mu$, and hence, with the same probability, finds a non-trivial polynomial that vanishes at the challenge point $s = s_i$; in such a case, A' thus guesses s correctly, from among at most Δ roots, with probability at least $1/\Delta$. The overall probability of success of A' is at lest $\varepsilon/\mu\Delta$, and this concludes the proof of Proposition 5.15. □

Linear-only homomorphism. The linear-only property of linear-only one-way encoding schemes is defined analogously to the case of linear-only encryption. Essentially, it says that, given the public key pk, encodings in *linear-only mode* $(\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m))$, and possibly additional encodings in *standard mode* $(\mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_1), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_{\tilde{m}}))$, it is infeasible to compute a new encoding c' in the image of $\mathsf{Enc}_{\mathsf{pk}}$, except by evaluating an affine combination of the encodings $(\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m))$; in particular, "standard mode" encodings in the image of $\mathsf{SEnc}_{\mathsf{pk}}$ cannot be "moved into" the image of $\mathsf{Enc}_{\mathsf{pk}}$. Formally, the property is captured by guaranteeing that, whenever the prover produces a valid new encoding, it is possible to efficiently extract the corresponding affine combination.

Definition 5.17. *A linear encoding scheme has the* **linear-only (homomorphism)** *property if for any polynomial-size adversary A there is a polynomial-size extractor E such that for any sufficiently large $\lambda \in \mathbb{N}$, any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any plaintext generator $\mathcal{M}$:*

$$\Pr\left[\begin{array}{c} (a'_1, \ldots, a'_k)^\top = \Pi \cdot (a_1, \ldots, a_m)^\top + \boldsymbol{b} \\ \text{and} \\ \exists\, i \in [k] : \mathsf{ImVer}_{\mathsf{sk}}(c'_i) = 1 \text{ but } c'_i \notin \mathsf{Enc}_{\mathsf{pk}}(a'_i) \end{array} \;\middle|\; \begin{array}{r} \mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda) \\ (a_1, \ldots, a_m, \tilde{a}_1, \ldots, \tilde{a}_{\tilde{m}}) \leftarrow \mathcal{M}(\mathsf{pk}) \\ (c_1, \ldots, c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m)) \\ (\tilde{c}_1, \ldots, \tilde{c}_{\tilde{m}}) \leftarrow (\mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_1), \ldots, \mathsf{SEnc}_{\mathsf{pk}}(\tilde{a}_{\tilde{m}})) \\ (c'_1, \ldots, c'_k) \leftarrow A(\mathsf{pk}, c_1, \ldots, c_m, \tilde{c}_1, \ldots, \tilde{c}_{\tilde{m}}; z) \\ (\Pi, \boldsymbol{b}) \leftarrow E(\mathsf{pk}, c_1, \ldots, c_m, \tilde{c}_1, \ldots, \tilde{c}_{\tilde{m}}; z) \end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where $\Pi \in \mathbb{F}^{k \times m}$ and $\boldsymbol{b} \in \mathbb{F}^k$.

5.3 Instantiations

We discuss candidates for our notions of linear-only encryption and one-way encoding schemes.

Linear-only property (and linear targeted malleability) from Paillier encryption. Paillier encryption [Pai99] has plaintext group $(\mathbb{Z}_N, +)$, where N is a product of two λ-bit primes $\mathfrak{p}$ and $\mathfrak{q}$. (See Remark (2.6) and Remark (5.1).) We consider two variants of Paillier encryption:

- **A "single-ciphertext" variant with linear targeted malleability.** We assume that standard Paillier encryption satisfies Definition 5.8. Note that this variant *cannot* satisfy Definition 5.4 (which is stronger, as shown in Lemma 5.10), because it is easy to "obliviously sample" valid Paillier ciphertexts without "knowing" the corresponding plaintext. (See Remark (5.6).)
- **A "two-ciphertext" variant with linear-only property.** In order to (heuristically) prevent oblivious sampling, we can "sparsify" the ciphertext space of Paillier encryption by following the template of knowledge-of-exponent assumptions. Concretely, an encryption of a plaintext a consists of $\mathsf{Enc}_{\mathsf{pk}}(a)$ and $\mathsf{Enc}_{\mathsf{pk}}(\alpha \cdot a)$ for a secret random $\alpha \in \mathbb{Z}_N$; additionally, an image verification algorithm $\mathsf{ImVer}_{\mathsf{sk}}$ checks this linear relation. (This candidate is also considered in [GGPR13].) We then assume that this variant of Paillier encryption satisfies Definition 5.4.

Because Paillier encryption is based on the decisional composite residuosity assumption, it suffers from factoring attacks, and thus security for succinct arguments based on the above instantiations can only be assumed to hold against subexponential-time provers (specifically, running in time $2^{o(\lambda^{1/3})}$).

Linear-only property (and linear targeted malleability) from Elgamal encryption. Elgamal encryption [EG85] has plaintext group $(\mathbb{Z}_{\mathfrak{p}}, \times)$ for a large prime $\mathfrak{p}$, and is conjectured to resist subexponential-time attacks when implemented over elliptic curves [PQ12].

We are interested in *additive*, rather than multiplicative, homomorphism for plaintexts that belong to the field $\mathbb{F}_{\mathfrak{p}}$ (whose elements coincide with those of $\mathbb{Z}_{\mathfrak{p}}$). Thus, we would like the plaintext group to be $(\mathbb{Z}_{\mathfrak{p}}, +)$ instead. The two groups $(\mathbb{Z}_{\mathfrak{p}}, \times)$ and $(\mathbb{Z}_{\mathfrak{p}}, +)$ are in fact isomorphic via the function that maps a plaintext a to a new plaintext $g^a (\mathrm{mod}\, \mathfrak{p})$, where g is a primitive element of $\mathbb{F}_{\mathfrak{p}}$. Unfortunately, inverting this mapping is computationally inefficient: in order to recover the plaintext a from $g^a (\mathrm{mod}\, \mathfrak{p})$, the decryption algorithm has to compute a discrete logarithm base g; doing so is inefficient when a can be any value. Thus, a naive use Elgamal encryption in our context presents a problem.

Nonetheless, as explained in Section 1.3.3, we can still use Elgamal encryption in our context by ensuring that the distribution of the honest LIP prover's answers, conditioned on any choice of verifier randomness, has a polynomial-size support. Doing so comes with two caveats: it results in succinct arguments with only $1/\mathrm{poly}(\lambda)$ security and (possibly) a slow online verification time (but, of course, with proofs that are still succinct).

Here too, to prove security, we can consider single-ciphertext and two-ciphertext variants of Elgamal encryption that we assume satisfy Definition 5.8 and Definition 5.4 respectively.

Linear-only property (and linear targeted malleability) from Benaloh encryption. Benaloh encryption [Ben94] generalizes the quadratic-residuosity-based encryption scheme of Goldwasser and Micali [GM84] to higher residue classes; it can support any plaintext group $(\mathbb{Z}_{\mathfrak{p}}, +)$ where $\mathfrak{p}$ is *polynomial* in the security parameter. Unlike Elgamal encryption (implemented over elliptic curves) and similarly to Paillier encryption, Benaloh encryption is susceptible to subexponential-time attacks.

As before, we can consider single-ciphertext and two-ciphertext variants of Benaloh encryption that we assume satisfy Definition 5.8 and Definition 5.4 respectively. Because we are restricted to $\mathfrak{p} = \mathrm{poly}(\lambda)$, succinct arguments based on Benaloh encryption can only yield $1/\mathrm{poly}(\lambda)$ security.

Linear-only one-way encodings from KEA in bilinear groups. In order to obtain publicly-verifiable preprocessing SNARKs (see Section 6.2), we seek linear-only encodings that have $\mathrm{poly}(\lambda)$-power one-wayness and allow to publicly test for zeroes of $\mathrm{poly}(\lambda)$-degree polynomials. For this, we use the same candidate encoding over bilinear groups, and essentially the same assumptions, as in [Gro10, Lip12, GGPR13]; because of the use of bilinear maps, we will in fact only be able to publicly test for zeros of quadratic polynomials.

For the sake of completeness, and since the construction does not correspond directly to a known encryption scheme as in the examples above, we give the basic construction and relevant assumptions.

The encoding is defined over a bilinear group ensemble $\{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$ where each $(\mathbb{G},\mathbb{G}_\mathsf{T})\in\mathcal{G}_\lambda$ is a pair of groups of prime order $\mathfrak{p}\in(2^{\lambda-1},2^\lambda)$ with an efficiently-computatable pairing $e\colon \mathbb{G}\times\mathbb{G}\to\mathbb{G}_\mathsf{T}$. A public key pk includes the description of the groups and $g, g^\alpha\in\mathbb{G}$, where $g\in\mathbb{G}^*$ is a generator and $\alpha\leftarrow\mathbb{F}_\mathfrak{p}$ is random. The encoding is *deterministic*: the linear-only mode encoding is $\mathsf{Enc}_\mathsf{pk}(a) := (g^a, g^{\alpha a})$, and the standard-mode encoding is $\mathsf{SEnc}_\mathsf{pk}(a) := g^a$. Public image verification is as follows: $\mathsf{ImVer}_\mathsf{pk}(f, f')$ outputs 1 if and only if $e(f, g^\alpha) = e(g, f')$. Public testing of quadratic polynomials can also be done using the pairing: for $\{(g_i, g_i^\alpha)\}_{i\in[m]} = \{\mathsf{Enc}_\mathsf{pk}(a_i)\}_{i\in[m]}$ and $\{\tilde{g}_i\}_{i\in[\tilde{m}]} = \{\mathsf{SEnc}_\mathsf{pk}(\tilde{a}_i)\}_{i\in[\tilde{m}]}$, Test uses $g_1,\ldots,g_m$ and $\tilde{g}_1,\ldots,\tilde{g}_{\tilde{m}}$ and the pairing to test zeros for a quadratic polynomial $\boldsymbol{t}$. The required cryptographic assumptions are:

Assumption 5.18 (KEA and poly-power DL in bilinear groups)**.** *There exists an efficiently-samplable group ensemble $\{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $(\mathbb{G},\mathbb{G}_\mathsf{T})\in\mathcal{G}_\lambda$ are groups of prime order $\mathfrak{p}\in(2^{\lambda-1},2^\lambda)$ having a corresponding efficiently-computable pairing $e\colon\mathbb{G}\times\mathbb{G}\to\mathbb{G}_\mathsf{T}$, such that the following properties hold.*

1. ***Knowledge of exponent:*** *For any polynomial-size adversary A there exists a polynomial-size extractor E such that for all large enough $\lambda\in\mathbb{N}$, any auxiliary input $z\in\{0,1\}^{\mathrm{poly}(\lambda)}$,[8] and any group element sampler S,*

$$\Pr\left[\begin{array}{c|r} f'=f^\alpha & (\mathbb{G},\mathbb{G}_\mathsf{T})\leftarrow\mathcal{G}_\lambda \\ \prod_{i\in[t]} g_i^{\pi_i}\neq f & (g_1,\ldots,g_t)\leftarrow S(\mathbb{G},\mathbb{G}_\mathsf{T}) \\ & \alpha\leftarrow\mathbb{F}_\mathfrak{p} \\ & (f,f')\leftarrow A(\mathbb{G},\mathbb{G}_\mathsf{T},g_1,g_1^\alpha,\ldots,g_t,g_t^\alpha;z) \\ & (\pi_1,\ldots,\pi_t)\leftarrow E(\mathbb{G},\mathbb{G}_\mathsf{T},g_1,g_1^\alpha,\ldots,g_t,g_t^\alpha;z) \end{array}\right] \le \mathsf{negl}(\lambda)\ .$$

2. ***Hardness of poly-power discrete logarithms:*** *For any polynomial-size adversary A, polynomial $t=\mathrm{poly}(\lambda)$, all large enough $\lambda\in\mathbb{N}$, and generator sampler S:*

$$\Pr\left[\begin{array}{c|r} s'=s & (\mathbb{G},\mathbb{G}_\mathsf{T})\leftarrow\mathcal{G}_\lambda \\ & s\leftarrow\mathbb{F}_\mathfrak{p} \\ & g\leftarrow S(\mathbb{G}) \text{ where } \langle g\rangle=\mathbb{G} \\ & s'\leftarrow A(\mathbb{G},\mathbb{G}_\mathsf{T},g,g^s,g^{s^2}\ldots,g^{s^t}) \end{array}\right] \le \mathsf{negl}(\lambda)\ .$$

Remark 5.19 (lattice-based candidates)**.** In principle, we may also consider as candidates lattice-based encryption schemes (e.g., [Reg05]). However, our confidence that these schemes satisfy linear-only properties may be more limited, as they can be tweaked to yield *fully*-homomorphic encryption schemes [BV11].

[8] It is possible to restrict the definition to specific auxiliary input distributions. See Remark (5.5).

6 Preprocessing SNARKs from LIPs

We describe how to combine LIPs and linear-only encryption and encodings in order to construct preprocessing SNARKs. Before describing our transformations, we make two technical remarks.

SNARKs and LIPs for boolean circuit families. Since the LIPs that we have presented so far are for boolean circuit satisfaction problems, it will be convenient to construct here preprocessing SNARKs for boolean circuit satisfaction problems. As explained in Section 4.1, such preprocessing SNARKs imply preprocessing SNARKs for the universal relation $\mathcal{R}_{\mathcal{U}}$ with similar efficiency.

Also, for the sake of simplicity, the LIP constructions that we have presented so far are for satisfiability of specific boolean circuits. However, all of these constructions directly extend to work for any *family* of boolean circuits $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$, in which case all the LIP algorithms (e.g., $V_{\mathsf{LIP}} = (Q_{\mathsf{LIP}}, D_{\mathsf{LIP}})$ and P_{LIP}) will also get as input 1^ℓ (as foreshadowed in Remark (2.3)). If the circuit family $\mathcal{C}$ is uniform, all the LIP algorithms are uniform as well. If the circuit family $\mathcal{C}$ is non-uniform, then Q_{LIP} and P_{LIP} will also get a circuit C_ℓ as auxiliary input (in addition to 1^ℓ).

Field size depending on λ. Definition 2.5 (and Definition 2.2) are with respect to a fixed field $\mathbb{F}$. However, since the knowledge error of a LIP (or LPCP) typically decreases with the field size, it is often convenient to let the size of $\mathbb{F}$ scale with a security parameter λ. In fact, when combining a LIP with some of our linear-only encryption and encoding candidates, letting $\mathbb{F}$ scale with λ is essential, because security will only hold for a large enough plaintext space. (For example, this is the case for the Elgamal-like linear-only encoding described in Section 5.3). All of the LIP constructions described in Sections 3.1 and 3.2 do work for arbitrarily large fields, and we can assume that $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ simply get as additional input the description of the field; abusing notation, we will just denote this description by $\mathbb{F}_\lambda$.

6.1 Designated-Verifier Preprocessing SNARKs from Arbitrary LIPs

We describe how to combine a LIP and linear-only encryption to obtain a designated-verifier preprocessing SNARK.

