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

Dan Boneh[*] Yuval Ishai[†] Amit Sahai[‡] David J. Wu[§]

**Abstract**

Succinct non-interactive arguments (SNARGs) enable verifying $\mathsf{NP}$ computations with significantly less complexity than that required for classical $\mathsf{NP}$ verification. In this work, we focus on simultaneously minimizing the proof size and the prover complexity of SNARGs. Concretely, for a security parameter $\lambda$, we measure the asymptotic cost of achieving soundness error $2^{-\lambda}$ against provers of size $2^{\lambda}$. 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 $\mathsf{NP}$ prover. We show that this definition is the best we could hope for assuming that $\mathsf{NP}$ 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.

## 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 [GMR85, LFKN90, Sha90], zero-knowledge proofs [GMR85], probabilistically checkable proofs [BFLS91, FGL+91, ALM+98], and computationally sound proofs [Kil92, Mic00]. In this work, we study one such aspect: $\mathsf{NP}$ proof

[*]Stanford and Center for Encrypted Functionalities. Email: `dabo@cs.stanford.edu`.
[†]Technion, UCLA, and Center for Encrypted Functionalities. Email: `yuvali@cs.technion.ac.il`.
[‡]UCLA and Center for Encrypted Functionalities. Email: `sahai@cs.ucla.edu`.
[§]Stanford and Center for Encrypted Functionalities. Email: `dwu4@cs.stanford.edu`.

systems where the proofs can be significantly shorter than the $\mathsf{NP}$ witness and can be verified much faster than the time needed to check the $\mathsf{NP}$ witness. We say that such proof systems are *succinct*.

In interactive proof systems for $\mathsf{NP}$ with statistical soundness, non-trivial savings in communication and verification time are highly unlikely [BHZ87, GH98, GVW01, Wee05]. However, if we relax the requirements and consider proof systems with computational soundness, also known as *argument systems* [BCC88], significant efficiency improvements become possible. Kilian [Kil92] gave the first succinct four-round interactive argument system for $\mathsf{NP}$ based on collision-resistant hash functions and probabilistically checkable proofs (PCPs). Subsequently, Micali [Mic00] showed how to convert Kilian's four-round argument into a single-round argument for $\mathsf{NP}$ by applying the Fiat-Shamir heuristic [FS86] 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" [GW11]). In the standard model, single-round succinct arguments are highly unlikely for sufficiently hard languages [BP04a, Wee05], 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 $\lambda$, we measure the asymptotic cost of achieving soundness against provers of size $2^\lambda$ 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 $\mathsf{NP}$ 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 [BISW17], which imposed a weaker soundness requirement on the SNARG. Notably, under the definition in [BISW17], 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 Remark 5.3.

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 [BCI+13, BISW17]. To our knowledge, all previous candidates are either not quasi-optimal or rely on a heuristic security argument (Remark 5.4). Similar to previous works [BCI+13, BISW17], 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 [IKO07]). We then leverage cryptographic tools (e.g., *linear-only* encryption [BCI+13, BISW17]) 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 [BISW17]. 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. [BCI+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 $2^{\lambda}$. 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 $\mathsf{NP}$ 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 [BISW17] which only required soundness error $\mathrm{negl}(\lambda)$ against provers of size $2^{\lambda}$. In fact, we show in Section 5.1 that the previous notion of quasi-optimality from [BISW17] is not tight. If we only want $\rho$ bits of soundness where $\rho = o(\lambda)$, it is possible to construct a designated-verifier SNARG where the proofs are exactly $\rho$ bits. This means that there exists a designated-verifier SNARG which meet the soundness requirements in [BISW17], but whose size is strictly shorter than what would be considered "optimal." We provide a more detailed comparison of the two notions in Remark 5.3.

**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 [Mic00]. 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 [Gro10, Lip12, GGPR13, BCI+13, Lip13, DFGK14, Gro16, BISW17]. With the exception of [BISW17], 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 $\lambda$. Consequently, none of these existing constructions satisfy our notion of quasi-optimal prover complexity. The lattice-based construction in [BISW17] 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.

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

**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 [BCI$^+$13, BISW17], 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 [IKO07, BCI$^+$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 $\ell$ queries are made to the linear PCP oracle, the $\ell$ 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 $\ell$ 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 $\ell$-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 Appendix A.1.

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 $\widetilde{O}(|C|) + \operatorname{poly}(\lambda, \log |C|)$, and the linear MIP provides soundness error $2^{-\lambda}$. As we note in Remark A.6, existing linear PCP constructions [BCI$^+$13, BISW17] (which can be viewed as linear MIPs with a single prover) are not quasi-optimal: they either require embedding the Boolean circuit into an arithmetic circuit over a large field [BCI$^+$13], or rely on making $O(\lambda)$ queries, each of length $m = O(|C|)$ [BISW17].

**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 Remark B.7.

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 $1/\text{poly}(\lambda)$ 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 Appendix B.1 by combining MPC protocols with polylogarithmic overhead [DIK10] with the "MPC-in-the-head" paradigm [IKOS07]. 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. [BISW17]. 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 5.8 and 5.9).

**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 [BCI+13, BISW17], which use different information-theoretic primitives as the underlying building block (namely, linear interactive proofs [BCI+13] and linear PCPs [BISW17]). After all, any $k$-query, $\ell$-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 $\ell$. 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 [Mic00, BCCT12, BCC+14] 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 [Mic00], this kind of construction achieves quasi-optimal prover complexity, but not quasi-optimal succinctness [BISW17, Remark 4.16]. In the standard model [BCCT13, BCC+14], 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 [BCI+13, BISW17], 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 (and in Remark A.6), existing linear PCP constructions [BCI+13, BISW17] 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 $\rho$ bits of soundness where $\rho$ 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 [Gro10, GGPR13, BCI+13, BISW17]. 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 $\rho$ 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 $\rho$: namely, to build a SNARG with soundness error $2^{-\rho}$, we concatenate $\rho$ independent instances of a 1-bit SNARG.

In Section 6, we show that the designated-verifier analog of the Sahai-Waters [SW14] 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 [GGSW13] allows anyone to encrypt a message $m$ with respect to a statement $x$ in an $\mathsf{NP}$ language; then, anyone who holds a witness $w$ for $x$ is able to decrypt the ciphertext. In Section 6.2, 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

[2]Note that for some special languages such as graph non-isomorphism, we do have 1-bit laconic arguments [Gol01].

[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.

this is a relaxation of the usual notion of witness encryption, it already suffices to realize some of the powerful applications of witness encryption described in [GGSW13]. 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 $\pi$ 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. [AIK10] 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. [BDRV17] 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., [FNV17]). A predictable argument is one where there is exactly one accepting proof for any statement. Faonio et al. [FNV17] 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. We discuss this in greater detail in Remark 6.13.

## 1.3 Additional Related Work

Gentry and Wichs [GW11] showed that no construction of an *adaptively-secure* SNARG (for general $\mathsf{NP}$ languages) can be proven secure via a black-box reduction from any falsifiable cryptographic assumption [Nao03].[4] As a result, most existing SNARG constructions (for general $\mathsf{NP}$ languages) in the standard model have relied on non-falsifiable assumptions such as knowledge-of-exponent assumptions [Dam91, HT98, BP04b, Mie08, Gro10, Lip12, GGPR13, Lip13, DFGK14, Lip16, GM17], extractable collision-resistant hashing [BCCT12, DFH12, BCC$^+$14], extractable homomorphic encryption [BC12, GGPR13], and linear-only encryption [BCI$^+$13, BISW17]. Other constructions have relied on showing security in idealized models such as the random oracle model [Mic00, Val08] or the generic group model [Gro16]. 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 [Gro10, Lip12, GGPR13, BCI$^+$13, Lip13, DFGK14, Lip16, GM17] which says that the proof $\boldsymbol{\pi}$ 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

[4]In the case of non-adaptive SNARGs, Sahai and Waters give a construction from indistinguishability obfuscation and one-way functions [SW14].

setting [GKR08, CMT12, TRMP12, SMBW12, SVP+12, Tha13, VSBW13, WHG+16, WJB+17] as well as the non-interactive setting [PHGR13, BCG+13, BFR+13, BCTV14, WSR+15, CFH+15]. We refer to [WB15] 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 Appendix B.1, we describe one way of instantiating the robust decomposition by applying the "MPC-in-the-head" paradigm of [IKOS07] to MPC protocols with polylogarithmic overhead [DIK10]. 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., [DIK10]), 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 Appendix B.1.

**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 $i^{\text{th}}$ 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 $j^{\text{th}}$ 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 [Gro09], and subsequently used in [IPS09, BISW17] 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 $\Pi$ be a permutation over the entries in the matrix $\mathbf{A}$ such that $\Pi$ 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 [Gro09] 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 $\Pi$, if $\mathbf{W} = \Pi(\mathbf{W})$, this check always succeeds. However, if $\mathbf{W} \neq \Pi(\mathbf{W})$, then by the Schwartz-Zippel lemma (Lemma A.1), 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 $\lambda$ 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 $\lambda$ (since we would require $\Omega(\lambda)$ repetitions). This is one of the key reasons the construction in [BISW17] 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 [Gro09, IPS09, BISW17] 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 $\Pi$ 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 $\Pi$ 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 Appendix B.2.

**Decomposing the permutation.** Suppose $\Pi$ 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 $\Pi$ into a sequence of $\alpha$ permutations $(\Pi_1, \ldots, \Pi_\alpha)$ where $\Pi = \Pi_\alpha \circ \cdots \circ \Pi_1$. Moreover, each of the intermediate permutations $\Pi_i$ 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 $\Pi_i$ 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 $\Pi$. There are a number of technicalities that arise in the construction and its analysis, and we refer to Appendix B.2 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 $x \overset{\text{R}}{\leftarrow} S$ 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 $\lambda$ to denote the security parameter. We say that a function $f(\lambda)$ is negligible in $\lambda$ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We write $f(\lambda) = \mathrm{poly}(\lambda)$ to denote that $f$ is bounded by some (fixed) polynomial in $\lambda$, and $f = \mathrm{polylog}(\lambda)$ 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 Appendix A.

## 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 Appendices B.1 and B.2.

### 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 [IKOS07] with robust MPC protocols with polylogarithmic overhead [DIK10]. We provide the complete description in Appendix B.1.

**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 $(t, \delta)$-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 $i \in [t]$.
- **$\delta$-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 $\delta$-fraction of the constraints.
- **Efficiency:** The mappings $\mathsf{inp}$ and $\mathsf{wit}$ 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. [BCG+13, 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** ([BCG+13, 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 $q \le m$ 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 $\mathbf{W}_{i_1,j_1} \ne \mathbf{W}_{i_2,j_2}$. 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 $(t, \delta)$-robust decomposition (Definition 4.1). For a constant $\delta > 0$, let $(f_1, \ldots, f_t, \mathsf{inp}, \mathsf{wit})$ be a $(t, \delta)$-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 $\delta$-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 Lemma B.21.

**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 $\Pi$ 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 $\Pi'$ 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 $i^{\text{th}}$ row of $\mathbf{W}$ is $\boldsymbol{\pi}_i$. Similarly, we associate a matrix $\mathbf{W}'$ with provers $(P'_1, \ldots, P'_t)$. Let $\Pi$ 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 $\mathbf{R} \xleftarrow{\text{R}} \mathbb{F}^{t \times d}$, and sets $\mathbf{R}' \leftarrow \Pi(\mathbf{R})$. Let $\mathbf{r}_i$ and $\mathbf{r}'_i$ denote the $i^{\text{th}}$ 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 $\Pi$ is 2-locally decomposable, we can decompose $\Pi$ into $t' = t/2$ independent permutations, $\Pi_1, \ldots, \Pi_{t'}$, where each $\Pi_i$ 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 $t$, $\Pi$, $\mathbf{W}$, and $\mathbf{W}'$ as in Construction 4.8. Then, if the matrix $\mathbf{W}'$ disagrees with $\Pi(\mathbf{W})$ on $\kappa$ 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 $\kappa$ 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:

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

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 $\Pi$ 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 [Gro09] (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 $\Pi_i$ be a permutation over 2-by-$q$ matrices such that $\Pi_i$ splits into a disjoint set of Hamiltonian cycles based on the entries of $\mathbf{A}_{[2i-1,2i]}$. Define a permutation $\Pi$ on $t$-by-$q$ matrices where the action of $\Pi$ on rows $2i-1$ and $2i$ is given by $\Pi_i$ for all $i \in [t']$. By construction, the permutation $\Pi$ 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 $\Pi$ 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 $\Pi$, 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 $\Pi$ 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 $\Pi$ 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 $\Pi$ 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 $\Pi$ is *$\rho$-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 $s/\rho$. Moreover, there is a one-to-one correspondence between the edges in $M'$ and a subset of the edges in $M$ (as determined by $\Pi$). We say that $(\Pi_1, \ldots, \Pi_z)$ is a collection of $\rho$-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 $\Pi_\beta$ is $\rho$-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 $\rho$-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 $\Pi_\beta$ 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 $\Pi$. 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 $\Pi$ is 2-locally decomposable, so a natural starting point is to decompose the permutation $\Pi$ into a sequence of 2-locally-decomposable permutations $\Pi_1, \ldots, \Pi_\alpha$, where $\Pi = \Pi_\alpha \circ \cdots \circ \Pi_1$. This is possible, for instance, by first applying Lemma A.8 and Construction A.12 to $\Pi$. 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 $\Pi$, 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 $\Pi$ 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 $\kappa_1$ 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 $\kappa_1$ 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 $\kappa_1$ 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 $\Gamma$ is a sequence of $(\kappa_1, \kappa_2)$-*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 $\kappa_1$ rows contains more than $\kappa_2$ 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 $(\kappa_1, \kappa_2)$-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 $(\kappa_1, \kappa_2)$-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 $\Pi_\beta$, 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 $\Pi_\beta$, 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 Appendix B.2), 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 (Theorem B.20) and analysis to Appendix B.3. By instantiating Construction 4.14 with the appropriate primitives (described in Appendices B.1 and B.2), 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 $(t, \delta)$-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 $\varepsilon$ 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 $\rho$-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 $\Pi_\beta$ that is $(\kappa_1, \kappa_2)$-non-concentrating for $\mathcal{B}_\beta$ (Definition 4.13). In particular, each $\Gamma_\beta^{(j)}$ is a sequence of $\alpha$ 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 $P^{(j)}_{\beta,\ell,i}$ 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 (as in Construction 4.8). 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 $i^{\text{th}}$ 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 $i^{\text{th}}$ 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 $i^{\text{th}}$ row is the vector $\mathbf{w}_i$. Now, for all $j \in [\gamma]$, $\beta \in [z]$, and $\ell \in [\alpha]$, let $\mathbf{W}^{(j)}_{\beta,\ell} = \big(\Pi^{(j)}_{\beta,\ell} \circ \Pi^{(j)}_{\beta,\ell-1} \circ \cdots \circ \Pi^{(j)}_{\beta,1}\big)(\mathbf{W})$. Let $\hat{\mathbf{W}}^{(j)}_{\beta,\ell} = \big[\mathbf{W}^{(j)}_{\beta,\ell}, \mathbf{0}^{t \times (d-q)}\big]$. The prover associates $\hat{\mathbf{W}}^{(j)}_{\beta,\ell}$ with provers $(P^{(j)}_{\beta,\ell,1}, \ldots, P^{(j)}_{\beta,\ell,t})$.

- **Verifier's query algorithm:** To simplify the description, we will sometimes state the query vectors the verifier submits to each prover $P_i$ and $P^{(j)}_{\beta,\ell,i}$ 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 $\mathsf{st}_i$. 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 $\Pi^{(j)}_{\beta,\ell}$ to obtain two query matrices $\mathbf{R}^{(j)}_{\beta,\ell}$ and $\mathbf{S}^{(j)}_{\beta,\ell} \in \mathbb{F}^{t \times q}$. The verifier pads the matrices to obtain $\hat{\mathbf{R}}^{(j)}_{\beta,\ell} = \big[\mathbf{R}^{(j)}_{\beta,\ell}, \mathbf{0}^{t \times (d-q)}\big]$ and $\hat{\mathbf{S}}^{(j)}_{\beta,\ell} = \big[\mathbf{S}^{(j)}_{\beta,\ell}, \mathbf{0}^{t \times (d-q)}\big]$. 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 $\big(P^{(j)}_{\beta,\ell-1,1}, \ldots, P^{(j)}_{\beta,\ell-1,t}\big)$.

