Quasi-Optimal SNARGs via Linear Multi-Prover Interactive Proofs*

Dan Boneh[1,4], Yuval Ishai[2,3,4], Amit Sahai[3,4], and David J. Wu[1,4]

[1] Stanford University
[2] Technion
[3] UCLA
[4] Center for Encrypted Functionalities

Abstract. Succinct non-interactive arguments (SNARGs) enable verifying NP computations with significantly less complexity than that required for classical NP verification. In this work, we focus on simultaneously minimizing the proof size and the prover complexity of SNARGs. Concretely, for a security parameter λ, we measure the asymptotic cost of achieving soundness error $2^{-\lambda}$ against provers of size 2^{λ}. We say a SNARG is *quasi-optimally succinct* if its proof length is $\widetilde{O}(\lambda)$, and that it is *quasi-optimal*, if moreover, its prover complexity is only polylogarithmically greater than the running time of the classical NP prover. We show that this definition is the best we could hope for assuming that NP does not have succinct proofs. Our definition strictly strengthens the previous notion of quasi-optimality introduced in the work of Boneh et al. (Eurocrypt 2017).

This work gives the first quasi-optimal SNARG for Boolean circuit satisfiability from a concrete cryptographic assumption. Our construction takes a two-step approach. The first is an information-theoretic construction of a quasi-optimal linear multi-prover interactive proof (linear MIP) for circuit satisfiability. Then, we describe a generic cryptographic compiler that transforms our quasi-optimal linear MIP into a quasi-optimal SNARG by relying on the notion of linear-only vector encryption over rings introduced by Boneh et al. Combining these two primitives yields the first quasi-optimal SNARG based on linear-only vector encryption. Moreover, our linear MIP construction leverages a new *robust* circuit decomposition primitive that allows us to decompose a circuit satisfiability instance into several smaller circuit satisfiability instances. This primitive may be of independent interest.

Finally, we consider (designated-verifier) SNARGs that provide *optimal* succinctness for a non-negligible soundness error. Concretely, we put forward the notion of "1-bit SNARGs" that achieve soundness error 1/2 with only one bit of proof. We first show how to build 1-bit SNARGs from indistinguishability obfuscation, and then show that 1-bit SNARGs also suffice for realizing a form of witness encryption. The latter result highlights a two-way connection between the soundness of very succinct argument systems and powerful forms of encryption.

*The full version of this paper is available at `https://eprint.iacr.org/2018/133.pdf`.

1 Introduction

Proof systems are fundamental to modern cryptography. Many works over the last few decades have explored different aspects of proof systems, including interactive proofs [35, 48, 56], zero-knowledge proofs [35], probabilistically checkable proofs [3, 26, 2], and computationally sound proofs [44, 49]. In this work, we study one such aspect: NP proof systems where the proofs can be significantly shorter than the NP witness and can be verified much faster than the time needed to check the NP witness. We say that such proof systems are *succinct*.

In interactive proof systems for NP with statistical soundness, non-trivial savings in communication and verification time are highly unlikely [16, 32, 33, 65]. However, if we relax the requirements and consider proof systems with computational soundness, also known as *argument systems* [17], significant efficiency improvements become possible. Kilian [44] gave the first succinct four-round interactive argument system for NP based on collision-resistant hash functions and probabilistically checkable proofs (PCPs). Subsequently, Micali [49] showed how to convert Kilian's four-round argument into a single-round argument for NP by applying the Fiat-Shamir heuristic [27] to Kilian's interactive protocol. Micali's "computationally-sound proofs" (CS proofs) represents the first candidate construction of a *succinct non-interactive argument* (that is, a "SNARG" [30]). In the standard model, single-round succinct arguments are highly unlikely for sufficiently hard languages [4, 65], so we consider the weaker goal of two-message succinct arguments systems where the initial message from the verifier is independent of the statement being verified. We refer to this message as the common reference string (CRS).

In this work, we focus on simultaneously minimizing both the proof size and the prover complexity of succinct non-interactive arguments. For a security parameter λ, we measure the asymptotic cost of achieving soundness against provers of size 2^λ with $2^{-\lambda}$ error. We say that a SNARG is *quasi-optimally succinct* if its proof length is $\widetilde{O}(\lambda)$, and that it is *quasi-optimal* if in addition, the prover's runtime is only polylogarithmically greater than the the running time of the classical prover. In Section 5.1, we show that this notion of quasi-optimal succinctness is tight (up to polylogarithmic factors): assuming NP does not have succinct proofs, no succinct argument system can provide the same soundness guarantees with proofs of size $o(\lambda)$. Our notion of quasi-optimality is a strict strengthening of the previous notion from [14], which imposed a weaker soundness requirement on the SNARG. Notably, under the definition in [14], we show that it is possible to construct SNARGs with even shorter proofs than what they consider to be (quasi)-optimally succinct. We discuss the differences in these notions of quasi-optimality in Section 1.1 as well as the full version of this paper [15].

In this paper, we construct the first quasi-optimal SNARG whose security is based on a concrete cryptographic assumption similar in flavor to those of previous works [13, 14]. To our knowledge, all previous candidates are either not quasi-optimal or rely on a heuristic security argument. Similar to previous works [13, 14], we take a two-step approach to construct our quasi-optimal SNARGs. First, we construct an information-theoretic proof system that provides soundness

against a restricted class of provers (e.g., *linearly*-bounded provers [41]). We then leverage cryptographic tools (e.g., *linear-only* encryption [13, 14]) to compile the information-theoretic primitive into a succinct argument system. In this work, the core information-theoretic primitive we use is a linear multi-prover interactive proof (linear MIP). One of the main contributions in this work is a new construction of a quasi-optimal linear MIP that can be compiled to a quasi-optimal SNARG using similar cryptographic tools as those in [14]. We give an overview of our quasi-optimal linear MIP construction in Section 2, and the formal construction in Section 4.

Background on SNARGs. We briefly introduce several properties of succinct non-interactive argument systems. In this work, we focus on constructing SNARGs for the problem of Boolean circuit satisfiability. (This suffices for building SNARGs for general RAM computations, cf. [13].) A SNARG is *publicly verifiable* if anyone can verify the proofs, and it is *designated-verifier* if only the holder of a secret verification state (generated along with the CRS) can verify proofs. In this work, we focus on constructing quasi-optimal designated-verifier SNARGs. In addition, we say a SNARG is fully succinct if the setup algorithm (i.e., the algorithm that generates the CRS, and in the designated-verifier setting, the secret verification state), is also efficient (i.e., runs in time that is only polylogarithmic in the circuit size). A weaker notion is the concept of a *preprocessing SNARG*, where the setup algorithm is allowed to run in time that is polynomial in the size of the circuit being verified. In this work, we consider preprocessing SNARGs. We provide additional background on SNARGs and other related work in Section 1.3.

1.1 Quasi-Optimal SNARGs

In this section, we summarize the main results of this work on defining and constructing quasi-optimal SNARGs. In Section 2, we provide a more technical survey of our main techniques.

Defining quasi-optimality. In this work, we are interested in minimizing the prover complexity and proof size in succinct non-interactive argument systems. To reiterate, our definition of quasi-optimality considers the prover complexity and proof size needed to ensure soundness error $2^{-\lambda}$ against provers of size 2^{λ}. We say a SNARG (for Boolean circuit satisfiability) is quasi-optimal if the proof size is $\widetilde{O}(\lambda)$ and the prover complexity is $\widetilde{O}(|C|) + \mathrm{poly}(\lambda, \log |C|)$, where C is the Boolean circuit.[1] In Lemma 5.2, we show that this notion of quasi-optimality is the "right" one in the following sense: assuming NP does not have succinct *proofs*, the length of any succinct argument system that provides this soundness guarantee is necessarily $\Omega(\lambda)$. Thus, SNARG systems with strictly better parameters are unlikely to exist. Our notion is a strict strengthening of the previous notion of quasi-optimality from [14] which only required soundness error $\mathrm{negl}(\lambda)$ against

[1] We write $\widetilde{O}(\cdot)$ to suppress factors that are *polylogarithmic* in the circuit size $|C|$ and the security parameter λ.

provers of size 2^λ. In fact, we show in the full version [15] that the previous notion of quasi-optimality from [14] is not tight. If we only want ρ bits of soundness where $\rho = o(\lambda)$, it is possible to construct a designated-verifier SNARG where the proofs are exactly ρ bits. This means that there exists a designated-verifier SNARG which meet the soundness requirements in [14], but whose size is strictly shorter than what would be considered "optimal."

Previous SNARG constructions. Prior to this work, the only SNARG candidate that satisfies our notion of quasi-optimal prover complexity is Micali's CS proofs [49]. However, to achieve $2^{-\lambda}$ soundness, the length of a CS proof is $\Omega(\lambda^2)$, which does not satisfy our notion of quasi-optimal succinctness. Conversely, if we just consider SNARGs that provide quasi-optimal succinctness, we have many candidates [37, 45, 29, 13, 46, 24, 38, 14]. With the exception of [14], the SNARG proof in all of these candidates contains a constant number of bilinear group elements, and so, is quasi-optimally succinct. The drawback is that to construct the proof, the prover has to perform a group operation for every gate in the underlying circuit. Since each group element is $\Omega(\lambda)$ bits, the prover overhead is at least multiplicative in λ. Consequently, none of these existing constructions satisfy our notion of quasi-optimal prover complexity. The lattice-based construction in [14] has the same limitation: the prover needs to operate on an LWE ciphertext per gate in the circuit, which introduces a multiplicative overhead $\Omega(\lambda)$ in the prover's computational cost.

Quasi-optimal linear MIPs. This work gives the first construction of a quasi-optimal SNARG for Boolean circuit satisfiability from a concrete cryptographic assumption. Following previous works on constructing SNARGs [13, 14], our construction can be broken down into two components: an information-theoretic component (linear MIPs), and a cryptographic component (linear-only vector encryption). We give a brief description of the information-theoretic primitive we construct in this work: a *quasi-optimal* linear MIP. At the end of this section, we discuss why the general PCPs and linear PCPs that have featured in previous SNARG constructions do not seem sufficient for building quasi-optimal SNARGs.

We first review the notion of a linear PCP [41, 13]. A linear PCP over a finite field $\mathbb{F}$ is an oracle computing a linear function $\boldsymbol{\pi} \colon \mathbb{F}^m \to \mathbb{F}$. On any query $\mathbf{q} \in \mathbb{F}^m$, the linear PCP oracle responds with the inner product $\mathbf{q}^\top \boldsymbol{\pi} = \langle \mathbf{q}, \boldsymbol{\pi} \rangle \in \mathbb{F}$. More generally, if ℓ queries are made to the linear PCP oracle, the ℓ queries can be packed into the columns of a query matrix $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$. In this case, we can express the response of the linear PCP oracle as the matrix-vector product $\mathbf{Q}^\top \boldsymbol{\pi}$.

Linear MIPs are a direct generalization of linear PCPs to the setting where there are ℓ independent proof oracles $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell)$, each implementing a linear function $\boldsymbol{\pi}_i \colon \mathbb{F}^m \to \mathbb{F}$. In the linear MIP model, the verifier's queries consist of a ℓ-tuple $(\mathbf{q}_1, \ldots, \mathbf{q}_\ell)$ where each $\mathbf{q}_i \in \mathbb{F}^m$. For each query $\mathbf{q}_i \in \mathbb{F}^m$ to the proof oracle $\boldsymbol{\pi}_i$, the verifier receives the response $\langle \mathbf{q}_i, \boldsymbol{\pi}_i \rangle$. We review the formal definitions of linear PCPs and linear MIPs in the full version [15].

In this work, we say that a linear MIP for Boolean circuit satisfiability is quasi-optimal if the MIP prover (for proving satisfiability of a circuit C) can be

implemented by a circuit of size $\widetilde{O}(|C|) + \mathrm{poly}(\lambda, \log |C|)$, and the linear MIP provides soundness error $2^{-\lambda}$. Existing linear PCP constructions [13, 14] (which can be viewed as linear MIPs with a single prover) are not quasi-optimal: they either require embedding the Boolean circuit into an arithmetic circuit over a large field [13], or rely on making $O(\lambda)$ queries, each of length $m = O(|C|)$ [14].

Constructing quasi-optimal linear MIPs. Our work gives the first construction of a quasi-optimal linear MIP for Boolean circuit satisfiability. We refer to Section 2 for an overview of our construction and to Section 4 for the full description. At a high-level, our quasi-optimal linear MIP construction relies on two key ingredients: a robust circuit decomposition and a method for enforcing consistency.

Robust circuit decomposition. Our robust decomposition primitive takes a circuit C and produces from it a collection of constraints $f_1, \ldots, f_t$, each of which can be computed by a circuit of size roughly $|C|/t$. Each constraint reads a subset of the bits of a global witness (computed based on the statement-witness pair for C). The guarantee provided by the robust decomposition is that for any false statement $\mathbf{x}$ (that is, a statement $\mathbf{x}$ where for all witnesses $\mathbf{w}$, $C(\mathbf{x}, \mathbf{w}) = 0$), no single witness to $f_1, \ldots, f_t$ can simultaneously satisfy more than a *constant fraction* of the constraints. Now, to prove satisfiability of a circuit C, the prover instead proves that there is a consistent witness that simultaneously satisfies all of the constraints $f_1, \ldots, f_t$. Each of these proofs can be implemented by a standard linear PCP. The advantage of this approach is that for a false statement, only a constant fraction of the constraints can be satisfied (for any choice of witness), so even if each underlying linear PCP instance only provided *constant* soundness, the probability that the prover is able to satisfy *all* of the instances is amplified to $2^{-\Omega(t)} = 2^{-\Omega(\lambda)}$ if we let $t = \Theta(\lambda)$. Finally, even though the prover now has to construct t proofs for the t constraints, each of the constraints can themselves be computed by a circuit of size $\widetilde{O}(|C|/t)$. The robustness property of our decomposition is reminiscent of the relation between traditional PCPs and constraint-satisfaction problems, and one might expect that we could instantiate such a decomposition using PCPs. However, in our settings, we require that the decomposition be *input-independent*, which to the best of our knowledge, is not satisfied by existing (quasilinear) PCP constructions. We discuss this in more detail in the full version [15].