Construction 6.1. *Let $\{\mathbb{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a field ensemble (with efficient description and operations). Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of circuits. Let $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ be an input-oblivious two-message LIP for the relation $\mathcal{R}_{\mathcal{C}}$, where for the field $\mathbb{F}_\lambda$, the verifier message is in $\mathbb{F}_\lambda^m$, the prover message is in $\mathbb{F}_\lambda^k$, and the knowledge error is $\varepsilon(\lambda)$. Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{lmVer})$ be a linear-only encryption scheme whose plaintext field, for security parameter λ, is $\mathbb{F}_\lambda$. We define a preprocessing SNARK (G, P, V) for $\mathcal{R}_{\mathcal{C}}$ as follows.*

- *$G(1^\lambda, 1^\ell)$ invokes the LIP query algorithm $Q_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell)$ to generate an LIP message $\mathbf{q} \in \mathbb{F}_\lambda^m$ along with a secret state $\mathbf{u} \in \mathbb{F}^{m'}$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$, computes $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(q_i)$ for $i \in [m]$, defines $\sigma := (\mathsf{pk}, c_1, \ldots, c_m)$ and $\tau := (\mathsf{sk}, \mathbf{u})$, and outputs (σ, τ). (Assume that both (σ, τ) contain ℓ and the description of the field $\mathbb{F}_\lambda$).*

- *$P(\sigma, x, w)$ invokes the LIP prover algorithm $P_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell, x, w)$ to get a matrix $\Pi \in \mathbb{F}_\lambda^{k \times m}$ representing its message function, invokes the homomorphic Add to generate k ciphertexts $c'_1, \ldots, c'_k$ encrypting $\Pi \cdot \mathbf{q}$, defines $\pi := (c'_1, \ldots, c'_k)$, and outputs π.*

- *$V(\tau, x, \pi)$, verifies, for $i \in [k]$, that $\mathsf{lmVer}_{\mathsf{sk}}(c'_i) = 1$, lets $a_i := \mathsf{Dec}_{\mathsf{sk}}(c'_i)$ and outputs the decision of $D_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell, x, \mathbf{u}, (a_1, \ldots, a_k))$.*

Lemma 6.2. *Suppose that the LIP $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ has knowledge error $\varepsilon(\lambda)$ and $\mathcal{E}$ is a linear-only encryption scheme. Then, (G, P, V) from Construction 6.1 is a designated-verifier preprocessing SNARK with knowledge error $\varepsilon(\lambda) + \mathsf{negl}(\lambda)$. Furthermore:*

- $\mathsf{time}(G) = \mathsf{time}(Q_{\mathsf{LIP}}) + \mathrm{poly}(\lambda) \cdot m$,
- $\mathsf{time}(P) = \mathsf{time}(P_{\mathsf{LIP}}) + \mathrm{poly}(\lambda) \cdot k^2 \cdot m$,
- $\mathsf{time}(V) = \mathsf{time}(D_{\mathsf{LIP}}) + \mathrm{poly}(\lambda) \cdot k$,
- $|\sigma| = \mathrm{poly}(\lambda) \cdot m$, $|\tau| = \mathrm{poly}(\lambda) + m'$, *and* $|\pi| = \mathrm{poly}(\lambda) \cdot k$.

Proof sketch. Completeness easily follows from the completeness of $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ and the correctness of $\mathcal{E}$. Efficiency as claimed above is easy to see. We thus focus on establishing the knowledge property.

Let P^* be a malicious polynomial-size prover. We construct a knowledge extractor E for P^* in two steps: first invoke the linear-only extractor E' for P^* (on the same input as P^*) to obtain an LIP affine transformation Π^* "explaining" the encryptions output by P^*, and then invoke the LIP extractor E_{LIP} (with oracle access to Π^* and on input the statement chosen by P^*) to obtain an assignment for the circuit. We now argue that E works correctly.

First, we claim that, except with negligible probability, whenever $P^*(\sigma)$ produces a statement x and proof $\mathbf{c}' = (c'_1, \ldots, c'_k)$ accepted by the verifier, the extracted Π^* is such that $D_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell, x, \mathbf{u}, \mathbf{a}^*) = 1$, where $\mathbf{u}$ is the private state of the verifier and $\mathbf{a}^* = \Pi^*(\mathbf{q})$. Indeed, by the linear-only property of $\mathcal{E}$ (see Definition 5.4), except with negligible probability, whenever the verifier is convinced, $\mathbf{a}^* = \Pi^*(\mathbf{q})$ is equal to $\mathbf{a} = \mathsf{Dec}_{\mathsf{sk}}(\mathbf{c'})$, which is accepted by the LIP decision algorithm.

Second, we claim that, due to semantic security of $\mathcal{E}$, the extracted proof Π^* is not only "locally satisfying" (i.e., such that $D_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell, x, \mathbf{u}, \Pi^*(\mathbf{q})) = 1$ where $\mathbf{q}$ is the message encrypted in σ), but it is in fact satisfying for all but a negligible fraction of queries $\mathbf{q}$; otherwise, E could be used to break semantic security.

We can then conclude that Π^* convinces the LIP verifier for most messages, and thus can be used to extract a valid witness.

(The above proof is similar to proofs establishing security in other works where semantic security and extraction are used in conjunction. For a detailed such proof see, for example, [BCCT12].) □

Designated-verifier non-adaptive preprocessing SNARKs from linear targeted malleability. We also consider a notion that is weaker than linear-only encryption: encryption with linear targeted malleability (see Definition 5.8). For this notion, we are still able to obtain, via the same Construction 6.1, designated-verifier preprocessing SNARKs, but this time only against statements that are non-adaptively chosen.

Lemma 6.3. *Suppose that the LIP $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ has knowledge error $\varepsilon(\lambda)$ and $\mathcal{E}$ is an encryption scheme with linear targeted malleability. Then, (G, P, V) from Construction 6.1 is a designated-verifier* ***non-adaptive*** *preprocessing SNARK with knowledge error $\varepsilon(\lambda) + \mathsf{negl}(\lambda)$.*

Proof sketch. Let P^* be a malicious polynomial-size prover, which convinces the verifier for infinitely many false statements $x \in \mathcal{X}$. By the targeted malleability property (see Definition 5.8), there exists a polynomial-

size simulator S (depending on P^*) such that such that:

$$\left\{\begin{array}{l|r} & (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathsf{pk}, & (\mathbf{q},\mathbf{u}) \leftarrow Q_{\mathsf{LIP}} \\ \mathbf{q}, & \mathbf{c} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathbf{q}) \\ \mathbf{u}, & \mathbf{c}' \leftarrow P^*(\mathsf{pk},\mathbf{c};x) \\ \mathbf{a} & \text{where} \\ & \mathsf{ImVer}_{\mathsf{sk}}(\mathbf{c}') = 1 \\ & \mathbf{a} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathbf{c}') \end{array}\right\} \approx_c \left\{\begin{array}{l|r} \mathsf{pk}, & (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathbf{q}, & (\mathbf{q},\mathbf{u}) \leftarrow Q_{\mathsf{LIP}} \\ \mathbf{u}, & (\Pi,\boldsymbol{b}) \leftarrow S(\mathsf{pk};x) \\ \mathbf{a} & \mathbf{a} \leftarrow \Pi \cdot \mathbf{q} + \boldsymbol{b} \end{array}\right\},$$

where $\mathbf{q}$ is the LIP query and $\mathbf{u}$ is the LIP verification state.

If P^* convinces the verifier to accept with probability at least $\varepsilon(\lambda)$, then, with at least the same probability, the distribution on the left satisfies that $D_{\mathsf{LIP}}(x,\mathbf{u},\mathbf{a}) = 1$. Because this condition is efficiently testable, the simulated distribution on the right satisfies the same condition with probability at least $\varepsilon(\lambda) - \mathsf{negl}(\lambda)$. However, in this distribution the generation of $\mathbf{q}$ and $\mathbf{u}$ is independent of the generation of the simulated affine function $\Pi' = (\Pi, \boldsymbol{b})$. Therefore, by averaging, there is some (in fact, roughly an $\varepsilon(\lambda)/2$-fraction) Π' such that, with probability at least $\varepsilon(\lambda)/2$ over the choice of $\mathbf{q}$ and $\mathbf{u}$, it holds that $D_{\mathsf{LIP}}(x,\mathbf{u},\Pi\mathbf{q}+\boldsymbol{b}) = 1$. We can use the LIP extractor E_{LIP} to extract a valid witness from any such Π'. □

Remark 6.4 (inefficient simulator)**.** As mentioned in Remark (5.9), Definition 5.8 can be weakened by allowing the simulator to be *inefficient*. In such a case, we are able to obtain designated-verifier non-adaptive preprocessing SNARGs (note the lack of the knowledge property), via essentially the same proof as the one we gave above for Lemma 6.3.

Remark 6.5 (a word on adaptivity)**.** One can strengthen Definition 5.8 by allowing the adversary to output an additional (arbitrary) string y, which the simulator must be able to simulate as well. Interpreting this additional output as the adversary's choice of statement, a natural question is whether the strengthened definition suffices to prove security against adaptively-chosen statements as well.

Unfortunately, to answer this question in the positive, it seems that a polynomial-size distinguisher should be able to test whether a statement y output by the adversary is a true or false statement. This may not be possible if y encodes an arbitrary NP statement (and for the restricted case of deterministic polynomial-time computations, the approach we just described does in fact work.)

We stress that, while we do not know how to prove security against adaptively-chosen statements, we also do not know of any attack on the construction in the adaptive case.

6.2 Publicly-Verifiable Preprocessing SNARKs from Algebraic LIPs

We show how to transform any LIP with degree $(d_Q, d_D) = (\mathrm{poly}(\lambda), 2)$ to a *publicly-verifiable* preprocessing SNARK using linear-only one-way encodings with quadratic tests. The restriction to quadratic tests (i.e., $d_D \leq 2$) is made for simplicity, because we only have one-way encoding candidates based on bilinear maps. As noted in Remark (5.12), the transformation can in fact support any $d_D = \mathrm{poly}(\lambda)$, given one-way encodings with corresponding d_D-degree tests.

Construction 6.6. *Let $\{\mathbb{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a field ensemble (with efficient description and operations). Let $\mathcal{C} = \{C_\ell\}_{\ell\in\mathbb{N}}$ be a family of circuits. Let $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ be an input-oblivious two-message LIP for the relation $\mathcal{R}_{\mathcal{C}}$, where for field $\mathbb{F}_\lambda$, the verifier message is in $\mathbb{F}_\lambda^m$, the prover message is in $\mathbb{F}_\lambda^k$, and the knowledge error is $\varepsilon(\lambda)$; assume that the verifier degree is $(d_Q, d_D) = (\mathrm{poly}(\lambda), 2)$. Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{SEnc}, \mathsf{Test}, \mathsf{Add}, \mathsf{ImVer})$ be a linear-only one-way encoding scheme whose plaintext field, for security parameter λ, is $\mathbb{F}_\lambda$. We define a preprocessing SNARK (G, P, V) for $\mathcal{R}_{\mathcal{C}}$ as follows.*

- $G(1^\lambda, 1^\ell)$ *invokes the LIP query algorithm* $Q_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell)$ *to generate an LIP message* $\mathbf{q} \in \mathbb{F}_\lambda^m$ *along with a secret state* $\mathbf{u} \in \mathbb{F}^{m'}$*, generates* $\mathsf{pk} \leftarrow \mathsf{Gen}(1^\lambda)$*, lets* $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(q_i)$ *for* $i \in [m]$*,* $\tilde{c}_i \leftarrow \mathsf{SEnc}_{\mathsf{pk}}(u_i)$ *for* $i \in [m']$*, defines* $\sigma := (\mathsf{pk}, c_1, \ldots, c_m)$ *and* $\tau := (\mathsf{pk}, \tilde{c}_1, \ldots, \tilde{c}_{m'})$*, and outputs* (σ, τ)*. (Assume that both* (σ, τ) *contain* ℓ *and the description of the field* $\mathbb{F}_\lambda$*).*

- $P(\sigma, x, w)$ *invokes the LIP prover algorithm* $P_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell, x, w)$ *to get a matrix* $\Pi \in \mathbb{F}_\lambda^{k \times m}$ *representing its message function, invokes the homomorphic* Add *to generate* k *encodings* $c'_1, \ldots, c'_k$ *for* $\Pi \cdot \mathbf{q}$*, defines* $\pi := (c'_1, \ldots, c'_k)$*, and outputs* π*.*

- $V(\tau, x, \pi)$ *verifies that* $\mathsf{ImVer}_{\mathsf{pk}}(c'_i) = 1$ *for* $i \in [k]$*, lets* $\boldsymbol{t}_x : \mathbb{F}^{k+m'} \to \mathbb{F}^\eta$ *be the quadratic polynomial given by* $D_{\mathsf{LIP}}(\mathbb{F}_\lambda, 1^\ell, x, \cdots)$*, and accepts if and only if* $\mathsf{Test}\big(\mathsf{pk}, \boldsymbol{t}_x, c'_1, \ldots, c'_k, \tilde{c}_1, \ldots, \tilde{c}_{m'}\big) = 1$.

Lemma 6.7. *Suppose that the LIP* $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ *has knowledge error* $\varepsilon(\lambda)$ *and* $\mathcal{E}$ *is a linear-only one-way encoding scheme. Then* (G, P, V) *from Construction 6.6 is a publicly-verifiable preprocessing SNARK with knowledge error* $\varepsilon(\lambda) + \mathsf{negl}(\lambda)$*. Furthermore:*

- $\mathsf{time}(G) = \mathsf{time}(Q_{\mathsf{LIP}}) + \mathrm{poly}(\lambda) \cdot m$*,*
- $\mathsf{time}(P) = \mathsf{time}(P_{\mathsf{LIP}}) + \mathrm{poly}(\lambda) \cdot k^2 \cdot m$*,*
- $\mathsf{time}(V) = \mathrm{poly}(\lambda) \cdot \mathsf{time}(D_{\mathsf{LIP}})$*,*
- $|\sigma| = \mathrm{poly}(\lambda) \cdot m$*,* $|\tau| = \mathrm{poly}(\lambda) \cdot m'$*, and* $|\pi| = \mathrm{poly}(\lambda) \cdot k$*.*

Proof sketch. Completeness easily follows from the completeness of $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ and the correctness of $\mathcal{E}$. Efficiency as claimed above is easy to see. We thus focus on establishing the knowledge property.

Let P^* be a malicious polynomial-size prover. As in the designated-verifier case, we construct its knowledge extractor E in two steps: first invoke the linear-only extractor E' for P^* (on the same input as P^*) to obtain an LIP affine transformation Π^* "explaining" the encryptions output by P^*, and then the LIP extractor E_{LIP} (with oracle access to Π^* and on input the statement chosen by P^*) to obtain an assignment for the circuit. We now argue that E works correctly. Note that this part must be different than the designated-verifier case (see proof of Lemma 6.2) because it relies on $\mathrm{poly}(\lambda)$-power one-wayness (see Definition 5.13) of the linear-only encoding scheme instead of semantic security.