    In addition, the verifier distributes the queries $\hat{\mathbf{S}}^{(j)}_{\beta,\ell}$ to provers $\big(P^{(j)}_{\beta,\ell,1}, \ldots, P^{(j)}_{\beta,\ell,t}\big)$. 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 $\Pi'_\beta$ 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 $\Pi'_\beta$ 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 $\mathbf{W}_{\beta,\alpha}^{(j)}$ 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 $\mathsf{st}_i$, 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 $\Pi_{\beta,\ell}^{(j)}$), and for each set of pairwise consistency query responses (for checking consistency of the final configurations $\Pi'_\beta$), 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 Appendix B.3. In Appendices B.1 and B.2, we 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 Appendix B.3.

**Theorem 4.15** (Quasi-Optimal Linear MIP)**.** *Fix a security parameter $\lambda$. 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 the underlying linear PCPs can be made robust against affine strategies with minimal overhead by applying the transformation in Remark A.3. 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 [BISW17]. 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 Appendix A.3 for the formal definition of a succinct non-interactive argument (SNARG), and to Appendix A.4 for the definitions of a linear-only vector encryption that we use in our construction.

## 5.1 Defining Quasi-Optimality

In this section, we formally define our notion of a quasi-optimal SNARG. Then, in Remarks 5.3 and 5.4, we compare our notion to the previous notion of quasi-optimality introduced in [BISW17], 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, $\Pi_{\mathsf{SNARG}}$ is *quasi-optimal* if it achieves $2^{-\lambda}$ soundness error against provers of size $2^{\lambda}$ and satisfies the following properties:

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

Next, in Lemma 5.2, we show that our notion of quasi-optimality is tight in the following sense: assuming $\mathsf{NP}$ does not have succinct *proofs*, any argument system for $\mathsf{NP}$ that provides soundness error $2^{-\lambda}$ must have proofs of length $\Omega(\lambda)$. .

**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 $\rho$ and a security parameter $\lambda$. 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 $\Pi_{\mathsf{SNARG}}$ is $\Omega(\rho)$.*

*Proof.* Let $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a SNARG for $\mathcal{L}$ with soundness error $2^{-\rho}$ against provers of size $\mathrm{poly}(\lambda)$ and argument length $\ell(\rho) \le c\rho$ for all constants $c > 0$. Let $1/2^{\delta}$ denote the probability that there exists a statement $\mathbf{x} \notin \mathcal{L}_{C_n}$ and a proof $\boldsymbol{\pi}$ such that $\mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) = 1$, where $(\sigma, \tau) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^n)$, and the probability is taken over the coins of the $\mathsf{Setup}$ algorithm. Since $\Pi_{\mathsf{SNARG}}$ has soundness error $2^{-\rho}$, it follows that $2^{-(\delta+\ell)} \le 2^{-\rho}$. Otherwise, a prover with $\mathbf{x}$ hard-wired inside it can guess $\boldsymbol{\pi}$ and break soundness with probability $2^{-(\delta+\ell)}$. Equivalently, this means that $\delta + \ell \ge \rho$. Since $\ell \le c\rho$ for all $c > 0$, this means that $\delta = \Omega(\rho)$. We use $\Pi_{\mathsf{SNARG}}$ to construct a *proof* system for $\mathcal{L}$ by concatenating $2 \cdot n/\delta$ instances of $\Pi_{\mathsf{SNARG}}$. The length of the proofs in this new system is then $2 \cdot n\ell/\delta = o(n)$, which is succinct. Moreover, for any false statement $\mathbf{x} \in \{0,1\}^n$, the probability that there exists a proof $\boldsymbol{\pi}$ that causes the verifier to accept is now $(1/2^{\delta})^{(2n/\delta)} = 1/2^{2n}$. Taking a union bound over all $2^n$ possible statements, the probability that there exists any false statement with a proof that convinces the verifier is at most $2^{-n}$. This yields a succinct proof system for $\mathcal{L}$ with soundness error $2^{-n}$, which contradicts the assumption that $\mathcal{L}$ does not have succinct proofs. Thus, there must exist some constant $c > 0$ such that $\ell > c\rho$, from which we conclude that $\ell = \Omega(\rho)$. □

**Remark 5.3** (Previous Notions of Quasi-Optimality)**.** Previously, Boneh et al. [BISW17] introduced a weaker notion of quasi-optimality that only required the prover complexity and succinctness properties in Definition 5.1 to hold for SNARG constructions that provide $\text{negl}(\lambda)$ soundness (as opposed to $2^{-\lambda}$ soundness) against provers of size $2^\lambda$. In the case of publicly-verifiable SNARGs, this notion still captures the right notion of quasi-optimal succinctness, since achieving any level of soundness against $2^\lambda$-bounded provers requires proofs of length $\Omega(\lambda)$. Otherwise, a cheating prover can just enumerate all of the possible proofs and run the verification algorithm to check whether a particular proof is valid or not. In the (non-reusable) designated-verifier setting, however, the prover does not learn whether a candidate proof is valid or not. This negates the basic enumeration strategy, and as we show in Construction 6.1 (Theorem 6.2), in the designated-verifier setting, achieving soundness $2^{-\rho}$ against $2^\lambda$-bounded provers is possible with proofs of length exactly $\rho$. Moreover, Lemma 5.2 shows that for general $\mathsf{NP}$ languages, this is the best we can hope to do. Thus, the notion of succinctness given in Definition 5.1 is the "correct" definition of quasi-optimal succinctness in the designated-verifier setting, and represents a strict strengthening of the corresponding notion introduced in [BISW17].

**Remark 5.4** (Heuristic Construction of Quasi-Optimal SNARGs)**.** One approach for constructing a quasi-optimal SNARG is to compose a SNARG that provides quasi-optimal prover complexity with one that is quasi-optimally succinct. To prove that $C(\mathbf{x}, \mathbf{w}) = 1$, the prover first constructs a proof $\boldsymbol{\pi}$ for the statement using the "inner" SNARG that provides quasi-optimal prover complexity. Then, the prover uses the "outer" SNARG that is quasi-optimally succinct to prove that it knows a proof $\boldsymbol{\pi}$ of the statement $\mathbf{x}$ under the inner SNARG. The additional cost of generating this second proof is proportional to the size of the verifier for the inner proof system, which is polylogarithmic in the size of $C$. Thus, this composition is quasi-optimal. Note that to show soundness of the composition, we additionally require that both SNARGs satisfies a knowledge property (namely, that they are SNARKs). However, to our knowledge, the only candidate SNARK with quasi-optimal prover efficiency is Micali's CS proofs [Mic00] in the random oracle model. Thus, composing CS proofs with a quasi-optimally succinct SNARK (e.g., [GGPR13, BCI$^+$13, BISW17]) can only yield a construction whose security is heuristic. This is because the prover in the outer SNARK needs access to the circuit description of the verification algorithm of the inner SNARK (in order to prove knowledge of an accepting proof $\boldsymbol{\pi}$), but the verification algorithm in the inner SNARK necessarily makes random oracle queries.

## 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 Appendix A.4 for the definition of a linear-only vector encryption from [BISW17]. We give our construction and security analysis below.

**Construction 5.5** (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 $\ell$ 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 $\lambda$ 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 $\mathsf{st}$.
2. Generate a secret key $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell)$ for the vector encryption scheme.
3. Pack the $\ell$ 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 $\ell$ 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 $i^{\text{th}}$ 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 $\ell$ 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 $\mathsf{ct}'$.

- $\mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) \to \{0, 1\}$: 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 $\mathsf{ct}'$ using the secret key $\mathsf{sk}$ 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 $\mathsf{st}$ and outputs $\mathcal{D}(\mathsf{st}, \mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_\ell)$.

**Theorem 5.6.** *Fix a security parameter $\lambda$ 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 $\ell$ 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 (Definition A.20). Then, applying Construction 5.5 to $(\mathcal{P}, \mathcal{V})$ and $\Pi_{\mathsf{venc}}$ yields a non-adaptive designated-verifier preprocessing SNARG with soundness error $2 \cdot \varepsilon(\lambda) + \operatorname{negl}(\lambda)$.*

*Proof.* The proof proceeds similarly to the analogous proofs in [BCI+13, Lemma 6.3] and [BISW17, Theorem 4.6]. Let $\mathcal{P}^*$ be a (non-adaptive) malicious prover that successfully convinces the verifier of some false statement $\mathbf{x} \notin \mathcal{L}_\mathcal{C}$ with some non-negligible probability $\varepsilon$. Let $\mathcal{V} = (\mathcal{Q}, \mathcal{D})$, where $\mathcal{Q}$ is the query-generation algorithm, and $\mathcal{D}$ be the verifier's decision algorithm. Next, since $\Pi_{\mathsf{venc}}$ satisfies linear targeted malleability, there exists a simulator $\mathcal{S}$ such that the following distributions are computationally indistinguishable:

**Real Distribution:**

1. $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n)$
2. $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}(1^\lambda, 1^n)$ where $\mathbf{Q} \in R^{d \times k}$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{q}_i)$ for all $i \in [d]$ where $\mathbf{q}_i \in R^k$ is the $i^{\text{th}}$ row of $\mathbf{Q}$
4. $\mathsf{ct}' \leftarrow \mathcal{P}^*(\mathsf{ct}_1, \ldots, \mathsf{ct}_q; \mathbf{x})$ where $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \neq \perp$
5. $\mathbf{y} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \in R^k$
6. Output $(\mathbf{Q}, \mathsf{st}, \mathbf{y})$

**Ideal Distribution:**

1. $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}(1^\lambda, 1^n)$ where $\mathbf{Q} \in R^{d \times k}$
2. $(\boldsymbol{\pi}, \mathbf{b}) \leftarrow \mathcal{S}(\mathbf{x})$ where $\boldsymbol{\pi} \in R^d$, $\mathbf{b} \in R^k$
3. $\hat{\mathbf{y}} \leftarrow \mathbf{Q}^\top \boldsymbol{\pi} + \mathbf{b}$
4. Output $(\mathbf{Q}, \mathsf{st}, \hat{\mathbf{y}})$

By assumption, $\mathcal{P}^*$ outputs a proof $\mathsf{ct}'$ that convinces the verifier with probability $\varepsilon'(\lambda)$. This means that, in the real distribution, $\mathcal{D}(\mathsf{st}, \mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_\ell) = 1$, where $\mathbf{y}_1, \ldots, \mathbf{y}_\ell \in \mathbb{F}_p^k$ is the decomposition of the vector $\mathbf{y} \in R^k$ (recall that each ring element in $R$ can be viewed as a vector of $\ell$ values in $\mathbb{F}_p$). Since the decision algorithm $\mathcal{D}$ is efficiently computable, this means that $\mathcal{D}(\mathsf{st}, \mathbf{x}, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell) = 1$ with probability at least $\varepsilon'(\lambda) - \operatorname{negl}(\lambda)$, where $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_\ell \in \mathbb{F}_p^k$ is the decomposition of $\hat{\mathbf{y}} \in R^k$. By construction, in the ideal distribution, the affine function $(\boldsymbol{\pi}, \mathbf{b})$ is generated *independently* of the verifier's queries $\mathbf{Q}$ and state $\mathsf{st}$. Thus, by an averaging argument, there exists some affine function $(\boldsymbol{\pi}^*, \mathbf{b}^*)$ such that with probability at least $\varepsilon'(\lambda)/2 - \operatorname{negl}(\lambda)$, we have that $\mathcal{D}(\mathbf{x}, \mathsf{st}, \mathbf{y}_1^*, \ldots, \mathbf{y}_\ell^*) = 1$, where $\mathbf{y}_1^*, \ldots, \mathbf{y}_\ell^* \in \mathbb{F}_p^k$ are the components of $\mathbf{y}^* = \mathbf{Q}^\top \boldsymbol{\pi}^* + \mathbf{b}^*$. Since the linear MIP has soundness error $\varepsilon(\lambda)$, it must be the case that $\varepsilon'(\lambda)/2 - \operatorname{negl}(\lambda) < \varepsilon(\lambda)$, and the claim follows. □

**Theorem 5.7.** *Fix a security parameter $\lambda$ 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 $\ell$ 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 (Definition A.21). Then, applying Construction 5.5 to $(\mathcal{P}, \mathcal{V})$ and $\Pi_{\mathsf{venc}}$ yields an adaptive designated-verifier preprocessing SNARG with soundness error $\varepsilon(\lambda) + \operatorname{negl}(\lambda)$.*

*Proof (Sketch).* The proof proceeds similarly to the analogous proof in [BCI+13, Lemma 6.2]. Let $\mathcal{P}^*$ be a malicious prover that takes as input the common reference string $\sigma = (\mathsf{ct}_1, \ldots, \mathsf{ct}_d)$ and outputs a statement $\mathbf{x} \notin \mathcal{L}_C$ and a proof $\mathsf{ct}'$ that is accepted by the verifier with probability $\varepsilon'(\lambda)$. By construction, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_d$ are encryptions of the rows $\mathbf{q}_i \in R^k$ of a query matrix $\mathbf{Q} \in R^{d \times k}$. Since $\Pi_{\mathsf{venc}}$ is a linear-only vector encryption scheme, there exists an efficient extractor $\mathcal{E}$ that extracts an affine function $(\boldsymbol{\pi}^*, \mathbf{b}^*) \in R^d \times R^k$ from the ciphertexts $\{\mathsf{ct}_i\}_{i \in [d]}$ such that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') = \mathbf{Q}^\top \boldsymbol{\pi}^* + \mathbf{b}^* \in R^k$ with probability $1 - \operatorname{negl}(\lambda)$. Let $\mathbf{y}^* = \mathbf{Q}^\top \boldsymbol{\pi}^* + \mathbf{b}^* \in R^k$ and let $\mathbf{y}_1^*, \ldots, \mathbf{y}_\ell^* \in \mathbb{F}_p^k$ be the components of $\mathbf{y}^* \in R^k$. Since the verifier accepts the proof $\mathsf{ct}'$, it must be the case that $\mathcal{D}(\mathsf{st}, \mathbf{x}, \mathbf{y}_1^*, \ldots, \mathbf{y}_\ell^*) = 1$. Moreover, by semantic security of $\Pi_{\mathsf{venc}}$, the extracted proof $(\boldsymbol{\pi}^*, \mathbf{b}^*)$ cannot be locally satisfying (i.e., $(\boldsymbol{\pi}^*, \mathbf{b}^*)$ is a valid proof only with respect to the particular $\mathbf{Q}$ encrypted in $\sigma$), but rather, it must be satisfying for all but a negligible fraction of queries $\mathbf{Q}$. Otherwise, if the output distribution of the extractor $\mathcal{E}$ is noticeably different when run on different choices of the query matrix $\mathbf{Q}$, then $\mathcal{E}$ can be used to break semantic security of $\Pi_{\mathsf{venc}}$. Thus, if $\mathcal{P}^*$ succeeds in convincing the verifier of a statement $\mathbf{x} \notin \mathcal{L}_\mathcal{C}$ with probability $\varepsilon'(\lambda)$, then with probability $\varepsilon'(\lambda) - \operatorname{negl}(\lambda)$, the extractor $\mathcal{E}$ for $\Pi_{\mathsf{venc}}$ can be used to produce a proof $(\boldsymbol{\pi}^*, \mathbf{b}^*)$ of $\mathbf{x} \notin \mathcal{L}_\mathcal{C}$ that is accepted by the linear MIP verifier. Since the linear MIP has soundness error $\varepsilon(\lambda)$, we conclude that $\varepsilon'(\lambda) \leq \varepsilon(\lambda) + \operatorname{negl}(\lambda)$. □

**Instantiating the construction.** To conclude this section, we show that combining the candidate vector encryption scheme $\Pi_{\mathsf{venc}}$ over polynomial rings $R^k$, where $R \cong \mathbb{F}_p^\ell$ from [BISW17, §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 [BISW17, §4.4] that the candidate vector encryption scheme $\Pi_{\mathsf{venc}}$ has the following properties:

- When $k = \operatorname{polylog}(\lambda)$, $\ell = \widetilde{O}(\lambda)$, and $|\mathbb{F}| = \operatorname{poly}(\lambda)$, 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 5.5 to the linear MIP from Theorem 4.15 and $\Pi_{\mathsf{venc}}$, 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 $\widetilde{O}(s) + \operatorname{poly}(\lambda, \log s)$, where $s$ is the size of the circuit. To construct the proof, the prover has to perform $d$ homomorphic operations, where $d = \widetilde{O}(s/\lambda) + \operatorname{poly}(\lambda, \log s)$ 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 $\widetilde{O}(s) + \operatorname{poly}(\lambda, \log s)$.
- **Proof size:** The proof in Construction 5.5 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 5.8.** *Assuming the vector encryption scheme* $\Pi_{\mathsf{venc}}$ *from [BISW17, §4.4] satisfies linear targeted malleability (with exponential security), then applying Construction 5.5 to the quasi-optimal linear MIP from Theorem 4.15 and* $\Pi_{\mathsf{venc}}$ *yields a non-adaptive designated-verifier quasi-optimal SNARG for Boolean circuit satisfiability in the preprocessing model.*

**Corollary 5.9.** *Assuming the vector encryption scheme* $\Pi_{\mathsf{venc}}$ *from [BISW17, §4.4] (with the "double-encryption" transformation described in [BISW17, Remark C.4]) is linear-only (with exponential security), then applying Construction 5.5 to the quasi-optimal linear MIP from Theorem 4.15 and* $\Pi_{\mathsf{venc}}$ *yield an adaptive designated-verifier quasi-optimal SNARG for Boolean circuit satisfiability in the preprocessing model.*

Construction 5.5 gives a construction of a *single-theorem* SNARG from any linear MIP system. Below, we discuss some of the challenges in extending our construction to provide multi-theorem security.