The robust decomposition can amplify soundness without introducing much additional overhead. The alternative approach of directly applying a constant-query linear PCP to check satisfiability of C has the drawback of only providing $1/\mathrm{poly}(\lambda)$ soundness when working over a small field (i.e., as would be the case with Boolean circuit satisfiability). We state the formal requirements of our robust decomposition in Section 4.1, and give one instantiation in the full version by combining MPC protocols with polylogarithmic overhead [23] with the "MPC-in-the-head" paradigm [42]. Since the notion of a robust decomposition is a very natural one, we believe that our construction is of independent interest and will have applications beyond quasi-optimal linear MIP constructions.

Enforcing consistency. The second ingredient we require is a way for the verifier to check that the individual proofs the prover constructs (for showing satisfiability of each constraint $f_1, \ldots, f_t$) are self-consistent. Our construction here relies on constructing randomized permutation decompositions, and we refer to Section 2 for the technical overview, and Section 4 for the full description.

Preprocessing SNARGs from linear MIPs. To complete our construction of quasi-optimal SNARGs, we show a generic compiler from linear MIPs to preprocessing SNARGs by relying on the notion of a linear-only vector encryption scheme over rings introduced by Boneh et al. [14]. We give our construction in Section 5. Our primary contribution here is recasting the Boneh et al. construction, which satisfies the weaker notion of quasi-optimality, as a generic framework for compiling linear MIPs into preprocessing SNARGs. Combined with our information-theoretic construction of quasi-optimal linear MIPs, this yields the first quasi-optimal designated-verifier SNARG for Boolean circuit satisfiability in the preprocessing model (Corollaries 5.6 and 5.7).

Why linear MIPs? A natural question to ask is whether our new linear MIP to preprocessing SNARG compiler provides any advantage over the existing compilers in [13, 14], which use different information-theoretic primitives as the underlying building block (namely, linear interactive proofs [13] and linear PCPs [14]). After all, any k-query, ℓ-prover linear MIP with query length m can be transformed into a $(k\ell)$-query linear PCP with query length $m\ell$ by concatenating the proofs of the different provers together, and likewise, padding the queries accordingly. While this still yields a quasi-optimal linear PCP (with sparse queries), applying the existing cryptographic compilers to this linear PCP incurs an additional prover overhead that is proportional to ℓ. In our settings, $\ell = \Theta(\lambda)$, so the resulting SNARG is no longer quasi-optimal. By directly compiling linear MIPs to preprocessing SNARGs, our compiler *preserves* the prover complexity of the underlying linear MIP, and so, combined with our quasi-optimal linear MIP construction, yields a quasi-optimal SNARG for Boolean circuit satisfiability.

Alternatively, one might ask whether a similar construction of quasi-optimal SNARGs is possible starting from standard PCPs or linear PCPs with quasi-optimal prover complexity. Existing techniques for compiling general PCPs [49, 10, 9] to succinct argument systems all rely on some form of cryptographic hashing to commit to the proof and then open up a small number of bits chosen by the verifier. In the random oracle model [49], this kind of construction achieves quasi-optimal prover complexity, but not quasi-optimal succinctness [14, Remark 4.16]. In the standard model [11, 9], additional cryptographic tools (notably, a private information retrieval protocol) are needed in the construction, which do not preserve the prover complexity of the underlying construction.

If instead we start with linear PCPs and apply the compilers in [13, 14], the challenge is in constructing a quasi-optimal linear PCP that provides soundness error $2^{-\lambda}$ over a small field $\mathbb{F}$. As noted above, existing linear PCP constructions [13, 14] are not quasi-optimal for Boolean circuit satisfiability.

1.2 Optimally-Laconic Arguments and 1-Bit SNARGs

More broadly, we can view our quasi-optimal SNARGs in the preprocessing model as a quasi-optimal *interactive* argument system with a maximally *laconic* prover. Here, we allow the verifier to send an arbitrarily long string (namely, the CRS), and our goal is to minimize the prover's computational cost and the number of bits the prover communicates to the verifier. Our quasi-optimal SNARG thus gives the first interactive argument system with a *quasi-optimal* laconic prover.

Optimally-laconic arguments and 1-bit SNARGs. Independent of our results on constructing quasi-optimal SNARGs, we also ask the question of what is the minimal proof length needed to ensure ρ bits of soundness where ρ is a concrete soundness parameter. Lemma 5.2 shows that achieving $2^{-\rho}$ soundness error only requires proofs of length $\Omega(\rho)$. When $\rho = \Omega(\lambda)$, many existing SNARG candidates, including the one we construct in this paper, are quasi-optimally succinct [37, 29, 13, 14]. More generally, this question remains interesting when $\rho = o(\lambda)$, and even independently of achieving quasi-optimal prover complexity. A natural question to ask is whether there exist SNARGs where the size of the proofs achieves the lower bound of $\Omega(\rho)$ for providing ρ bits of soundness. Taken to the extreme, we ask whether there exists a 1-bit SNARG with soundness error $1/2 + \mathrm{negl}(\lambda)$. We note that a 1-bit SNARG immediately implies an *optimally-succinct* SNARG for all soundness parameters ρ: namely, to build a SNARG with soundness error $2^{-\rho}$, we concatenate ρ independent instances of a 1-bit SNARG.

In the full version [15], we show that the designated-verifier analog of the Sahai-Waters [53] construction of non-interactive zero-knowledge proofs from indistinguishability obfuscation and one-way functions is a 1-bit SNARG. In the *interactive* setting, we show that we can construct 1-bit laconic arguments from witness encryption. We do not know how to build 1-bit SNARGs and 1-bit laconic arguments for general languages from weaker assumptions,[2] and leave this as an open problem.

The power of optimally-laconic arguments. Finally, we show an intriguing connection between 1-bit laconic arguments and a variant of witness encryption. Briefly, a witness encryption scheme [28] allows anyone to encrypt a message m with respect to a statement x in an NP language; then, anyone who holds a witness w for x is able to decrypt the ciphertext. In the full version [15], we show that a 1-bit laconic argument (or SNARG) for a cryptographically-hard[3] language $\mathcal{L}$ implies a relaxed form of witness encryption for $\mathcal{L}$ where semantic security holds for messages encrypted to a *random* false instance (as opposed to an arbitrary false instance in the standard definition). While this is a relaxation of the usual notion of witness encryption, it already suffices to realize some of the

[2] Note that for some special languages such as graph non-isomorphism, we do have 1-bit laconic arguments [31].

[3] Here, we say a language is cryptographically-hard if there exists a distribution over YES instances that is computationally indistinguishable from a distribution of NO instances for the language.

powerful applications of witness encryption described in [28]. This implication thus demonstrates the power of optimally-laconic arguments, as well as some of the potential challenges in constructing them from simple assumptions.

Our construction of witness encryption from 1-bit arguments relies on the observation that for a (random) false statement $\mathbf{x}$, any computationally-bounded prover can only produce a valid proof $\pi \in \{0,1\}$ with probability that is negligibly close to $1/2$. Thus, the proof π can be used to hide the message m in a witness encryption scheme (when encrypting to the statement $\mathbf{x}$). Here, we implicitly assume that a (random) statement $\mathbf{x}$ has exactly one accepting proof—this assumption holds for any cryptographically-hard language. Essentially, our construction shows how to leverage the soundness property of a proof system to obtain a secrecy property in an encryption scheme. Previously, Applebaum et al. [1] showed how to leverage secrecy to obtain soundness, so in some sense, we can view our construction as a dual of their secrecy-to-soundness construction. The recent work of Berman et al. [8] also showed how to obtain public-key encryption from laconic *zero-knowledge* arguments. While their construction relies on the additional assumption of zero-knowledge, their construction does not require the argument system be optimally laconic.

We can also view a 1-bit argument for a cryptographically-hard language as a "predictable argument" (c.f., [25]). A predictable argument is one where there is exactly one accepting proof for any statement. Faonio et al. [25] show that any predictable argument gives a witness encryption scheme. In this work, we show that soundness *alone* suffices for this transformation, provided we make suitable restrictions on the underlying language.

1.3 Additional Related Work

Gentry and Wichs [30] showed that no construction of an *adaptively-secure* SNARG (for general NP languages) can be proven secure via a black-box reduction from any falsifiable cryptographic assumption [51].[4] As a result, most existing SNARG constructions (for general NP languages) in the standard model have relied on non-falsifiable assumptions such as knowledge-of-exponent assumptions [21, 40, 5, 50, 37, 45, 29, 46, 24, 47, 39], extractable collision-resistant hashing [10, 22, 9], extractable homomorphic encryption [12, 29], and linear-only encryption [13, 14]. Other constructions have relied on showing security in idealized models such as the random oracle model [49, 59] or the generic group model [38]. In many of these constructions, the underlying SNARGs also satisfy a knowledge property, which says that whenever a prover generates an accepting proof $\boldsymbol{\pi}$ of a statement $\mathbf{x}$, there is an efficient extractor that can extract a witness $\mathbf{w}$ from $\boldsymbol{\pi}$ such that $C(\mathbf{x}, \mathbf{w}) = 1$. SNARGs with this property are called SNARGs of knowledge, or more commonly, SNARKs. In many cases, SNARGs also have a zero-knowledge property [37, 45, 29, 13, 46, 24, 47, 39] which says that the proof $\boldsymbol{\pi}$

[4] In the case of non-adaptive SNARGs, Sahai and Waters give a construction from indistinguishability obfuscation and one-way functions [53].

does not reveal any additional information about the witness $\mathbf{w}$ other than the fact that $C(\mathbf{x}, \mathbf{w}) = 1$.

A compelling application of succinct argument systems is to verifiable delegation of computation. Over the last few years, there has been significant progress in leveraging SNARGs (and their variants) for implementing scalable systems for verifiable computation both in the interactive setting [34, 19, 58, 54, 55, 57, 60–62] as well as the non-interactive setting [52, 6, 18, 7, 63, 20]. We refer to [64] and the references therein for a more comprehensive survey of this area.

2 Quasi-Optimal Linear MIP Construction Overview

In this section, we give a technical overview of our quasi-optimal linear MIP construction for arithmetic circuit satisfiability over a finite field $\mathbb{F}$. Combined with our cryptographic compiler based on linear-only vector encryption over rings, this gives the first construction of a quasi-optimal SNARG from a concrete cryptographic assumption.

Robust circuit decomposition. The first ingredient we require in our quasi-optimal linear MIP construction is a *robust* way to decompose an arithmetic circuit $C\colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^{h'}$ into a collection of t constraint functions $f_1, \ldots, f_t$, where each constraint $f_i\colon \mathbb{F}^n \times \mathbb{F}^m \to \{0,1\}$ takes as input a common statement $\mathbf{x} \in \mathbb{F}^n$ and witness $\mathbf{w} \in \mathbb{F}^m$. More importantly, each constraint f_i can be computed by a small arithmetic circuit C_i of size roughly $|C|/t$. This means that each arithmetic circuit C_i may only need to read some subset of the components in $\mathbf{x}$ and $\mathbf{w}$. There is a mapping $\mathsf{inp}\colon \mathbb{F}^{n'} \to \mathbb{F}^n$ that takes as input a statement $\mathbf{x}'$ for C and outputs a statement $\mathbf{x}$ for $f_1, \ldots, f_t$, and another mapping $\mathsf{wit}\colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^m$ that takes as input a statement-witness pair $(\mathbf{x}', \mathbf{w}')$ for C, and outputs a witness $\mathbf{w}$ for $f_1, \ldots, f_t$. The decomposition must satisfy two properties: completeness and robustness. Completeness says that whenever a statement-witness pair $(\mathbf{x}', \mathbf{w}')$ is accepted by C, then $f_i(\mathbf{x}, \mathbf{w}) = 1$ for all i if we set $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$ and $\mathbf{w} = \mathsf{wit}(\mathbf{x}', \mathbf{w}')$. Robustness says that for a false statement $\mathbf{x}' \in \mathbb{F}^{n'}$, there are no valid witnesses $\mathbf{w} \in \mathbb{F}^m$ that can simultaneously satisfy more than a constant fraction of the constraints $f_1(\mathbf{x}, \cdot), \ldots, f_t(\mathbf{x}, \cdot)$, where $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$.

Roughly speaking, a robust decomposition allows us to reduce checking satisfiability of a large circuit C to checking satisfiability of many smaller circuits $C_1, \ldots, C_t$. The gain in performance will be due to our ability to check satisfiability of all of the $C_1, \ldots, C_t$ in parallel. The importance of robustness will be critical for soundness amplification. We give the formal definition of a robust decomposition in Section 4.1.

Instantiating the robust decomposition. In the full version [15], we describe one way of instantiating the robust decomposition by applying the "MPC-in-the-head" paradigm of [42] to MPC protocols with polylogarithmic overhead [23]. We give a brief overview here. For an arithmetic circuit $C\colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^{h'}$, the encoding of a statement-witness pair $(\mathbf{x}, \mathbf{w})$ will be the *views* of each party

in a (simulated) t-party MPC protocol computing C on $(\mathbf{x}, \mathbf{w})$, where the bits of the input and witness are evenly distributed across the parties. Each of the constraint functions f_i checks that party i outputs 1 in the protocol execution (indicating an accepting input), and that the view of party i is *consistent* with the views of the other parties. This means that the only bits of the encoded witness that each constraint f_i needs to read are those that correspond to messages that were sent or received by party i. Then, using an MPC protocol where the computation and communication overhead is polylogarithmic in the circuit size (c.f., [23]), and where the computational burden is evenly distributed across the computing parties, each $f_1, \ldots, f_t$ can be implemented by a circuit of size $\widetilde{O}(|C|/t)$. Robustness of the decomposition follows from security of the underlying MPC protocol. We give the complete description and analysis in the full version [15].

Blueprint for linear MIP construction. The high-level idea behind our quasi-optimal linear MIP construction is as follows. We first apply a robust circuit decomposition to the input circuit to obtain a collection of constraints $f_1, \ldots, f_t$, which can be computed by smaller arithmetic circuits $C_1, \ldots, C_t$, respectively. Each arithmetic circuit takes as input a subset of the components of the statement $\mathbf{x} \in \mathbb{F}^n$ and the witness $\mathbf{w} \in \mathbb{F}^m$. In the following, we write $\mathbf{x}_i$ and $\mathbf{w}_i$ to denote the subset of the components of $\mathbf{x}$ and $\mathbf{w}$, respectively, that circuit C_i reads. We can now construct a linear MIP with t provers as follows. A proof of a true statement $\mathbf{x}'$ with witness $\mathbf{w}'$ consists of t proof vectors $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t)$, where each proof $\boldsymbol{\pi}_i$ is a linear PCP proof that $C_i(\mathbf{x}_i, \cdot)$ is satisfiable. Then, in the linear MIP model, the verifier has oracle access to the linear functions $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$, which it can use to check satisfiability of $C_i(\mathbf{x}_i, \cdot)$. Completeness of this construction is immediate from completeness of the robust decomposition.