First, we claim that, except with negligible probability, whenever $P^*(\sigma, \tau)$ produces a statement x and proof $\mathbf{c}' = (c'_1, \ldots, c'_k)$ accepted by the verifier, the extracted Π^* is such that $\boldsymbol{t}_x(\Pi^*(\mathbf{q}), \mathbf{u}) = \mathbf{0}$. Indeed, by the linear-only property of $\mathcal{E}$ (see Definition 5.17), except with negligible probability, whenever the verifier is convinced, it holds that $\mathbf{c}' \in \mathsf{Enc}_{\mathsf{pk}}(\Pi^*(\mathbf{q}))$ (i.e., $\mathbf{c}'$ encodes the plaintext $\Pi^*(\mathbf{q})$); moreover, since the verifier only accepts if $\mathsf{Test}\big(\mathsf{pk}, \boldsymbol{t}_x, \mathbf{c}', \tilde{\mathbf{c}}\big) = 1$, where $\tilde{\mathbf{c}} = (\tilde{c}_1, \ldots, \tilde{c}_{m'})$ is the (standard-mode) encoding of $\mathbf{u}$, it indeed holds that $\boldsymbol{t}_x(\Pi^*(\mathbf{q}), \mathbf{u}) = \mathbf{0}$.

Second, we claim that, thinking about $\boldsymbol{t}_x(\Pi^*(\mathbf{q}(\mathbf{r})), \mathbf{u}(\mathbf{r}))$ as a $(d_Q d_D)$-degree polynomial in the randomness $\mathbf{r}$ of the query algorithm Q_{LIP}, it holds that $\boldsymbol{t}_x(\Pi^*(\mathbf{q}(\mathbf{r})), \mathbf{u}(\mathbf{r})) \equiv \mathbf{0}$, i.e. it is vanishes everywhere; in particular, Π^* is a (perfectly) convincing LIP affine function. Indeed, if that is not the case, then, since $\boldsymbol{t}$ is of degree $d_Q \cdot d_D = \mathrm{poly}(\lambda)$, we can use the extractor E to break the $\mathrm{poly}(\lambda)$-power one-wayness of the linear only scheme (see Definition 5.13 and Definition 5.14). □

6.3 Resulting Preprocessing SNARKs

We now state what preprocessing SNARKs we get by applying our different transformations. Let $\mathcal{C} = \{C_\ell\}$ be a circuit family where C_ℓ is of size $s = s(\ell)$ and input size $n = n(\ell)$. Table 2 summarizes (most) of the preprocessing SNARKs obtained from our LIP constructions (from Sections 3.1 and 3.2) and computational transformations (from Sections 6.1 and 6.2).

LIP Thm.	starting point of LIP construction	# ciphertexts (or encodings) in reference string	# ciphertexts (or encodings) in proof	verification time	adaptive or nonadaptive	public or designated	assumption
3.3	Hadamard PCP	$O(s^2)$	4	$n \cdot \text{poly}(\lambda)$	nonadaptive	designated	Paillier TM
"	"	"	8	"	adaptive	designated	Paillier LOEC
"	"	"	8	"	adaptive	public	Bilinear LOED
3.4	QSPs of [GGPR13]	$O(s)$	4	$n \cdot \text{poly}(\lambda)$	nonadaptive	designated	Paillier TM
"	"	"	8	"	adaptive	designated	Paillier LOEC
"	"	"	8	"	adaptive	public	Bilinear LOED
3.9	PCPs of [BSCGT13b]	$\widetilde{O}(s)$	1	$n \cdot \text{poly}(\lambda)$	nonadaptive	designated	Paillier TM
"	"	"	2	"	adaptive	designated	Paillier LOEC

Table 2: Summary of most of our preprocessing SNARK constructions. The gray-row constructions achieve new features compared to previous work. Above, *Paillier TM* stands for Paillier encryption assumed to satisfy Definition 5.8, *Paillier LOEC* stands for a variant of Paillier encryption assumed to satisfy Definition 5.4, and *Bilinear LOED* stands for one-way encodings in bilinear groups that we assume satisfy Definition 5.17. See Section 5.3 for a discussion about instantiations. Recall that adaptivity is a crucial property in order to benefit from the recursive composition techniques of Bitansky et al. [BCCT13].

Zero-knowledge and ZAPs. As mentioned before, if the LIP is HVZK then the corresponding preprocessing SNARK is zero-knowledge (against malicious verifiers in the CRS model), provided that linear only-encryption (or one-way encoding) are rerandomizable; all of our candidates constructions are rerandomizable.

As mentioned in Section 3.1.1, both of our LIP constructions based on LPCPs can be made HVZK (either via the general transformation described in Appendix B or via construction-specific modifications discussed in Appendix A). As for the LIP constructions based on traditional PCPs, we need to start with an HVZK PCP. For efficient such constructions, see [DFK$^+$92].

The zero-knowledge preprocessing SNARKs we obtain are arguments of knowledge where the witness can be extracted *without a trapdoor on the CRS*; this is unlike what happens in typical NIZKs (based on standard assumptions). This property is crucial when recursively composing SNARKs as in [BCCT13].

Finally, the zero-knowledge SNARKs we obtain are, in fact, perfect zero-knowledge. Moreover, for the case of *publicly-verifiable* (perfect) zero-knowledge preprocessing SNARKs, the CRS can be tested, so that (similarly to previous works [Gro10, Lip12, GGPR13]) we also obtain succinct ZAPs.

Acknowledgements

We thank Eli Ben-Sasson, Ran Canetti, Prahladh Harsha, Eran Tromer, and Daniel Wichs for helpful discussions and comments.

A Two LPCPs For Boolean Circuit Satisfaction Problems

We describe two ways of obtaining algebraic LPCPs for boolean circuit satisfaction problems (see Definition 4.5). For simplicity, in this section (as well as in Sections 3.1 and 3.2), we focus on relations $\mathcal{R}_C$ where C is a fixed boolean circuit. All the discussions can be naturally extended to relations $\mathcal{R}_{\mathcal{C}}$ where $\mathcal{C}$ is a uniform boolean circuit family.

A.1 An LPCP From The Hadamard Code

We begin with a very simple LPCP, which is the well-known Hadamard-based PCP of Arora et al. [ALM+98a] (ALMSS), naturally extended to work over an arbitrary finite field $\mathbb{F}$. Since we only require soundness against linear proof oracles, there is no need to apply linearity testing and self-correction as in the original PCP of ALMSS. This makes the construction of the LPCP and its analysis even simpler, and has the advantage of yielding knowledge error $O(1/|\mathbb{F}|)$ with a constant number of queries. The resulting LPCP verifier will have degree $(2, 2)$, so that the LPCP is *algebraic*.

We now formulate the properties of the simplified version of the Hadamard-based PCP in the arithmetic setting:

Claim A.1. *Let $\mathbb{F}$ be a finite field and $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ a boolean circuit of size s. There is a 3-query LPCP for $\mathcal{R}_C$ with knowledge error $2/|\mathbb{F}|$, query length $s + s^2$, and degree $(2, 2)$. Furthermore:*

- *the LPCP prover P_{LPCP} is an arithmetic circuit of size $O(s^2)$;*
- *the LPCP query algorithm Q_{LPCP} is an arithmetic circuit of size $O(s^2)$;*
- *the LPCP decision algorithm D_{LPCP} is an arithmetic circuit of size $O(n)$.*

Construction and analysis. It is convenient to formulate our variant of the ALMSS construction for a relation $\mathcal{R}(x, w)$ defined by an *arithmetic* (rather than boolean) circuit over $\mathbb{F}$. In the following, an arithmetic circuit may contain addition gates of unbounded fan-in and multiplication gates of fan-in 2; both types of gates can have unbounded fan-out. A sequence of one or more nodes (gates or inputs) defines the output of the circuit. The *size* of the circuit is defined as the total number of nodes. We say that an arithmetic circuit $C\colon \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^\ell$ is *satisfied* on a given input if all of the outputs are 0; that is, the corresponding relation is defined as $\mathcal{R}_C := \{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^h : C(\mathbf{x}, \mathbf{w}) = 0^\ell\}$.

The standard problem for boolean circuit satisfaction can be easily reduced to the problem of arithmetic circuit satisfaction with only constant overhead.

Claim A.2 (from boolean circuit satisfaction to arithmetic circuit satisfaction)**.** *Let $\mathbb{F}$ be a finite field, n an input length parameter, and h an output length parameter. There exist efficient (in fact, linear-time) transformations $(\mathsf{arith}, \mathsf{inp}, \mathsf{wit}, \mathsf{wit}^{-1})$ such that, for any boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ (with* AND*,* OR*, and* NOT *gates) and input $x \in \{0,1\}^n$, the following conditions hold:*

- $\mathsf{arith}(C)$ *outputs an arithmetic circuit $C'\colon \mathbb{F}^{n+1} \times \mathbb{F}^h \to \mathbb{F}^{h+1}$;*
- $\mathsf{inp}(x)$ *outputs an input $\mathbf{x} \in \mathbb{F}^{n+1}$;*
- *there is $w \in \{0,1\}^h$ s.t. $C(x, w) = 1$ if and only if there is $\mathbf{w} \in \mathbb{F}^h$ s.t. $C'(\mathbf{x}, \mathbf{w}) = 0^{h+1}$;*
- *if $(x, w) \in \mathcal{R}_C$ then $\mathsf{wit}(C, x, w)$ outputs a witness $\mathbf{w} \in \mathbb{F}^h$ such that $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_{C'}$;*
- *if $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_{C'}$ then $\mathsf{wit}^{-1}(C', \mathbf{x}, \mathbf{w})$ outputs a witness $w \in \{0,1\}^h$ such that $(x, w) \in \mathcal{R}_C$.*

Proof sketch. Let $\mathbf{x} = (-1, x_1, \ldots, x_n)$. The circuit $C'(\mathbf{x}, \mathbf{w})$ emulates the computation of C on $(x, \mathbf{w})$, using the constant -1 to emulate OR and NOT gates, and treating the entries of $\mathbf{w}$ as bits of w. (For instance, $\mathrm{NOT}(z)$ can be computed as $1 + (-1) \cdot z$.) The first output of C' is equal to $1 - C(x, \mathbf{w})$ (for all $\mathbf{w} \in \{0,1\}^h$). The remaining h outputs of C' are used to ensure that any satisfying $\mathbf{w} \in \mathbb{F}^h$ is a bit vector. This is done by outputting $w_i \cdot (1 - w_i)$ for $i \in \{1, \ldots, h\}$. □

We now proceed to describe the construction of the LPCP for an arithmetic circuit $C \colon \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^\ell$. We let z_i denote the value of the i-th wire of C given input $\mathbf{x} \in \mathbb{F}^n$ and witness $\mathbf{w} \in \mathbb{F}^h$, where we assume that $z_i = x_i$ for $i \in \{1, \ldots, n\}$ and $z_{s-\ell+1}, \ldots, z_s$ are the values of the output wires (which are supposedly 0). The honest prover uses a linear oracle $\boldsymbol{\pi}$ whose coefficient vector includes $z_1, \ldots, z_s$, and $z_i \cdot z_j$ for all $i, j \in \{1, \ldots, s\}$. The verifier needs to check that: (1) the coefficient vector is consistent with itself (namely, the entry which supposedly contained $z_i \cdot z_j$ is indeed the product of the entries containing z_i and z_j); (2) $z_i = x_i$ for $i \in \{1, \ldots, n\}$; (3) $z_{s-i} = 0$ for $i \in \{0, \ldots, \ell - 1\}$; and (4) for every gate, the given value for its output wire is consistent with the given values for its input wires.

The first condition can be verified with two queries: the first query asks for a random linear combination of the z_i and the second query asks for the linear combination of the $z_i \cdot z_j$ that corresponds to the square of the first query. The verifier checks that the second answer is the square of the first answer. It follows from the Schwartz–Zippel Lemma (cf. Lemma 2.1) that if condition (1) is violated then this test will fail except with at most $2/|\mathbb{F}|$ probability. Conditions 2,3,4 are tested together using a single query by taking a random linear combination with coefficient vector $\boldsymbol{r}$ of the left hand sides of the following equations:

- $z_i = x_i$ for $i = 1, \ldots, n$;
- $z_j = 0$ for $j = s - (\ell - 1), \ldots, s$;
- $\left(\sum_{j=1}^{k} z_{i_j}\right) - z_k = 0$ for each addition gate with input wires $i_1, \ldots, i_k$ and output wire k;
- $z_i \cdot z_j - z_k = 0$ for each multiplication gate with input wires i, j and output wire k.

Note that this random linear combination can be efficiently transformed into a random linear combination of the coefficients z_i and $z_i \cdot z_j$. The verifier accepts if the answer is equal to the corresponding linear combination of the right hand side, namely to $\sum_{i=1}^{n} r_i x_i$. Assuming that condition (1) is satisfied, this test fails with probability at most $1/|\mathbb{F}|$.