**Remark 5.10** (Multi-Theorem SNARGs)**.** Construction 5.5 gives a construction of a *single-theorem* SNARG from any linear MIP system. The works of [BCI$^+$13, BISW17] show how to construct multi-theorem designated-verifier SNARGs by relying on a stronger notion of soundness at the linear PCP level coupled with a stronger interactive linear-only encryption assumption. While we could rely on the same type of cryptographic assumption as in [BISW17], our linear MIP from Section 4 does not satisfy the notion of "reusable" or "strong" soundness from [BCI$^+$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.

## 6 Optimally-Succinct SNARGs and Laconic Arguments

Recall from Section 1 that a "1-bit SNARG" is a SNARG that achieves soundness error $1/2 + \text{negl}(\lambda)$ with just a *single* bit of proof. In this section, we show that a variant of the Sahai-Waters NIZK construction from indistinguishability obfuscation (and one-way functions) [SW14] can be used to construct a 1-bit SNARG. As noted in the introduction (Section 1), we can also view a 1-bit SNARG as a 1-bit laconic interactive argument system. Thus, our results also gives the first 1-bit laconic argument system for $\mathsf{NP}$ assuming indistinguishability obfuscation and one-way functions. Then, in Section 6.2, we show that in the interactive setting, we can also build 1-bit laconic arguments from witness encryption [GGSW13]. Here, we also demonstrate that a variant of the converse holds: namely, a 1-bit argument system for a "cryptographically-hard" language implies a relaxed notion of witness encryption for the same language. While the notion of witness encryption we obtain is weaker than the traditional one, we show that it still suffices for instantiating some of the main applications of witness encryption.

### 6.1 1-Bit SNARGs from Indistinguishability Obfuscation

In this section, we show how to construct 1-bit SNARGs from indistinguishability obfuscation ($i\mathcal{O}$). We refer to Appendix A.5 for the definitions of $i\mathcal{O}$ and puncturable PRFs that we rely on for our construction. Our construction is essentially the Sahai-Water NIZK [SW14] specialized to the designated-verifier setting. The CRS is an obfuscated program that takes as input a statement $\mathbf{x}$ and a witness $\mathbf{w}$, and outputs a 1-bit PRF on $\mathbf{x}$ if $C(\mathbf{x}, \mathbf{w}) = 1$, and $\perp$ otherwise. The PRF key is hard-coded inside the obfuscated program. The secret verification key is the PRF key. We can essentially view the CRS as an obfuscated program that checks whether $(\mathbf{x}, \mathbf{w})$ is a satisfying assignment, and if so, outputs a 1-bit message authenticated code (MAC) on the statement $\mathbf{x}$. We now give the construction and its security analysis.

**Construction 6.1** (1-Bit SNARG from $i\mathcal{O}$)**.** Let $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ be a family of Boolean circuits, where each $C_n \colon \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}$ for all $n \in \mathbb{N}$. Let $\mathcal{R}_{\mathcal{C}}$ be the associated relation and $\mathcal{L}_{\mathcal{C}}$ be the associated language. Let $\mathsf{F}_n \colon \mathcal{K}_n \times \{0,1\}^n \to \{0,1\}$ be a puncturable PRF family (indexed by a parameter $n$). We construct a 1-bit designated-verifier SNARG $\Pi_{\mathsf{SNARG}} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ in the preprocessing model for $\mathcal{R}_{\mathcal{C}}$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm samples a puncturable PRF key $k \leftarrow \mathsf{F}_n.\mathsf{Setup}(1^\lambda)$, and constructs the obfuscated program $P \leftarrow i\mathcal{O}(\mathsf{Prove}[C_n, k])$,[5] where the program $\mathsf{Prove}[C_n, k]$ is defined as follows:

[5]Note that we pad the program $\mathsf{Prove}[C_n, k]$ to the maximum size of any program that appears in the proof of Theorem 6.2.

**Constants:** a circuit $C_n \colon \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}$ and a key $k$ for $\mathsf{F}_n$

On input $\mathbf{x} \in \{0,1\}^n, \mathbf{w} \in \{0,1\}^{m(n)}$:

1. If $C(\mathbf{x}, \mathbf{w}) = 1$, then output $\mathsf{F}_n(k, \mathbf{x})$. Otherwise, output $\perp$.

Figure 1: The program $\mathsf{Prove}[C_n, k]$

The setup algorithm outputs the common reference string $\sigma = P$ and the verification state $\tau = k$.

- $\mathsf{Prove}(\sigma, \mathbf{x}, \mathbf{w}) \to \pi$: On input the common reference string $\sigma = P$ a statement $\mathbf{x} \in \{0,1\}^n$ and a witness $\mathbf{w} \in \{0,1\}^m$, the prover runs $P$ on $(\mathbf{x}, \mathbf{w})$ to obtain a proof $\pi \leftarrow P(\mathbf{x}, \mathbf{w})$ and outputs $\pi \in \{0,1\}$.
- $\mathsf{Verify}(\tau, \mathbf{x}, \pi) \to \{0,1\}$: On input the secret verification state $\tau = k$, a statement $\mathbf{x} \in \{0,1\}^n$, and a proof $\pi \in \{0,1\}$, the verifier outputs 1 if $\pi = \mathsf{F}_n.\mathsf{Eval}(k, \mathbf{x})$, and 0 otherwise.

**Theorem 6.2.** *Suppose $\mathsf{F}_n$ is a family of puncturable PRFs and $i\mathcal{O}$ is an indistinguishability obfuscator. Then, Construction 6.1 is a non-adaptive designated-verifier 1-bit SNARG for $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ in the preprocessing model. In particular, $\Pi_{\mathsf{SNARG}}$ achieves soundness error $1/2 + \operatorname{negl}(\lambda)$ against polynomial-time bounded provers.*[6]

*Proof.* Completeness of the construction follows immediately by correctness of the indistinguishability obfuscator, so it suffices to show soundness. Take any statement $\mathbf{x}^* \notin \mathcal{L}_{C_n}$. We show that no efficient prover can produce a proof $\pi^* \in \{0,1\}$ where $\mathsf{Verify}(\tau, \mathbf{x}^*, \pi^*) = 1$ except with probability $1/2 + \operatorname{negl}(\lambda)$ over the randomness used to sample $(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$. Our proof follows the same structure as the corresponding proof in [SW14, Theorem 9]. In particular, we define a sequence of hybrid arguments:

- $\mathsf{Hyb}_0$: This is the real game, where the challenger generates $(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, and gives $\sigma$ to the prover. The prover outputs $\pi^* \in \{0,1\}$. The output of the experiment is 1 if $\mathsf{Verify}(\tau, \mathbf{x}^*, \boldsymbol{\pi}^*) = 1$, and 0 otherwise.
- $\mathsf{Hyb}_1$: This is the same game as $\mathsf{Hyb}_0$, except during the setup algorithm, the challenger first computes $k_{\mathbf{x}^*} \leftarrow \mathsf{F}_n.\mathsf{Puncture}(k, \mathbf{x}^*)$. When constructing the obfuscated program $P$, the challenger replaces the invocation $\mathsf{F}_n(k, \mathbf{x})$ with $\mathsf{F}_n.\mathsf{Eval}(k_{\mathbf{x}^*}, \mathbf{x})$.
- $\mathsf{Hyb}_2$: This is the same game as $\mathsf{Hyb}_1$, except in the verification procedure, the challenger samples $b \xleftarrow{\text{R}} \{0,1\}$ and accept if $\pi^* = b$.

We now argue that each pair of consecutive hybrid experiments is computationally indistinguishable.

- Hybrids $\mathsf{Hyb}_0$ and $\mathsf{Hyb}_1$ are computationally indistinguishable by security of $i\mathcal{O}$ and correctness of the puncturable PRF family. In particular, by correctness of the puncturable PRF, $\mathsf{F}_n(k, \cdot)$

[6]If we make the stronger assumption that the underlying primitives (the indistinguishability obfuscator and the puncturable PRF family) are secure against subexponential-time adversaries, then we correspondingly achieve soundness error $1/2 + \operatorname{negl}(\lambda)$ against all subexponential-time provers.

and $\mathsf{F}_n.\mathsf{Eval}(k_{\mathbf{x}^*}, \cdot)$ agree on all inputs $\mathbf{x} \in \{0,1\}^n$ where $\mathbf{x} \neq \mathbf{x}^*$. Moreover, the programs $P$ in $\mathsf{Hyb}_0$ and $\mathsf{Hyb}_1$ never needs to evaluate the PRF at $\mathbf{x}^*$, since by assumption, for all $\mathbf{w} \in \{0,1\}^m$, $C(\mathbf{x}^*, \mathbf{w}) = 0$. This means that the outputs of $P$ in $\mathsf{Hyb}_0$ and $\mathsf{Hyb}_1$ are identical on all inputs $(\mathbf{x}, \mathbf{w})$. Indistinguishability then follows by security of $i\mathcal{O}$.

- Hybrids $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$ are computationally indistinguishable by security of the punctured PRF. Concretely, suppose there is an adversary $\mathcal{A}$ that can distinguish the outputs of $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$. We can build an adversary $\mathcal{B}$ that breaks the puncturing security of $\mathsf{F}_n$ as follows. Algorithm $\mathcal{B}$ submits $\mathbf{x}^*$ as the punctured point to the puncturing security challenger and receives a punctured key $k_{\mathbf{x}^*}$ and a challenge value $y \in \{0,1\}$. Algorithm $\mathcal{B}$ constructs the obfuscated program $P$ as in $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$ using the punctured key $k_{\mathbf{x}^*}$, and gives $\sigma = P$ to adversary $\mathcal{A}$. At the end of the experiment, after $\mathcal{A}$ output $\pi^* \in \{0,1\}$, $\mathcal{B}$ outputs 1 if $y = \pi^*$ and 0 otherwise. By construction, $\mathcal{B}$ perfectly simulates the output distribution of $\mathsf{Hyb}_1$ if $y = \mathsf{F}_n(k, \mathbf{x}^*)$ is pseudorandom and the output distribution of $\mathsf{Hyb}_2$ if $y$ is truly random. We conclude that by puncturing security of $\mathsf{F}_n$, hybrids $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$ are computationally indistinguishable.

To conclude the proof, we note that the output distribution of $\mathsf{Hyb}_2$ is 1 with probability 1/2 (since $y$ is uniform and independent of the prover's view). Since $\mathsf{Hyb}_0$ and $\mathsf{Hyb}_2$ are computationally indistinguishable, this means that the output of $\mathsf{Hyb}_0$ is 1 with probability at most $1/2 + \mathrm{negl}(\lambda)$. We conclude that $\Pi_{\mathsf{SNARG}}$ provides soundness error $1/2 + \mathrm{negl}(\lambda)$. □

**Remark 6.3** (Additional Properties)**.** By the same argument in [SW14, Theorem 8], the 1-bit SNARG in Construction 6.1 is perfect zero-knowledge. Moreover, by a standard hybrid argument, we can show that it is non-adaptively reusable in the following sense. For any set of statements $\mathbf{x}_1, \ldots, \mathbf{x}_k \notin \mathcal{L}_{C_n}$, the probability that a malicious prover can produce proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_k$ such that $\mathcal{V}(\tau, \mathbf{x}_i, \boldsymbol{\pi}_i) = 1$ for all $i \in [k]$ and $(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ is bounded by $1/2^k + \mathrm{negl}(\lambda)$.

**Remark 6.4** (Adaptivity via VBB Obfuscation)**.** If we replace the indistinguishability obfuscator in Construction 6.1 with a VBB obfuscator [BGI+01], then it is straightforward to prove that the resulting construction is adaptively sound. We leave it as an open problem to construct adaptively sound 1-bit SNARGs without relying on such strong forms of obfuscation.

## 6.2 1-Bit Laconic Arguments and Witness Encryption

A 1-bit SNARG immediately implies a 2-round laconic argument where the prover communicates just a *single* bit. Namely, in the first round, the verifier runs the setup algorithm for the 1-bit SNARG, and sends the CRS to the prover. The prover's response consists of the 1-bit SNARG proof for the statement. Thus, Theorem 6.2 shows that assuming the existence of indistinguishability obfuscation and one-way functions, there exists a 1-bit laconic argument for $\mathsf{NP}$.

While we do not know of alternative constructions of 1-bit SNARGs from weaker assumptions, such constructions are possible in the interactive setting. Importantly, in the interactive setting, the verifier's initial message (i.e., the $\mathsf{Setup}$ algorithm) can depend on the statement $\mathbf{x}$, while in the standard non-interactive setting, the setup algorithm (that generates the CRS) is independent of the statement. We now show how to construct a 2-round laconic argument for $\mathsf{NP}$ from any semantically-secure witness encryption scheme for $\mathsf{NP}$ [GGSW13]. Recall that in a witness encryption scheme for an $\mathsf{NP}$ relation $\mathcal{R}$ (and associated language $\mathcal{L}$), the encrypter can encrypt a message $m$ with respect to a statement $\mathbf{x}$. Then, anyone who knows a witness $\mathbf{w}$ such that $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$ is able to decrypt. The security guarantee states that ciphertexts encrypted to a statement $\mathbf{x} \notin \mathcal{L}$ are semantically secure. We review the definition of witness encryption below, and then describe our 2-round laconic argument construction from witness encryption.

**Definition 6.5** (Witness Encryption [GGSW13])**.** A witness encryption for an $\mathsf{NP}$ language $\mathcal{L}$ (with corresponding $\mathsf{NP}$ relation $\mathcal{R}$) is a tuple of algorithms $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ with the following properties:

- $\mathsf{Encrypt}(1^\lambda, \mathbf{x}, m) \to \mathsf{ct}$: On input the security parameter $\lambda$, a statement $\mathbf{x}$ and a message $m \in \{0,1\}$, the encryption algorithm outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Decrypt}(\mathsf{ct}, \mathbf{w}) \to m'$: On input a ciphertext $\mathsf{ct}$ and a witness $\mathbf{w}$, the decryption algorithm outputs a message $m' \in \{0,1\} \cup \{\perp\}$.