Soundness is more challenging to argue. For any false statement $\mathbf{x}'$, robustness of the decomposition of C only ensures that for any witness $\mathbf{w} \in \mathbb{F}^m$, at least a constant fraction of the constraints $f_i(\mathbf{x}, \mathbf{w})$ will not be satisfied, where $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$. However, this does *not* imply that a constant fraction of the individual circuits $C_i(\mathbf{x}_i, \cdot)$ is unsatisfiable. For instance, for all i, there could exist some witness $\mathbf{w}_i$ such that $C_i(\mathbf{x}_i, \mathbf{w}_i) = 1$. This does *not* contradict the robustness of the decomposition so long as the set of all satisfying witnesses $\{\mathbf{w}_i\}$ contain many "inconsistent" assignments. More specifically, we can view each $\mathbf{w}_i$ as assigning values to some subset of the components of the overall witness $\mathbf{w}$, and we say that a collection of witnesses $\{\mathbf{w}_i\}$ is consistent if whenever two witnesses $\mathbf{w}_i$ and $\mathbf{w}_j$ assign a value to the same component of $\mathbf{w}$, they assign the *same* value. Thus, robustness only ensures that the prover cannot find a *consistent* set of witnesses $\{\mathbf{w}_i\}$ that can simultaneously satisfy more than a fraction of the circuits C_i. Or equivalently, if $\mathbf{x}$ is the encoding of a false statement $\mathbf{x}'$, then a constant fraction of any set of witnesses $\{\mathbf{w}_i\}$ where $C_i(\mathbf{x}_i, \mathbf{w}_i) = 1$ must be mutually inconsistent.

The above analysis shows that it is insufficient for the prover to independently argue satisfiability of each circuit $C_i(\mathbf{x}_i, \cdot)$. Instead, we need the stronger requirement that the prover uses a *consistent* set of witnesses $\{\mathbf{w}_i\}$ when constructing its proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$. Thus, we need a way to bind each proof $\boldsymbol{\pi}_i$ to a specific witness

$\mathbf{w}_i$, as well as a way for the verifier to check that the complete set of witnesses $\{\mathbf{w}_i\}$ are mutually consistent. For the first requirement, we introduce the notion of a *systematic linear PCP*, which is a linear PCP where the linear PCP proof vector $\boldsymbol{\pi}_i$ contains a copy of a witness $\mathbf{w}_i$ where $C_i(\mathbf{x}_i, \mathbf{w}_i) = 1$ (Definition 4.2). Now, given a collection of systematic linear PCP proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$, the verifier's goal is to decide whether the witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_t$ embedded within $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$ are mutually consistent. Since the witnesses $\mathbf{w}_i$ are part of the proof vectors $\boldsymbol{\pi}_i$, in the remainder of this section, we will simply assume that the verifier has oracle access to the linear function $\langle \mathbf{w}_i, \cdot \rangle$ for all i since such queries can be simulated using the proof oracle $\langle \boldsymbol{\pi}_i, \cdot \rangle$.

2.1 Consistency Checking

The robust decomposition ensures that for a false statement $\mathbf{x}'$, any collection of witnesses $\{\mathbf{w}_i\}$ where $C_i(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all i is guaranteed to have many inconsistencies. In fact, there must always exists $\Omega(t)$ (mutually disjoint) pairs of witnesses that contain some inconsistency in their assignments. Ensuring soundness thus reduces to developing an efficient method for testing whether $\mathbf{w}_1, \ldots, \mathbf{w}_t$ constitute a consistent assignment to the components of $\mathbf{w}$ or not. This is the main technical challenge in constructing quasi-optimal linear MIPs, and our construction proceeds in several steps, which we describe below.

Notation. We begin by introducing some notation. First, we pack the different witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_t \in \mathbb{F}^q$ into the rows of an *assignment matrix* $\mathbf{W} \in \mathbb{F}^{t \times q}$. Specifically, the i^{th} row of $\mathbf{W}$ is the witness $\mathbf{w}_i$. Next, we define the *replication structure* for the circuits $C_1, \ldots, C_t$ to be a matrix $\mathbf{A} \in [m]^{t \times q}$. Here, the $(i,j)^{\text{th}}$ entry $\mathbf{A}_{i,j}$ encodes the index in $\mathbf{w} \in \mathbb{F}^m$ to which the j^{th} entry in $\mathbf{w}_i$ corresponds. With this notation, we say that the collection of witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_t$ are consistent if for all indices (i_1, j_1) and (i_2, j_2) where $\mathbf{A}_{i_1,j_1} = \mathbf{A}_{i_2,j_2}$, the assignment matrix satisfies $\mathbf{W}_{i_1,j_1} = \mathbf{W}_{i_2,j_2}$.

Checking global consistency. To check whether an assignment matrix $\mathbf{W} \in \mathbb{F}^{t \times q}$ is consistent with respect to the replication structure $\mathbf{A} \in [m]^{t \times q}$, we can leverage an idea from Groth [36], and subsequently used in [43, 14] for performing similar kinds of consistency checks. The high-level idea is as follows. Take any index $z \in [m]$ and consider the positions $(i_1, j_1), \ldots, (i_d, j_d)$ where z appears in $\mathbf{A}$. In this way, we associate a disjoint set of Hamiltonian cycles over the entries of $\mathbf{A}$, one for each of the m components of $\mathbf{w}$. Let Π be a permutation over the entries in the matrix $\mathbf{A}$ such that Π splits into a product of the Hamiltonian cycles induced by the entries of $\mathbf{A}$. In particular, this means $\mathbf{A} = \Pi(\mathbf{A})$, and moreover, $\mathbf{W}$ is consistent with respect to $\mathbf{A}$ if and only if $\mathbf{W} = \Pi(\mathbf{W})$. The insight in [36] is that the relation $\mathbf{W} = \Pi(\mathbf{W})$ can be checked using two sets of linear queries. First, the verifier draws vectors $\mathbf{r}_1, \ldots, \mathbf{r}_t \xleftarrow{\text{R}} \mathbb{F}^q$ and defines the matrix $\mathbf{R} \in \mathbb{F}^{t \times q}$ to be the matrix whose rows are $\mathbf{r}_1, \ldots, \mathbf{r}_t$. Next, the verifier computes the permuted matrix $\mathbf{R}' \leftarrow \Pi(\mathbf{R})$. Let $\mathbf{r}'_1, \ldots, \mathbf{r}'_t$ be the rows of $\mathbf{R}'$.

Similarly, let $\mathbf{w}_1, \ldots, \mathbf{w}_t$ be the rows of $\mathbf{W}$. Finally, the verifier queries the linear MIP oracles $\langle \mathbf{w}_i, \cdot \rangle$ on $\mathbf{r}_i$ and $\mathbf{r}'_i$ for all i and checks the relation

$$\sum_{i \in [t]} \langle \mathbf{w}_i, \mathbf{r}_i \rangle \stackrel{?}{=} \sum_{i \in [t]} \langle \mathbf{w}_i, \mathbf{r}'_i \rangle \in \mathbb{F}. \tag{2.1}$$

By construction of Π, if $\mathbf{W} = \Pi(\mathbf{W})$, this check always succeeds. However, if $\mathbf{W} \neq \Pi(\mathbf{W})$, then by the Schwartz-Zippel lemma, this check rejects with probability $1/|\mathbb{F}|$. When working over a polynomial-size field, this consistency check achieves $1/\text{poly}(\lambda)$ soundness (where λ is a security parameter). We can use repeated queries to amplify the soundness to $\text{negl}(\lambda)$ without sacrificing quasi-optimality. However, this approach cannot give a linear MIP with $2^{-\lambda}$ soundness and still retain prover overhead that is only polylogarithmic in λ (since we would require $\Omega(\lambda)$ repetitions). This is one of the key reasons the construction in [14] only achieves $\text{negl}(\lambda)$ soundness rather than $2^{-\lambda}$ soundness. To overcome this problem, we require a more robust consistency checking procedure.

Checking pairwise consistency. The consistency check described above and used in [36, 43, 14] is designed for checking *global* consistency of all of the assignments in $\mathbf{W} \in \mathbb{F}^{t \times q}$. The main disadvantage of performing the global consistency check in Eq. (2.1) is that it only provides soundness $1/|\mathbb{F}|$, which is insufficient when $\mathbb{F}$ is small (e.g., in the case of Boolean circuit satisfiability). One way to amplify soundness is to replace the single global consistency check with $t/2$ *pairwise* consistency checks, where each pairwise consistency check affirms that the assignments in a (mutually disjoint) pair of rows of $\mathbf{W}$ are self-consistent. Specifically, each of the $t/2$ checks consists of two queries $(\mathbf{r}_i, \mathbf{r}_j)$ and $(\mathbf{r}'_i, \mathbf{r}'_j)$ to $\langle \mathbf{w}_i, \cdot \rangle$ and $\langle \mathbf{w}_j, \cdot \rangle$, constructed in exactly the same manner as in the global consistency check, except specialized to only checking for consistency in the assignments to the variables in rows i and j. Since all of the pairwise consistency checks are independent, if there are $\Omega(t)$ pairs of inconsistent rows, the probability that all $t/2$ checks pass is bounded by $2^{-\Omega(t)}$. This means that for the same cost as performing a *single* global consistency check, the verifier can perform $\Omega(t)$ pairwise consistency checks. As long as many of the pairs of rows the verifier checks contain inconsistencies, we achieve soundness amplification.

Recall from earlier that our robust decomposition guarantees that whenever $\mathbf{x}_1, \ldots, \mathbf{x}_t$ correspond to a false statement, any collection of witnesses $\{\mathbf{w}_i\}$ where $C_i(\mathbf{x}_i, \mathbf{w}_i)$ is satisfied for all i necessarily contains many pairs $\mathbf{w}_i$ and $\mathbf{w}_j$ that are inconsistent. Equivalently, many pairs of rows in the assignment matrix $\mathbf{W}$ contain inconsistencies. Now, if the verifier knew which pairs of rows of $\mathbf{W}$ are inconsistent, then the verifier can apply a pairwise consistency check to detect an inconsistent $\mathbf{W}$ with high probability. The problem, however, is that the verifier does not know *a priori* which pairs of rows in $\mathbf{W}$ are inconsistent, and so, it is unclear how to choose the rows to check in the pairwise consistency test. However, if we make the stronger assumption that not only are there many pairs of rows in $\mathbf{W}$ that contain inconsistent assignments, but also, that most of these inconsistencies appear in *adjacent* rows, then we can use a pairwise consistency

test (where each test checks for consistency between an adjacent pair of rows) to decide if $\mathbf{W}$ is consistent or not. When the assignment matrix $\mathbf{W}$ has many inconsistencies in pairs of adjacent rows, we say that the inconsistency pattern of $\mathbf{W}$ is "regular," and can be checked using a pairwise consistency test.

Regularity-inducing permutations. To leverage the pairwise consistency check, we require that the assignment matrix $\mathbf{W}$ has a regular inconsistency structure that is amenable to a pairwise consistency check. To ensure this, we introduce the notion of a *regularity-inducing permutation*. Our construction relies on the observation that the assignment matrix $\mathbf{W}$ is consistent with a replication structure $\mathbf{A}$ if and only if $\Pi(\mathbf{W})$ is consistent with $\Pi(\mathbf{A})$, where Π is an arbitrary permutation over the entries of a t-by-q matrix. Thus, if we want to check consistency of $\mathbf{W}$ with respect to $\mathbf{A}$, it suffices to check consistency of $\Pi(\mathbf{W})$ with respect to $\Pi(\mathbf{A})$. Then, we say that a specific permutation Π is regularity-inducing with respect to a replication structure $\mathbf{A}$ if whenever $\mathbf{W}$ has many pairs of inconsistent rows with respect to $\mathbf{A}$ (e.g., $\mathbf{W}$ is a set of accepting witnesses to a false statement), then $\Pi(\mathbf{W})$ has many inconsistencies in pairs of *adjacent* rows with respect to $\Pi(\mathbf{A})$. In other words, a regularity-inducing permutation shuffles the entries of the assignment matrix such that any inconsistency pattern in $\mathbf{W}$ maps to a regular inconsistency pattern according to the replication structure $\Pi(\mathbf{A})$. In the construction, instead of performing the pairwise consistency test on $\mathbf{W}$, which can have an arbitrary inconsistency pattern, we perform it on $\Pi(\mathbf{W})$, which has a regular inconsistency pattern. We define the notion more formally in Section 4.2 and show how to construct regularity-inducing permutations in the full version.

Decomposing the permutation. Suppose Π is a regularity-inducing permutation for the replication structure $\mathbf{A}$ associated with the circuits $C_1, \ldots, C_t$ from the robust decomposition of C. Robustness ensures that for any false statement $\mathbf{x}'$, for all collections of witnesses $\{\mathbf{w}_i\}$ where $C_i(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all i, and $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$, the permuted assignment matrix $\Pi(\mathbf{W})$ has inconsistencies in $\Omega(t)$ pairs of adjacent rows with respect to $\Pi(\mathbf{A})$. This can be detected with probability $1 - 2^{-\Omega(t)}$ by performing a pairwise consistency test on the matrix $\mathbf{W}' = \Pi(\mathbf{W})$. The problem, however, is that the verifier only has oracle access to $\langle \mathbf{w}_i, \cdot \rangle$, and it is unclear how to *efficiently* perform the pairwise consistency test on the permuted matrix $\mathbf{W}'$ given just oracle access to the rows $\mathbf{w}_i$ of the unpermuted matrix. Our solution here is to introduce another set of t linear MIP provers for each row $\mathbf{w}'_i$ of $\mathbf{W}' = \Pi(\mathbf{W})$. Thus, the verifier has oracle access to both the rows of the original assignment matrix $\mathbf{W}$, which it uses to check satisfiability of $C_i(\mathbf{x}_i, \cdot)$, as well as the rows of the permuted assignment matrix $\mathbf{W}'$, which it uses to check consistency of the assignments in $\mathbf{W}$. The verifier accepts only if both sets of checks pass. The problem with this basic approach is that there is no reason the prover chooses the matrix $\mathbf{W}'$ so as to satisfy the relation $\mathbf{W}' = \Pi(\mathbf{W})$. Thus, to ensure soundness from this approach, the verifier needs a mechanism to also check that $\mathbf{W}' = \Pi(\mathbf{W})$, given oracle access to the rows of $\mathbf{W}$ and $\mathbf{W}'$.