Overall, we get a 3-query LPCP with knowledge error $2/|\mathbb{F}| = O(1/|\mathbb{F}|)$, query length $s + s^2 = O(s^2)$ and degree $(2, 2)$. The construction is summarized as follows:

Construction A.3. *Let $\mathbb{F}$ be a finite field and $C \colon \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^\ell$ an arithmetic circuit over $\mathbb{F}$ of size s.*

LPCP prover algorithm P_{LPCP}. *Given $(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^h$ such that $C(\mathbf{x}, \mathbf{w}) = 1$, compute the values $z_1, \ldots, z_s$ of all wires in $C(\mathbf{x}, \mathbf{w})$, and output the linear function $\boldsymbol{\pi} \colon \mathbb{F}^{s+s^2} \to \mathbb{F}$ defined by the coefficients z_i for all $i \in [s]$ and $z_i \cdot z_j$ for all $i, j \in [s]$.*

LPCP query algorithm Q_{LPCP}. *The query algorithm Q_{LPCP} has hardcoded in it:*

- *A matrix $A_C \in \mathbb{F}^{s \times (s+s^2)}$*
- *A vector $\boldsymbol{b}_C \in \mathbb{F}^{s-n}$*

Both A_C and $\boldsymbol{b}_C$ can be computed efficiently (in fact, in linear time) from C. The query algorithm Q_{LPCP} outputs queries $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3 \in \mathbb{F}^{s+s^2}$ that are computed as follows:

1. *sample a random vector* $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2) \in \mathbb{F}^{2s}$*; denote by* $\boldsymbol{r}_{\mathbf{x}}$ *the first* n *coordinates of* $\boldsymbol{r}_1$ *and by* $\boldsymbol{r}_C$ *the last* $s - n$ *coordinates of* $\boldsymbol{r}_1$*;*
2. *set* $\boldsymbol{q}_1 := \boldsymbol{r}_1 \cdot A_C$*;*
3. *the first* s *elements of* $\boldsymbol{q}_2$ *is the vector* $\boldsymbol{r}_2$ *and the last* s^2 *elements are* 0*;*
4. *the first* s *elements of* $\boldsymbol{q}_3$ *are* 0 *and the last* s^2 *element are* $\boldsymbol{r}_2[i] \cdot \boldsymbol{r}_2[j]$ *for all* $i, j \in [s]$*.*

Additionally the query algorithm Q_{LPCP} *outputs the state information* $\mathbf{u} = (u_C, \boldsymbol{r}_{\mathbf{x}})$ *where* $\mathbf{u}_C = \langle \boldsymbol{r}_C, \boldsymbol{b}_C \rangle$*.*

LPCP decision algorithm D_{LPCP}**.** *Given input* $\mathbf{x}$*, state* $\mathbf{u} = (u_C, \boldsymbol{r}_{\mathbf{x}})$*, and answers* $a_1, a_2, a_3 \in \mathbb{F}$*, verify that*

$$\boldsymbol{t}_x(\mathbf{u}, a_1, a_2, a_3) := \left(a_1 - (u_C + \langle \boldsymbol{r}_{\mathbf{x}}, \mathbf{x} \rangle)\,,\; a_2^2 - a_3\right) = (0, 0) \;\; .$$

Remark A.4 (HVZK variant)**.** To make the LPCP of Claim A.1 an HVZK LPCP, we could apply the general transformation presented in Appendix B. However, there is a transformation specific to the LPCP of Claim A.1 that introduces almost no overhead. Essentially, the prover can simply concatenate a random element to the linear function he generates; this can be interpreted as adding a dummy wire to the circuit and assigning to it a random value; the verifier then reasons in terms of this new circuit. In fact, the idea we just described is a very simple instance of the transformation in Appendix B, which generalizes it to work for *any* LPCP.

A.2 An LPCP From Quadratic Span Programs

Next, we note that an LPCP with linear query algorithm and quasilinear prover algorithm can be easily obtained by going through the quadratic span programs of Gennaro et al. [GGPR13]; this gain is at the expense of the degree of the query algorithm which now becomes $O(s)$ instead of 2.

Claim A.5. *Let* $\mathbb{F}$ *be a finite field and* $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ *a boolean circuit of size* s*. There is a* 3*-query LPCP for* $\mathcal{R}_C$ *with knowledge error* $O(s/|\mathbb{F}|)$*, query length* $O(s)$*, and degree* $(O(s), 2)$*. Furthermore:*

- *the LPCP prover* P_{LPCP} *is an arithmetic circuit of size* $\widetilde{O}(s)$*;*
- *the LPCP query algorithm* Q_{LPCP} *is an arithmetic circuit of size* $O(s)$*;*
- *the LPCP decision algorithm* D_{LPCP} *is an arithmetic circuit of size* $O(n)$*.*

Construction and analysis. We present the definition of a quadratic span program satisfiability problem, based on the definition of a strong quadratic-span program (QSP) in [GGPR13].

Definition A.6. *Let* $\mathbb{F}$ *be a finite field. The* **QSP satisfiability problem** *over* $\mathbb{F}$ *of a (strong) QSP* $Q = (n, m, d, \mathbb{V}, \mathbb{W}, t)$*, where* $\mathbb{V} = \left(v_i(z)\right)_{0 \le i \le m}$ *and* $\mathbb{W} = \left(w_i(z)\right)_{0 \le i \le m}$ *are two vectors of polynomials of degree* d *over* $\mathbb{F}$ *and* t *is a polynomial over* $\mathbb{F}$*, is the relation:*

$$\mathcal{R}_Q := \left\{ \left(\mathbf{x}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{h})\right) : \;\; t(z) \cdot \left(\sum_{i=0}^{d-1} h_i \cdot z^i \right) = \left(v_0(z) + \sum_{i=1}^{n} x_i \cdot v_i(z) + \sum_{i=1}^{m-n} a_i \cdot v_{i+n}(z) \right) \cdot \left(w_0(z) + \sum_{i=1}^{m} b_i \cdot w_i(z) \right) \right\} \;\; ,$$

where $\mathbf{x} \in \mathbb{F}^n$*,* $\boldsymbol{a} \in \mathbb{F}^{m-n}$*,* $\boldsymbol{b} \in \mathbb{F}^m$*, and* $\boldsymbol{h} \in \mathbb{F}^d$*.*

Gennaro et al. [GGPR13] have shown that there is a very efficient reduction from boolean circuit satisfiability problems to QSP satisfiability problems:

Claim A.7. *Let $\mathbb{F}$ be a field. There exist transformations $(\mathsf{qsp}, \mathsf{inp}, \mathsf{wit}, \mathsf{wit}^{-1})$ such that, for any boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ and input $x \in \{0,1\}^n$, the following conditions hold:*

- $\mathsf{qsp}(C)$ *outputs a QSP Q where $m = O(|C|)$, $d = O(|C|)$, and $n' = O(n)$; moreover, Q is sparsely represented ;*
- $\mathsf{inp}(x)$ *outputs an input $\mathbf{x} \in \mathbb{F}^{n'}$;*
- *there is $w \in \{0,1\}^h$ s.t. $C(x,w) = 1$ if and only if there is $\mathbf{w}$ s.t. $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_Q$;*
- *if $(x,w) \in \mathcal{R}_C$ then $\mathsf{wit}(C, x, w)$ outputs a witness $\mathbf{w}$ such that $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_Q$;*
- *if $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_Q$ then $\mathsf{wit}^{-1}(Q, \mathbf{x}, \mathbf{w})$ outputs a witness $w \in \{0,1\}^h$ such that $(x,w) \in \mathcal{R}_C$.*

Moreover, $\mathsf{qsp}, \mathsf{inp}, \mathsf{wit}^{-1}$ run in linear time and wit runs in quasilinear time.

We now give a construction of an LPCP for a QSP satisfiability problem:

Construction A.8. *Let $\mathbb{F}$ be a field and $Q = (n, m, d, \mathbb{V}, \mathbb{W}, t)$ a (strong) QSP.*

LPCP prover algorithm P_{LPCP}. *Given $(\mathbf{x}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{h}))$ such that $(\mathbf{x}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{h})) \in \mathcal{R}_Q$, output the linear function $\boldsymbol{\pi}\colon \mathbb{F}^{2m-n+d} \to \mathbb{F}$ defined by $\boldsymbol{\pi} := (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{h})$ (i.e., the concatenation of the coefficients vectors).*

LPCP query algorithm Q_{LPCP}. *Select a random point $r \in \mathbb{F}$ and output the queries $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3 \in \mathbb{F}^{2m-n+d}$ defined as follows:*

- *the first $m-n$ elements of $\boldsymbol{q}_1$ are $v_i(r)$ for all $n < i \leq m$ and the last $m+d$ element are 0;*
- *the first $m-n$ and the last d elements of $\boldsymbol{q}_2$ are 0 and the remaining m elements are $w_i(r)$ for all $0 < i \leq m$;*
- *the first $(m-n)+m$ elements of $\boldsymbol{q}_3$ are 0 and the last d elements are r^i for all $0 \leq i < d$.*

Additionally the algorithm outputs the state information $\mathbf{u} := (\{v_i(r)\}_{0 \leq i \leq n}, w_0(r), t(r)) \in \mathbb{F}^{n+2}$.

LPCP decision algorithm D_{LPCP}. *Given input x, state $\mathbf{u} = (\{v_i(r)\}_{0 \leq i \leq n}, w_0(r), t(r))$, and answers a_1, a_2, a_3, verify that*

$$\boldsymbol{t}_x(\mathbf{u}, a_1, a_2, a_3) := \left(v_0(r) + \sum_{i=1}^{n} x_i \cdot v_i(r) + a_1 \right) \cdot (w_0(r) + a_2) - a_3 \cdot t(r) = 0 \ .$$

We now prove that Construction A.8 has the desired properties. Correctness follows from the construction. Next, fix a linear function $\boldsymbol{\pi}^*\colon \mathbb{F}^{2m-n+d} \to \mathbb{F}$ and view it as $\boldsymbol{\pi}^* = (\boldsymbol{a}^*, \boldsymbol{b}^*, \boldsymbol{h}^*)$ for some $\boldsymbol{a}^* \in \mathbb{F}^{m-n}$, $\boldsymbol{b}^* \in \mathbb{F}^m$, and $\boldsymbol{h}^* \in \mathbb{F}^d$. Suppose that $V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x)$ accepts with more than $2d/|\mathbb{F}|$ probability. By construction, we have that:

$$\Pr_{r \leftarrow \mathbb{F}} \left[\left(v_0(r) + \sum_{i=1}^{n} x_i \cdot v_i(r) + \sum_{i=1}^{m-n} a^*_i \cdot v_{i+n}(r) \right) \cdot \left(w_0(r) + \sum_{i=1}^{m} b^*_i \cdot w_i(r) \right) - \left(\sum_{i=0}^{d-1} h^*_i \cdot r^i \right) \cdot t(r) = 0 \right] > \frac{2d}{|\mathbb{F}|} \ .$$

By the Schwartz–Zippel Lemma (cf. Lemma 2.1) we deduce that:

$$\left(v_0(z) + \sum_{i=1}^{n} x_i \cdot v_i(z) + \sum_{i=1}^{m-n} a_i^* \cdot v_{i+n}(z)\right) \cdot \left(w_0(z) + \sum_{i=1}^{m} b_i^* \cdot w_i(z)\right) = \left(\sum_{i=0}^{d-1} h_i^* \cdot z^i\right) \cdot t(z) \ .$$

We conclude that $(\mathbf{x}, (\boldsymbol{a}^*, \boldsymbol{b}^*, \boldsymbol{h}^*)) \in \mathcal{R}_Q$. We thus have a 3-query LPCP for $\mathcal{R}_Q$ with knowledge error $2d/|\mathbb{F}|$, query length $2m - n + d$, and degree $(d, 2)$. Furthermore, the LPCP prover P_{LPCP} runs in time $O(m + d)$, the LPCP query algorithm Q_{LPCP} runs in time $O(m + d)$ (since it only has to produce random evaluations of polynomials), and the LPCP decision algorithm D_{LPCP} runs in time $O(n)$.

Invoking Claim A.7, we obtain 3-query LPCP for $\mathcal{R}_C$, where $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ is a boolean circuit of size s, with knowledge error $O(s/|\mathbb{F}|)$, query length $O(s)$, and degree $(O(s), 2)$. Furthermore, the LPCP prover P_{LPCP} runs in time $\widetilde{O}(s)$, the LPCP query algorithm Q_{LPCP} runs in time $O(s)$, and the LPCP decision algorithm D_{LPCP} runs in time $O(n)$.

Remark A.9 (HVZK variant)**.** To make the LPCP of Claim A.5 an HVZK LPCP, we could apply the general transformation presented in Appendix B. However, Gennaro et al. [GGPR13] give a transformation for the specific case of QSPs that introduces almost no overhead. Their transformation is rather different from our general transformation, and exploits special features of QSPs. At a very high-level, we first pad each of the LPCP answers with a random field element, and then add terms to the proof that allow a verifier to cancel the noise, but without leaking any further information. In particular, in our transformation, the verifier has to make sure that these additional terms are properly structured, and doing so requires additional tests. Instead, [GGPR13] use the special QSP structure: instead of padding the LPCP answers with random field elements, they add randomized factors of the target polynomial t to the answers, and this is already sufficient to force the prover to stick to the proper structure, and does not require additional structure tests.

B HVZK For LPCPs With Low-Degree Decision Algorithm

We describe a *general* transformation that takes any LPCP with $d_D = 2$ to a corresponding LPCP that is HVZK, with only a small overhead in parameters. (With additional work, the transformation can in fact be extended to work for $d_D = O(1)$.)

Theorem B.1. *There exists an efficient compiler that takes any LPCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ with $d_D = 2$ into an LPCP $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ with the following properties:*

- *if $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has knowledge error ε then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has knowledge error $\varepsilon + O(1/|\mathbb{F}|)$;*
- *$(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ is $O(1/|\mathbb{F}|)$-statistical HVZK (and can be made perfect HVZK if V'_{LPCP} is allowed to sample its randomness from $\mathbb{F} - \{0\}$ instead of $\mathbb{F}$);*
- *if $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has query length m, k queries, private state of length m', η test polynomials, and degree $(d_Q, 2)$, then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has query length $O(\eta k^4(m+m'))$, $O(k+\eta)$ queries, private state of length $O(k^2 m')$, $O(\eta)$ test polynomials, and degree $(d_Q, 2)$.*

Remark B.2 (parameters)**.** Typically (and this is the case for both of the LPCP constructions of Section A), k and η are constants and $m' < m$ (e.g., m is proportional to the size of the circuit to be verified and m' is proportional to the input size of the circuit). For such typical choices, $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has the same parameters as $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$, up to constant factors.

Of course, because the transformation guaranteed by Theorem B.1 is generic, optimizations are often possible in specific cases to, e.g., reduce the number of queries. For instance, the two LPCP constructions described in Appendix A have HVZK variants with almost no overhead (see Remarks A.4 and A.9).

The rest of this section contains the proof of Theorem B.1. Let us begin with the high-level idea.

The high-level idea. The LPCP prover adds to each of the original LPCP answers a_i a random element ξ_i; clearly, the distribution of these new answers is independent of the witness used by the prover, and thus the new answers are HVZK. Unfortunately, this modification allows the LPCP verifier to only compute a *noisy evaluation* of the test polynomial, namely $\boldsymbol{t}_x(\mathbf{u}, \mathbf{a} + \boldsymbol{\xi})$, when instead the verifier needed a noiseless evaluation. To recover soundness without breaking HVZK, the verifier needs to learn the difference $\boldsymbol{\Delta} = \boldsymbol{t}_x(\mathbf{u}, \mathbf{a}+\boldsymbol{\xi}) - \boldsymbol{t}_x(\mathbf{u}, \mathbf{a})$, but nothing else. We show how the prover can do this by adding suitable crossterms to the linear function, and letting the verifier test that the new linear function, which includes these cross terms, has the correct "structure". We now concretize this high-level idea by proceeding in three steps.

Step 1: soundness against provers respecting quadratic tensor relations. In our transformation, an honest LPCP proof $\boldsymbol{\pi}$ is mapped to a new proof $\boldsymbol{\pi}' = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_c)$ that must satisfy a set $\mathcal{T}$ of *quadratic tensor relations*; a quadratic tensor relation is a triple $(i, j, k) \in [c]^3$, that corresponds to the requirement $\boldsymbol{\pi}_i = \boldsymbol{\pi}_j \otimes \boldsymbol{\pi}_k$.[9] Thus we can write:

$$\mathcal{T} = \{\boldsymbol{\pi}_{i_\ell} = \boldsymbol{\pi}_{j_\ell} \otimes \boldsymbol{\pi}_{k_\ell}\}_{\ell=1}^{\beta} \quad ,$$

where $\mathbf{a} \otimes \mathbf{b} = (a_i \cdot b_j)_{i,j}$ is the tensor product of vectors $\mathbf{a}$ and $\mathbf{b}$. We say that β is the *size* of $\mathcal{T}$. In our construction, $\mathcal{T}$ is independent of the input x, so we focus our attention to this case.