Moreover, $\Pi_{\mathsf{WE}}$ must satisfy the following properties:

- **Correctness:** For all messages $m \in \{0,1\}$, and any statement-witness pair $(\mathbf{x}, \mathbf{w})$ where $R(\mathbf{x}, \mathbf{w}) = 1$, it follows that
$$\Pr[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, m), \mathbf{w}) = m] = 1.$$
- **Semantic Security:** For all efficient adversaries $\mathcal{A}$, and all statements $\mathbf{x} \notin \mathcal{L}$,
$$\left| \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, 0)) = 1] - \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, 1)) = 1] \right| = \operatorname{negl}(\lambda). \tag{6.1}$$

**Remark 6.6** (Equivalent Security Notion)**.** In our analysis, it will often be easier to work with the following equivalent notion of security for witness encryption:

- **Unguessable:** For all efficient adversaries $\mathcal{A}$, and all statements $\mathbf{x} \notin \mathcal{L}$,
$$\left| \Pr[m \xleftarrow{\text{R}} \{0,1\} : \mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, m)) = m] - \frac{1}{2} \right| = \operatorname{negl}(\lambda). \tag{6.2}$$

To see the equivalence, take any adversary $\mathcal{A}$. Without loss of generality, assume that $\mathcal{A}$ always outputs a bit. Let $p_0 = \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, 0)) = 0]$ and $p_1 = \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, 1)) = 1]$. Then, the guessing advantage (Eq. (6.2)) of $\mathcal{A}$ is $|(1 - p_0 - p_1)/2|$, and the distinguishing advantage (Eq. (6.1)) of $\mathcal{A}$ is $|1 - p_0 - p_1|$. In particular, this means that if the guessing advantage of $\mathcal{A}$ is $\varepsilon$, then the distinguishing advantage of $\mathcal{A}$ is $2\varepsilon$.

**Construction 6.7** (1-Bit Laconic Argument for $\mathsf{NP}$ from Witness Encryption)**.** Let $\mathcal{L}$ be an $\mathsf{NP}$ language, and let $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ be a witness encryption scheme for $\mathcal{L}$. We construct an interactive 1-bit laconic argument system $\Pi_{\mathsf{arg}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ as follows:[7]

- $\mathsf{Setup}(1^\lambda, \mathbf{x}) \to (\sigma_{\mathbf{x}}, \tau_{\mathbf{x}})$: On input the security parameter $\lambda$ and a statement $\mathbf{x}$, the setup algorithm chooses a random message $m \xleftarrow{\text{R}} \{0,1\}$ and computes the ciphertext $\mathsf{ct} \leftarrow \mathsf{Encrypt}(1^\lambda, \mathbf{x}, m)$. It outputs the initial message $\sigma_{\mathbf{x}} = \mathsf{ct}$ and the verification state $\tau_{\mathbf{x}} = m$.

[7]We use the same schema as a SNARG (Definition A.18) to describe our 2-round interactive argument. The main difference is that the verifier's first message (i.e., the output of the $\mathsf{Setup}$ algorithm) can depend on the statement $\mathbf{x}$.

- $\mathsf{Prove}(\sigma_{\mathbf{x}}, \mathbf{w}) \to \pi$: On input the verifier's initial message $\sigma_{\mathbf{x}} = \mathsf{ct}$ and a witness $\mathbf{w}$, the prover computes $m' \leftarrow \mathsf{Decrypt}(\mathsf{ct}, \mathbf{w})$, and outputs the proof $\pi = m'$.
- $\mathsf{Verify}(\tau_{\mathbf{x}}, \pi) \to \{0,1\}$: On input the verification state $\tau = m$ and the proof $\pi = m'$, the verifier outputs 1 if $m = m'$, and 0 otherwise.

**Theorem 6.8** (Laconic Arguments from Witness Encryption)**.** *Let $\mathcal{L}$ be an* NP *language. If $\Pi_{\mathsf{WE}}$ is a witness encryption scheme for $\mathcal{L}$, then Construction 6.7 is a 1-bit laconic argument for $\mathcal{L}$.*

*Proof.* Completeness follows from correctness of the witness encryption scheme. Soundness follows from security of the witness encryption scheme. Namely, if $\mathbf{x} \notin \mathcal{L}$, then $\mathsf{ct}$ is a semantically-secure encryption of the message $m \in \{0,1\}$. Since $m$ is sampled uniformly at random, the probability that the adversary outputs $m'$ where $m' = m$ is at most $1/2 + \mathrm{negl}(\lambda)$. □

This construction shows that any witness encryption for a language $\mathcal{L}$ yields a 1-bit laconic interactive argument system for the same language $\mathcal{L}$. It is unclear how to leverage this construction to construct a 1-bit preprocessing SNARG (critically, the verifier's message is *not* oblivious, and depends on the underlying statement). We leave it as an open construction to construct 1-bit SNARGs from witness encryption or other weaker assumptions. Along those same lines, it is also interesting to construct 1-bit laconic interactive arguments from weaker assumptions.

**Witness encryption from 1-bit laconic arguments.** Next, we show that a 1-bit laconic argument implies a weaker variant of witness encryption where semantic security is only required to hold when encrypting to a *randomly* sampled statement $\mathbf{x} \notin \mathcal{L}$ rather than any statement $\mathbf{x} \notin \mathcal{L}$. Our results here are conceptually similar to those of Faonio et al. [FNV17], who previously showed how to construct witness encryption from any *predictable* argument system. In our setting, we do not impose any additional restriction on the underlying argument system. Instead, we show that for a class of "cryptographically-hard" languages, soundness of an optimally laconic argument alone already implies a "predictability" property, which suffices to give our relaxed variant of witness encryption We discuss the connection between our 1-bit arguments for cryptographically-hard languages and the notion of predictable arguments from [FNV17] in greater detail in Remark 6.13.

**Definition 6.9** (Distributional Witness Encryption)**.** Fix a parameter $n \in \mathbb{N}$. Let $\mathcal{L} \subseteq \{0,1\}^n$ be an NP language, and let $\mathcal{D}$ be a probability distribution over $\{0,1\}^n \setminus \mathcal{L}$. A distributional witness encryption scheme for $\mathcal{L}$ with respect to $\mathcal{D}$ is a tuple of algorithms $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ with the same properties and requirements as Definition 6.5, except the semantic security requirement is replaced by a weaker $\mathcal{D}$-semantic security requirement:

- **$\mathcal{D}$-Semantic Security:** For all efficient adversaries $\mathcal{A}$ and $\mathbf{x} \leftarrow \mathcal{D}$
$$\left| \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, 0)) = 1] - \Pr[\mathcal{A}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, 1)) = 1] \right| = \mathrm{negl}(\lambda),$$
where the probability is taken over the randomness of sampling $\mathbf{x}$, the encryption randomness, as well as the adversary's randomness.

As described in Remark 6.6, we can replace $\mathcal{D}$-semantic security with an equivalent notion of $\mathcal{D}$-unguessability.

**Construction 6.10.** Fix a parameter $n$. Let $\mathcal{L} \subseteq \{0,1\}^n$ be an $\mathsf{NP}$ language and let $\mathcal{D}$ be a distribution over $\{0,1\}^n \setminus \mathcal{L}$. Let $\Pi_{\mathsf{arg}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a 1-bit laconic argument for $\mathcal{L}$. We construct a distributional witness encryption scheme $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ for $\mathcal{L}$ with respect to $\mathcal{D}$ as follows:

- $\mathsf{Encrypt}(1^\lambda, \mathbf{x}, m)$: On input the security parameter $\lambda$, a statement $\mathbf{x}$, and a message $m \in \{0,1\}$, the encryption algorithm samples parameters $(\sigma_{\mathbf{x}}, \tau_{\mathbf{x}}) \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{x})$, and computes $\alpha_0 \leftarrow \mathsf{Verify}(\tau_{\mathbf{x}}, 0)$ and $\alpha_1 \leftarrow \mathsf{Verify}(\tau_{\mathbf{x}}, 1)$. Then, it does the following:
  - If $\alpha_0 = 1 = \alpha_1$, then the encryption algorithm outputs the message $m$ in the clear.
  - If $\alpha_0 = 0 = \alpha_1$, then the encryption algorithm outputs $\perp$.
  - Otherwise, the encryption algorithm outputs $(\sigma_{\mathbf{x}}, m \oplus b)$ where $b \in \{0,1\}$ is such that $\alpha_b = 1$.
- $\mathsf{Decrypt}(\mathsf{ct}, \mathbf{w}) \to \{0,1\} \cup \perp$. If $\mathsf{ct} = 0$, $\mathsf{ct} = 1$, or $\mathsf{ct} = \perp$, then the decryption algorithm outputs $\mathsf{ct}$. Otherwise, it parses $\mathsf{ct} = (\sigma_{\mathbf{x}}, \beta)$, computes $b \leftarrow \mathsf{Prove}(\sigma_{\mathbf{x}}, \mathbf{w})$, and outputs $\beta \oplus b$.

Next, we show that Construction 6.10 gives a distributional witness encryption scheme for any language that is "cryptographically-hard." Intuitively, we say that an $\mathsf{NP}$ language $\mathcal{L}$ is cryptographically-hard if there exists a distribution $\mathcal{D}_{\text{YES}}$ over YES instances that is computationally indistinguishable from a distribution $\mathcal{D}_{\text{NO}}$ of NO instances.

**Definition 6.11** (Cryptographically-Hard Language)**.** Let $\mathcal{L} \subseteq \{0,1\}^n$ be an $\mathsf{NP}$ language. We say that $\mathcal{L}$ is a *cryptographically-hard* language if there exists distributions $\mathcal{D}_{\text{YES}}$ over $\mathcal{L}$ and $\mathcal{D}_{\text{NO}}$ over $\{0,1\}^n \setminus \mathcal{L}$ such that $\mathcal{D}_{\text{YES}} \overset{c}{\approx} \mathcal{D}_{\text{NO}}$.

**Theorem 6.12.** *Fix a security parameter $\lambda$ and let $\mathcal{L} \subseteq \{0,1\}^{n(\lambda)}$ be an $\mathsf{NP}$ language, and suppose that $\mathcal{L}$ is cryptographically-hard (Definition 6.11). Let $\mathcal{D}_{\text{YES}}$ and $\mathcal{D}_{\text{NO}}$ be the distributions over* YES *instances and* NO *instances, respectively, for $\mathcal{L}$ from Definition 6.11. Assume moreover that $\Pi_{\mathsf{arg}}$ is a 1-bit laconic argument for $\mathcal{L}$. Then, $\Pi_{\mathsf{WE}}$ from Construction 6.10 is a distributional witness encryption scheme for $\mathcal{L}$ with respect to $\mathcal{D}_{\text{NO}}$.*

*Proof.* We show correctness and security separately.

**Correctness.** Take any statement $\mathbf{x} \in \mathcal{L}$ and any witness $\mathbf{w}$ where $R(\mathbf{x}, \mathbf{w}) = 1$. We show that

$$\Pr[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, \mathbf{x}, m), \mathbf{w}) = m] = 1.$$

Let $\mathsf{ct} \leftarrow \mathsf{Encrypt}(1^\lambda, \mathbf{x}, m)$, and $(\sigma_{\mathbf{x}}, \tau_{\mathbf{x}})$ be the parameters sampled by the encryption algorithm. By perfect completeness of $\Pi_{\mathsf{arg}}$, if $b \leftarrow \mathsf{Prove}(\sigma_{\mathbf{x}}, \mathbf{x}, \mathbf{w})$, then $\mathsf{Verify}(\tau_{\mathbf{x}}, b) = 1$. We consider two possible scenarios:

- Suppose $\mathsf{Verify}(\tau_{\mathbf{x}}, 1-b) = 1$. In this case, $\mathsf{ct} = m$, and the decryption algorithm also outputs $m$. Correctness holds.
- Suppose $\mathsf{Verify}(\tau_{\mathbf{x}}, 1-b) = 0$. In this case, $\mathsf{ct} = (\sigma_{\mathbf{x}}, m \oplus b)$, and the decryption algorithm outputs $(m \oplus b) \oplus b = m$.

**Security.** By assumption, $\mathcal{D}_{\text{YES}} \overset{c}{\approx} \mathcal{D}_{\text{NO}}$. Now, we show that

$$\Pr[\mathbf{x} \leftarrow \mathcal{D}_{\text{NO}}; (\sigma_{\mathbf{x}}, \tau_{\mathbf{x}}) \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{x}) : \mathsf{Verify}(\tau_{\mathbf{x}}, 0) = \mathsf{Verify}(\tau_{\mathbf{x}}, 1)] = \operatorname{negl}(\lambda).$$

We consider the two possibilities separately:

- Suppose $\mathsf{Verify}(\tau_{\mathbf{x}}, 0) = 0 = \mathsf{Verify}(\tau_{\mathbf{x}}, 1)$ with probability $\varepsilon$. This implies that $\mathcal{D}_{\text{YES}}$ and $\mathcal{D}_{\text{NO}}$ are distinguishable with the same advantage $\varepsilon$. Specifically, on input an instance $\mathbf{x}$, the distinguisher samples $(\sigma_{\mathbf{x}}, \tau_{\mathbf{x}}) \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{x})$ and outputs 1 if $\mathsf{Verify}(\tau_{\mathbf{x}}, 0) = 0 = \mathsf{Verify}(\tau_{\mathbf{x}}, 1)$. If $\mathbf{x} \leftarrow \mathcal{D}_{\text{YES}}$, then $\mathbf{x} \in \mathcal{L}$ and by perfect completeness of $\Pi_{\mathsf{arg}}$, the distinguisher outputs 1 with probability 0. Conversely, if $\mathbf{x} \leftarrow \mathcal{D}_{\text{NO}}$, then by assumption, the distinguisher outputs 1 with probability $\varepsilon$.

- Suppose $\mathsf{Verify}(\mathbf{x}, 0) = 1 = \mathsf{Verify}(\mathbf{x}, 1)$ with probability $\varepsilon$. Then, we can construct an adversary that breaks soundness of $\Pi_{\mathsf{arg}}$ with advantage $1/2 + \varepsilon/2 - \operatorname{negl}(\lambda)$. Consider the adversary that samples a statement $\mathbf{x} \leftarrow \mathcal{D}_{\text{NO}}$ and outputs a random bit $b \xleftarrow{\text{R}} \{0,1\}$ as its proof. We compute the probability that $\mathsf{Verify}(\tau_{\mathbf{x}}, b) = 1$, where $(\sigma_{\mathbf{x}}, \tau_{\mathbf{x}}) \leftarrow \mathsf{Setup}(1^\lambda, \mathbf{x})$. From the first case, we have that $\mathsf{Verify}(\tau_{\mathbf{x}}, 0) = 0 = \mathsf{Verify}(\tau_{\mathbf{x}}, 1)$ with negligible probability. Thus, with probability $1 - \operatorname{negl}(\lambda)$, at least one of $b \in \{0,1\}$ is a valid proof for $\mathbf{x}$. The probability that the guessing adversary succeeds in breaking soundness is then

$$\Pr[\mathsf{Verify}(\tau_{\mathbf{x}}, b) = 1] \geq \varepsilon + \frac{1}{2}(1 - \varepsilon - \operatorname{negl}(\lambda)) = \frac{1}{2} + \frac{\varepsilon}{2} - \operatorname{negl}(\lambda).$$

  This contradicts soundness of $\Pi_{\mathsf{arg}}$.