To facilitate this check, we decompose the permutation Π into a sequence of α permutations $(\Pi_1, \ldots, \Pi_\alpha)$ where $\Pi = \Pi_\alpha \circ \cdots \circ \Pi_1$. Moreover, each

of the intermediate permutations Π_i has the property that they themselves can be decomposed into $t/2$ independent permutations, each of which only permutes entries that appear in 2 distinct rows of the matrix. This "2-locality" property on permutations is amenable to the linear MIP model, and we show in Construction 4.8 a way for the verifier to efficiently check that two matrices $\mathbf{W}$ and $\mathbf{W}'$ (approximately) satisfy the relation $\mathbf{W} = \Pi_i(\mathbf{W}')$, where Π_i is 2-locally decomposable. To complete the construction, we have the prover provide not just the matrix $\mathbf{W}$ and its permutation $\mathbf{W}'$, but all of the intermediate matrices $\mathbf{W}_i = (\Pi_i \circ \Pi_{i-1} \circ \cdots \circ \Pi_1)(\mathbf{W})$ for all $i = 1, \ldots, \alpha$. Since each of the intermediate permutations applied are 2-locally decomposable, there is an efficient procedure for the prover to check each relation $\mathbf{W}_i = \Pi_i(\mathbf{W}_{i-1})$, where we write $\mathbf{W}_0 = \mathbf{W}$ to denote the original assignment matrix. If each of the intermediate permutations are correctly implemented, then the verifier is assured that $\mathbf{W}' = \Pi(\mathbf{W})$, and it can apply the pairwise consistency check on $\mathbf{W}'$ to complete the verification process. We use a Beneš network to implement the decomposition. This ensures that the number of intermediate permutations required is only logarithmic in t, so introducing these additional steps only incurs logarithmic overhead, and does not compromise quasi-optimality of the resulting construction.

Randomized permutation decompositions. There is one additional complication in that the intermediate consistency checks $\mathbf{W}' \stackrel{?}{=} \Pi_i(\mathbf{W})$ are imperfect. They only ensure that *most* of the rows in $\mathbf{W}'$ agree with the corresponding rows in $\Pi_i(\mathbf{W})$. What this means is that when the prover crafts its sequence of permuted assignment matrices $\mathbf{W} = \mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_\alpha$, it is able to "correct" a small number of inconsistencies that appear in $\mathbf{W}$ in each step. Thus, we must ensure that for the particular inconsistency pattern that appears in $\mathbf{W}$, the prover is not able to find a sequence of matrices $\mathbf{W}_1, \ldots, \mathbf{W}_\alpha$, where each of them approximately implements the correct permutation at each step, but at the end, is able to correct all of the inconsistencies in $\mathbf{W}$. To achieve this, we rely on a *randomized permutation decomposition*, where the verifier samples a random sequence of intermediate permutations $\Pi_1, \ldots, \Pi_\alpha$ that collectively implement the target regularity-inducing permutation Π. There are a number of technicalities that arise in the construction and its analysis, and we refer to the full version [15] for the full description.

Putting the pieces together. To summarize, our quasi-optimal linear MIP for circuit satisfiability consists of two key components. First, we apply a robust decomposition to the circuit to obtain many constraints with the property that for a false statement, a malicious prover either cannot satisfy most of the constraints, or if it does satisfy all of the constraints, it must have used an assignment with many inconsistencies. The second key ingredient we introduce is an efficient way to check if there are many inconsistencies in the prover's assignments in the linear MIP model. Our construction here relies on first constructing a regularity-inducing permutation to enable a simple method for consistency checking, and then using a randomized permutation decomposition to enforce the consistency check. We give the formal description and analysis in Section 4.

3 Preliminaries

We begin by defining some notation. For an integer n, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. We use bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices and bold lowercase letters (e.g., $\mathbf{x}, \mathbf{y}$) to denote vectors. For a matrix $\mathbf{A} \in \mathbb{F}^{t \times q}$ over a finite field $\mathbb{F}$, we write $\mathbf{A}_{[i_1, i_2]}$ (where $i_1, i_2 \in [t]$) to denote the sub-matrix of $\mathbf{A}$ containing rows i_1 through i_2 of $\mathbf{A}$ (inclusive). For $i \in [t]$ and $j \in [q]$, we use $\mathbf{A}_{i,j}$ and $\mathbf{A}[i, j]$ to refer to the entry in row i and column j of $\mathbf{A}$.

For a graph $\mathcal{G}$ with n nodes, labeled with the integers $1, \ldots, n$, a matching M is a set of edges $(i, k) \in [n] \times [n]$ with no common vertices. For a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote that x is drawn uniformly at random from S. For a distribution D, we write $x \leftarrow D$ to denote a draw from distribution D. Unless otherwise noted, we write λ to denote the security parameter. We say that a function $f(\lambda)$ is negligible in λ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We write $f(\lambda) = \text{poly}(\lambda)$ to denote that f is bounded by some (fixed) polynomial in λ, and $f = \text{polylog}(\lambda)$ if f is bounded by a (fixed) polynomial in $\log \lambda$. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input.

For a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, the Boolean circuit satisfaction problem is defined by the relation $\mathcal{R}_C = \{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^m : C(\mathbf{x}, \mathbf{w}) = 1\}$. We refer to $\mathbf{x} \in \{0,1\}^n$ as the statement and $\mathbf{w} \in \{0,1\}^m$ as the witness. We write $\mathcal{L}_C$ to denote the language associated with $\mathcal{R}_C$: namely, the set of statements $\mathbf{x} \in \{0,1\}^n$ for which there exists a witness $\mathbf{w} \in \{0,1\}^m$ such that $C(\mathbf{x}, \mathbf{w}) = 1$. In many cases in this work, it will be more natural to work with arithmetic circuits. For an arithmetic circuit $C \colon \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^h$ over a finite field $\mathbb{F}$, we say that C is satisfied if on an input $(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^m$, all of the outputs are 0. Specifically, we define the relation for arithmetic circuit satisfiability to be $\mathcal{R}_C = \left\{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^m : C(\mathbf{x}, \mathbf{w}) = \mathbf{0}^h\right\}$. We include additional preliminaries in the full version [15].

4 Quasi-Optimal Linear MIPs

In this section, we present our core information-theoretic construction of a linear MIP with quasi-optimal prover complexity. We refer to Section 2 for a high-level overview of the construction. In Sections 4.1 and 4.2, we introduce the key building blocks underlying our construction. We give the full construction of our quasi-optimal linear MIP in Section 4.3. We show how to instantiate our core building blocks in the full version [15].

4.1 Robust Decomposition for Circuit Satisfiability

In this section, we formally define our notion of a robust decomposition of an arithmetic circuit. We refer to the technical overview in Section 2 for a high-level description of how we implement our decomposition by combining the MPC-in-the-head paradigm [42] with robust MPC protocols with polylogarithmic overhead [23]. We provide the complete description in the full version [15].

Definition 4.1 (Quasi-Optimal Robust Decomposition). *Let $C\colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^{h'}$ be an arithmetic circuit of size s over a finite field $\mathbb{F}$, $\mathcal{R}_C$ be its associated relation, and $\mathcal{L}_C \subseteq \mathbb{F}^{n'}$ be its associated language. A (t, δ)-robust decomposition of C consists of the following components:*

- *A collection of functions $f_1, \ldots, f_t$ where each function $f_i\colon \mathbb{F}^n \times \mathbb{F}^m \to \{0,1\}$ can be computed by an arithmetic circuit C_i of size $\widetilde{O}(s/t) + \mathrm{poly}(t, \log s)$. Note that a function f_i may only depend on a (fixed) subset of its input variables; in this case, its associated arithmetic circuit C_i only needs to take the (fixed) subset of dependent variables as input.*
- *An efficiently-computable mapping $\mathsf{inp}\colon \mathbb{F}^{n'} \to \mathbb{F}^n$ that maps between a statement $\mathbf{x}' \in \mathbb{F}^{n'}$ for C to a statement $\mathbf{x} \in \mathbb{F}^n$ for $f_1, \ldots, f_t$.*
- *An efficiently-computable mapping $\mathsf{wit}\colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^m$ that maps between a statement-witness pair $(\mathbf{x}', \mathbf{w}') \in \mathbb{F}^{n'} \times \mathbb{F}^{m'}$ to C to a witness $\mathbf{w} \in \mathbb{F}^m$ for $f_1, \ldots, f_t$.*

Moreover, the decomposition must satisfy the following properties:

- ***Completeness:*** *For all $(\mathbf{x}', \mathbf{w}') \in \mathcal{R}_C$, if we set $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$ and $\mathbf{w} = \mathsf{wit}(\mathbf{x}', \mathbf{w}')$, then $f_i(\mathbf{x}, \mathbf{w}) = 1$ for all $i \in [t]$.*
- ***δ-Robustness:*** *For all statements $\mathbf{x}' \notin \mathcal{L}_C$, if we set $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$, then it holds that for all $\mathbf{w} \in \mathbb{F}^m$, the set of indices $S_{\mathbf{w}} = \{i \in [t] : f_i(\mathbf{x}, \mathbf{w}) = 1\}$ satisfies $|S_{\mathbf{w}}| < \delta t$. In other words, any single witness $\mathbf{w}$ can only simultaneously satisfy at most a δ-fraction of the constraints.*
- ***Efficiency:*** *The mappings inp and wit can be computed by an arithmetic circuit of size $\widetilde{O}(s) + \mathrm{poly}(t, \log s)$.*

Systematic linear PCPs. Recall from Section 2 that our linear MIP for checking satisfiability of a circuit C begins by applying a robust decomposition to the circuit C. The MIP proof is comprised of linear PCP proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$ to show that each of the circuits $C_1(\mathbf{x}_1, \cdot), \ldots, C_t(\mathbf{x}_t, \cdot)$ in the robust decomposition of C is satisfiable. Here, $\mathbf{x}_i$ denotes the bits of the statement $\mathbf{x}$ that circuit C_i reads. To provide soundness, the verifier needs to perform a sequence of consistency checks to ensure that the proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$ are *consistent* with some witness $\mathbf{w}$. To facilitate this, we require that the underlying linear PCPs are *systematic*: namely, each proof $\boldsymbol{\pi}_i$ contains a copy of some witness $\mathbf{w}_i$ where $(\mathbf{x}_i, \mathbf{w}_i) \in \mathcal{R}_{C_i}$. The consistency check then affirms that the witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_t$ associated with $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$ are mutually consistent. We give the formal definition of a systematic linear PCP below, and then describe one such instantiation by Ben-Sasson et al. [6, Appendix E].

Definition 4.2 (Systematic Linear PCPs). *Let $(\mathcal{P}, \mathcal{V})$ be an input-oblivious k-query linear PCP for a relation $\mathcal{R}_C$ where $C\colon \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^h$. We say that $(\mathcal{P}, \mathcal{V})$ is* systematic *if the following conditions hold:*

- *On input a statement-witness pair $(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^m$ the prover's output of $\mathcal{P}(\mathbf{x}, \mathbf{w})$ has the form $\boldsymbol{\pi} = [\mathbf{w}, \mathbf{p}] \in \mathbb{F}^d$, for some $\mathbf{p} \in \mathbb{F}^{d-m}$. In other words, the witness is included as part of the linear PCP proof vector.*

– *On input a statement* $\mathbf{x}$ *and given oracle access to a proof* $\boldsymbol{\pi}^* = [\mathbf{w}^*, \mathbf{p}^*]$*, the knowledge extractor* $\mathcal{E}^{\boldsymbol{\pi}^*}(\mathbf{x})$ *outputs* $\mathbf{w}^*$*.*

Fact 4.3 ([6, Claim E.3]). Let $C\colon \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^h$ be an arithmetic circuit of size s over a finite field $\mathbb{F}$ where $|\mathbb{F}| > s$. There exists a systematic input-oblivious 5-query linear PCP $(\mathcal{P}, \mathcal{V})$ for $\mathcal{R}_{\mathcal{C}}$ over $\mathbb{F}$ with knowledge error $O(s/|\mathbb{F}|)$ and query length $O(s)$. Moreover, letting $\mathcal{V} = (\mathcal{Q}, \mathcal{D})$, the prover and verifier algorithms satisfy the following properties:

– the prover algorithm $\mathcal{P}$ is an arithmetic circuit of size $\widetilde{O}(s)$;
– the query-generation algorithm $\mathcal{Q}$ is an arithmetic circuit of size $O(s)$;
– the decision algorithm $\mathcal{D}$ is an arithmetic circuit of size $O(n)$.

4.2 Consistency Checking

As described in Section 2, in our linear MIP construction, we first apply a robust decomposition to the input circuit C to obtain smaller arithmetic circuits $C_1, \ldots, C_t$, each of which depends on some subset of the components of a witness $\mathbf{w} \in \mathbb{F}^m$. The proof then consists of a collection of systematic linear PCP proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$ that $C_1, \ldots, C_t$ are individually satisfiable. The second ingredient we require is a way for the verifier to check that the prover uses a consistent witness to construct the proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t$. In this section, we formally introduce the building blocks we use for the consistency check. We refer to Section 2.1 for an overview of our methods. We begin by defining the notion of a replication structure induced by the decomposition $C_1, \ldots, C_t$, and what it means for a collection of assignments to the circuit $C_1, \ldots, C_t$ to be consistent.