We begin by showing that any LPCP that is secure against $\mathcal{T}$-respecting provers (for some given $\mathcal{T}$) can be transformed into a corresponding LPCP against *arbitrary* provers, with a small overhead in parameters and while maintaining HVZK. For a given set of quadratic tensor relations $\mathcal{T}$, we say that an LPCP has soundness (or knowledge) error ε *against $\mathcal{T}$-respecting provers* if it has soundness (or knowledge) error ε against proofs that satisfy all the quadratic tensor relations in $\mathcal{T}$.

Claim B.3. *There exists an efficient LPCP transformation $\mathbb{T}$ such that, for any set of quadratic tensor relations $\mathcal{T}$, the following properties hold:*

- Security: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is an LPCP for a relation $\mathcal{R}$ with knowledge error ε against $\mathcal{T}$-respecting provers, then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}}) := \mathbb{T}(\mathcal{T}, P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is an LPCP for $\mathcal{R}$ with knowledge error $\varepsilon + O(1/|\mathbb{F}|)$.*
- Zero-Knowledge Preservation: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is δ-statistical HVZK then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ is $(\delta + 1/|\mathbb{F}|)$-statistical HVZK. (If V'_{LPCP} is allowed to sample randomness from $\mathbb{F} - \{0\}$ instead of $\mathbb{F}$ then $(P'_{\mathsf{LPCP}}$, $V'_{\mathsf{LPCP}})$ can be made δ-statistical HVZK.)*
- Efficiency: *If $\mathcal{T}$ has size β, $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has query length m, k queries, η test polynomials, and degree (d_Q, d_D), then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has query length $O(m)$, $k + O(\beta)$ queries, $\eta + \beta$ test polynomials, and degree $(d_Q, \max\{d_D, 2\})$.*

Proof sketch. We sketch the transformation $\mathbb{T}$ and an argument for its correctness; a detailed description of the transformation can be found after the proof sketch, in Construction B.4.

[9] More precisely, $\mathcal{T}$ must also specify $c \in \mathbb{N}$ and how to parse the vector $\boldsymbol{\pi}'$ into component vectors $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_c$. We notationally ignore this small detail. We assume that the same i cannot appear in more than one triple and that each vector $\boldsymbol{\pi}_i$ has at least two coordinates. Note that there cannot be "cycles" such as (i, j, k) and (k, j', i) both being in $\mathcal{T}$.

The transformation leverages the fact that any quadratic tensor relation can be checked with only 3 queries and a quadratic decision predicate. Concretely, given two vectors $\mathbf{a}$ and $\mathbf{b}$ and another vector $\mathbf{c}$, to test whether $\mathbf{c} = \mathbf{a} \otimes \mathbf{b}$, we can sample two random vectors $\mathbf{r_a}$ and $\mathbf{r_b}$, and test whether

$$\langle \mathbf{c}, \mathbf{r_a} \otimes \mathbf{r_b} \rangle = \langle \mathbf{a}, \mathbf{r_a} \rangle \cdot \langle \mathbf{b}, \mathbf{r_b} \rangle \quad .$$

By the Schwartz–Zippel Lemma (see Lemma 2.1), if the quadratic tensor relation does not hold, then the test passes with probability at most $2/|\mathbb{F}|$.

Thus, letting $\boldsymbol{\pi}^* = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_c)$ be a potentially malicious proof, if we want to test whether $\boldsymbol{\pi}^*$ satisfies the set of quadratic tensor relations $\mathcal{T}$, we proceed as follows. For each $(i, j, k) \in \mathcal{T}$, generate queries $\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k$ to $\boldsymbol{\pi}^*$ so that

$$\begin{aligned} \langle \boldsymbol{\pi}^*, \boldsymbol{q}_i \rangle &= \langle \boldsymbol{\pi}_i, \mathbf{r}_j \otimes \mathbf{r}_k \rangle \\ \langle \boldsymbol{\pi}^*, \boldsymbol{q}_j \rangle &= \langle \boldsymbol{\pi}_j, \mathbf{r}_j \rangle \\ \langle \boldsymbol{\pi}^*, \boldsymbol{q}_k \rangle &= \langle \boldsymbol{\pi}_k, \mathbf{r}_k \rangle \end{aligned}$$

where $\mathbf{r}_j$ and $\mathbf{r}_k$ are random vectors of suitable length,[10] and then test whether $\langle \boldsymbol{\pi}^*, \boldsymbol{q}_i \rangle = \langle \boldsymbol{\pi}^*, \boldsymbol{q}_j \rangle \cdot \langle \boldsymbol{\pi}^*, \boldsymbol{q}_k \rangle$.

While the solution described in the previous paragraph does provide suitable security and efficiency guarantees, it does not preserve HVZK, because the additional "structure" queries may reveal information about the witness. To fix this problem, we proceed as follows. For each quadratic tensor relation $\mathbf{c} = \mathbf{a} \otimes \mathbf{b}$, we modify the part of the proof corresponding to it, as well as the verifier's "structure" queries for it. Specifically, we extend the vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ to $\mathbf{a}' = (\mathbf{a}|\xi_{\mathbf{a}})$, $\mathbf{b}' = (\mathbf{b}|\xi_{\mathbf{b}})$, $\mathbf{c}' = \mathbf{a}' \otimes \mathbf{b}'$, where $\xi_{\mathbf{a}}$ and $\xi_{\mathbf{b}}$ are random field elements; similarly, the verifier's "structure" queries $\mathbf{r_a}, \mathbf{r_b}, \mathbf{r_a} \otimes \mathbf{r_b}$ are extended to $\mathbf{r}'_{\mathbf{a}} = (\mathbf{r_a}|r_{\mathbf{a}})$, $\mathbf{r}'_{\mathbf{b}} = (\mathbf{r_b}|r_{\mathbf{b}})$, $\mathbf{r}'_{\mathbf{a}} \otimes \mathbf{r}'_{\mathbf{b}}$, where $r_{\mathbf{a}}$ and $r_{\mathbf{b}}$ are random field elements. The two query answers $\langle \mathbf{a}', \mathbf{r}'_{\mathbf{a}} \rangle$ and $\langle \mathbf{b}', \mathbf{r}'_{\mathbf{b}} \rangle$ are now truly random conditioned on $r_{\mathbf{a}}$ and $r_{\mathbf{b}}$ being non-zero, and hence can be simulated.[11] In addition, $\langle \mathbf{c}', \mathbf{r}'_{\mathbf{a}} \otimes \mathbf{r}'_{\mathbf{b}} \rangle$ can be simulated by taking the product of the simulated answers. As for the queries of the original LPCP, these can be appropriately padded with zeros so that the answers to them are not affected by the changes to the proof; the answers to these queries are simulatable, when assuming that the original LPCP is HVZK. Overall, the modification to preserve HVZK does not change the number of queries, and increases their length m by at most a factor of $1 + \frac{1}{m} < 2$.

This concludes the proof sketch for Claim B.3. □

Construction B.4 (transformation for Claim B.3)**.** *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be an LPCP for $\mathcal{R}$ that is secure against $\mathcal{T}$-respecting provers, for some set of quadratic tensor relations $\mathcal{T}$ of size β. Parse a proof $\boldsymbol{\pi}$ as follows:*

$$\boldsymbol{\pi} = \begin{pmatrix} \boldsymbol{\pi}_1 \\ \vdots \\ \boldsymbol{\pi}_\alpha \\ \boldsymbol{\pi}_{\alpha+1} \\ \vdots \\ \boldsymbol{\pi}_{\alpha+\beta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\pi}_1 \\ \vdots \\ \boldsymbol{\pi}_\alpha \\ \boldsymbol{\pi}_{j_1} \otimes \boldsymbol{\pi}_{k_1} \\ \vdots \\ \boldsymbol{\pi}_{j_\beta} \otimes \boldsymbol{\pi}_{k_\beta} \end{pmatrix} ,$$

[10] If the same j or k appears in multiple triples in $\mathcal{T}$, there is no need to sample a new random vector.

[11] Here is where we lose $1/|\mathbb{F}|$ in statistical distance between the honest distribution and the simulated distribution. If, however, the honest verifier could sample randomness from the set $\mathbb{F} - \{0\}$ (instead of $\mathbb{F}$ as required by the definition), then this loss can be avoided at the expense of an additional $1/|\mathbb{F}|$ factor in the knowledge error. (Note that mapping the uniform distribution on $\mathbb{F}$ to the uniform distribution on $\mathbb{F} - \{0\}$ is not a low-degree operation so, if we want to ensure that $d'_Q = d_Q$, then we cannot simply let V'_{LIP} do this, but we must supply him with the uniform distribution on $\mathbb{F} - \{0\}$.)

where $c = \alpha + \beta$ is the number of components in $\boldsymbol{\pi}$ and $(\alpha+1, j_1, k_1), \ldots, (\alpha+\beta, j_\beta, k_\beta)$ are the β triples in $\mathcal{T}$. (We can always relabel triples in $\mathcal{T}$ so that the β components of $\boldsymbol{\pi}$ constrained by a tensor relation in $\mathcal{T}$ appear after any unconstrained component.) Note that each of $j_1, \ldots, j_\beta, k_1, \ldots, k_\beta$ can be any index in $[\alpha+\beta]$ (always subject to the condition that the are no "cycles" in $\mathcal{T}$).

Construct the LPCP $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ as follows.

- ***The prover P'_{LPCP}.*** *Given $(x, w) \in \mathcal{R}$, P'_{LPCP} invokes $P_{\mathsf{LPCP}}(x, w)$ to obtain a proof $\boldsymbol{\pi}$ as above, samples random $\boldsymbol{\xi} \in \mathbb{F}^{\alpha+\beta}$, and outputs the proof $\boldsymbol{\pi}'$ defined as follows:*

$$\boldsymbol{\pi}' = \begin{pmatrix} \boldsymbol{\pi}'_1 \\ \vdots \\ \boldsymbol{\pi}'_\alpha \\ \boldsymbol{\pi}'_{\alpha+1} \\ \vdots \\ \boldsymbol{\pi}'_{\alpha+\beta} \end{pmatrix} = \begin{pmatrix} (\boldsymbol{\pi}_1 | \xi_1) \\ \vdots \\ (\boldsymbol{\pi}_\alpha | \xi_\alpha) \\ (\boldsymbol{\pi}'_{j_1} \otimes \boldsymbol{\pi}'_{k_1} | \xi_{\alpha+1}) \\ \vdots \\ (\boldsymbol{\pi}'_{j_\beta} \otimes \boldsymbol{\pi}'_{k_\beta} | \xi_{\alpha+\beta}) \end{pmatrix} .$$

  Let us explain the mapping from $\boldsymbol{\pi}$ to $\boldsymbol{\pi}'$ in words. Let us assume without loss of generality that the triples $(\alpha+1, j_1, k_1), \ldots, (\alpha+\beta, j_\beta, k_\beta)$ are labeled so that they are in a "topological order": namely, if $\ell < \ell'$ then it cannot be that j_ℓ or k_ℓ is equal to $\alpha + \ell'$. (A topological order exists because there are no cycles in $\mathcal{T}$.) We first pad each of the components $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\alpha$ with a random element to obtain $\boldsymbol{\pi}'_1, \ldots, \boldsymbol{\pi}'_\alpha$. Then, for $\ell = 1, \ldots, \beta$ (in this order), we pad $\boldsymbol{\pi}'_{j_\ell} \otimes \boldsymbol{\pi}'_{k_\ell}$ with a random element to obtain $\boldsymbol{\pi}'_{\alpha+\ell}$; because of the topological order, when defining $\boldsymbol{\pi}'_{\alpha+\ell}$, we have already defined both $\boldsymbol{\pi}'_{j_\ell}$ and $\boldsymbol{\pi}'_{k_\ell}$.

- ***The query algorithm Q'_{LPCP}.*** *The query algorithm Q'_{LPCP} invokes Q_{LPCP} to obtain queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ and a state $\mathbf{u}$, and then produces and outputs (along with $\mathbf{u}$) new queries*

$$\boldsymbol{q}^{(1)}_1, \ldots, \boldsymbol{q}^{(1)}_k, \boldsymbol{q}^{(2)}_{s_1}, \ldots, \boldsymbol{q}^{(2)}_{s_\gamma}, \boldsymbol{q}^{(3)}_{\alpha+1}, \ldots, \boldsymbol{q}^{(3)}_{\alpha+\beta} \ ,$$

  where $s_1, \ldots, s_\gamma$ are the (distinct) indices in the set $\{j\}_{(i,j,k)\in\mathcal{T}} \cup \{k\}_{(i,j,k)\in\mathcal{T}}$; note that $\gamma \leq 2\beta$. The new queries are defined as follows.

  - *The queries $\boldsymbol{q}^{(1)}_1, \ldots, \boldsymbol{q}^{(1)}_k$ correspond to the original queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ to $\boldsymbol{\pi}$. In other words, we construct each $\boldsymbol{q}^{(1)}_j$ so that*

$$\left\langle \boldsymbol{\pi}', \boldsymbol{q}^{(1)}_j \right\rangle = \left\langle \boldsymbol{\pi}, \boldsymbol{q}_j \right\rangle \ .$$

    Since $\boldsymbol{\pi}'$ contains $\boldsymbol{\pi}$ (and some new terms), each $\boldsymbol{q}^{(1)}_j$ can be obtained from $\boldsymbol{q}_j$ via suitable padding with zeros (in the locations corresponding to the new terms).

  - *The remaining queries are for testing the tensor structure in a way that preserves HVZK:*
    - *For $\ell \in [\gamma]$, $\boldsymbol{q}^{(2)}_{s_\ell}$ is constructed so that*

$$\left\langle \boldsymbol{\pi}', \boldsymbol{q}^{(2)}_{s_\ell} \right\rangle = \left\langle \boldsymbol{\pi}'_{s_\ell}, (\mathbf{r}^{(2)}_{s_\ell} | r^{(2)}_{s_\ell}) \right\rangle = \begin{cases} \left\langle \boldsymbol{\pi}_{s_\ell}, \mathbf{r}^{(2)}_{s_\ell} \right\rangle + \xi_{s_\ell} r^{(2)}_{s_\ell} & \text{if } s_\ell \in \{1, \ldots, \alpha\} \\ \left\langle \boldsymbol{\pi}'_{j_{s_\ell - \alpha}} \otimes \boldsymbol{\pi}'_{k_{s_\ell - \alpha}}, \mathbf{r}^{(2)}_{s_\ell} \right\rangle + \xi_{s_\ell} r^{(2)}_{s_\ell} & \text{if } s_\ell \in \{\alpha+1, \ldots, \alpha+\beta\} \end{cases} ,$$

      where $\mathbf{r}^{(2)}_{s_\ell}$ is a random vector (of suitable length) and $r^{(2)}_{s_\ell}$ is a random field element.