We conclude that with overwhelming probability over the choice of $\mathbf{x}$ and the $\mathsf{Setup}$ randomness, there is exactly one proof $b \in \{0,1\}$ such that $\mathsf{Verify}(\tau_{\mathbf{x}}, b) = 1$. To show the claim, suppose there exists an efficient adversary $\mathcal{A}$ whose guessing advantage (Eq. (6.2)) is $\varepsilon$. Without loss of generality, suppose that given an encryption of $m \xleftarrow{\text{R}} \{0,1\}$, adversary $\mathcal{A}$ outputs $m$ with probability at least $1/2 + \varepsilon$ (if $\mathcal{A}$ outputs $m$ with probability less than $1/2 - \varepsilon$, we can consider an adversary that runs $\mathcal{A}$ and outputs the complement of $\mathcal{A}$'s output). We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks soundness of $\Pi_{\mathsf{arg}}$ with probability $1/2 + \varepsilon$. Algorithm $\mathcal{B}$ works as follows:

1. At the beginning of the game, algorithm $\mathcal{B}$ samples a statement $\mathbf{x} \leftarrow \mathcal{D}_{\text{NO}}$ and gives $\mathbf{x}$ to the challenger for the soundness game. It receives a common reference string $\sigma_{\mathbf{x}}$ from the soundness challenger.
2. Algorithm $\mathcal{B}$ samples a random bit $\beta \xleftarrow{\text{R}} \{0,1\}$ and sends $(\sigma_{\mathbf{x}}, \beta)$ to the guessing adversary $\mathcal{A}$.
3. When $\mathcal{A}$ outputs a guess $m \in \{0,1\}$, $\mathcal{B}$ submits $m \oplus \beta$ as its proof.

First, we argue that $\mathcal{B}$ correctly simulates the unguessability game for adversary $\mathcal{A}$. From above, we have that with overwhelming probability (over the choice of $\mathbf{x}$ and the randomness in the $\mathsf{Setup}$ algorithm), $\mathsf{Verify}(\tau_{\mathbf{x}}, 0) \neq \mathsf{Verify}(\tau_{\mathbf{x}}, 1)$. Let $b \in \{0,1\}$ be such that $\mathsf{Verify}(\tau_{\mathbf{x}}, b) = 1$. In this case, a valid ciphertext for a message $m$ consists of the tuple $(\sigma_{\mathbf{x}}, b \oplus m)$. In the unguessability game, the message $m$ is sampled uniformly at random, and so the bit $b \oplus m$ is also uniformly random. This is precisely the distribution $\mathcal{B}$ simulates in the reduction.

By assumption $\mathcal{A}$ is able to guess the message with probability at least $1/2 + \varepsilon$. In particular, this means that the bit $m$ output by $\mathcal{A}$ satisfies $m = \beta \oplus b$ where $\mathsf{Verify}(\tau_{\mathbf{x}}, b) = 1$. But in this case, $b = m \oplus \beta$, and algorithm $\mathcal{B}$ has produced an accepting proof for the statement $\mathbf{x}$. We conclude that if $\mathcal{A}$ has guessing advantage $\varepsilon$, then $\mathcal{B}$ breaks soundness with advantage $1/2 + \varepsilon$. □

**Remark 6.13** (1-Bit Arguments and Predictable Arguments)**.** We can interpret the first step in our soundness proof of Theorem 6.12 as showing that a 1-bit argument for a cryptographically-hard language is essentially a predictable argument (c.f., [FNV17]). Specifically, we show that for a randomly-sampled statement $\mathbf{x}$, there is exactly *one* proof that the verifier accepts. Previously, Faonio et al. [FNV17] showed that any predictable argument for a language $\mathcal{L}$ implies a witness encryption for the same language. Since our arguments are predictable (for a randomly-sampled instance) when the underlying language is cryptographically-hard, we obtain the distributional variant of witness encryption for cryptographically-hard languages.

**Distributional witness encryption to public-key encryption.** Although 1-bit laconic arguments only suffice for constructing a weaker distributional variant of witness encryption, this variant still suffices to instantiate some of the applications of witness encryption from [GGSW13]. Here, we recall the construction of public-key encryption from witness encryption from [GGSW13, §4.1] and show how we can instantiate it using a distributional witness encryption scheme for the same language. In particular, this means that a 1-bit SNARG, and more generally, a 1-bit laconic argument implies a public-key encryption scheme where the complexity of the key-generation algorithm is *independent* of the complexity of the underlying argument system. Key-generation in this scheme only requires a single evaluation of a pseudorandom generator. The only public-key encryption schemes that have this property rely on witness encryption (or stronger assumptions). This provides some evidence on the difficulty of realizing optimally laconic arguments from simpler assumptions.

**Construction 6.14** (Public Key Encryption from Witness Encryption [GGSW13, §4.1])**.** Fix a security parameter $\lambda$ and let $G\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a length-doubling PRG. Define the language $\mathcal{L} \subset \{0,1\}^{2\lambda}$ as $\mathcal{L} = \left\{y \in \{0,1\}^{2\lambda} : y = G(x) \text{ for some } x \in \{0,1\}^\lambda\right\}$. Let $\Pi_{\mathsf{WE}} = (\mathsf{WE.Encrypt}, \mathsf{WE.Decrypt})$ be a witness encryption scheme for $\mathcal{L}$. We define the public key encryption scheme $\Pi_{\mathsf{PKE}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter $\lambda$, the key-generation algorithm samples a seed $s \xleftarrow{\text{R}} \{0,1\}^\lambda$, computes $t \leftarrow G(s)$, and outputs $\mathsf{pk} = (\lambda, t)$ and $\mathsf{sk} = s$.
- $\mathsf{Encrypt}(\mathsf{pk}, m)$: On input the public key $\mathsf{pk} = (\lambda, t)$ and a message $m \in \{0,1\}$, the encryption algorithm outputs the ciphertext $\mathsf{ct} \leftarrow \mathsf{WE.Encrypt}(1^\lambda, t, m)$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})$: On input the secret key $\mathsf{sk} = s$ and a ciphertext $\mathsf{ct}$, the decryption algorithm outputs $\mathsf{WE.Decrypt}(\mathsf{ct}, s)$.

**Theorem 6.15.** *Fix a security parameter $\lambda$ and let $G\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a length-doubling PRG. Define the language $\mathcal{L} \subset \{0,1\}^{2\lambda}$ as in Construction 6.14. Let $\mathcal{D}_{\text{NO}}$ be the uniform distribution over $\{0,1\}^{2\lambda} \setminus \mathcal{L}$. If $\Pi_{\mathsf{WE}}$ is a witness encryption scheme for $\mathcal{L}$ with respect to $\mathcal{D}_{\text{NO}}$ and $G$ is a secure PRG, then $\Pi_{\mathsf{PKE}}$ from Construction 6.14 is a semantically-secure PKE scheme.*

*Proof.* The proof is essentially identical to the corresponding proof in [GGSW13, Appendix A.1]. We give the formal hybrid argument below:

- $\mathsf{Hyb}_0$: This is the semantic security game where the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and responds to the adversary's query with $\mathsf{Encrypt}(\mathsf{pk}, m_0)$.

- $\mathsf{Hyb}_1$: Same as $\mathsf{Hyb}_0$ except instead of setting $\mathsf{pk} = (\lambda, t)$ where $t = G(s)$ and $s \xleftarrow{R} \{0,1\}^\lambda$, the challenger samples $t \xleftarrow{R} \{0,1\}^{2\lambda}$. Hybrids $\mathsf{Hyb}_0$ and $\mathsf{Hyb}_1$ are computationally indistinguishable by PRG security of $G$.
- $\mathsf{Hyb}_2$: Same as $\mathsf{Hyb}_1$, except the challenger samples $t \xleftarrow{R} \{0,1\}^{2\lambda} \setminus \mathcal{L}$. $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$ are statistically indistinguishable.
- $\mathsf{Hyb}_3$: Same as $\mathsf{Hyb}_2$ except the challenger encrypts message $m_1$ when responding to the adversary's challenge. Hybrids $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$ are computationally indistinguishable by distributional semantic security of $\Pi_{\mathsf{WE}}$.
- $\mathsf{Hyb}_4$: Same as $\mathsf{Hyb}_3$ except the challenger samples $t \xleftarrow{R} \{0,1\}^{2\lambda}$. Hybrids $\mathsf{Hyb}_3$ and $\mathsf{Hyb}_4$ are statistically indistinguishable.
- $\mathsf{Hyb}_5$: Same as $\mathsf{Hyb}_4$ except the challenger sets $t = G(s)$ where $s \xleftarrow{R} \{0,1\}^\lambda$. This is the semantic security game where the challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and respond to the adversary's query with $\mathsf{Encrypt}(\mathsf{pk}, m_1)$. Hybrids $\mathsf{Hyb}_4$ and $\mathsf{Hyb}_5$ are computationally indistinguishable by PRG security of $G$. $\square$

To instantiate Construction 6.14 from a 1-bit laconic argument for $\mathsf{NP}$ (or more specifically, for the language $\mathcal{L}$ in Construction 6.14), it suffices to show that there exists a distribution $\mathcal{D}_{\text{YES}}$ over $\mathcal{L}$ such that $\mathcal{D}_{\text{YES}} \overset{c}{\approx} \mathcal{D}_{\text{NO}}$, where $\mathcal{D}_{\text{NO}}$ is the distribution from Construction 6.14. This follows from PRG security. Specifically, let $\mathcal{D}_{\text{YES}}$ be the distribution $\{s \xleftarrow{R} \{0,1\}^\lambda : G(s)\}$. By PRG security, $\mathcal{D}_{\text{YES}}$ is computationally indistinguishable from the uniform distribution over $\{0,1\}^{2\lambda}$. Finally, the uniform distribution over $\{0,1\}^{2\lambda}$ is statistically indistinguishable from $\mathcal{D}_{\text{NO}}$ and the claim follows. Thus, a 1-bit laconic argument for $\mathsf{NP}$ implies a public-key encryption scheme where the complexity of the key-generation algorithm is independent of the complexity of the witness encryption scheme.

## 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

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP*, 2010.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3), 1998.

[BC12] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *CRYPTO*, 2012.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2), 1988.

[BCC$^+$14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014, 2014.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *STOC*, 2013.

[BCG$^+$13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO*, 2013.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *STOC*, 2014.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium*, 2014.

[BDRV17] Itay Berman, Akshay Degwekar, Ron Rothblum, and Prashant Nalini Vasudevan. From laconic zero-knowledge to public-key cryptography. *Electronic Colloquium on Computational Complexity (ECCC)*, 2017, 2017.

[Ben64] Václad E Beneš. Optimal rearrangeable multistage connecting networks. *Bell Labs Technical Journal*, 43(4), 1964.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, 1991.

[BFR$^+$13] Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. Verifying computations with state. In *SOSP*, 2013.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, 2014.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2), 1987.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *EUROCRYPT*, 2017.

[BP04a] Boaz Barak and Rafael Pass. On the possibility of one-message weak zero-knowledge. In *TCC*, 2004.

[BP04b] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, 2004.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM J. Comput.*, 38(2), 2008.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In *ITCS*, 2012.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1), 2000.

[CFH$^+$15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *IEEE SP*, 2015.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *ITCS*, 2012.

[Dam91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, 1991.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, 2014.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, 2012.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, 2010.

[Din06] Irit Dinur. The PCP theorem by gap amplification. In *STOC*, 2006.

[FGL$^+$91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *FOCS*, 1991.

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *PKC*, 2017.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4), 1998.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, 2012.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, 1982.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In *CRYPTO*, 2017.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, 1985.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO*, 2009.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, 2016.

[GVW01] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. In *ICALP*, 2001.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *CRYPTO*, 1998.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *CCC*, 2007.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, 2009.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, 1992.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS*, 2013.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *FOCS*, 1990.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, 2012.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT*, 2013.

[Lip16] Helger Lipmaa. Prover-efficient commit-and-prove zero-knowledge snarks. In *AFRICACRYPT*, 2016.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4), 2000.

[Mie08] Thilo Mie. Polylogarithmic two-round argument systems. *J. Mathematical Cryptology*, 2(4), 2008.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, 2003.

[OTW71] D.C. Opferman and N.T. Tsao-Wu. On a class of rearrangeable switching networks part I: Control algorithm. *Bell Labs Technical Journal*, 50(5), 1971.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, 2013.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4), 1980.

[Sha90] Adi Shamir. IP=PSPACE. In *FOCS*, 1990.

[SMBW12] Srinath T. V. Setty, Richard McPherson, Andrew J. Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *NDSS*, 2012.

[SVP$^+$12] Srinath T. V. Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J. Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security Symposium*, 2012.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, 2014.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO*, 2013.

[TRMP12] Justin Thaler, Mike Roberts, Michael Mitzenmacher, and Hanspeter Pfister. Verifiable computation with massively parallel interactive proofs. In *HotCloud*, 2012.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, 2008.

[VSBW13] Victor Vu, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *IEEE SP*, 2013.

[Wak68] Abraham Waksman. A permutation network. *Journal of the ACM (JACM)*, 15(1), 1968.

[WB15] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2), 2015.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In *ICALP*, 2005.

[WHG$^+$16] Riad S. Wahby, Max Howald, Siddharth J. Garg, Abhi Shelat, and Michael Walfish. Verifiable asics. In *IEEE Symposium on Security and Privacy*, 2016.

[WJB$^+$17] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *ACM CCS*, 2017.

[WSR$^+$15] Riad S. Wahby, Srinath T. V. Setty, Zuocheng Ren, Andrew J. Blumberg, and Michael Walfish. Efficient RAM and control flow in verifiable outsourced computation. In *NDSS*, 2015.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, 1979.

## A Additional Preliminaries

In this section, we review some additional preliminaries. First, we review the Schwartz-Zippel lemma [Sch80, Zip79]:

**Lemma A.1** (Schwartz-Zippel [Sch80, Zip79])**.** *Let $p$ be a prime and let $f \in \mathbb{F}_p[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree $d$ that is not identically zero. Then,*

$$\Pr[\alpha_1, \ldots, \alpha_n \xleftarrow{\text{R}} \mathbb{F}_p \colon f(\alpha_1, \ldots, \alpha_n) = 0)] \le \frac{d}{p}.$$

## A.1 Linear PCPs and Linear MIPs

Similar to [IKO07, BCI+13, BISW17], our starting point in this paper is a *linear* proof system. In this section, we review the notion of linear probabilistically-checkable proofs (PCPs) and linear multi-prover interactive proofs (MIPs). In a linear PCP system for a relation $\mathcal{R}$ over a finite field $\mathbb{F}$, the PCP oracle is restricted to computing a linear function $\boldsymbol{\pi} \colon \mathbb{F}^d \to \mathbb{F}$ over the verifier's queries. The linear MIP model directly generalizes the linear PCP model to the setting where there are multiple provers, each implemented by a linear PCP. We now review the formal definitions. Afterwards, we introduce the notion of a quasi-optimal linear MIP.

**Definition A.2** (Linear PCP [IKO07, BCI+13, adapted])**.** Let $\mathcal{R}$ be a binary relation, $\mathcal{L}$ be the associated language, $\mathbb{F}$ be a finite field, $\mathcal{P}$ be a prover algorithm, and $\mathcal{V}$ be an oracle verifier algorithm. Then, the pair $(\mathcal{P}, \mathcal{V})$ is an (input-oblivious) $k$-query linear PCP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error $\varepsilon$ and query length $d$ if it satisfies the following requirements:

- **Syntax:** The prover algorithm $\mathcal{P}$ takes as input a statement $\mathbf{x}$ and a witness $\mathbf{w}$, and outputs a vector $\boldsymbol{\pi} \in \mathbb{F}^d$. The verification algorithm $\mathcal{V}^{\boldsymbol{\pi}} = (\mathcal{Q}, \mathcal{D})$ consists of an input-oblivious probabilistic query-generation algorithm $\mathcal{Q}$ and a deterministic decision algorithm $\mathcal{D}$. The query algorithm $\mathcal{Q}$ generate a query matrix $\mathbf{Q} \in \mathbb{F}^{d \times k}$ and some additional state information $\mathsf{st}$. The decision algorithm $\mathcal{D}$ takes as input the statement $\mathbf{x}$, the verification state $\mathsf{st}$, and the prover response $\mathbf{y} = \mathbf{Q}^\top \boldsymbol{\pi} \in \mathbb{F}^k$, and either "accepts" (with output 1) or rejects (with output 0).

- **Completeness:** For every $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, setting $\boldsymbol{\pi} \leftarrow \mathcal{P}(\mathbf{x}, \mathbf{w})$, it follows that $\mathcal{V}^{\boldsymbol{\pi}}(\mathbf{x})$ accepts with probability 1.