Definition 4.4 (Replication Structures and Inconsistency Matrices). *Fix integers* $m, t, q \in \mathbb{N}$*. A* replication structure *is a matrix* $\mathbf{A} \in [m]^{t \times q}$*. We say that a matrix* $\mathbf{W} \in \mathbb{F}^{t \times q}$ *is consistent with respect to a replication structure* $\mathbf{A}$ *if for all* $i_1, i_2 \in [t]$ *and* $j_1, j_2 \in [q]$*, whenever* $\mathbf{A}_{i_1,j_1} = \mathbf{A}_{i_2,j_2}$*,* $\mathbf{W}_{i_1,j_1} = \mathbf{W}_{i_2,j_2}$*. If there is a pair of indices* (i_1, j_1) *and* (i_2, j_2) *where this relation does not hold, then we say that there is an inconsistency in* $\mathbf{W}$ *(with respect to* $\mathbf{A}$*) at locations* (i_1, j_1) *and* (i_2, j_2)*. For a replication structure* $\mathbf{A} \in [m]^{t \times q}$ *and a matrix of values* $\mathbf{W} \in \mathbb{F}^{t \times q}$*, we define the inconsistency matrix* $\mathbf{B} \in \{0,1\}^{t \times q}$ *where* $\mathbf{B}_{i,j} = 1$ *if and only if there is an inconsistency in* $\mathbf{W}$ *at location* (i, j) *with respect to the replication structure* $\mathbf{A}$*. In the subsequent analysis, we will sometimes refer to an arbitrary inconsistency matrix* $\mathbf{B} \in \{0,1\}^{t \times q}$ *(independent of any particular set of values* $\mathbf{W}$ *or replication structure* $\mathbf{A}$*).*

Definition 4.5 (Consistent Inputs to Circuits). *Let* $C_1, \ldots, C_t$ *be a collection of circuits where each* $C_i\colon \mathbb{F}^m \to \mathbb{F}^h$ *only depends on at most* $q \leq m$ *components of an input vector* $\mathbf{w} \in \mathbb{F}^m$*. For each* $i \in [t]$*, let* $a_1^{(i)}, \ldots, a_q^{(i)} \in [m]$ *be the indices of the* q *components of the input* $\mathbf{w}$ *on which* C_i *depends. The* replication structure *of* $C_1, \ldots, C_t$ *is the matrix* $\mathbf{A} \in [m]^{t \times q}$*, where the* i^{th} *row of* $\mathbf{A}$ *is the vector* $a_1^{(i)}, \ldots, a_q^{(i)}$ *(namely, the subset of indices on which* C_i *depends). We say that a collection of inputs* $\mathbf{w}_1, \ldots, \mathbf{w}_t \in \mathbb{F}^q$ *to* $C_1, \ldots, C_t$ *is consistent if*

the assignment matrix $\mathbf{W}$*, where the* i^{th} *row of* $\mathbf{W}$ *is* $\mathbf{w}_i$ *for* $i \in [t]$*, is consistent with respect to the replication structure* $\mathbf{A}$*.*

To simplify the analysis, we introduce the notion of an inconsistency graph for an assignment matrix $\mathbf{W} \in \mathbb{F}^{t \times q}$ with respect to a replication structure $\mathbf{A} \in [m]^{t \times q}$. At a high level, the inconsistency graph of $\mathbf{W}$ with respect to $\mathbf{A}$ is a graph with t nodes, one for each row of $\mathbf{W}$, and there is an edge between two nodes $i, j \in [t]$ if assignments $\mathbf{w}_i$ and $\mathbf{w}_j$ (in rows i and j of $\mathbf{W}$, respectively) contain an inconsistent assignment with respect to $\mathbf{A}$.

Definition 4.6 (Inconsistency Graph). *Fix positive integers* $m, t, q \in \mathbb{N}$ *and take a replication structure* $\mathbf{A} \in [m]^{t \times q}$*. For any assignment matrix* $\mathbf{W} \in \mathbb{F}^{t \times q}$*, we define the inconsistency graph* $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ *of* $\mathbf{W}$ *with respect to* $\mathbf{A}$ *as follows:*

- *Graph* $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ *is an undirected graph with* t *nodes, with labels in* $[t]$*. We associate node* $i \in [t]$ *with the* i^{th} *row of* $\mathbf{A}$*.*
- *Graph* $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ *has an edge between nodes* i_1 *and* i_2 *if there exists* $j_1, j_2 \in [q]$ *such that* $\mathbf{A}_{i_1,j_1} = \mathbf{A}_{i_2,j_2}$ *but* $\mathbf{W}_{i_1,j_1} \neq \mathbf{W}_{i_2,j_2}$*. In other words, there is an edge in* $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ *whenever there is an inconsistency in the assignments to rows* i_1 *and* i_2 *in* $\mathbf{W}$ *(with respect to the replication structure* $\mathbf{A}$*).*

Definition 4.7 (Regular Matchings). *Fix integers* $m, t, q \in \mathbb{N}$ *where* t *is even, and take any replication structure* $\mathbf{A} \in [m]^{t \times q}$ *and assignment matrix* $\mathbf{W} \in \mathbb{F}^{t \times q}$*. We say that the inconsistency graph* $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ *contains a* regular *matching of size* s *if* $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ *contains a matching* M *of size* s*, where each edge* $(v_1, v_2) \in M$ *satisfies* $(v_1, v_2) = (2i - 1, 2i)$ *for some* $i \in [t/2]$*. In other words, all matched edges are between nodes corresponding to* adjacent *rows in* $\mathbf{W}$*.*

Having defined these notions, we can reformulate the guarantees provided by the (t, δ)-robust decomposition (Definition 4.1). For a constant $\delta > 0$, let $(f_1, \ldots, f_t, \mathsf{inp}, \mathsf{wit})$ be a (t, δ)-robust decomposition of a circuit C. Let $\mathbf{A}$ be the replication structure of the circuits $C_1, \ldots, C_t$ computing $f_1, \ldots, f_t$. Take any statement $\mathbf{x}' \notin \mathcal{L}_C$, and consider any collection of witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_t$ where $C_i(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [t]$. As usual, $\mathbf{x}_i$ denotes the bits of $\mathbf{x} = \mathsf{inp}(\mathbf{x}')$ that C_i reads. Robustness of the decomposition ensures that no single $\mathbf{w}$ can be used to simultaneously satisfy more than a δ-fraction of the constraints. In particular, this means that there must exist $\Omega(t)$ pairs of witnesses $\mathbf{w}_i$ and $\mathbf{w}_j$ which are inconsistent. Equivalently, we say that the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching of size $\Omega(t)$. We prove this statement formally in the full version [15].

Approximate consistency check. By relying on the robust decomposition, it suffices to construct a protocol where the verifier can detect whether the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ of the prover's assignments $\mathbf{W}$ with respect to a replication structure $\mathbf{A}$ contains a large matching. To facilitate this, we first describe an algorithm to check whether two assignment matrices $\mathbf{W}, \mathbf{W}' \in \mathbb{F}^{t \times q}$ (approximately) satisfy the relation $\mathbf{W}' = \Pi(\mathbf{W})$ in the linear MIP model, where Π is a 2-locally decomposable permutation. This primitive can then be used directly

to detect whether an inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a *regular* matching (Corollary 4.11). Subsequently, we show how to permute the entries in $\mathbf{W}$ according to a permutation Π' so as to convert an arbitrary matching in $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ into a regular matching in $\mathcal{G}_{\Pi'(\mathbf{W}),\Pi'(\mathbf{A})}$. Our construction of the approximate consistency check is a direct generalization of the pairwise consistency check procedure described in Section 2.1.

Construction 4.8 (Approximate Consistency Check). Fix an even integer $t \in \mathbb{N}$, and let $P_1, \ldots, P_t, P'_1, \ldots, P'_t$ be a collection of $2 \cdot t$ provers in a linear MIP system. For $i \in [t]$, let $\boldsymbol{\pi}_i \in \mathbb{F}^d$ be the proof vector associated with prover P_i and $\boldsymbol{\pi}'_i \in \mathbb{F}^d$ be the proof vector associated with prover P'_i. We can associate a matrix $\mathbf{W} \in \mathbb{F}^{t \times d}$ with provers $(P_1, \ldots, P_t)$, where the i^{th} row of $\mathbf{W}$ is $\boldsymbol{\pi}_i$. Similarly, we associate a matrix $\mathbf{W}'$ with provers $(P'_1, \ldots, P'_t)$. Let Π be a 2-locally decomposable permutation on the entries of a t-by-d matrix. Then, we describe the following linear MIP verification procedure for checking that $\mathbf{W}' \approx \Pi(\mathbf{W})$.

- **Verifier's query algorithm:** The verifier chooses a random matrix $\mathbf{R} \xleftarrow{R} \mathbb{F}^{t \times d}$, and sets $\mathbf{R}' \leftarrow \Pi(\mathbf{R})$. Let $\mathbf{r}_i$ and $\mathbf{r}'_i$ denote the i^{th} row of $\mathbf{R}$ and $\mathbf{R}'$, respectively. The query algorithm outputs the query $\mathbf{r}_i$ for prover P_i and the query $\mathbf{r}'_i$ to prover P'_i.
- **Verifier's decision algorithm:** Since Π is 2-locally decomposable, we can decompose Π into $t' = t/2$ independent permutations, $\Pi_1, \ldots, \Pi_{t'}$, where each Π_i only operates on a pair of rows (j_{2i-1}, j_{2i}), for all $i \in [t']$. Given responses $\mathbf{y}_i = \langle \boldsymbol{\pi}_i, \mathbf{r}_i \rangle \in \mathbb{F}$ and $\mathbf{y}'_i = \langle \boldsymbol{\pi}'_i, \mathbf{r}'_i \rangle \in \mathbb{F}$ for $i \in [t]$, the verifier checks that the relation
$$\mathbf{y}_{j_{2i-1}} + \mathbf{y}_{j_{2i}} \stackrel{?}{=} \mathbf{y}'_{j_{2i-1}} + \mathbf{y}'_{j_{2i}},$$
for all $i \in [t']$. The verifier accepts if the relations hold for all $i \in [t']$. Otherwise, it rejects.

By construction, we see that if $\mathbf{W}' = \Pi(\mathbf{W})$, then the verifier always accepts.

Lemma 4.9 (Consistency Check Soundness). *Define t, Π, $\mathbf{W}$, and $\mathbf{W}'$ as in Construction 4.8. Then, if the matrix $\mathbf{W}'$ disagrees with $\Pi(\mathbf{W})$ on κ rows, the verifier in Construction 4.8 will reject with probability at least $1 - 2^{-\Omega(\kappa)}$.*

Proof. Consider the event where $\mathbf{W}'$ disagrees with $\hat{\mathbf{W}} = \Pi(\mathbf{W})$ on κ rows. We show that the probability of the verifier accepting in this case is bounded by $2^{-\Omega(\kappa)}$. In the linear MIP model, the verifier's decision algorithm corresponds to checking the following relation:

$$\left\langle \boldsymbol{\pi}_{j_{2i}}, \mathbf{r}_{j_{2i}} \right\rangle + \left\langle \boldsymbol{\pi}_{j_{2i+1}}, \mathbf{r}_{j_{2i+1}} \right\rangle \stackrel{?}{=} \left\langle \boldsymbol{\pi}'_{j_{2i}}, \mathbf{r}'_{j_{2i}} \right\rangle + \left\langle \boldsymbol{\pi}'_{j_{2i+1}}, \mathbf{r}'_{j_{2i+1}} \right\rangle. \tag{4.1}$$

By assumption, there are at least $\kappa/2$ indices $i \in [t]$ where $\mathbf{W}'_{[j_{2i-1}, j_{2i}]} \neq \hat{\mathbf{W}}_{[j_{2i-1}, j_{2i}]}$. By the Schwartz-Zippel lemma, for the indices $i \in [t]$ where $\mathbf{W}'_{[j_{2i}, j_{2i+1}]} \neq \hat{\mathbf{W}}_{[j_{2i}, j_{2i+1}]}$, the relation in Eq. (4.1) holds with probability at most $1/|\mathbb{F}|$ (over the randomness used to sample $\mathbf{r}_{j_{2i-1}}$ and $\mathbf{r}_{j_{2i}}$) Since there are at least $\kappa/2$ such indices, the probability that Eq. (4.1) holds for all $i \in [t']$ is at most $(1/|\mathbb{F}|)^{\kappa/2} = 2^{-\Omega(\kappa)}$. Hence, the verifier rejects with probability $1 - 2^{-\Omega(\kappa)}$. □

The approximate consistency check from Construction 4.8 immediately gives a way to check whether an inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a regular matching of size $\Omega(t)$. To show this, it suffices to exhibit a 2-locally decomposable permutation Π where the assignment matrix $\mathbf{W}$ is consistent on adjacent pairs of rows if and only if $\mathbf{W} = \Pi(\mathbf{W})$. The construction can be viewed as composing many copies of the global consistency check permutation used in [36] (and described in Section 2.1), each applied to a pair of adjacent rows. We give the construction below.

Construction 4.10 (Pairwise Consistency in Adjacent Rows). Fix integers $m, t, q \in \mathbb{N}$ with t even, and let $\mathbf{A} \in [m]^{t \times q}$ be a replication structure. Let $t' = t/2$. For each $i \in [t']$, let Π_i be a permutation over 2-by-q matrices such that Π_i splits into a disjoint set of Hamiltonian cycles based on the entries of $\mathbf{A}_{[2i-1,2i]}$. Define a permutation Π on t-by-q matrices where the action of Π on rows $2i-1$ and $2i$ is given by Π_i for all $i \in [t']$. By construction, the permutation Π is 2-locally decomposable, and moreover, $\mathbf{W} \in \mathbb{F}^{t \times q}$ is pairwise consistent on adjacent rows with respect to $\mathbf{A}$ if and only if $\mathbf{W} = \Pi(\mathbf{W})$.

Corollary 4.11. *Fix integers $m, t, q \in \mathbb{N}$ with t even. Let $\mathbf{A} \in [m]^{t \times q}$ be a replication structure, and Π be the pairwise consistency test permutation for $\mathbf{A}$ from Construction 4.10. Then, for any assignment matrix $\mathbf{W} \in \mathbb{F}^{t \times q}$ where the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a regular matching of size $\Omega(t)$, the verifier Construction 4.8 will reject the relation $\mathbf{W} \stackrel{?}{=} \Pi(\mathbf{W})$ with probability $1 - 2^{-\Omega(t)}$.*

Proof. Since $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a regular matching of size $\Omega(t)$, there are inconsistencies in $\Omega(t)$ pairs of adjacent rows of $\mathbf{W}$. By construction of Π, this means that $\mathbf{W}$ and $\Pi(\mathbf{W})$ differ on $\Omega(t)$ rows. The claim then follows by Lemma 4.9. □

Regularity-inducing permutations. Recall that our objective in the consistency check is to give an algorithm that detects whether an inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching of size $\Omega(t)$. Corollary 4.11 gives a way to detect if the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a *regular* matching of size $\Omega(t)$ with soundness error $2^{-\Omega(t)}$. Thus, to perform the consistency check, we first construct a permutation Π on $\mathbf{W}$ such that whenever $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contain a matching of size $\Omega(t)$, the inconsistency graph $\mathcal{G}_{\Pi(\mathbf{W}),\Pi(\mathbf{A})}$ contains a regular matching of similar size $\Omega(t)$. We say that such permutations are *regularity-inducing*. While we are not able to construct a single permutation Π that is regularity-inducing for all assignment matrices $\mathbf{W}$, we are able to construct a *family* of permutations $(\Pi_1, \ldots, \Pi_z)$ for a fixed replication structure $\mathbf{A}$ such that for all assignment matrices $\mathbf{W} \in \mathbb{F}^{t \times q}$, there is at least one $\beta \in [z]$ where $\mathcal{G}_{\Pi_\beta(\mathbf{W}),\Pi_\beta(\mathbf{A})}$ contains a regular matching of size $\Omega(t)$.