∗ *For $\ell \in [\beta]$, $\boldsymbol{q}^{(3)}_{\alpha+\ell}$ is constructed so that*

$$\left\langle \boldsymbol{\pi}', \boldsymbol{q}^{(3)}_{\alpha+\ell} \right\rangle = \left\langle \boldsymbol{\pi}'_{\alpha+\ell}, ((\mathbf{r}^{(2)}_{j_\ell} | r^{(2)}_{j_\ell}) \otimes (\mathbf{r}^{(2)}_{k_\ell} | r^{(2)}_{k_\ell}) | r^{(3)}_{\alpha+\ell}) \right\rangle = \left\langle \boldsymbol{\pi}'_{j_\ell} \otimes \boldsymbol{\pi}'_{k_\ell}, (\mathbf{r}^{(2)}_{j_\ell} | r^{(2)}_{j_\ell}) \otimes (\mathbf{r}^{(2)}_{k_\ell} | r^{(2)}_{k_\ell}) \right\rangle + \xi_{\alpha+\ell} r^{(3)}_{\alpha+\ell} \ ,$$

where $r^{(3)}_{\alpha+\ell}$ is a random field element.

- ***The decision algorithm*** D'_{LPCP}***.*** *The decision algorithm D'_{LPCP} invokes D_{LPCP} on the answers $a^{(1)}_1, \ldots, a^{(1)}_k$ and checks that $a^{(3)}_{\alpha+1} = a^{(2)}_{j_1} a^{(2)}_{k_1}, \ldots, a^{(3)}_{\alpha+\beta} = a^{(2)}_{j_\beta} a^{(2)}_{k_\beta}$.*

Step 2: from higher-degree tensor relations to quadratic tensor relations. The notion of a set of tensor relations introduced in the previous step can of course be extended to relations of more than 2 vectors. In general, a set of tensor relations $\mathcal{T}$ is a set of tuples of potentially different length; we say that a vector $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_c)$ satisfies a set of tensor relations in $\mathcal{T}$ if for every tuple $(i, j, k, \ldots, z) \in \mathcal{T}$ it holds that $\boldsymbol{\pi}_i = \boldsymbol{\pi}_j \otimes \boldsymbol{\pi}_k \otimes \cdots \otimes \boldsymbol{\pi}_z$. (As before, we require tuples in $\mathcal{T}$ to not form cycles, etc.; see Footnote 9.)

We describe a (zero-knowledge preserving) transformation that maps any LPCP that is secure against $\mathcal{T}$-respecting provers, for some $\mathcal{T}$ containing tensor relations that are *at most cubic*, into a corresponding LPCP that is secure against $\mathcal{T}'$-respecting provers, for a corresponding set of *quadratic* tensor relations $\mathcal{T}'$. (While not needed here, the transformation can be extended to deal with arbitrary sets of tensor relations; see Remark (B.6) below.)

Claim B.5. *There exists an efficient LPCP transformation $\mathbb{Q}$ such that, for any set of at-most-cubic tensor relations $\mathcal{T}$, the following properties hold:*

- Security: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is an LPCP for a relation $\mathcal{R}$ with knowledge error ε against $\mathcal{T}$-respecting provers, then $(\mathcal{T}', P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}}) := \mathbb{Q}(\mathcal{T}, P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is an LPCP for $\mathcal{R}$ with knowledge error ε against $\mathcal{T}'$-respecting provers, where $\mathcal{T}'$ is a set of quadratic tensor relations.*
- Zero-Knowledge Preservation: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is δ-statistical HVZK then so is $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$. (In particular, perfect HVZK is preserved.)*
- Efficiency: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has query length m, k queries, η test polynomials, and degree (d_Q, d_D), then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has query length $O(m)$, k queries, η test polynomials, and degree (d_Q, d_D). Furthermore, if $\mathcal{T}$ has size β then $\mathcal{T}'$ has size $O(\beta)$.*

Proof sketch. The idea is to simply split every cubic tensor into two quadratic tensors. Namely, for every tensor relation $\boldsymbol{\pi}_i = \boldsymbol{\pi}_j \otimes \boldsymbol{\pi}_k \otimes \boldsymbol{\pi}_\ell$ in $\mathcal{T}$, we let $\mathcal{T}'$ require the two tensor relations $\boldsymbol{\pi}_{i'} = \boldsymbol{\pi}_j \otimes \boldsymbol{\pi}_k$ and $\boldsymbol{\pi}_i = \boldsymbol{\pi}_{i'} \otimes \boldsymbol{\pi}_\ell$, for some new index i'. With this modification the length of the proof at most doubles and, moreover, the size of $\mathcal{T}'$ is at most $2\beta = O(\beta)$. □

While it is possible to test cubic tensor relations "directly", in a manner that is analogous to the way we test quadratic tensor relations in the proof of Claim B.3, doing so requires decision predicates of cubic degree, which we wish to avoid. Claim B.5 thus lets us move from cubic tensor relations to quadratic tensor relations before any tensor relations are tested.

Remark B.6. Claim B.5 can be extended to handle any set of tensor relations $\mathcal{T}$, and not only quadratic ones. Concretely, a tensor relation over d vectors can be split into $(d-1)$ quadratic tensor relations. The corresponding parameter changes to the new proof length and size of $\mathcal{T}'$ can be easily deduced.

Step 3: from LPCP to HVZK LPCP against tensor-respecting provers. We describe a transformation that maps any LPCP into a corresponding perfect HVZK LPCP, with similar parameters, that is secure against $\mathcal{T}$-respecting provers for a certain set of at-most-cubic tensor relations $\mathcal{T}$.

Claim B.7. *There exists an efficient LPCP transformation $\mathbb{Z}$ with the following properties:*

- Security: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is an LPCP for a relation $\mathcal{R}$ with knowledge error ε and $d_D = 2$, then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}}) := \mathbb{Z}(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is a perfect HVZK LPCP for $\mathcal{R}$ with knowledge error $\varepsilon + O(1/|\mathbb{F}|)$ against $\mathcal{T}$-respecting provers for some set of at-most-cubic tensor relations $\mathcal{T}$.*
- Efficiency: *If $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has query length m, k queries, state length m', η test polynomials, and degree $(d_Q, 2)$, then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has query length $O(\eta k^4(m+m'))$, $k+\eta+1$ queries, state length $O(k^2 m')$, $\eta+1$ test polynomials, and degree $(d_Q, 2)$. Furthermore, $\mathcal{T}$ has size $\beta = 3\eta$.*

Proof. Recall that, for an input x, we denote by $\boldsymbol{t}_x(u_1, \ldots, u_{m'}, a_1, \ldots, a_k)$ the (quadratic) test polynomial of the decision algorithm $D_{\mathsf{LPCP}}(x, \cdots)$. In general, $\boldsymbol{t}_x$ is a η-dimensional multivariate polynomial; however, to simplify notation, we describe our construction for the case $\eta = 1$ and then, in Remark (B.9), explain how the construction can be extended to the case $\eta > 1$ (and what is the effect on the parameters).

Define the *padding polynomial* (of t_x) to be:

$$\boldsymbol{\Delta}_{t_x}(u_1, \ldots, u_{m'}, a_1, \ldots, a_k, \xi_1, \ldots, \xi_k) = t_x(u_1, \ldots, u_{m'}, a_1 + \xi_1, \ldots, a_k + \xi_k) - t_x(u_1, \ldots, u_{m'}, a_1, \ldots, a_k) \ .$$

Note that $\boldsymbol{\Delta}_{t_x}$ can be expanded to:

$$\boldsymbol{\Delta}_{t_x}(u_1, \ldots, u_{m'}, a_1, \ldots, a_k, \xi_1, \ldots, \xi_k) = \sum_{(i,j) \in [m'] \times [k]} c^{\mathbf{u} \otimes \boldsymbol{\xi}}_{x,i,j} u_i \xi_j + \sum_{(i,j) \in [k] \times [k]} c^{\mathbf{a} \otimes \boldsymbol{\xi}}_{x,i,j} a_i \xi_j + \sum_{(i,j) \in [k] \times [k]} c^{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}_{x,i,j} \xi_i \xi_j \ ,$$

for some coefficients $c^{\mathbf{u} \otimes \boldsymbol{\xi}}_{x,i,j}$, $c^{\mathbf{a} \otimes \boldsymbol{\xi}}_{x,i,j}$, and $c^{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}_{x,i,j}$; we denote the corresponding coefficients vectors by

$$\mathbf{c}^{\mathbf{u} \otimes \boldsymbol{\xi}}_x = \left(c^{\mathbf{u} \otimes \boldsymbol{\xi}}_{x,i,j}\right)_{(i,j) \in [m'] \times [k]}, \mathbf{c}^{\mathbf{a} \otimes \boldsymbol{\xi}}_x = \left(c^{\mathbf{a} \otimes \boldsymbol{\xi}}_{x,i,j}\right)_{(i,j) \in [k] \times [k]}, \mathbf{c}^{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}_x = \left(c^{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}_{x,i,j}\right)_{(i,j) \in [k] \times [k]} \ .$$

Construction B.8. *Construct $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ from $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ as follows:*

- ***The prover P'_{LPCP}.*** *Given $(x, w) \in \mathcal{R}$, P'_{LPCP} invokes $P_{\mathsf{LPCP}}(x, w)$ to obtain a proof $\boldsymbol{\pi} \in \mathbb{F}^m$, samples a random $\boldsymbol{\xi} \in \mathbb{F}^k$, and outputs the proof $\boldsymbol{\pi}'$ defined as follows:*

$$\boldsymbol{\pi}' = \begin{pmatrix} \boldsymbol{\pi} \\ \boldsymbol{\xi} \\ \mathbf{c}^{\mathbf{u} \otimes \boldsymbol{\xi}}_x \\ \mathbf{c}^{\mathbf{a} \otimes \boldsymbol{\xi}}_x \\ \mathbf{c}^{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}_x \\ \boldsymbol{\xi} \otimes \mathbf{c}^{\mathbf{u} \otimes \boldsymbol{\xi}}_x \\ \boldsymbol{\pi} \otimes \boldsymbol{\xi} \otimes \mathbf{c}^{\mathbf{a} \otimes \boldsymbol{\xi}}_x \\ \boldsymbol{\xi} \otimes \boldsymbol{\xi} \otimes \mathbf{c}^{\boldsymbol{\xi} \otimes \boldsymbol{\xi}}_x \end{pmatrix} \ .$$

In other words, $\boldsymbol{\pi}'$ has 8 component vectors, and we can define $\mathcal{T}$ as the set

$$\mathcal{T} := \{(6,2,3), (7,1,2,4), (8,2,2,5)\} \ .$$

- ***The query algorithm*** Q'_{LPCP}***.*** *The query algorithm Q'_{LPCP} invokes Q_{LPCP} to obtain queries $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ and a state $\mathbf{u}$, and then produces new queries $\boldsymbol{q}'_1, \ldots, \boldsymbol{q}'_k, \boldsymbol{q}'_{k+1}, \boldsymbol{q}'_{k+2}$. Specifically:*

  - *For each $j \in [k]$, the prefix of $\boldsymbol{q}'_j$ is $\boldsymbol{q}_j$, and the suffix is a unit vector $\mathbf{e}_j$, which is 1 in the j-th position and zero otherwise. In other words, we construct $\boldsymbol{q}'_j$ so that:*

$$\left\langle \boldsymbol{\pi}', \boldsymbol{q}'_j \right\rangle = \left\langle \boldsymbol{\pi}, \boldsymbol{q}_j \right\rangle + \xi_j \ .$$

  - *The vector $\boldsymbol{q}'_{k+1}$ takes a random combination of the elements $\mathbf{c}_x^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\mathbf{a}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}}$. In other words, $\boldsymbol{q}'_{k+1}$ consists of a random vector $\mathbf{r}$ padded with zeros so that*

$$\left\langle \boldsymbol{\pi}', \boldsymbol{q}'_{k+1} \right\rangle = \left\langle \left(\mathbf{c}_x^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\mathbf{a}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}} \right), \mathbf{r} \right\rangle \ .$$

  - *The vector $\boldsymbol{q}'_{k+2}$ consists of several copies of each of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ and $\mathbf{u}$, and of zeros, so that*

$$\left\langle \boldsymbol{\pi}', \boldsymbol{q}'_{k+2} \right\rangle = \boldsymbol{\Delta}_{t_x}(\mathbf{u}, \langle \boldsymbol{\pi}, \boldsymbol{q}_1 \rangle, \ldots, \langle \boldsymbol{\pi}, \boldsymbol{q}_k \rangle, \boldsymbol{\xi}) = \boldsymbol{\Delta}_{t_x}(\mathbf{u}, \mathbf{a}, \boldsymbol{\xi}) \ .$$

  As for the state, Q'_{LPCP} outputs $\mathbf{u}' = (\mathbf{u}, \mathbf{r})$.

- ***The decision algorithm*** D'_{LPCP}***.*** *Given $\mathbf{u}' = (\mathbf{u}, \mathbf{r})$ and answers $\left(a'_1, \ldots, a'_{k+2}\right)$, D'_{LPCP} checks that $t_x(\mathbf{u}, a'_1, \ldots, a'_k) = a'_{k+2}$ and $a'_{k+1} = \left\langle \left(\mathbf{c}_x^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\mathbf{a}'\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}} \right), \mathbf{r} \right\rangle$, where t_x is the test polynomial of $D_{\mathsf{LPCP}}(x, \cdots)$.*

Remark B.9 (multi-dimensional test polynomials)**.** As mentioned above, to simplify notation, we have defined the padding polynomial and given Construction B.8 for the case $\eta = 1$. For the general case of η-dimensional polynomial $\boldsymbol{t}_x$ with $\eta > 1$, we proceed as follows. Instead of defining a single padding polynomial, we define η padding polynomials $\boldsymbol{\Delta}_{\boldsymbol{t}_x,i}$, each with its own coefficients $\mathbf{c}_i = \left(\mathbf{c}_{x,i}^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_{x,i}^{\mathbf{a}\otimes\boldsymbol{\xi}}, \mathbf{c}_{x,i}^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}} \right)$. Accordingly, the third, fourth, and fifth row of the proof $\boldsymbol{\pi}'$ are extended to η corresponding rows; similarly for the sixth, eight, and tenth row of $\boldsymbol{\pi}'$. The query $\boldsymbol{q}'_{k+1}$ is modified to include η random vectors $\mathbf{r}_1, \ldots, \mathbf{r}_\eta$; indeed, the linear combination checking the correctness of $\mathbf{c}_1, \ldots, \mathbf{c}_\eta$ can be taken simultaneously for all of the $\mathbf{c}_i$ (rather than for each $\mathbf{c}_i$ separately). The last query $\boldsymbol{q}'_{k+2}$ is replaced by η queries $\boldsymbol{q}'_{k+2}, \ldots, \boldsymbol{q}'_{k+\eta+1}$, each according to the appropriate $\boldsymbol{\Delta}_{\boldsymbol{t}_x,i}$.