- **Soundness:** For every $\mathbf{x} \notin \mathcal{L}$, and all proofs $\boldsymbol{\pi}^* \in \mathbb{F}^d$, the probability that $\mathcal{V}^{\boldsymbol{\pi}^*}(\mathbf{x})$ accepts is at most $\varepsilon$. We say that $(\mathcal{P}, \mathcal{V})$ satisfy *soundness against affine provers* if soundness holds even against affine adversarial strategies $(\boldsymbol{\pi}^*, \mathbf{b}^*)$ where $\boldsymbol{\pi}^* \in \mathbb{F}^d$ and $\mathbf{b}^* \in \mathbb{F}^k$, and the prover's response is computed as $\mathbf{y} = \mathbf{Q}^\top \boldsymbol{\pi}^* + \mathbf{b}^* \in \mathbb{F}^k$.

We say that $(\mathcal{P}, \mathcal{V})$ is an input-oblivious $k$-query linear PCP for $\mathcal{R}$ over $\mathbb{F}$ with *knowledge error* $\varepsilon$ and query length $d$ if $(\mathcal{P}, \mathcal{V})$ satisfies the properties above, but the soundness property is replaced by the following (stronger) knowledge property:

- **Knowledge:** There exists a knowledge extractor $\mathcal{E}$ such that for every vector $\boldsymbol{\pi}^* \in \mathbb{F}^d$, if $\mathcal{V}^{\boldsymbol{\pi}^*}(\mathbf{x})$ accepts with probability at least $\varepsilon$, then $\mathcal{E}^{\boldsymbol{\pi}^*}(\mathbf{x})$ outputs $\mathbf{w}$ such that $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$. As with soundness, we can correspondingly define a notion of *knowledge against affine strategies.*

**Remark A.3** (Soundness Against Affine Provers)**.** Given a $k$-query linear PCP over a finite field $\mathbb{F}$ with soundness error $\varepsilon$ against linear provers, Bitansky et al. [BCI+13, Construction 3.1] give a generic construction of a $k + 1$ query linear PCP over $\mathbb{F}$ with soundness error $\varepsilon + 1/|\mathbb{F}|$ against affine provers by introducing an additional consistency check. In fact, the construction in [BCI+13] provides even stronger soundness guarantees, but those will not be needed in this work.

**Definition A.4** (Linear MIPs [IKO07, adapted])**.** Let $\mathcal{R}$ be a binary relation, $\mathbb{F}$ be a finite field, $\mathcal{P} = (P_1, \ldots, P_\ell)$ be a tuple of $\ell$ prover algorithms, and $\mathcal{V}$ be an oracle verifier algorithm. Then, the pair $(\mathcal{P}, \mathcal{V})$ is an (input-oblivious) $k$-query linear multi-prover interactive proof (MIP) with $\ell$ provers for $\mathcal{R}$ over $\mathbb{F}$ with soundness error $\varepsilon$ and query length $d$ if it satisfies the following requirements:

- **Syntax:** Each prover algorithm $P_i$ (for $i \in [\ell]$) takes as input a statement $\mathbf{x}$ and a witness $\mathbf{w}$ and outputs a vector $\boldsymbol{\pi}_i \in \mathbb{F}^d$. We write $\mathcal{P}(\mathbf{x}, \mathbf{w})$ to denote the tuple $(P_1(\mathbf{x}, \mathbf{w}), \ldots, P_\ell(\mathbf{x}, \mathbf{w}))$. The verification algorithm $\mathcal{V}^{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell} = (\mathcal{Q}, \mathcal{D})$ consists of an input-oblivious probabilistic query-generation algorithm $\mathcal{Q}$ and a deterministic decision algorithm $\mathcal{D}$. The query algorithm $\mathcal{Q}$ generates a tuple of query matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_\ell \in \mathbb{F}^{d \times k}$ and some additional state information $\mathsf{st}$. The decision algorithm $\mathcal{D}$ takes as input the statement $\mathbf{x}$, the verification state $\mathsf{st}$, and the prover responses $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$ where each $\mathbf{y}_i = \mathbf{Q}_i^\top \boldsymbol{\pi}_i \in \mathbb{F}^k$, and either "accepts" (with output 1) or "rejects" (with output 0).

- **Completeness:** For every $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, and setting $\boldsymbol{\pi}_i \leftarrow P_i(\mathbf{x}, \mathbf{w})$ for all $i \in [\ell]$, we have that $\mathcal{V}^{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell}(\mathbf{x})$ accepts with probability 1.

- **Soundness:** For every $\mathbf{x} \notin \mathcal{L}$, and all proof vectors $(\boldsymbol{\pi}_1^*, \ldots, \boldsymbol{\pi}_\ell^*)$ where each $\boldsymbol{\pi}_i^* \in \mathbb{F}^d$, the probability that $\mathcal{V}^{\boldsymbol{\pi}_1^*, \ldots, \boldsymbol{\pi}_\ell^*}(\mathbf{x})$ accepts is at most $\varepsilon$. As in Definition A.2, we can define a corresponding notion of *soundness against affine provers* where soundness holds against provers who each implement a different affine strategy $(\boldsymbol{\pi}_i^*, \mathbf{b}_i^*) \in \mathbb{F}^d \times \mathbb{F}^k$.

Similar to Definition A.2, we can replace the soundness property with a stronger knowledge property.

**Definition A.5** (Quasi-Optimal Linear MIPs)**.** Let $\lambda$ be a security parameter, and $C$ be an arithmetic circuit of size $s$ over a finite field $\mathbb{F}$. A $k$-query linear MIP $(\mathcal{P}, \mathcal{V})$ with $\ell$ provers for $\mathcal{R}_C$ with soundness error $2^{-\lambda}$ is *quasi-optimal* if the prover $\mathcal{P} = (P_1, \ldots, P_\ell)$ can be implemented by an arithmetic circuit of size $\widetilde{O}(s) + \operatorname{poly}(\lambda, \log s)$, where the $\widetilde{O}(\cdot)$ notation is suppressing terms that are polylogarithmic in $s$ and $\lambda$.

**Remark A.6** (Existing Linear PCP Construction)**.** Existing linear PCP constructions [BCI+13, BISW17] (which can be viewed as linear MIPs with a single prover) are *not* quasi-optimal for arithmetic circuit satisfiability over a polynomial-size field (i.e., for Boolean circuit satisfiability). To provide soundness error $2^{-\lambda}$, the linear PCP constructions in [BCI+13, BISW17] either embed the circuit satisfiability instance inside a field of size $2^{\Omega(\lambda)}$, or have query complexity $O(\lambda)$. In both cases, the prover complexity becomes $\Omega(\lambda s) + \operatorname{poly}(\lambda, \log s)$. Thus, the existing SNARG constructions in [BCI+13, BISW17] are not quasi-optimal for Boolean circuit satisfiability.

## A.2 Routing Networks

Our quasi-optimal linear MIP construction in Section 4 relies on an efficient method for checking whether two matrices $\mathbf{W}, \mathbf{W}' \in \mathbb{F}^{t \times q}$ satisfy $\mathbf{W} = \Pi(\mathbf{W}')$ where $\Pi$ is an arbitrary permutation over the entries of a $t$-by-$q$ matrix. We begin by stating a lemma from [GHS12] that states that an arbitrary permutation $\Pi$ over the entries of a $t$-by-$q$ matrix can be decomposed into the composition of a small number of permutations, where each permutation implements a row-wise permutation or a column-wise permutation of the matrix entries.

**Definition A.7** (Matrix Permutations)**.** Fix integers $t, q$ and let $\Pi$ be a permutation over the entries of a $t$-by-$q$ matrix. We say that $\Pi$ is *row-wise restricted* if $\Pi$ only permutes elements within the rows of the matrix (that is, the permutation only changes the column, and not the row, of each element). Similarly, we say that $\Pi$ is *column-wise restricted* if $\Pi$ only permutes elements within the columns of the matrix.

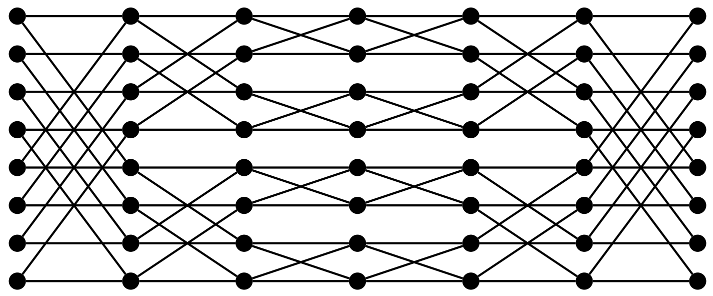

Figure 2: A Beneš network over $8 = 2^3$ nodes (BENEŠ$_3$)

**Lemma A.8** ([GHS12, Lemma 1]). *Fix positive integers $t, q \in \mathbb{N}$, and let $\Pi$ be a permutation over the entries of a $t$-by-$q$ matrix. Then, there exist permutations $\Pi_1, \Pi_2, \Pi_3$ such that $\Pi = \Pi_3 \circ \Pi_2 \circ \Pi_1$, where $\Pi_1$ and $\Pi_3$ are row-wise restricted, and $\Pi_2$ is column-wise restricted. Moreover, there is an efficient algorithm to compute $\Pi_1, \Pi_2, \Pi_3$ given $\Pi$.*

**Beneš networks.** A Beneš permutation network [Ben64] is a special graph that can model all permutations $\Pi$ on a collection of $m = 2^d$ elements. We give the precise definition below, and then state an elementary property on the structure of Beneš networks. We show an example of a Beneš network in Figure 2.

**Definition A.9** (Beneš Network [Ben64]). For a non-negative integer $d$, a $d$-dimensional Beneš network, denoted BENEŠ$_d$, is a directed graph with $2d + 1$ layers (labeled $0, \ldots, 2d$). Each layer contains $m = 2^d$ nodes (labeled $0, \ldots, 2^d - 1$), and edges only go from nodes in layer $i - 1$ to nodes in layer $i$ for $i \in [2d]$. The first layer (layer 0) is the input layer and the final layer (layer $2d$) is the output layer. The graph structure is defined recursively as follows:

- A 0-dimensional Beneš network BENEŠ$_0$ consists of a single node.
- A $d$-dimensional Beneš network BENEŠ$_d$ consists of an input layer (with $2^d$ nodes) that feeds into two stacked BENEŠ$_{d-1}$ networks, which feed into an output layer. The edge configuration of the input and output layers are then defined as follows, for $i = 0, \ldots, 2^{d-1} - 1$:
  - **Input edges:** There is an edge from the $i^{\text{th}}$ input of BENEŠ$_d$ to the $i^{\text{th}}$ input of each of the BENEŠ$_{d-1}$ networks. There is also an edge from the $(i + 2^{d-1})^{\text{th}}$ input of BENEŠ$_d$ to the $i^{\text{th}}$ input of each of the BENEŠ$_{d-1}$ networks.
  - **Output edges:** There is an edge from the $i^{\text{th}}$ output of each of the BENEŠ$_{d-1}$ networks to the $i^{\text{th}}$ output of BENEŠ$_d$. There is also an edge from the $i^{\text{th}}$ output of each BENEŠ$_{d-1}$ network to the $(i + 2^{d-1})^{\text{th}}$ output of the BENEŠ$_d$ network.

**Fact A.10** (Structure of Beneš Networks). Fix a positive integer $d \in \mathbb{N}$, and let $S = \{0, \ldots, 2^d - 1\}$. Then, a BENEŠ$_d$ network has the following structural properties:

- For $j \in \{0, \ldots, d\}$, we can partition the nodes in layer $j$ into $\ell_j = 2^{d-j}$ disjoint sets $S_0^{(j)}, \ldots, S_{\ell_j - 1}^{(j)} \subseteq S$, each containing $2^j$ nodes, with the following properties:

- For $k \in \{0, \ldots, 2^{d-j} - 1\}$, $S_k^{(j)}$ contains all indices $i \in S$ where the least-significant $d - j$ bits of $i$ is equal to $k$.
- For all nodes $i_s, i_t \in S_k^{(j)}$ for some $k$, there is a unique path of length $j$ from the $i_s^{\text{th}}$ input node to the $i_t^{\text{th}}$ node in layer $j$ of $\text{BENEŠ}_d$.
- For any two nodes $i_s \in S_k^{(j)}$, $i_t \in S_{k'}^{(j)}$ where $k \neq k'$, there are no paths from the $i_s^{\text{th}}$ input node to the $i_t^{\text{th}}$ node in layer $j$ of $\text{BENEŠ}_d$.

- For $j \in \{d, \ldots, 2d\}$, we can partition the nodes in layer $j$ into $\ell_j = 2^{j-d}$ disjoint sets $S_0^{(j)}, \ldots, S_{\ell-1}^{(j)} \subseteq S$, each containing $2^{2d-j}$ nodes, with the following properties:
  - For $k \in \{0, \ldots, 2^{j-d} - 1\}$, $S_k^{(j)}$ contains all indices $i \in S$ where the least-significant $j - d$ bits of $i$ is equal to $k$.
  - For all nodes $i_s, i_t \in S_k^{(j)}$ for some $k$, there is a unique path of length $j$ from the $i_s^{\text{th}}$ node in layer $j$ to the $i_t^{\text{th}}$ output node in $\text{BENEŠ}_d$.
  - For all nodes $i_s \in S_k^{(j)}$, $i_t \in S_{k'}^{(j)}$ where $k \neq k'$, there are no paths from the $i_s^{\text{th}}$ node in layer $j$ to the $i_t^{\text{th}}$ output node of $\text{BENEŠ}_d$.

A key property of Beneš networks is that they are rearrangeable: any permutation $\Pi$ on $m = 2^d$ values can be mapped to a set of $2^d$ node-disjoint paths in a $d$-dimensional Beneš network $\text{BENEŠ}_d$ where the $i^{\text{th}}$ path maps from input $i$ to output $\Pi(i)$. We state the following fact from [Wak68, OTW71].

**Fact A.11** (Rearrangeability of Beneš Networks [Wak68, OTW71])**.** Let $d \in \mathbb{N}$ be a positive integer and let $S = \{0, \ldots, 2^d - 1\}$. For all permutations $\Pi \colon S \to S$ there exists a set of $2^d$ *node-disjoint* paths from each input node $i \in S$ in $\text{BENEŠ}_d$ to its corresponding output node $\Pi(i)$. Moreover, these paths can be computed in time $O(d \cdot 2^d)$. We say that the paths $\mathcal{P}$ implement the permutation $\Pi$.

**Randomized routing in Beneš networks.** Fact A.11 states that we can route any permutation $\Pi$ on $[2^d]$ elements using a $d$-dimensional Beneš network. In our construction, we will use the following procedure to randomize the routing configuration.

**Construction A.12** (Randomized Routing)**.** Let $d \in \mathbb{N}$ be a positive integer and $S = \{0, \ldots, 2^d - 1\}$. For a permutation $\Pi \colon S \to S$ on $S$, let $\mathcal{P}$ be a collection of node-disjoint paths in $\text{BENEŠ}_d$ that routes each input node $i \in S$ to its corresponding output node $\Pi(i)$. Namely, for each $i \in S$ and $j \in \{0, \ldots, 2d\}$, let $\mathcal{P}[i, j]$ be the value of node $i$ in layer $j$ of the $\text{BENEŠ}_d$ network. We construct a new collection of paths $\mathcal{P}'$ as follows:

1. Initialize $\mathcal{P}' \leftarrow \mathcal{P}$. Then for $j \in [d]$ and $k = 0, \ldots, 2^{j-1} - 1$, do the following:
   (a) Choose a random bit $b_{j,k} \xleftarrow{\text{R}} \{0, 1\}$.
   (b) For $i \in S$, let $i_1 i_2 \cdots i_d$ be the binary representation of $i$. Then, if $b_{j,k} = 1$, for all nodes $i \in S$ where $i_1 i_2 \cdots i_{j-1} = k$ and $i_j = 0$, swap the values $\mathcal{P}'[i, j']$ and $\mathcal{P}'[i + 2^{d-j}, j']$, for all $j' = j, \ldots, d - j$.
2. Output the randomized collection of paths $\mathcal{P}'$.