Definition 4.12 (Regularity-Inducing Permutations). *Fix integers $m, t, q \in \mathbb{N}$, and let $\mathbf{A} \in [m]^{t \times q}$ be a replication structure. Let Π be a permutation on t-by-q matrices and $\mathbf{W} \in \mathbb{F}^{t \times q}$ be a matrix such that the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching M of size s. We say that Π is* ρ-regularity-inducing *for $\mathbf{W}$ with respect to $\mathbf{A}$ if the inconsistency graph $\mathcal{G}_{\Pi(\mathbf{W}),\Pi(\mathbf{A})}$ contains a* regular

matching M' of size at least s/ρ. Moreover, there is a one-to-one correspondence between the edges in M' and a subset of the edges in M (as determined by Π). We say that $(\Pi_1, \ldots, \Pi_z)$ is a collection of ρ-regularity-inducing permutations with respect to a replication structure $\mathbf{A}$ if for all $\mathbf{W} \in \mathbb{F}^{t \times q}$, there exists $\beta \in [z]$ such that Π_β is ρ-regularity-inducing for $\mathbf{W}$.

In this work, we will construct regularity-inducing permutations where $\rho = O(1)$. To simplify the following description, we will implicitly assume that $\rho = O(1)$. Given an assignment matrix $\mathbf{W}$ and a collection of ρ-regularity-inducing permutations $(\Pi_1, \ldots, \Pi_z)$ for a replication structure $\mathbf{A}$, we can affirm that the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ does not contain a matching of size $\Omega(t)$ by checking that each of the graphs $\mathcal{G}_{\Pi_\beta(\mathbf{W}),\Pi_\beta(\mathbf{A})}$ does not contain a regular matching of size $\Omega(t/\rho) = \Omega(t)$ for all $\beta \in [z]$ and assuming $\rho = O(1)$. By Corollary 4.11, each of these checks can be implemented in the linear MIP model using Construction 4.8. However, to apply the protocol in Construction 4.8 to $\Pi_\beta(\mathbf{W})$, the verifier requires oracle access to the individual rows of $\Pi_\beta(\mathbf{W})$. Thus, in the linear MIP construction, in addition to providing oracle access to the rows of the assignment matrix $\mathbf{W}$, we also provide the verifier oracle access to the rows of $\Pi_\beta(\mathbf{W})$ for all $\beta \in [z]$. Of course, a malicious MIP prover may provide the rows of a different matrix $\mathbf{W}' \in \mathbb{F}^{t \times q}$ (so as to pass the consistency check). Thus, the final ingredient we require is a way for the verifier to check that two matrices $\mathbf{W}, \mathbf{W}' \in \mathbb{F}^{t \times q}$ satisfy the relation $\mathbf{W}' = \Pi_\beta(\mathbf{W})$. Note that Construction 4.8 does not directly apply because the permutation Π_β is not necessarily 2-locally decomposable.

Decomposing the permutation. To complete the description, we now describe a way for the verifier to check that two matrices $\mathbf{W}, \mathbf{W}' \in \mathbb{F}^{t \times q}$ satisfy the relation $\mathbf{W}' = \Pi(\mathbf{W})$, for an *arbitrary* permutation Π. We assume that the verifier is given oracle access to the rows of $\mathbf{W}$ and $\mathbf{W}'$ in the linear MIP model. Construction 4.8 provides a way to check the relation whenever Π is 2-locally decomposable, so a natural starting point is to decompose the permutation Π into a sequence of 2-locally-decomposable permutations $\Pi_1, \ldots, \Pi_\alpha$, where $\Pi = \Pi_\alpha \circ \cdots \circ \Pi_1$. Then, the linear MIP proof consists of the initial and final matrices $\mathbf{W}$ and $\mathbf{W}'$, as well as the intermediate matrices $\mathbf{W}_i = (\Pi_i \circ \cdots \circ \Pi_1)(\mathbf{W})$. The linear MIP proof would consist of the rows of all of the matrices $\mathbf{W} = \mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_\alpha = \mathbf{W}'$, and the verifier would apply Construction 4.8 to check that for all $\ell \in [\alpha]$, $\mathbf{W}_i = \Pi_i(\mathbf{W}_{i-1})$.

While this general approach seems sound, there is a subtle problem. The soundness guarantee for the consistency check in Construction 4.8 only states that on input $\mathbf{W}, \mathbf{W}'$ and a permutation Π, the verifier will only reject with probability $1 - 2^{\Omega(t)}$ when $\mathbf{W}'$ and $\Pi(\mathbf{W})$ differ on $\Omega(t)$ rows. This means that a malicious prover can provide a sequence of matrices $\mathbf{W}, \mathbf{W}_1, \ldots, \mathbf{W}_\alpha$ where each $\mathbf{W}_\ell$ differs from $\Pi_\ell(\mathbf{W}_{\ell-1})$ on a small number of rows (e.g., $o(t)$ rows), and in doing so, correct all of the inconsistent assignments that appear in the final matrix $\mathbf{W}_\alpha$.

Randomizing the decomposition. Abstractly, we can view the problem as follows. Let $\mathbf{B} \in \{0,1\}^{t \times q}$ be the inconsistency matrix for $\mathbf{W}$ with respect to $\mathbf{A}$

(Definition 4.4). In other words, $\mathbf{B}_{i,j} = 1$ whenever $\mathbf{W}_{i,j}$ encodes a value that is inconsistent with another assignment elsewhere in $\mathbf{W}$. Since $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching of size $\Omega(t)$, we know that there are at least $\Omega(t)$ rows in $\mathbf{B}$ that contain a 1. The permutation Π is chosen so that $\Pi(\mathbf{W})$ has a regular matching of size $\Omega(t)$ with respect to $\Pi(\mathbf{A})$. In particular, this means that the permuted inconsistency matrix $\Pi(\mathbf{B})$ contains a 1 in $\Omega(t)$ adjacent pairs of rows.

Consider the sequence of matrices $\mathbf{W}_1, \ldots, \mathbf{W}_\alpha$ chosen by the prover. Using the approximate pairwise consistency check, we can ensure that $\mathbf{W}_i$ agrees with $\Pi_i(\mathbf{W}_{i-1})$ on all but some κ_1 rows. Now suppose that there exists some $\ell \in [\alpha]$ where $\mathbf{B}_\ell = (\Pi_\ell \circ \cdots \circ \Pi_1)(\mathbf{B})$ has the property that all of the locations with a 1 in $\mathbf{B}$ appear in just κ_1 rows of $\mathbf{B}_\ell$. If this happens, then the malicious prover can construct $\mathbf{W}_1, \ldots, \mathbf{W}_{\ell-1}$ honestly, and then choose $\mathbf{W}_\ell$ such that $\mathbf{W}_\ell = \Pi_\ell(\mathbf{W}_{\ell-1})$ on all rows where $\mathbf{B}_\ell$ does not contain a 1, and set the values in the rows where $\mathbf{B}_\ell$ does contain a 1 to be consistent with the other rows of $\mathbf{W}$. Notably, all the entries in $\mathbf{W}_\ell$ are now consistent, and moreover, $\mathbf{W}_\ell$ differs from $\Pi_\ell(\mathbf{W}_{\ell-1})$ on at most κ_1 rows (and so, will not be detected with high probability by the pairwise consistency check). This means that from the verifier's perspective, the final matrix $\Pi(\mathbf{W})$ has no inconsistencies, and thus, the verifier's final pairwise consistency check passes with probability 1 (even though the original inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching of size $\Omega(t)$). Thus, we require a stronger property on the permutation decomposition. It is not sufficient that there is a matching of size $\Omega(t)$ in the starting and ending configurations $\mathbf{W}$ and $\mathbf{W}'$. Rather, we need that the size of the matching in *every* step of the decomposition cannot shrink by too much, or equivalently, the intermediate permutations $\Pi_1, \ldots, \Pi_\alpha$ cannot "concentrate" all of the inconsistencies in $\mathbf{W}$ into a small number of rows (which the malicious prover can fix without being detected). We say permutation decompositions with this property are *non-concentrating*. We now formally define the notion of a non-concentrating permutation decomposition and what it means for a collection of permutation sequences to be non-concentrating.

Definition 4.13 (Non-Concentrating Permutations). *Fix positive integers $t, q \in \mathbb{N}$, and let $\Gamma = (\Pi_1, \ldots, \Pi_\alpha)$ be a sequence of permutations over t-by-q matrices. Let $\mathbf{B} \in \{0,1\}^{t \times q}$ be an inconsistency matrix. For $\ell \in [\alpha]$, define $\mathbf{B}_\ell = (\Pi_\ell \circ \cdots \circ \Pi_1)(\mathbf{B})$. We say that Γ is a sequence of* (κ_1, κ_2)-non-concentrating permutations *with respect to $\mathbf{B}$ if for all $\ell \in [\alpha]$, the inconsistency matrix $\mathbf{B}_\ell$ has the property that no subset of κ_1 rows contains more than κ_2 inconsistencies (indices where the value is 1). Next, we say a collection of permutation sequences $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ where each $\Gamma^{(j)} = \left(\Pi_1^{(j)}, \ldots, \Pi_\alpha^{(j)}\right)$ is (κ_1, κ_2)-non-concentrating for a set $\mathcal{B} \subseteq \{0,1\}^{t \times q}$ of inconsistency matrices if for all $\mathbf{B} \in \mathcal{B}$, there is some $j \in [\gamma]$ such that $\Gamma^{(j)}$ is (κ_1, κ_2)-non-concentrating with respect to $\mathbf{B}$.*

Putting the pieces together. To summarize, the goal of the consistency check is to decide whether the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ of some assignment matrix $\mathbf{W}$ with respect to a replication structure $\mathbf{A}$ contains a matching of size $\Omega(t)$. Our strategy relies on the following:

– Let $(\Pi_1, \ldots, \Pi_z)$ be a collection of regularity-inducing permutations with respect to $\mathbf{A}$.
– For each $\beta \in [z]$, let $\Gamma_\beta^{(1)}, \ldots, \Gamma_\beta^{(\gamma)}$ be a collection of non-concentrating permutations that implement Π_β, where $\Gamma_\beta^{(j)} = (\Pi_{\beta,1}^{(j)}, \ldots, \Pi_{\beta,\alpha}^{(j)})$ for all $j \in [\gamma]$, and each of the intermediate permutations $\Pi_{\beta,\ell}^{(j)}$ are 2-locally decomposable for all $j \in [\gamma]$, $\beta \in [z]$, and $\ell \in [\alpha]$.

The proof then consists of the initial assignment matrix $\mathbf{W}$ in addition to all of the intermediate matrices $\mathbf{W}_{\beta,\ell}^{(j)} = \Pi_{\beta,\ell}^{(j)}(\mathbf{W}_{\beta,\ell-1}^{(j)})$, where we define $\mathbf{W}_{\beta,0}^{(j)} = \mathbf{W}$ for all $j \in [\gamma]$, $\beta \in [z]$. The verifier checks consistency of all of the intermediate matrices using Construction 4.8, and applies a pairwise consistency test (Construction 4.10) to each of $\mathbf{W}_{\beta,\alpha}^{(j)}$ for all $j \in [\gamma]$ and $\beta \in [z]$. The soundness argument then proceeds roughly as follows:

– Since $(\Pi_1, \ldots, \Pi_z)$ is regularity-inducing, there is some $\beta \in [z]$ where $\mathcal{G}_{\Pi_\beta(\mathbf{W}),\Pi_\beta(\mathbf{A})}$ contains a regular matching.
– Since $\Gamma_\beta^{(1)}, \ldots, \Gamma_\beta^{(\gamma)}$ is a collection of non-concentrating permutations that implement Π_β, and all of the intermediate consistency checks pass, then there must be some $j \in [\gamma]$ such that $\mathcal{G}_{\mathbf{W}_{\beta,\alpha}^{(j)},\Pi_\beta(\mathbf{A})}$ contains a regular matching of size $\Omega(t)$. The verifier then rejects with exponentially-small probability (in t) by soundness of the pairwise consistency test.

Finally, in our concrete instantiation (described in the full version [15]), we show how to construct our collection of regularity-inducing permutations and non-concentrating permutations sequences where $z = O(1)$, $\gamma = O(\log^3 t)$, $\alpha = \Theta(\log t)$. For this setting of parameters, the overall consistency check only incurs *polylogarithmic* overhead to the prover complexity and the proof size. In Section 4.3, we give the formal description and analysis of our linear MIP construction.

4.3 Quasi-Optimal Linear MIP Construction

In this section, we describe our quasi-optimal linear MIP for circuit satisfiability. We give our construction (Construction 4.14) but defer the security theorem and analysis to the full version. By instantiating Construction 4.14 with the appropriate primitives, we obtain the first quasi-optimal linear MIP (Theorem 4.15).

Construction 4.14 (Linear MIP). Fix parameters $t, \delta, k, \varepsilon, d, \rho, \kappa_1, \kappa_2$, and let C be an arithmetic circuit of size s over a finite field $\mathbb{F}$. The construction relies on the following ingredients:

– Let $(f_1, \ldots, f_t, \mathsf{inp}, \mathsf{wit})$ be a quasi-optimal (t, δ)-robust decomposition of C. Let C_i be the arithmetic circuit that computes each constraint $f_i \colon \mathbb{F}^n \times \mathbb{F}^m \to \{0,1\}$.
– Let $(\mathcal{P}_1, \mathcal{V}_1), \ldots, (\mathcal{P}_t, \mathcal{V}_t)$ be k-query systematic linear PCP systems for circuits $C_1, \ldots, C_t$, respectively, with knowledge error ε and query length d.