First, note that the degree of $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ is the same as that of $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$: we have only introduced linear operations to both query and decision polynomials.

Next, let us argue that $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has the claimed knowledge error. Assuming that the vectors $(\mathbf{c}_x^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\mathbf{a}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}})$ appear correctly in the proof $\boldsymbol{\pi}'$ and the prover is $\mathcal{T}$-respecting, then $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ has the exact same knowledge error as in $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$, because then the verifier exactly computes $t_x(\mathbf{u}, \mathbf{a})$. If instead the vectors $(\mathbf{c}_x^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\mathbf{a}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}})$ are not computed properly, then the random linear combination test will fail except with probability $1/|\mathbb{F}|$. Thus, the knowledge error of $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ is at most $\varepsilon + 1/|\mathbb{F}|$.

To see that $(P'_{\mathsf{LPCP}}, V'_{\mathsf{LPCP}})$ is perfect HVZK, note that the honest verifier does not learn from an honest proof any information except for the fact that $\boldsymbol{t}_x(\mathbf{u}, \mathbf{a}) = \boldsymbol{t}_x(\mathbf{u}, \langle \boldsymbol{\pi}, \boldsymbol{q}_1 \rangle, \ldots, \langle \boldsymbol{\pi}, \boldsymbol{q}_k \rangle) = 0$. More formally, the answers $a'_1, \ldots, a'_k$ can all be simulated by random independent elements; the answer a'_{k+1} can be simulated by just taking a random combination of $(\mathbf{c}_x^{\mathbf{u}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\mathbf{a}\otimes\boldsymbol{\xi}}, \mathbf{c}_x^{\boldsymbol{\xi}\otimes\boldsymbol{\xi}})$; and the difference $a'_{k+2} = \boldsymbol{\Delta}_{t_x}(\mathbf{u}, \mathbf{a}, \boldsymbol{\xi})$ can be simulated simply as $\boldsymbol{t}_x(\mathbf{u}, \mathbf{a}')$, where $\mathbf{u}$ is an honestly generated state for the verifier and $\mathbf{a}'$ are the simulated answers $a'_1, \ldots, a'_k$. □

Combining Claim B.7, Claim B.5, and Claim B.3 completes the proof of Theorem B.1.

C Multi-Theorem Designated-Verifier SNARKs via Strong Knowledge

A desirable property of SNARKs is the ability to generate the reference string σ, once and for all, and then reuse it to produce polynomially-many proofs (potentially by different provers). Doing so is especially desirable for preprocessing SNARKs, where generating σ afresh is expensive.

However, being able to securely reuse σ requires security also against provers that *have access to a proof-verification oracle*. For publicly-verifiable SNARKs, this *multi-theorem proof of knowledge* is automatically guaranteed. For designated-verifier SNARKs, however, multi-theorem proof of knowledge needs to be required explicitly as an additional property. Intuitively, this is achieved by ensuring that the verifier's response "leaks" only a negligible amount of information about the verification state τ (for then malicious prover strategies that create a significant correlation between τ and the event of the verifier rejecting are ruled out).[12]

Security against such provers can be formulated for (computational) soundness or proof of knowledge, both in the non-adaptive and adaptive settings. Because in this paper we are typically interested in adaptive proof of knowledge, we formulate it in this setting.

Definition C.1. *A triple of algorithms (G, P, V) is a* **multi-theorem SNARK** *for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if it is a SNARK for $\mathcal{R}$ where adaptive proof of knowledge (Definition 4.3) is replaced by the following stronger requirement:*

- *Multi-theorem adaptive proof of knowledge*

  For every polynomial-size prover P^*, there exists a polynomial-size extractor E such that for every large enough security parameter $\lambda \in \mathbb{N}$, auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and time bound $T \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c|r} V(\tau, y, \pi) = 1 & (\sigma, \tau) \leftarrow G(1^\lambda, T) \\ (y, w) \notin \mathcal{R} & (y, \pi) \leftarrow P^{*\,V(\tau,\cdot,\cdot)}(z, \sigma) \\ & w \leftarrow E(z, \sigma) \end{array}\right] \leq \mathsf{negl}(\lambda) \ .^{13}$$

As discussed in Section 1.3.3, the PCP-based (or MIP-based) SNARKs of [DCL08, Mie08, BCCT12, GLR11, DFH12, BC12] are *not* multi-theorem SNARKs, because a malicious prover can adaptively learn the encrypted PCP (or MIP) queries (whose secrecy is crucial for security), just by feeding different proofs to the verifier and learning his responses. In this paper, some of the designated-verifier preprocessing SNARKs that we construct satisfy multi-theorem proof of knowledge (under suitable assumptions), and some do not.

Concretely, an interesting property that is satisfied by algebraic LIPs, which we call *strong knowledge*, is that such "correlation attacks" are impossible: roughly, every LIP prover either makes the LIP verifier accept with probability 1 or with probability less than $O(\mathrm{poly}(\lambda)/|\mathbb{F}|)$. *Designated-verifier preprocessing SNARKs constructed from such LIPs can thus be shown to have the multi-theorem property.* Details follow.

Definition C.2. *An LPCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ with knowledge error ε has* **strong knowledge error** *if for every input x and every linear function $\boldsymbol{\pi}^*\colon \mathbb{F}^m \to \mathbb{F}$, the probability that $V_{\mathsf{LPCP}}^{\boldsymbol{\pi}^*}(x)$ accepts is either 1 or at most ε. (When we are not paying attention to the knowledge properties of the LPCP, we shall call this property* **strong soundness***.) An analogous definition holds for LIPs.*

[12] Note that $O(\log \lambda)$-theorem soundness always holds; the "non-trivial" case is whenever $\omega(\log \lambda)$. Weaker solutions to support more theorems include simply assuming that the verifier's responses remain secret, or re-generating σ every logarithmically-many rejections, e.g., as in [KR06, GKR08, KR09, GGP10, CKV10].

[13] One can also consider a weaker variant of Definition C.1 where the extractor, just like the prover, has oracle access to the proof-verification oracle $V(\tau, \cdot, \cdot)$. Doing so allows the extractor, for instance, to run the prover.

For sufficiently large field $\mathbb{F}$, algebraic LPCPs and LIPs have the strong knowledge error property, as proved in the following lemma.

Lemma C.3. *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be an LPCP over $\mathbb{F}$ with knowledge error ε; if $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has degree (d_Q, d_D), then $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has strong knowledge error $\max\{\varepsilon, \frac{d_Q d_D}{|\mathbb{F}|}\}$. An analogous statement holds for (input-oblivious two-message) LIPs.*

Proof. Since $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ has knowledge error ε, it also has knowledge error $\max\{\varepsilon, \frac{d_Q d_D}{|\mathbb{F}|}\}$. We are now only left to show that, for every input x and linear function $\boldsymbol{\pi}^*$, if $\Pr[V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x) = 1] > \max\{\varepsilon, \frac{d_Q d_D}{|\mathbb{F}|}\} \geq \frac{d_Q d_D}{|\mathbb{F}|}$ then $\Pr[V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x) = 1] = 1$. Indeed, letting $\boldsymbol{t}_x$ be the test polynomial, $\boldsymbol{p}$ the state polynomial, $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$ the query polynomials of V_{LPCP},

$$\Pr\left[\, V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x) = 1 \,\right] = \Pr_{\boldsymbol{r} \leftarrow \mathbb{F}^\mu}\left[\boldsymbol{t}_x\big(\boldsymbol{p}(\boldsymbol{r}), \langle \boldsymbol{\pi}^*, \boldsymbol{p}_1(\boldsymbol{r})\rangle, \ldots, \langle \boldsymbol{\pi}^*, \boldsymbol{p}_k(\boldsymbol{r})\rangle\big) = 0^\eta\right] = \Pr_{\boldsymbol{r} \leftarrow \mathbb{F}^\mu}\left[\, \mathbf{a}(\boldsymbol{r}) = 0^\eta \,\right] \ ,$$

where $\mathbf{a}$ is the polynomial of degree $d_Q d_D$ defined by $\mathbf{a}(\boldsymbol{r}) := \boldsymbol{t}_x\big(\boldsymbol{p}(\boldsymbol{r}), \langle \boldsymbol{\pi}^*, \boldsymbol{p}_1(\boldsymbol{r})\rangle, \ldots, \langle \boldsymbol{\pi}^*, \boldsymbol{p}_k(\boldsymbol{r})\rangle\big)$. By the Schwartz–Zippel Lemma (cf. Lemma 2.1), if $\Pr[V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x) = 1] > \frac{d_Q d_D}{|\mathbb{F}|}$ then $\mathbf{a} \equiv 0^\eta$ and thus $\Pr[V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(x) = 1] = 1$.

A similar argument proves the analogous statement for LIPs. □

Do LIPs with strong knowledge exist? In Section 3, we presented two types of LIP constructions.

- Both LIPs constructed in Section 3.1 are algebraic (as they are based on algebraic LPCPs) and hence, because of Lemma C.3, do enjoy strong knowledge.
- In contrast, the LIPs constructed Section 3.2 are not algebraic and also do not enjoy strong knowledge (or soundness). The reason is that those LIPs are based on traditional PCPs that do not enjoy strong knowledge (or soundness).

In fact, we now prove that *no* (traditional) PCP (for a hard enough language) can enjoy strong soundness, so that the lack of strong knowledge (or soundness) for the LIPs constructed in Section 3.2 is inherent. Concretely, we show that if a language L has a (traditional) PCP with strong soundness, then it can be decided quite easily.

Definition C.4. *Let $\ell\colon \mathbb{N} \to \mathbb{N}$ be a function. The complexity class $\mathsf{MA}(\ell)$ is the set of languages L for which there exists a probabilistic polynomial-time Turing machine M such that, for every instance x,*

- *if $x \in L$, then there is $y \in \{0,1\}^{\ell(|x|)}$ such that $\Pr[M(x, y) = 1] > 2/3$;*
- *if $x \notin L$, then for every $y \in \{0,1\}^{\ell(|x|)}$ it holds that $\Pr[M(x, y) = 0] > 2/3$.*

Theorem C.5. *Let $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ be a k-query PCP with proof length m for a language L, where k and m are functions of the input size. If $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ has strong soundness error $1/3$, then $L \in \mathsf{MA}(2k(\log m + 1))$.*

Proof. Because $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ has strong soundness error $1/3$, for every x and π, either $\Pr[V^{\pi}_{\mathsf{PCP}}(x) = 1] = 1$ or $\Pr[V^{\pi}_{\mathsf{PCP}}(x) = 1] \leq 1/3$. (See Definition 2.2.) It suffices to show that, for every $x \in L$ and π such that $\Pr[V^{\pi}_{\mathsf{PCP}}(x) = 1] = 1$, there is $S \subseteq [m]$ of size at most $2k$ for which $\Pr[V^{(\pi|_S, \mathbf{1}|_{[m]\setminus S})}_{\mathsf{PCP}}(x) = 1] = 1$, where $(\pi|_S, \mathbf{1}|_{[m]\setminus S})$ is the PCP oracle that is the same as π at the locations in S and is equal to 1 at all other locations. Indeed, to see that the latter is sufficient, consider the MA verifier that, on input x and candidate witness $(S, \pi|_S) \in \{0,1\}^{2k(\log m + 1)}$, runs V_{PCP} with $(\pi|_S, \mathbf{1}|_{[m]\setminus S})$ and accepts if and only if V_{PCP} does;

by the above claim and the completeness of $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$, for any $x \in L$, there is a witness that makes the MA verifier accept with probability 1; furthermore, for every $x \notin L$, the (strong) soundness of $(P_{\mathsf{PCP}}, V_{\mathsf{PCP}})$ implies that the MA verifier accepts with probability at most $1/3$ regardless of the witness.

To prove the aforementioned claim, fix any such $x \in L$ and define $S \subseteq [m]$ to be the set of positions that are queried by V_{PCP} with probability at least $1/2$; by averaging $|S| \leq 2k$. (While S may depend on x, it may not depend on the PCP proof.) Let $\pi \in \{0,1\}^m$ be a PCP oracle for which $\Pr[V_{\mathsf{PCP}}^{\pi}(x) = 1] = 1$. We show that also $\Pr[V_{\mathsf{PCP}}^{(\pi|_S, \mathbf{1}|_{[m]\setminus S})}(x) = 1] = 1$, or else strong soundness is violated. Indeed, assume towards contradiction that this is not the case, then by strong soundness $\Pr[V_{\mathsf{PCP}}^{(\pi|_S, \mathbf{1}|_{[m]\setminus S})}(x) = 1] \leq 1/3$. Consider a sequence of hybrid PCP oracles $\pi_0, \ldots, \pi_{m-|S|}$ defined as follows: starting from $\pi_0 = \pi$, we set the bits outside of S one by one (in any fixed order) to 1, until we get $\pi_{m-|S|} = (\pi|_S, \mathbf{1}|_{[m]\setminus S})$. By strong soundness, there exist $i \in [m - |S|]$ such that $\Pr[V_{\mathsf{PCP}}^{\pi_{i-1}}(x) = 1] = 1$ and $\Pr[V_{\mathsf{PCP}}^{\pi_i}(x) = 1] \leq 1/3$. However, π_{i-1} and π_i only differ on a single position $j \in [m] \setminus S$ that, in particular, is queried with probability at most $1/2$, implying that $\Pr[V_{\mathsf{PCP}}^{\pi_{i-1}}(x) = 1] - \Pr[V_{\mathsf{PCP}}^{\pi_i}(x) = 1] \leq 1/2$, leading to a contradiction. □

From LIPs with strong knowledge to multi-theorem designated-verifier preprocessing SNARKs. At high level, the fact that LIPs with strong knowledge make "correlation attacks" impossible gives strong intuition for why such LIPs should give rise (through our transformation from Section 6.1) to designated-verifier preprocessing SNARKs with the multi-theorem property.

However, formalizing this intuition seems to require a notion of linear-only encryption that is stronger than the one introduced in Section 5.1. Specifically, we need to be able to *repeatedly* extract from a malicious prover in order to simulate its queries to the proof-verification oracle. Doing so raises difficulties similar to the case of plaintext-aware encryption (see [BP04a, BP04b]), and seems to be solvable via "interactive extractability assumptions" [BP04a, BP04b, DFH12].

We now provide a definition of linear-only encryption that, together with strong knowledge, suffices to obtain multi-theorem SNARKs. We thus obtain a confirmation that using strong knowledge is a "conceptually correct" method serving as a good heuristic towards the construction of multi-theorem SNARKs (despite the fact that we need to make fairly strong cryptographic assumptions to formalize this step).