At a high level, Construction A.12 takes a set of paths $\mathcal{P}$ implementing a specific permutation $\Pi$ in a $\textsc{beneš}_d$ network and produces a new set of paths $\mathcal{P}'$ in $\textsc{beneš}_d$ that implement the same permutation. The procedure relies on the recursive structure of the Beneš network. For example, in the first layer of a $\textsc{beneš}_d$ network, the input nodes are partitioned into two disjoint sets, one of which is routed using the top $\textsc{beneš}_{d-1}$ network, and the other is routed using the bottom $\textsc{beneš}_{d-1}$ network. The randomization procedure in Construction A.12 maintains the same partitioning of input nodes, but each partition is either routed using the top $\textsc{beneš}_{d-1}$ network or the bottom $\textsc{beneš}_{d-1}$ network with equal probability. This process is then iteratively applied to permute the routing configuration for each of the $\textsc{beneš}_{d-1}$ networks in the first layer, and so on. We state the formal correctness guarantee in the following lemma:

**Lemma A.13.** *Let $d \in \mathbb{N}$ be a positive integer, $S = \{0, \ldots, 2^d - 1\}$, and $\Pi \colon S \to S$ be a permutation on $S$. Let $\mathcal{P}$ be a collection of node-disjoint paths in $\textsc{beneš}_d$ that implements the permutation $\Pi$. Then, the new collection of paths $\mathcal{P}'$ obtained by applying the randomized routing procedure in Construction A.12 to $\mathcal{P}$ is also a collection of node-disjoint paths in $\textsc{beneš}_d$ that implements the same permutation $\Pi$.*

For any sequence of paths $\mathcal{P}$ implementing a permutation $\Pi$ in a $\textsc{beneš}_d$ network, the set $\mathcal{P}'$ of randomized paths output by Construction A.12 has the property that if we consider the path of any *single* input node $i \in \{0, \ldots, 2^d - 1\}$ to $\Pi(i)$ in $\mathcal{P}'$, its path is distributed uniformly over all of the $2^d$ possible paths from $i$ to $\Pi(i)$ in $\textsc{beneš}_d$. We state the precise guarantee in the following lemma:

**Lemma A.14.** *Let $d \in \mathbb{N}$ be a positive integer, $S = \{0, \ldots, 2^d - 1\}$, and $\Pi \colon S \to S$ be a permutation on $S$. Let $\mathcal{P}$ be a collection of node-disjoint paths in $\textsc{beneš}_d$ that implements $\Pi$, and let $\mathcal{P}'$ be the set of randomized paths output by Construction A.12 applied to $\mathcal{P}$. For a node $i \in S$, let $i = i_0, i_1, \ldots, i_{2d} = \Pi(i)$ denote the path of $i$ in $\mathcal{P}'$. Then the following holds:*

- *For $j \in \{0, \ldots, d\}$, let $S_0^{(j)}, \ldots, S_{\ell_j - 1}^{(j)}$ be the partition of the nodes in layer $j$ from Fact A.10. Let $k \in \{0, \ldots, \ell_j - 1\}$ such that $i \in S_k^{(j)}$. Then, for all $i' \in S_k^{(j)}$, $\Pr[i_j = i'] = 1/\lvert S_k^{(j)} \rvert = 1/2^j$.*
- *For $j \in \{d, \ldots, 2d\}$, let $S_0^{(j)}, \ldots, S_{\ell_j - 1}^{(j)}$ be the partition of the nodes in layer $j$ from Fact A.10. Let $k \in \{0, \ldots, \ell_j - 1\}$ such that $\Pi(i) \in S_k^{(j)}$. Then, for all $i' \in S_k^{(j)}$, $\Pr[i_j = i'] = 1/\lvert S_k^{(j)} \rvert = 1/2^{2d-j}$.*

*In both cases, the probability is taken over the random coins in the randomization algorithm (Construction A.12).*

**Permutations from Beneš networks.** We can view a collection of paths in a Beneš network as providing a systematic decomposition of an arbitrary permutation $\Pi$ on $t = 2^d$ elements into a sequence of $\ell = O(\log t)$ permutations $\Pi_1, \ldots, \Pi_\ell$, where each $\Pi_1, \ldots, \Pi_\ell$ can be expressed as a product of disjoint 2-cycles, and $\Pi = \Pi_\ell \circ \cdots \circ \Pi_1$. More precisely, we can associate values $x_0, \ldots, x_{t-1}$ with the input nodes of the $\textsc{beneš}_d$ network (e.g., value $x_i$ with the $i^{\text{th}}$ input node). Given a path from an input node to an output node, we associate the value of the input node with every node along the path. Then, any collection of $t$ node-disjoint paths $\mathcal{P}$ from input nodes to output nodes induces an assignment to every node in the network. Now, for any permutation $\Pi$ on $t$ values $x_0, \ldots, x_{t-1}$, we define $\Pi_1, \ldots, \Pi_\ell$ so that $(\Pi_i \circ \cdots \circ \Pi_1)(x_0, \ldots, x_{t-1})$ gives the values of the nodes in layer $i + 1$ of $\textsc{beneš}_d$ on input $x_0, \ldots, x_{t-1}$ and paths determined by $\mathcal{P}$. The structure of the Beneš network ensures that each of the $\Pi_1, \ldots, \Pi_\ell$ is a product of *disjoint* 2-cycles. We say that permutations with this property (generalized to permutations over the entries of a matrix) are 2-locally decomposable (Definition A.15). We give a more precise description of this decomposition in Construction A.16. We formalize two properties satisfied by the decomposition in Lemma A.17.

**Definition A.15** (2-Local Decomposability)**.** Fix an even integer $t \in \mathbb{N}$, an integer $q \in \mathbb{N}$, and set $t' = t/2$. Let $\Pi$ be a permutation on the entries of a $t$-by-$q$ matrix. We say that $\Pi$ is 2-*locally decomposable* if there exists a partition $\{j_1, j_2\}, \ldots, \{j_{t'-1}, j_{t'}\}$ of $[t]$ and permutations $\Pi_1, \ldots, \Pi_{t'}$ over 2-by-$q$ matrices such that for all matrices $\mathbf{W} \in \mathbb{F}^{q \times t}$, we have that $\hat{\mathbf{W}} = \Pi(\mathbf{W})$ if and only if for all $i \in [t']$,

$$\hat{\mathbf{W}}_{[j_{2i}, j_{2i+1}]} = \Pi_i\big(\mathbf{W}_{[j_{2i}, j_{2i+1}]}\big).$$

In other words, a permutation $\Pi$ is 2-locally decomposable if $\Pi$ can be written as a composition of $t' = t/2$ disjoint permutations that each operate on exactly two rows of the matrix.

**Construction A.16** (Randomized 2-Local Decomposition)**.** Let $t, q \in \mathbb{N}$ be integers where $t = 2^d$ for some $d \in \mathbb{N}$. Let $\Pi$ be an arbitrary column-wise restricted permutation on the entries of a $t$-by-$q$ matrix. The randomized 2-local decomposition of $\Pi$ is a sequence of $2d$ matrices $\Pi_1, \ldots, \Pi_{2d}$ constructed as follows:

- For each column $i \in [q]$, let $\mathcal{P}_i$ be a collection of paths in a $\textsc{beneš}_d$ network that implements the permutation $\Pi$ on the entries in column $i$. Let $\mathcal{P}'_i$ be the output of the randomized routing procedure in Construction A.12 applied to $\mathcal{P}_i$.
- For $j \in [2d]$, we take $\Pi_j$ to be a column-wise restricted permutation on $t$-by-$q$ matrices. We write $\Pi_j^{(i)}$ to denote the permutation $\Pi_j$ implements on column $i \in [q]$. We define $\Pi_1, \ldots, \Pi_{2d}$ so that for all $i \in [q]$ and $j \in [2d]$, $(\Pi_j^{(i)} \circ \cdots \circ \Pi_1^{(i)})(x_0, \ldots, x_{t-1})$ gives the values of the nodes in layer $j$ of a $\textsc{beneš}_d$ network on input $(x_0, \ldots, x_{t-1})$ and using paths defined by $\mathcal{P}'_i$.

**Lemma A.17.** *Let $t, q \in \mathbb{N}$ be integers where $t = 2^d$ for some $d \in \mathbb{N}$. Let $\Pi$ be an arbitrary column-wise restricted permutation on the entries of a $t$-by-$q$ matrix, and let $(\Pi_1, \ldots, \Pi_{2d})$ be the randomized 2-local decomposition of $\Pi$ from Construction A.16. The local decomposition satisfies the following properties:*

- *Each $\Pi_1, \ldots, \Pi_{2d}$ is a column-wise restricted and 2-locally decomposable.*
- $\Pi = \Pi_{2d} \circ \cdots \circ \Pi_1$.

*Proof.* The first property follows immediately from the structure of Beneš networks, and the second follows by construction. □

## A.3 Succinct Non-Interactive Arguments

In this section, we review the definitions of succinct non-interactive arguments for the problem of Boolean circuit satisfiability.

**Definition A.18** (Succinct Non-Interactive Argument [BCCT12])**.** Let $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ be a family of Boolean circuits indexed by a parameter $n$. A succinct non-interactive argument (SNARG) for the relation $\mathcal{R}_\mathcal{C} = \bigcup_{n \in \mathbb{N}} \mathcal{R}_{C_n}$ (and corresponding language $\mathcal{L}_\mathcal{C} = \bigcup_{n \in \mathbb{N}} \mathcal{L}_{C_n}$) is a tuple of algorithms $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^n) \to (\sigma, \tau)$: On input a security parameter $\lambda$ and a circuit family parameter $n$, the setup algorithm outputs a reference string $\sigma$ and a verification state $\tau$.
- $\mathsf{Prove}(\sigma, \mathbf{x}, \mathbf{w}) \to \boldsymbol{\pi}$: On input the reference string $\sigma$, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prove algorithm outputs a proof $\boldsymbol{\pi}$.
- $\mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) \to \{0, 1\}$: On input the verification state $\tau$, a statement $\mathbf{x}$, and a proof $\boldsymbol{\pi}$, the verification algorithm outputs 1 if it "accepts" and 0 otherwise.

Moreover, $\Pi_{\mathsf{SNARG}}$ satisfies the following additional properties.

- **Completeness:** For all $\lambda \in \mathbb{N}$, all $n \in \mathbb{N}$, and all instances $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_{C_n}$,

$$\Pr[(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \pi \leftarrow \mathsf{Prove}(\sigma, \mathbf{x}, \mathbf{w}) : \mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) = 1] = 1.$$

- **Soundness:** Depending on the notion of soundness:
  - **Non-adaptive Soundness:** For all $n \in \mathbb{N}$, every polynomial-size prover $\mathcal{P}^*$ and all statements $\mathbf{x} \notin \mathcal{L}_{C_n}$,

$$\Pr[(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \boldsymbol{\pi} \leftarrow \mathcal{P}^*(\sigma, \mathbf{x}) : \mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) = 1] = \operatorname{negl}(\lambda).$$

  - **Adaptive Soundness:** For all $n \in \mathbb{N}$, and every polynomial-size prover $\mathcal{P}^*$,

$$\Pr[(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); (\mathbf{x}, \boldsymbol{\pi}) \leftarrow \mathcal{P}^*(\sigma) : \mathsf{Verify}(\tau, \mathbf{x}, \boldsymbol{\pi}) = 1 \text{ and } \mathbf{x} \notin \mathcal{L}_{C_n}] = \operatorname{negl}(\lambda).$$

- **Succinctness:** Depending on the notion of succinctness:
  - **Fully Succinct:** There exists a universal polynomial $p$ (independent of $\mathcal{R}_\mathcal{C}$) such that $\mathsf{Setup}$ runs in time $p(\lambda + \log |C_n|)$, $\mathsf{Prove}$ runs in time $p(\lambda + |C_n|)$, $\mathsf{Verify}$ runs in time $p(\lambda + |x| + \log |C_n|)$, and the length of the proof output by $\mathsf{Prove}$ is bounded by $p(\lambda + \log |C_n|)$.
  - **Preprocessing:** There exists a universal polynomial $p$ (independent of $\mathcal{R}_\mathcal{C}$) such that $\mathsf{Setup}$ runs in time $p(\lambda + |C_n|)$, $\mathsf{Prove}$ runs in time $p(\lambda + |C_n|)$, and $\mathsf{Verify}$ runs in time $p(\lambda + |x| + \log |C_n|)$, and the length of the proof output by $\mathsf{Prove}$ is bounded by $p(\lambda + \log |C_n|)$.

## A.4 Linear-Only Vector Encryption over Rings

In this section, we review the notion of linear-only vector encryption over rings introduced in [BISW17]. In Section 5.2, we show how to combine linear-only vector encryption over rings with quasi-optimal linear MIPs to obtain quasi-optimal SNARGs. A vector encryption encryption over a ring $R$ is an encryption scheme where the message space $R^\ell$ is a vector of ring elements, where $\ell$ here denotes the dimension of the vector. In our constructions, the ring $R = \mathbb{F}_p[x]/\Phi_m(x)$ is a polynomial ring (where $\Phi_m(x)$ here denote the $m^{\text{th}}$ cyclotomic polynomial), and the parameters $m$ and $p$ are chosen so that $R$ splits into $m$ isomorphic copies of $\mathbb{F}_p$. We now introduce the basic schema.

**Definition A.19** (Vector Encryption Scheme over $R$ [BISW17]). Fix a ring $R$. A secret-key vector encryption scheme over $R^\ell$ is a tuple of algorithms $\Pi_{\mathsf{venc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following properties:

- $\mathsf{KeyGen}(1^\lambda, 1^\ell) \to \mathsf{sk}$: On input the security parameter $\lambda$ and the dimension $\ell$ of the message space, the key-generation algorithm outputs a secret key $\mathsf{sk}$.
- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}) \to \mathsf{ct}$: On input the secret key $\mathsf{sk}$ and a vector $\mathbf{v} \in R^\ell$, the encryption algorithm outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to R^\ell \cup \{\perp\}$: On input the secret key $\mathsf{sk}$ and a ciphertext $\mathsf{ct}$, the decryption algorithm either outputs a vector $\mathbf{v} \in R^\ell$ or a special symbol $\perp$ (to denote an invalid ciphertext).

We can define the usual notions of correctness and semantic security [GM82] for a vector encryption scheme. We say a vector encryption scheme over $R^\ell$ is additively homomorphic if given encryptions $\mathsf{ct}_1, \mathsf{ct}_2$ of two vectors $\mathbf{v}_1, \mathbf{v}_2 \in R^\ell$, there is a public operation for computing $\mathsf{ct}'$ of the component-wise sum $\mathbf{v}_1 + \mathbf{v}_2 \in R^\ell$. We now recall the notions of linear targeted malleability and its strengthened analog, linear-only vector encryption that will be essential for constructing SNARGs. Intuitively, both of these definitions capture the property that the vector encryption scheme only supports affine homomorphisms. In other words, given a collection of ciphertexts, any new ciphertext produced by the adversary can be explained as taking a linear combination of existing ciphertexts.