– Let $\mathbf{A} \in [m]^{t \times q}$ be the replication structure of $C_1, \ldots, C_t$ (where q is a bound on the number of indices in a witness $\mathbf{w} \in \mathbb{F}^m$ on which each circuit depends). Let $\Pi_1, \ldots, \Pi_z$ be a collection of ρ-regularity-inducing permutations on t-by-q matrices with respect to the replication structure $\mathbf{A}$ (Definition 4.12).
– For $\beta \in [z]$, let $\mathcal{B}_\beta \subseteq \{0,1\}^{t \times q}$ be the set of inconsistency patterns where $\mathbf{B}$ and $\Pi_\beta(\mathbf{B})$ have at most one inconsistency in each row. Let $\Gamma_\beta^{(1)}, \ldots, \Gamma_\beta^{(\gamma)}$ be a collection of permutation sequences implementing Π_β that is (κ_1, κ_2)-non-concentrating for $\mathcal{B}_\beta$ (Definition 4.13). In particular, each $\Gamma_\beta^{(j)}$ is a sequence of α permutations $\left(\Pi_{\beta,1}^{(j)}, \ldots, \Pi_{\beta,\alpha}^{(j)}\right)$, where each intermediate permutation $\Pi_{\beta,\ell}^{(j)}$ is 2-locally decomposable.

The linear MIP with $t \cdot (1 + \alpha\gamma z)$ provers and query length d is defined as follows:

– **Syntax:** The linear MIP consists of $t \cdot (1 + \alpha\gamma z)$ provers. We label the provers as P_i and $P_{\beta,\ell,i}^{(j)}$ for $i \in [t]$, $j \in [\gamma]$, $\beta \in [z]$, and $\ell \in [\alpha]$. To simplify the description, we will often pack the proof vectors from different provers into the rows of a matrix. To recall, when we say we associate a matrix $\hat{\mathbf{W}} \in \mathbb{F}^{t \times d}$ with provers $(P_1, \ldots, P_t)$, we mean that the i^{th} row of $\hat{\mathbf{W}}$ is the proof vector assigned to prover P_i for all $i \in [t]$. Similarly, when we say the verifier distributes a query matrix $\mathbf{Q} \in \mathbb{F}^{t \times d}$ to provers $(P_1, \ldots, P_t)$, we mean that it submits the i^{th} row of $\mathbf{Q}$ as a query to P_i for all $i \in [t]$.
– **Prover's algorithm:** On input the statement $\mathbf{x}' \in \mathbb{F}^{n'}$ and witness $\mathbf{w}' \in \mathbb{F}^{m'}$, the prover prepares the proof vectors as follows:
  • **Linear PCP proofs.** First, the prover computes $\mathbf{x} \leftarrow \mathsf{inp}(\mathbf{x}')$ and $\mathbf{w} \leftarrow \mathsf{wit}(\mathbf{x}', \mathbf{w}')$. For each $i \in [t]$, it computes a proof $\boldsymbol{\pi}_i \leftarrow \mathcal{P}_i(\mathbf{x}_i, \mathbf{w}_i)$, where $\mathbf{x}_i$ and $\mathbf{w}_i$ denote the bits of the statement $\mathbf{x}$ and witness $\mathbf{w}$ on which circuit C_i depends, respectively. Since $(\mathcal{P}_i, \mathcal{V}_i)$ is a systematic linear PCP, we can write $\boldsymbol{\pi}_i = [\mathbf{w}_i, \mathbf{p}_i]$ where $\mathbf{w}_i \in \mathbb{F}^q$ and $\mathbf{p}_i \in \mathbb{F}^{d-q}$. For $i \in [t]$, the prover associates the vector $\boldsymbol{\pi}_i$ with P_i.
  • **Consistency proofs.** Let $\mathbf{W} \in \mathbb{F}^{t \times q}$ be the matrix where the i^{th} row is the vector $\mathbf{w}_i$. Now, for all $j \in [\gamma]$, $\beta \in [z]$, and $\ell \in [\alpha]$, let $\mathbf{W}_{\beta,\ell}^{(j)} = \left(\Pi_{\beta,\ell}^{(j)} \circ \Pi_{\beta,\ell-1}^{(j)} \circ \cdots \circ \Pi_{\beta,1}^{(j)}\right)(\mathbf{W})$. Let $\hat{\mathbf{W}}_{\beta,\ell}^{(j)} = \left[\mathbf{W}_{\beta,\ell}^{(j)}, \mathbf{0}^{t \times (d-q)}\right]$. The prover associates $\hat{\mathbf{W}}_{\beta,\ell}^{(j)}$ with provers $(P_{\beta,\ell,1}^{(j)}, \ldots, P_{\beta,\ell,t}^{(j)})$.
– **Verifier's query algorithm:** To simplify the description, we will sometimes state the query vectors the verifier submits to each prover P_i and $P_{\beta,\ell,i}^{(j)}$ rather than the explicit query matrices. The verifier's queries are constructed as follows:
  • **Linear PCP queries.** For $i \in [t]$, the verifier invokes the query generation algorithm $\mathcal{Q}_i$ for each of the underlying linear PCP instances $(\mathcal{P}_i, \mathcal{V}_i)$ to obtain a query matrix $\mathbf{Q}_i \in \mathbb{F}^{d \times k}$ and some state information st_i. The verifier gives $\mathbf{Q}_i$ to prover P_i, and saves the state $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_t)$.
  • **Routing consistency queries.** For all $j \in [\gamma]$, $\beta \in [z]$, and $\ell \in [\alpha]$, the verifier invokes the query generation algorithm of Construction 4.8 on permutation $\Pi_{\beta,\ell}^{(j)}$ to obtain two query matrices $\mathbf{R}_{\beta,\ell}^{(j)}$ and $\mathbf{S}_{\beta,\ell}^{(j)} \in \mathbb{F}^{t \times q}$.

The verifier pads the matrices to obtain $\hat{\mathbf{R}}^{(j)}_{\beta,\ell} = \left[\mathbf{R}^{(j)}_{\beta,\ell}, \mathbf{0}^{t\times(d-q)}\right]$ and $\hat{\mathbf{S}}^{(j)}_{\beta,\ell} = \left[\mathbf{S}^{(j)}_{\beta,\ell}, \mathbf{0}^{t\times(d-q)}\right]$. There are two cases:

  * If $\ell = 1$, the verifier distributes the queries $\hat{\mathbf{R}}^{(j)}_{\beta,\ell}$ to provers $(P_1, \ldots, P_t)$.
  * If $\ell > 1$, the verifier distributes the queries $\hat{\mathbf{R}}^{(j)}_{\beta,\ell}$ to provers $\left(P^{(j)}_{\beta,\ell-1,1}, \ldots, P^{(j)}_{\beta,\ell-1,t}\right)$.

  In addition, the verifier distributes the queries $\hat{\mathbf{S}}^{(j)}_{\beta,\ell}$ to provers $\left(P^{(j)}_{\beta,\ell,1}, \ldots, P^{(j)}_{\beta,\ell,t}\right)$. Intuitively, the verifier is applying the approximate consistency check from Construction 4.8 to every permutation $\Pi^{(j)}_{\beta,\ell}$.
- **Pairwise consistency queries.** For each $\beta \in [z]$, let $\mathbf{A}_\beta = \Pi_\beta(\mathbf{A})$, and let Π'_β be the pairwise consistency test matrix for $\mathbf{A}_\beta$ (Construction 4.10). The verifier invokes the query generation algorithm of Construction 4.8 on permutation Π'_β to obtain two query matrices $\mathbf{R}_\beta$ and $\mathbf{S}_\beta \in \mathbb{F}^{t\times q}$. It pads the matrices to obtain $\hat{\mathbf{R}}_\beta = [\mathbf{R}_\beta, \mathbf{0}^{t\times(d-q)}]$ and $\hat{\mathbf{S}}_\beta = [\mathbf{S}_\beta, \mathbf{0}^{t\times(d-q)}]$. Next, it distributes $\hat{\mathbf{R}}_\beta$ and $\hat{\mathbf{S}}_\beta$ to $(P^{(j)}_{\beta,\alpha,1}, \ldots, P^{(j)}_{\beta,\alpha,t})$ for all $j \in [\gamma]$. In this step, the verifier is checking pairwise consistency of the permuted assignment matrices $\mathbf{W}^{(j)}_{\beta,\alpha}$ for all $j \in [\gamma]$ and $\beta \in [z]$.

In total, the verifier makes a total of $k + \alpha\gamma z$ queries to each prover P_i for $i \in [t]$. It makes $O(1)$ queries to the other provers.

– **Verifier's decision algorithm:** First, the verifier computes the statement $\mathbf{x} \leftarrow \mathsf{inp}(\mathbf{x}')$. For $i \in [t]$, let $\mathbf{x}_i$ denote the bits of $\mathbf{x}$ on which circuit C_i depends. The verifier processes the responses from each set of queries as follows:

- **Linear PCP queries.** For $i \in [t]$, let $\mathbf{y}_i \in \mathbb{F}^k$ be the response of prover P_i to the linear PCP queries. For $i \in [t]$, the verifier invokes the decision algorithm $\mathcal{D}_i$ for each of the underlying linear PCP instances $(\mathcal{P}_i, \mathcal{V}_i)$ on the state st_i, the statement $\mathbf{x}_i$, and the response $\mathbf{y}_i$. It rejects the proof if $\mathcal{D}_i(\mathsf{st}_i, \mathbf{x}_i, \mathbf{y}_i) = 0$ for any $i \in [t]$.
- **Consistency queries.** For each set of routing consistency query responses (for checking consistency of the intermediate permutations $\Pi^{(j)}_{\beta,\ell}$), and for each set of pairwise consistency query responses (for checking consistency of the final configurations Π'_β), the verifier applies the decision algorithm from Construction 4.8, and rejects if any check fails.

If all of the checks pass, then the verifier accepts the proof.

Instantiating the construction. We defer the security analysis of Construction 4.14 to the full version [15]. In the full version, we additionally show how to instantiate the robust decomposition, the regularity-inducing permutations, and the non-concentrating permutation sequences needed to apply Construction 4.14. Combining Construction 4.14 with our concrete instantiations, we obtain a quasi-optimal linear MIP. We state the formal theorem below, and give the proof in the full version.

Theorem 4.15 (Quasi-Optimal Linear MIP). *Fix a security parameter λ. Let $C\colon \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^h$ be an arithmetic circuit of size s over a $\mathrm{poly}(\lambda)$-size finite field $\mathbb{F}$ where $|\mathbb{F}| > s$. Then, there exists an input-oblivious k-query linear MIP*

$(\mathcal{P}, \mathcal{V})$ *with* $\ell = \widetilde{O}(\lambda)$ *provers for* $\mathcal{R}_C$ *with soundness error* $2^{-\lambda}$*, query length* $\widetilde{O}(s/\lambda) + \mathrm{poly}(\lambda, \log s)$*, and* $k = \mathrm{polylog}(\lambda)$*. Moreover, letting* $\mathcal{V} = (\mathcal{Q}, \mathcal{D})$*, the prover and verifier algorithms satisfy the following properties:*

- *the prover algorithm* $\mathcal{P}$ *is an arithmetic circuit of size* $\widetilde{O}(s) + \mathrm{poly}(\lambda, \log s)$*;*
- *the query-generation algorithm* $\mathcal{Q}$ *is an arithmetic circuit of size* $\widetilde{O}(s) + \mathrm{poly}(\lambda, \log s)$*;*
- *the decision algorithm* $\mathcal{D}$ *is an arithmetic circuit of size* $\widetilde{O}(\lambda n)$*.*

Remark 4.16 (Soundness Against Affine Provers). To leverage our linear MIP to construct a SNARG, we often require that the linear MIP provide soundness against affine provers. We note that Construction 4.14 inherits this property as long as the underlying linear PCPs and approximate consistency check primitives provide soundness against affine strategies. It is straightforward to see that Construction 4.8 remains sound even against affine adversarial strategies, and in the full version, we show how the underlying linear PCPs can be made robust against affine strategies with minimal overhead. Importantly, these modifications do not increase the asymptotic complexity of Construction 4.14.

5 Quasi-Optimal SNARGs

In this section, we formally introduce the notion of a quasi-optimal SNARG. Next, in Section 5.2, we show how to compile a linear MIP into a designated-verifier SNARG in the preprocessing model using the notion of a linear-only vector encryption over rings introduced in [14]. Combined with our quasi-optimal linear MIP from Section 4, this yields a quasi-optimal designated-verifier SNARG for Boolean circuit satisfiability in the preprocessing model. We refer to the full version [15] for the formal definition of a succinct non-interactive argument (SNARG) and for the definitions of a linear-only vector encryption that we use in our construction. We also introduce the notion of a 1-bit SNARG in the full version.

5.1 Defining Quasi-Optimality

In this section, we formally define our notion of a quasi-optimal SNARG. Then, in the full version, we compare our notion to the previous notion of quasi-optimality introduced in [14], as well as describe a heuristic approach for instantiating quasi-optimal SNARGs.

Definition 5.1 (Quasi-Optimal SNARG). *Let* $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a SNARG for a family of Boolean circuits* $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$*. Then,* Π_{SNARG} *is* quasi-optimal *if it achieves* $2^{-\lambda}$ *soundness error against provers of size* 2^{λ} *and satisfies the following properties:*

- ***Prover Complexity:*** *The running time of* Prove *is* $\widetilde{O}(|C_n|) + \mathrm{poly}(\lambda, \log |C_n|)$*.*
- ***Succinctness:*** *The length of the proof output by* Prove *is* $\widetilde{O}(\lambda)$*.*

Next, in Lemma 5.2, we show that our notion of quasi-optimality is tight in the following sense: assuming NP does not have succinct *proofs*, any argument system for NP that provides soundness error $2^{-\lambda}$ must have proofs of length $\Omega(\lambda)$. We state the lemma below and give the proof in the full version [15].

Lemma 5.2. *Let $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ be a family of Boolean circuits for some language $\mathcal{L} = \bigcup_{n\in\mathbb{N}} \mathcal{L}_{C_n}$, where $C_n \colon \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}$ for all $n \in \mathbb{N}$. Fix a soundness parameter ρ and a security parameter λ. Let $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a SNARG for $\mathcal{C}$ with soundness $2^{-\rho}$ against provers of size $\mathrm{poly}(\lambda)$. If $\mathcal{L}_{C_n} \not\subseteq \mathbf{DTIME}(2^{o(n)})$, then the length $\ell(\rho)$ of an argument in Π_{SNARG} is $\Omega(\rho)$.*

5.2 Quasi-Optimal SNARGs from Quasi-Optimal Linear MIPs

In this section, we show how to combine a linear MIPs with linear-only vector encryption over rings to obtain a quasi-optimal SNARG. We refer to the full version for the definition of a linear-only vector encryption from [14]. We describe the construction and state its security theorems here, but defer the security proofs to the full version [15].