Linear-only homomorphism with interactive extraction. A linear-only encryption scheme with interactive extraction is the same as a (standard) linear-only encryption (Definition 5.4), except that it has a stronger extraction guarantee. Recall that the (standard) linear-only property says that whenever an efficient adversary, given a public key pk and ciphertexts $(c_1, \ldots, c_m)$, produces a ciphertext c' in the image of $\mathsf{Enc}_{\mathsf{pk}}$, there is an efficient extractor that outputs a corresponding affine function "explaining" the ciphertext (or, more accurately, the underlying plaintext) as an affine combination of $(c_1, \ldots, c_m)$ (or, more accurately, their underlying plaintexts). While the standard definition only guarantees "one-time" extraction, the interactive definition gives the adversary the option to interact with the extractor and try to repeatedly sample additional ciphertexts $c'_2, c'_3, \ldots$ given the previously-extracted affine combinations. The definition below is along the same lines as definitions in [BP04a, BP04b, DFH12].

Definition C.6. *An encryption scheme has the* **linear-only property with interactive extraction** *if, for any polynomial-size (interactive) adversary A, there is a polynomial-size (interactive) extractor E such that, for any sufficiently large $\lambda \in \mathbb{N}$, any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any plaintext generator $\mathcal{M}$, A wins the following game with negligible probability:*

1. *Generation step:*

   - $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$;

- $(a_1, \ldots, a_m) \leftarrow \mathcal{M}(\mathsf{pk})$*;*
- $(c_1, \ldots, c_m) \leftarrow (\mathsf{Enc}_{\mathsf{pk}}(a_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_m))$.

2. *For* $i \in \{1, \ldots, |A|\}$*:*

   - $(c'_{i,1}, \ldots, c'_{i,k}) \leftarrow A(\mathsf{pk}, c_1, \ldots, c_m; e_1, \ldots, e_{i-1}; z)$*;*
   - $e_i \leftarrow E(\mathsf{pk}, c_1, \ldots, c_m; i; z)$*, where* e_i *is either an affine function* $(\Pi_i, \boldsymbol{b}_i)$ *or it is* $\perp$.

3. A *wins if there exists some* $i \in [|A|]$*, such that any one of the following holds:*

   - *The extractor fails to identify that* A *outputs an invalid cipher. Namely, there exists* $j \in [k]$ *such that* $\mathsf{ImVer}_{\mathsf{sk}}(c'_{i,j}) \neq 1$ *and* $e_i \neq \perp$.
   - *The extractor fails to produce an affine function that explains the ciphertext produced by* A*. Namely, there exists* $j \in [k]$ *such that* $\mathsf{ImVer}_{\mathsf{sk}}(c'_{i,j}) = 1$ *and one of the following conditions hold: (1)* $e_i = \perp$*, or (ii)* $\mathsf{Dec}_{\mathsf{sk}}(c'_{i,j}) \neq a'_j$ *where* $e_i = (\Pi_i, \boldsymbol{b}_i) \neq \perp$ *and* $(a'_{i,1}, \ldots, a'_{i,k})^\top \leftarrow \Pi_i \cdot (a_1, \ldots, a_m)^\top + \boldsymbol{b}_i$.

Remark C.7 (instantiations)**.** While Definition C.6 seems stronger than Definition 5.4, all the instantiations described in Section 5.3 are plausible candidates for satisfying it. (In particular, [BP04a, BP04b, DFH12] considered "knowledge of exponent assumptions" satisfying a similar requirement.)

We now show that in Lemma 6.2, provided that the linear-only encryption satisfies Definition C.6 and the LIP has strong knowledge error, we obtain a multi-theorem SNARK (again through Construction 6.1).

Lemma C.8. *Suppose that the LIP* $(P_{\mathsf{LIP}}, V_{\mathsf{LIP}})$ *has strong knowledge error* $\mathrm{poly}(\lambda)/|\mathbb{F}|$ *and* $\mathcal{E}$ *is a linear-only encryption scheme with interactive extraction (Definition C.6). Then,* (G, P, V) *from Construction 6.1 is a multi-theorem designated-verifier preprocessing SNARK.*

Proof sketch. We show that any polynomial-size adversary A that can access the proof-verification oracle $V(\tau, \cdot, \cdot)$ can be transformed into a new polynomial-size adversary A' that cannot access $V(\tau, \cdot, \cdot)$ such that, except with negligible probability, the output of A' is equal to the (final) output of A. The lemma then follows from Lemma 6.2.

First, we use A to define a new interactive adversary I_A that (following the template of Definition C.6) works as follows. Given a public key pk and ciphertexts $(c_1, \ldots, c_m)$, I_A runs A and simulates the proof-verification oracle $V(\tau, \cdot, \cdot)$ for A. Specifically, when A outputs the first query (y_1, π_1) to $V(\tau, \cdot, \cdot)$, I_A proceeds as follows:

1. I_A outputs the tuple of ciphers π_1 and then receives e_1 from the extractor;
2. if $e_1 = \perp$, I_A answers A's query (y_1, π_1) with 0;
3. otherwise, e_i is an affine function $(\Pi_1, \boldsymbol{b}_1)$, and I_A runs the LIP verification procedure using fresh coins; namely, it samples $(\mathbf{u}, \boldsymbol{q}) \leftarrow Q_{\mathsf{LIP}}$, computes $d_1 = D_{\mathsf{LIP}}(\mathbf{u}, \Pi_1 \cdot \boldsymbol{q}^\top + \boldsymbol{b}_1)$, and then answers A's query (y_1, π_1) with d_1. (Note that $\mathbf{u}$ is sampled *independently* of the state contained in the verification state τ, which is unknown to I_A.)

Then, I_A continues to run A, each time simulating the answers of $V(\tau, \cdot, \cdot)$ the same way it simulated its first answer. Finally, when A produces its final output (y_f, π_f), I_A also outputs (y_f, π_f). By the interactive extraction guarantee (see Definition C.6), I_A has a corresponding polynomial-size extractor E such that, when I_A and E play the extraction game, I_A wins with negligible probability.[14]

Next, we use I_A and E to define a new (non-interactive) adversary A' that works as follows. Given a public key pk and ciphertexts $(c_1, \ldots, c_m)$, A' runs the extraction game between I_A and E, and then outputs the final output (y_f, π_f) of I_A. Note that A' does not require access to $V(\tau, \cdot, \cdot)$.

We claim that, except with negligible probability, the output of A' is equal to the (final) output of A. To show this, it suffices to argue that, except with negligible probability, the answers provided by I_A to A and those provided by $V(\tau, \cdot, \cdot)$ are equal. And indeed:

- Whenever A makes a query (y_i, π_i) where π_i contains a ciphertext c' such that $\mathsf{ImVer}_{\mathsf{sk}}(c') \neq 1$, $V(\tau, \cdot, \cdot)$ returns 0. In such a case, the extractor E outputs $\perp$, and so I_A answers A's query with 0.

- For any other query (y_i, π_i) of A, except with negligible probability, E outputs $(\Pi_i, \boldsymbol{b}_i)$ that explains A's output; i.e., $\mathsf{Dec}_{\mathsf{sk}}(c'_{i,j}) = a'_j$ for all $j \in [k]$, where $(a'_{i,1}, \ldots, a'_{i,k})^\top \leftarrow \Pi_i \cdot (a_1, \ldots, a_m)^\top + \boldsymbol{b}_i$. We now consider two cases.

  The first case is where $(\Pi_i, \boldsymbol{b}_i)$ convinces the LIP verifier with probability 1; in this case, both $V(\tau, \cdot, \cdot)$ and I_A return 1.

  The second case is where $(\Pi_i, \boldsymbol{b}_i)$ does not always convince the LIP verifier; in particular, by the strong knowledge guarantee, $(\Pi_i, \boldsymbol{b}_i)$ convinces the LIP verifier with only negligible probability. We argue that, in this case, except with negligible probability, both $V(\tau, \cdot, \cdot)$ and I_A return 0. This clearly holds for I_A, because it samples fresh queries and state from Q_{LIP}. The fact that this is also the case for $V(\tau, \cdot, \cdot)$ follows from semantic security. For, if this were not the case, I_A and E could be used to produce an affine function that causes $V(\tau, \cdot, \cdot)$ to accept and yet does not satisfy all but a negligible fraction of $(\mathbf{u}, \boldsymbol{q}) \in Q_{\mathsf{LIP}}$; this would allow to efficiently distinguish this encrypted LIP query from an encrypted random LIP query.

The proof sketch of the lemma is now complete. □

[14] More precisely, we should invoke the interactive extraction guarantee on the interactive adversary I'_A that is equal to I_A except that it outputs only π_f instead of (y_f, π_f).

References

[ABOR00] William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, ICALP '00, pages 463–474, 2000.

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *Proceedings of the 4th Theory of Cryptography Conference*, TCC '07, pages 118–136, 2007.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, ICALP '10, pages 152–163, 2010.

[ALM+98a] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92.

[ALM+98b] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS '92.

[AV77] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *Proceedings on 9th Annual ACM Symposium on Theory of Computing*, STOC '77, pages 30–41, 1977.

[BC12] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *Proceedings of the 32nd Annual International Cryptology Conference*, CRYPTO '12, pages 255–272, 2012.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 326–349, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, pages 111–120, 2013.

[Ben94] Josh Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.

[Ber70] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008. Preliminary version appeared in CCC '02.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *Proceedings of the 31st Annual International Cryptology Conference*, CRYPTO '11, pages 111–131, 2011.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[BO81] Michael Ben-Or. Probabilistic algorithms in finite fields. In *Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '81, pages 394–398, 1981.

[BP04a] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Proceedings of the 24th Annual International Cryptology Conference*, CRYPTO '04, pages 273–289, 2004.

[BP04b] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *Proceedings of the 10th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT '04, pages 48–62, 2004.

[BSCGT13a] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*, ITCS '13, pages 401–414, 2013.

[BSCGT13b] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, 2013.

[BSGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 1–10, 2004.

[BSGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC '05, pages 120–134, 2005.

[BSHLM09] Eli Ben-Sasson, Prahladh Harsha, Oded Lachish, and Arie Matsliah. Sound 3-query PCPPs are long. *ACM Transactions on Computation Theory*, 1(2):7:1–7:49, 2009. Preliminary version appeared in ICALP '08.

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008. Preliminary version appeared in STOC '05.

[BSSVW03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC '03, pages 612–621, 2003.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: Homomorphic encryption for restricted computations. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 350–366, 2012.

[BV07] Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 41–51, 2007.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, FOCS '11, 2011.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO '10, pages 483–501, 2010.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112, 2012.

[CR72] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing*, STOC '72, pages 73–80, 1972.

[CRR11] Ran Canetti, Ben Riva, and Guy N. Rothblum. Two 1-round protocols for delegation of computation. Cryptology ePrint Archive, Report 2011/518, 2011.

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment*, 5(1):25–36, 2011.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Proceedings of the 11th Annual International Cryptology Conference*, CRYPTO '92, pages 445–456, 1992.

[DCL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP proofs from an extractability assumption. In *Proceedings of the 4th Conference on Computability in Europe*, CiE '08, pages 175–185, 2008.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 54–74, 2012.

[DFK+92] Cynthia Dwork, Uriel Feige, Joe Kilian, Moni Naor, and Shmuel Safra. Low communication 2-prover zero-knowledge proofs for NP. In *Proceedings of the 11th Annual International Cryptology Conference*, CRYPTO '92, pages 215–227, 1992.

[DGW09] Zeev Dvir, Ariel Gabizon, and Avi Wigderson. Extractors and rank extractors for polynomial sources. *Computational Complexity*, 18(1):1–58, 2009.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[DLN+04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct NP proofs and spooky interactions, December 2004. Available at `www.openu.ac.il/home/mikel/papers/spooky.ps`.

[EG85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[FG12] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. Cryptology ePrint Archive, Report 2012/281, 2012.

[FGL+96] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in FOCS '91.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO '87, pages 186–194, 1987.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO '10, pages 465–482, 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '13, pages 626–645, 2013.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for Muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, 2008.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456, 2011.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC '85.

[GR05] Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 407–418, 2005.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT '10, pages 321–340, 2010.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, 53:558–655, July 2006. Preliminary version in STOC '02.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1/2):1–53, 2002.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, 2011.

[HK05] Johan Håstad and Subhash Khot. Query efficient PCPs with perfect completeness. *Theory of Computing*, 1(1):119–148, 2005.

[HS00] Prahladh Harsha and Madhu Sudan. Small PCPs with low query complexity. *Computational Complexity*, 9(3–4):157–201, Dec 2000. Preliminary version in STACS '91.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *Proceedings of the 18th Annual International Cryptology Conference*, CRYPTO '98, pages 408–423, 1998.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 278–291, 2007.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, 1992.

[KR06] Yael Tauman Kalai and Ran Raz. Succinct non-interactive zero-knowledge proofs with preprocessing for LOGSNP. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 355–366, 2006.

[KR08] Yael Kalai and Ran Raz. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP '08, pages 536–547, 2008.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Proceedings of the 29th Annual International Cryptology Conference*, CCC '09, pages 143–159, 2009.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 169–189, 2012.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '10, pages 1–23, 2010.

[Mei12] Or Meir. Combinatorial PCPs with short proofs. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity*, CCC '12, 2012.

[MH78] Ralph C. Merkle and Martin E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, Sep 1978.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version appeared in FOCS '94.

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

[MR08] Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *Journal of the ACM*, 57:1–29, June 2008. Preliminary version appeared in FOCS '08.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, CRYPTO '03, pages 96–109, 2003.

[NN90] Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, pages 213–223, 1990.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference On Theory And Application Of Cryptographic Techniques*, EUROCRYPT '99, pages 223–238, 1999.

[PQ12] Christophe Petit and Jean-Jacques Quisquater. On polynomial systems arising from a Weil descent. In *Proceedings of the 18th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT '12, 2012.

[PS94] Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC '94, pages 194–203, 1994.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, 2005.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Proceedings of the 8th Theory of Cryptography Conference*, TCC '11, pages 219–234, 2011.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, STOC '97, pages 475–484, 1997.

[SBV+12] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J. Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. Cryptology ePrint Archive, Report 2012/622, 2012.

[SBW11] Srinath Setty, Andrew J. Blumberg, and Michael Walfish. Toward practical and unconditional verification of remote computations. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, HotOS '13, pages 29–29, 2011.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[SMBW12] Srinath Setty, Michael McPherson, Andrew J. Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *Proceedings of the 2012 Network and Distributed System Security Symposium*, NDSS '12, pages ???–???, 2012.

[SVP+12] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J. Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *Proceedings of the 21st USENIX Security Symposium*, Security '12, pages 253–268, 2012.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical Foundations of Computer Science*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. 1977.

[VZGP01] Joachim Von Zur Gathen and Daniel Panario. Factoring polynomials over finite fields: a survey. *Journal of Symbolic Computation*, 31(1-2):3–17, Jan 2001.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, ICALP '05, pages 140–152, 2005.