Construction 5.3 (SNARG from Linear MIP). Fix a prime p and let $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ be a family of arithmetic circuits over $\mathbb{F}_p$. Let $\mathcal{R}_{\mathcal{C}}$ be the relation associated with $\mathcal{C}$. Let $(\mathcal{P}, \mathcal{V})$ be a k-query linear MIP with ℓ provers and query length d for the relation $\mathcal{R}_{\mathcal{C}}$. Let $\Pi_{\mathsf{venc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a secret-key vector encryption scheme over R^k where $R \cong \mathbb{F}_p^\ell$. Our single-theorem, designated-verifier SNARG $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ in the preprocessing model for $\mathcal{R}_{\mathcal{C}}$ is given below:

- $\mathsf{Setup}(1^\lambda, 1^n) \to (\sigma, \tau)$: On input the security parameter λ and the circuit family parameter n, the setup algorithm does the following:
  1. Invoke the query-generation algorithm $\mathcal{Q}$ for the linear MIP to obtain a tuple of query matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_\ell \in \mathbb{F}_p^{d\times k}$ and state information st.
  2. Generate a secret key $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell)$ for the vector encryption scheme.
  3. Pack the ℓ query matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_\ell$ into a single query matrix $\mathbf{Q} \in R^{d\times k}$ (recall that the ring R splits into ℓ isomorphic copies of $\mathbb{F}_p$).
  4. Encrypt each row of $\mathbf{Q}$ (an element of R^k) using the vector encryption scheme. In other words, for $i \in [d]$, let $\mathbf{q}_i \in R^d$ be the i^{th} row of $\mathbf{Q}$. In this step, the setup algorithm computes ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{q}_i)$.
  5. Output the common reference string $\sigma = (\mathsf{ct}_1, \ldots, \mathsf{ct}_d)$ and the verification state $\tau = (\mathsf{sk}, \mathsf{st})$.
- $\mathsf{Prove}(\sigma, \mathbf{x}, \mathbf{w}) \to \boldsymbol{\pi}$. On input the common reference string $\sigma = (\mathsf{ct}_1, \ldots, \mathsf{ct}_d)$, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prover's algorithm works as follows:
  1. For each $i \in [\ell]$, invoke the linear MIP prover algorithm P_i on input $\mathbf{x}$ and $\mathbf{w}$ to obtain a proof $\boldsymbol{\pi}_i \leftarrow P_i(\mathbf{x}, \mathbf{w}) \in \mathbb{F}_p^d$.

2. Pack the ℓ proof vectors $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell \in \mathbb{F}_p^d$ into a single proof vector $\boldsymbol{\pi} \in R^d$. Then, viewing the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ as vector encryptions of the rows of the query matrix $\mathbf{Q} \in R^{d \times k}$, homomorphically compute an encryption of the matrix-vector product $\mathbf{Q}^\top \boldsymbol{\pi} \in R^k$. In particular, the prover homomorphically computes the sum $\mathsf{ct}' = \sum_{i \in d} \boldsymbol{\pi}_i \cdot \mathsf{ct}_i$.
3. Output the proof ct'.

– $\mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) \to \{0, 1\}$: On input the verification state $\tau = (\mathsf{sk}, \mathsf{st})$, the statement $\mathbf{x}$, and the proof $\pi = \mathsf{ct}'$, the verifier does the following:
1. Decrypt the proof ct' using the secret key sk to obtain the prover's responses $\mathbf{y} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}')$. If $\mathbf{y} = \perp$, the verifier terminates with output 0.
2. The verifier decomposes $\mathbf{y} \in R^k$ into vectors $\mathbf{y}_1, \ldots, \mathbf{y}_\ell \in \mathbb{F}_p^k$. It then invokes the linear MIP decision algorithm $\mathcal{D}$ on the statement $\mathbf{x}$, the responses $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$, and the verification state st and outputs $\mathcal{D}(\mathsf{st}, \mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_\ell)$.

Theorem 5.4. *Fix a security parameter λ and a prime p. Let $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ be a family of arithmetic circuits over $\mathbb{F}_p$, $\mathcal{R}_\mathcal{C}$ be the relation associated with $\mathcal{C}$, and $(\mathcal{P}, \mathcal{V})$ be a k-query linear MIP with ℓ provers, query length d, and soundness error $\varepsilon(\lambda)$ against affine provers for the relation $\mathcal{R}_\mathcal{C}$. Let $\Pi_{\mathsf{venc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a vector encryption scheme over a ring $R \cong \mathbb{F}_p^\ell$ with linear targeted malleability. Then, applying Construction 5.3 to $(\mathcal{P}, \mathcal{V})$ and Π_{venc} yields a non-adaptive designated-verifier preprocessing SNARG with soundness error $2 \cdot \varepsilon(\lambda) + \mathrm{negl}(\lambda)$.*

Theorem 5.5. *Fix a security parameter λ and a prime p. Let $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ be a family of arithmetic circuits over $\mathbb{F}_p$, $\mathcal{R}_\mathcal{C}$ be the relation associated with $\mathcal{C}$, and $(\mathcal{P}, \mathcal{V})$ be a k-query linear MIP with ℓ provers, query length d, and soundness error $\varepsilon(\lambda)$ against affine provers for the relation $\mathcal{R}_\mathcal{C}$. Let $\Pi_{\mathsf{venc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a linear-only vector encryption scheme. Then, applying Construction 5.3 to $(\mathcal{P}, \mathcal{V})$ and Π_{venc} yields an adaptive designated-verifier preprocessing SNARG with soundness error $\varepsilon(\lambda) + \mathrm{negl}(\lambda)$.*

Instantiating the construction. To conclude this section, we show that combining the candidate vector encryption scheme Π_{venc} over polynomial rings R^k, where $R \cong \mathbb{F}_p^\ell$ from [14, §4.4] with our quasi-optimal linear MIP construction from Theorem 4.15 yields a quasi-optimal SNARG from linear-only vector encryption. We first recall from [14, §4.4] that the candidate vector encryption scheme Π_{venc} has the following properties:

– When $k = \mathrm{polylog}(\lambda)$, $\ell = \widetilde{O}(\lambda)$, and $|\mathbb{F}| = \mathrm{poly}(\lambda)$, each ciphertext encrypting an element of R^k has length $\widetilde{O}(\lambda)$.
– Scalar multiplication and homomorphic addition of two ciphertexts can be performed in time $\widetilde{O}(\lambda)$.

When we apply Construction 5.3 to the linear MIP from Theorem 4.15 and Π_{venc}, the prover complexity and proof sizes are then as follows (targeting soundness error $2^{-\lambda}$):

- **Prover complexity:** The SNARG prover first invokes the underlying linear MIP prover to obtain proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell$ for each of the $\ell = \widetilde{O}(\lambda)$ provers. From Theorem 4.15, this step requires time $\widetilde{O}(s) + \mathrm{poly}(\lambda, \log s)$, where s is the size of the circuit. To construct the proof, the prover has to perform d homomorphic operations, where $d = \widetilde{O}(s/\lambda) + \mathrm{poly}(\lambda, \log s)$ is the query length of the construction from Theorem 4.15. Since each homomorphic operation can be computed in $\widetilde{O}(\lambda)$ time, the overall prover complexity is $\widetilde{O}(s) + \mathrm{poly}(\lambda, \log s)$.
- **Proof size:** The proof in Construction 5.3 consists of a single ciphertext, which for our parameter settings, have length $\widetilde{O}(\lambda)$.

From this analysis, we obtain the following quasi-optimal SNARG instantiations:

Corollary 5.6. *Assuming the vector encryption scheme Π_{venc} from [14, §4.4] satisfies linear targeted malleability (with exponential security), then applying Construction 5.3 to the quasi-optimal linear MIP from Theorem 4.15 and Π_{venc} yields a non-adaptive designated-verifier quasi-optimal SNARG for Boolean circuit satisfiability in the preprocessing model.*

Corollary 5.7. *Assuming the vector encryption scheme Π_{venc} from [14, §4.4] (with the "double-encryption" transformation described in [14, Remark C.4]) is linear-only (with exponential security), then applying Construction 5.3 to the quasi-optimal linear MIP from Theorem 4.15 and Π_{venc} yield an adaptive designated-verifier quasi-optimal SNARG for Boolean circuit satisfiability in the preprocessing model.*

Construction 5.3 gives a construction of a *single-theorem* SNARG from any linear MIP system. In the full version [15], we discuss some of the challenges in extending our construction to provide multi-theorem security.

Remark 5.8 (Multi-Theorem SNARGs). Construction 5.3 gives a construction of a *single-theorem* SNARG from any linear MIP system. The works of [13, 14] show how to construct multi-theorem designated-verifier SNARGs by relying on a stronger notion of soundness at the linear PCP level coupled with a stronger interactive linear-only encryption assumption. While we could rely on the same type of cryptographic assumption as in [14], our linear MIP from Section 4 does not satisfy the notion of "reusable" or "strong" soundness from [13]. Strong soundness essentially says that for all proofs, the probability that the verifier accepts or that it rejects is negligible close to 1 (where the probability is taken over the randomness used to generate the queries). In particular, whether the verifier decides to accept or reject should be *uncorrelated* with the randomness associated with its secret verification state. In our linear MIP model, we operate over a polynomial-size field, so a prover making a local change will cause the verifier's decision procedure to change with noticeable probability. This reveals information about the secret verification state, which can enable the malicious prover to break soundness. We leave it as an open problem to construct a quasi-optimal linear MIP that provides strong soundness. Such a primitive would be useful in constructing a quasi-optimal multi-theorem SNARGs.

Acknowledgments

We thank the anonymous reviewers for helpful feedback on the presentation. D. Boneh and D. J. Wu are supported by NSF, DARPA, a grant from ONR, and the Simons Foundation. Y. Ishai and A. Sahai are supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, NSF-BSF grant 2015782, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. Y. Ishai is additionally supported by ISF grant 1709/14 and ERC grant 742754. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1. B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP*, 2010.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3), 1998.
3. L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *STOC*, 1991.
4. B. Barak and R. Pass. On the possibility of one-message weak zero-knowledge. In *TCC*, 2004.
5. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, 2004.
6. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO*, 2013.
7. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium*, 2014.
8. I. Berman, A. Degwekar, R. Rothblum, and P. N. Vasudevan. From laconic zero-knowledge to public-key cryptography. *Electronic Colloquium on Computational Complexity (ECCC)*, 2017, 2017.
9. N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer. The hunting of the SNARK. *J. Cryptology*, 30(4), 2017.
10. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.
11. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *STOC*, 2013.
12. N. Bitansky and A. Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *CRYPTO*, 2012.
13. N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, 2013.
14. D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *EUROCRYPT*, 2017.

15. D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. *IACR Cryptology ePrint Archive*, 2018, 2018. Available at `https://eprint.iacr.org/2018/133.pdf`.
16. R. B. Boppana, J. Håstad, and S. Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2), 1987.
17. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2), 1988.
18. B. Braun, A. J. Feldman, Z. Ren, S. T. V. Setty, A. J. Blumberg, and M. Walfish. Verifying computations with state. In *SOSP*, 2013.
19. G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In *ITCS*, 2012.
20. C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur. Geppetto: Versatile verifiable computation. In *IEEE SP*, 2015.
21. I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, 1991.
22. I. Damgård, S. Faust, and C. Hazay. Secure two-party computation with low communication. In *TCC*, 2012.
23. I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, 2010.
24. G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, 2014.
25. A. Faonio, J. B. Nielsen, and D. Venturi. Predictable arguments of knowledge. In *PKC*, 2017.
26. U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost np-complete (preliminary version). In *FOCS*, 1991.
27. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.
28. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *STOC*, 2013.
29. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, 2013.
30. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.
31. O. Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.
32. O. Goldreich and J. Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4), 1998.
33. O. Goldreich, S. P. Vadhan, and A. Wigderson. On interactive proofs with a laconic prover. In *ICALP*, 2001.
34. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.
35. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, 1985.
36. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO*, 2009.
37. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, 2010.
38. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, 2016.
39. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In *CRYPTO*, 2017.

40. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In *CRYPTO*, 1998.
41. Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Efficient arguments without short PCPs. In *CCC*, 2007.
42. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.
43. Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *TCC*, 2009.
44. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, 1992.
45. H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, 2012.
46. H. Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT*, 2013.
47. H. Lipmaa. Prover-efficient commit-and-prove zero-knowledge snarks. In *AFRICACRYPT*, 2016.
48. C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *FOCS*, 1990.
49. S. Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4), 2000.
50. T. Mie. Polylogarithmic two-round argument systems. *J. Mathematical Cryptology*, 2(4), 2008.
51. M. Naor. On cryptographic assumptions and challenges. In *CRYPTO*, 2003.
52. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, 2013.
53. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, 2014.
54. S. T. V. Setty, R. McPherson, A. J. Blumberg, and M. Walfish. Making argument systems for outsourced computation practical (sometimes). In *NDSS*, 2012.
55. S. T. V. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security Symposium*, 2012.
56. A. Shamir. IP=PSPACE. In *FOCS*, 1990.
57. J. Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO*, 2013.
58. J. Thaler, M. Roberts, M. Mitzenmacher, and H. Pfister. Verifiable computation with massively parallel interactive proofs. In *HotCloud*, 2012.
59. P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, 2008.
60. V. Vu, S. T. V. Setty, A. J. Blumberg, and M. Walfish. A hybrid architecture for interactive verifiable computation. In *IEEE SP*, 2013.
61. R. S. Wahby, M. Howald, S. J. Garg, A. Shelat, and M. Walfish. Verifiable asics. In *IEEE Symposium on Security and Privacy*, 2016.
62. R. S. Wahby, Y. Ji, A. J. Blumberg, A. Shelat, J. Thaler, M. Walfish, and T. Wies. Full accounting for verifiable outsourcing. In *ACM CCS*, 2017.
63. R. S. Wahby, S. T. V. Setty, Z. Ren, A. J. Blumberg, and M. Walfish. Efficient RAM and control flow in verifiable outsourced computation. In *NDSS*, 2015.
64. M. Walfish and A. J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2), 2015.
65. H. Wee. On round-efficient argument systems. In *ICALP*, 2005.