**Definition A.20** (Linear Targeted Malleability [BSW12, adapted]). Fix a security parameter $\lambda$. A secret-key vector encryption scheme $\Pi_{\mathsf{venc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ over a ring $R$ satisfies *linear targeted malleability* if for all efficient adversaries $\mathcal{A}$ and plaintext generation algorithms $\mathcal{M}$ (on input $1^\ell$, algorithm $\mathcal{M}$ outputs vectors in $R^\ell$), there exists a (possibly computationally unbounded) simulator $\mathcal{S}$ such that for any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, the following two distributions are computationally indistinguishable:

**Real Distribution:**

1. $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell)$
2. $(s, \mathbf{v}_1, \ldots, \mathbf{v}_m) \leftarrow \mathcal{M}(1^\ell)$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for all $i \in [m]$
4. $\mathsf{ct}' \leftarrow \mathcal{A}(\{\mathsf{ct}_i\}_{i\in[m]}\,; z)$ where $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \neq \perp$
5. Output $\left(\{\mathbf{v}_i\}_{i\in[m]}, s, \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}')\right)$

**Ideal Distribution:**

1. $(s, \mathbf{v}_1, \ldots, \mathbf{v}_m) \leftarrow \mathcal{M}(1^\ell)$
2. $(\boldsymbol{\pi}, \mathbf{b}) \leftarrow \mathcal{S}(z)$ where $\boldsymbol{\pi} \in R^m$, $\mathbf{b} \in R^\ell$
3. $\mathbf{v}' \leftarrow [\mathbf{v}_1 | \mathbf{v}_2 | \cdots | \mathbf{v}_m] \cdot \boldsymbol{\pi} + \mathbf{b}$
4. Output $\left(\{\mathbf{v}_i\}_{i\in[m]}, s, \mathbf{v}'\right)$

**Definition A.21** (Linear-Only Vector Encryption [BCI<sup></sup>+13, adapted]). Fix a security parameter $\lambda$. A secret-key vector encryption scheme $\Pi_{\mathsf{venc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ over a ring $R$ is a linear-only vector encryption scheme if for all efficient adversaries $\mathcal{A}$, there exists an efficient extractor $\mathcal{E}$ such that for all auxiliary inputs $z \in \{0,1\}^\lambda$, and any plaintext generation algorithms $\mathcal{M}$ (on input $1^\ell$, algorithm $\mathcal{M}$ outputs a vector in $R^\ell$), we have that for $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\ell)$, $(\mathbf{v}_1, \ldots, \mathbf{v}_m) \leftarrow \mathcal{M}(1^\ell)$, $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for all $i \in [m]$, $\mathsf{ct}' \leftarrow \mathcal{A}(\{\mathsf{ct}_i\}_{i\in[m]}\,; z)$, $(\boldsymbol{\pi}, \mathbf{b}) \leftarrow \mathcal{E}(\{\mathsf{ct}_i\}_{i\in[m]}\,; z)$, $\mathbf{v}' \leftarrow [\mathbf{v}_1 | \mathbf{v}_2 | \cdots | \mathbf{v}_m] \cdot \boldsymbol{\pi} + \mathbf{b}$,

$$\Pr[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \neq \mathbf{v}'] = \mathrm{negl}(\lambda).$$

**Remark A.22** (Multiple Ciphertexts)**.** Similar to [BSW12, BCI$^+$13], we can define analogs of linear targeted malleability and linear-only vector encryption where the adversary is allowed to output multiple ciphertexts $\mathsf{ct}'_1, \dots, \mathsf{ct}'_m$. To extend the definitions to this case, we modify the simulator $\mathcal{S}$ in Definition A.20 and the extractor $\mathcal{E}$ in Definition A.21 to output an affine function $(\mathbf{\Pi}, \mathbf{B})$ where $\mathbf{\Pi} \in R^{m \times m}$ and $\mathbf{B} \in R^{\ell \times m}$ that explains the ciphertexts $\mathsf{ct}'_1, \dots, \mathsf{ct}'_m$.

**Remark A.23** (Auxiliary Input Distributions)**.** In Definitions A.20 and A.21, the simulator $\mathcal{S}$ and the extractor $\mathcal{E}$, respectively, are required to succeed for all auxiliary inputs $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$. This seems like a very strong requirement since $z$ can be used to encode difficult problems that the simulator or extractor needs to solve in order to correctly simulate the output distribution [BCPR14]. However, the definitions can be relaxed to only consider "benign" auxiliary-input distributions for which the property holds. For instance, in many scenarios, it suffices that the auxiliary input $z$ is a uniformly random string.

## A.5 Indistinguishability Obfuscation and Puncturable PRFs

In this section, we review the definitions of indistinguishability obfuscation and puncturable PRFs.

**Definition A.24** (Indistinguishability Obfuscation [BGI$^+$01, GGH$^+$13])**.** An indistinguishability obfuscator $i\mathcal{O}$ for a circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is a uniform and efficient algorithm satisfying the following requirements:

- **Correctness.** For all $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_\lambda$, and all inputs $x$, we have that
$$\Pr[C' \leftarrow i\mathcal{O}(C) : C'(x) = C(x)] = 1.$$

- **Indistinguishability.** For all $\lambda \in \mathbb{N}$, and any two circuits $C_0, C_1 \in \mathcal{C}_\lambda$, if $C_0(x) = C_1(x)$ for all inputs $x$, then for all efficient adversaries $\mathcal{A}$, we have that
$$|\Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| = \mathrm{negl}(\lambda).$$

**Definition A.25** (Puncturable PRFs [BW13, KPTZ13, BGI14])**.** A puncturable pseudorandom function with key-space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is an efficiently computable function $\mathsf{F} \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with three additional algorithms $(\mathsf{F.Setup}, \mathsf{F.Puncture}, \mathsf{F.Eval})$ defined as follows:

- $\mathsf{F.Setup}(1^\lambda) \to k$: On input the security parameter $\lambda$, the setup algorithm outputs a PRF key $k \in \mathcal{K}$.

- $\mathsf{F.Puncture}(k, x^*) \to k_{x^*}$: On input the PRF key $k \in \mathcal{K}$ and a point $x^* \in \mathcal{X}$, the puncture algorithm outputs a punctured key $k_{x^*}$.

- $\mathsf{F.Eval}(k_{x^*}, x) \to y$: On input a punctured key $k_{x^*}$, the evaluation algorithm outputs a value $y \in \mathcal{Y} \cup \{\perp\}$.

Moreover $\mathsf{F}$ satisfies the following properties:

- **Correctness:** For all $x^* \in \mathcal{X}$ and $x \neq x^*$, and letting $k \leftarrow \mathsf{F.Setup}(1^\lambda)$, $k_{x^*} \leftarrow \mathsf{F.Puncture}(k, x^*)$, we have that
$$\Pr[\mathsf{F.Eval}(k_{x^*}, x) = \mathsf{F}(k, x)] = 1.$$

- **Pseudorandom at punctured points:** For all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and letting $k \leftarrow \mathsf{F.Setup}(1^\lambda)$, $(\mathsf{st}, x^*) \leftarrow \mathcal{A}_1^{\mathsf{F}(k,\cdot)}(1^\lambda)$, $k_{x^*} \leftarrow \mathsf{F.Puncture}(k, x^*)$, and $y \xleftarrow{\text{R}} \mathcal{Y}$, we have that
$$\left| \Pr\left[\mathcal{A}_2^{\mathsf{F}(k,\cdot)}(\mathsf{st}, k_{x^*}, \mathsf{F}(k, x^*))\right] - \Pr\left[\mathcal{A}_2^{\mathsf{F}(k,\cdot)}(\mathsf{st}, k_{x^*}, y)\right] \right| = \operatorname{negl}(\lambda),$$
provided that $\mathcal{A}_1$ and $\mathcal{A}_2$ do not query $\mathsf{F}(k, \cdot)$ on the challenge point $x^*$.

## B Quasi-Optimal Linear MIP Construction

In this section, we define and analyze the primitives needed for our quasi-optimal linear MIP construction. First, in Appendix B.1, we show how to instantiate our robust decomposition by combining MPC protocols with polylogarithmic overhead [DIK10] with the "MPC-in-the-head" paradigm [IKOS07]. Next, in Appendix B.2, we show how to construct our sequence of non-concentrating permutations that we use for consistency checking. Finally, in Appendix B.3, we provide the formal security analysis of our quasi-optimal linear MIP construction.

### B.1 Robust Decomposition via MPC

In this section, we show how to instantiate our robust decomposition using secure multiparty computation (MPC) protocols with polylogarithmic overhead [DIK10]. Our decomposition follows the "MPC-in-the-head" paradigm of Ishai et al. [IKOS07].

**MPC preliminaries.** We begin by reviewing some standard MPC definitions [Gol04, Can00]. Let $t$ be the number of players, denoted $P_1, \ldots, P_t$. We assume that all players communicate synchronously over secure point-to-point channels. We model the functionality $f$ computed by the $t$ parties as an arithmetic circuit $C$ over a finite field $\mathbb{F}$. In this section, it suffices to consider functionalities whose outputs consists of a single field element $\mathbb{F}$. We assume each party $P_i$ has a common input $\mathbf{x} \in \mathbb{F}^n$ and a local input $\mathbf{w}_i \in \mathbb{F}^m$.

We specify a $t$-party MPC protocol $\Pi$ by its next-message function. In particular, on input a party index $i \in [t]$, the public input $\mathbf{x}$, the party's local input $\mathbf{w}_i$ and randomness $\mathbf{r}_i$, and the messages $P_i$ received $(m_1, \ldots, m_j)$ in the first $j$ rounds of the protocol execution, $\Pi(i, \mathbf{s}, \mathbf{w}_i, \mathbf{r}_i, (m_1, \ldots, m_j))$ outputs a set of $t-1$ messages that $P_i$ sends to each of the other parties in round $j+1$. The view of a party $P_i$, denoted $\mathsf{view}_i$, in the protocol execution consists of its local input $\mathbf{w}_i$, randomness $\mathbf{r}_i$, and all of the messages that $P_i$ both sends and receives[8] during the execution of $\Pi$. At the end of the protocol execution (or if $\Pi$ signals an early termination), each party $P_i$ also computes some output (as a function of its local state). We say that a pair of views $\mathsf{view}_i$ and $\mathsf{view}_j$ for two distinct parties $P_i$ and $P_j$ is consistent (with respect to $\Pi$ and the public input $\mathbf{x}$) if the set of messages sent by $P_i$ (in $\mathsf{view}_i$) are identical to the messages $P_j$ receives (in $\mathsf{view}_j$). We now define the correctness and robustness requirement we require in our construction.

**Definition B.1** (Correctness)**.** An MPC protocol $\Pi$ realizes a deterministic $t$-party functionality $f(\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_n)$ with perfect correctness if on all inputs $\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_n$, the probability (taken over each party's randomness) that the output of some player $P_i$ is different from the output of $f$ is 0.

[8]Typically, one defines the view of a party to only consist of the messages it receives during the computation. In our setting, it will be useful to also include the messages the party sends as part of the view, even though those messages can be computed implicitly from the other components in the view.

**Definition B.2** ($\delta$-Robustness)**.** An MPC protocol $\Pi$ realizes a deterministic $t$-party functionality with perfect $\delta$-robustness if it is perfectly correct in the presence of a semi-honest adversary (as in Definition B.1), and furthermore, for any adversary that corrupts up to $\delta t$ parties, and for any input $(\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_t)$, the following robustness property holds: if there are no inputs $(\mathbf{w}'_1, \ldots, \mathbf{w}'_t)$ where $f(\mathbf{x}, \mathbf{w}'_1, \ldots, \mathbf{w}'_t) = 1$, then the probability (taken over each party's randomness) that some uncorrupted party outputs 1 in an execution of $\Pi$ where the inputs of the honest parties are consistent with $(\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_t)$ is 0.

Note that in our settings, we do not require an additional privacy property from the MPC protocol. With this in mind, we now present our robust decomposition for an arithmetic circuit $C$.

**Construction B.3** (Robust Decomposition via MPC)**.** Let $\delta > 0$ be a constant and $t \in \mathbb{N}$ be an integer. Let $C \colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^{h'}$ be an arithmetic circuit, and $\Pi_f$ be a $t$-party MPC protocol for the $t+1$-input function

$$f(\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_n) = \begin{cases} 1 & \text{if } C(\mathbf{x}, \mathbf{w}_1 \| \mathbf{w}_2 \| \cdots \| \mathbf{w}_n) = \mathbf{0}^{h'} \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

where each $\mathbf{w}_i$ is a vector of dimension $O(m'/t)$. Our $(t, \delta)$-robust decomposition $(f_1, \ldots, f_t, \mathsf{inp}, \mathsf{wit})$ of $C$ is then defined as follows:

- The input encoding function $\mathsf{inp} \colon \mathbb{F}^{n'} \to \mathbb{F}^n$, where $n' = n$, is the identity function.
- The witness encoding function $\mathsf{wit}$ takes as input a statement $\mathbf{x}' \in \mathbb{F}^{n'}$ and a witness $\mathbf{w}' \in \mathbb{F}^{m'}$ and simulates an execution of $\Pi_f$ on inputs $\mathbf{x}'$ and $\mathbf{w}'_1, \ldots, \mathbf{w}'_t$ where $\mathbf{w}' = \mathbf{w}'_1 \| \mathbf{w}'_2 \| \cdots \| \mathbf{w}'_t$. Let $\mathsf{view}_1, \ldots, \mathsf{view}_t$ be the views of each of the $t$ parties in the simulated MPC protocol. The output of the witness encoding functions is a new witness $\mathbf{w} = (\mathsf{view}_1, \ldots, \mathsf{view}_t)$ consisting of the views of the $t$ parties in the execution of $\Pi_f$.
- Each of the constraint functions $f_i$ for $i \in [t]$ takes as input the statement $\mathbf{x} \in \mathbb{F}^n$ and the witness $\mathbf{w} = (\mathsf{view}_1, \ldots, \mathsf{view}_t)$ and outputs 1 if the following conditions hold:
  - The output of party $P_i$ (as determined by $\mathsf{view}_i$) is 1 (indicating an accepting output).
  - The view $\mathsf{view}_i$ of party $P_i$ is consistent with an honest evaluation of $\Pi_f$ (with respect to the global input $\mathbf{x}'$). Recall that $\mathsf{view}_i$ includes not only the local state of party $P_i$ and the set of messages $P_i$ receives from the other parties during the protocol execution, but also the messages $P_i$ sends to each of the other parties during the protocol execution. This step verifies that the messages $P_i$ sends and its output are consistent with those that would be computed assuming an honest evaluation of $\Pi_f$.
  - The messages sent by party $P_i$ (as specified in $\mathsf{view}_i$) are consistent with the messages each of the other parties $P_j$ receives (as specified in $\mathsf{view}_j$).

  In particular, each $f_i$ only needs to read the components of the statement $\mathbf{x}$ that party $P_i$ needs to read during the protocol execution. In addition, $f_i$ only needs to read the components of $\mathbf{w}$ that pertain to party $P_i$: namely, $\mathsf{view}_i$ and the components of $\mathsf{view}_j$ (for all $j \neq i$) containing the messages $P_j$ received from $P_i$.

**Theorem B.4.** *Let $\delta > 0$ be a constant and $t \in \mathbb{N}$ be an integer. Let $C\colon \mathbb{F}^{n'} \times \mathbb{F}^{m'} \to \mathbb{F}^{h'}$ be an arithmetic circuit, and let $\Pi_f$ be a perfectly $\delta$-robust $t$-party MPC protocol for the function in Eq. (B.1). Then, the decomposition $(f_1, \ldots, f_t, \mathsf{inp}, \mathsf{wit})$ in Construction B.3 is $(t, 1-\delta)$-robust.*

*Proof.* We show completeness and robustness separately.

**Completeness.** If $(\mathbf{x}', \mathbf{w}') \in \mathcal{R}_C$, then by perfect correctness of $\Pi_f$, each of the honest parties will output 1. Moreover, in an honest protocol execution, all of the views $\mathsf{view}_i$ for $i \in [t]$ are internally and pairwise consistent.

**Robustness.** To show that the decomposition is $(1-\delta)$-robust, we need to show that for a statement $\mathbf{x}' \notin \mathcal{L}_C$, there is no setting of view $\mathbf{w} = (\mathsf{view}_1, \ldots, \mathsf{view}_t)$ that can satisfy more than a $(1-\delta)$-fraction of the constraints $f_i$. Our argument here is similar to the soundness analysis of the zero-knowledge proof systems from MPC in [IKOS07, Theorem 4.1]. At a high-level, suppose that all inconsistencies (if any) among the views $\mathsf{view}_1, \ldots, \mathsf{view}_t$ can be resolved by eliminating at most $\delta t$ views. In this case, the protocol execution defined by $(\mathsf{view}_1, \ldots, \mathsf{view}_t)$ can be realized by an adversary that corrupts at most $\delta t$ parties. But since $\Pi_f$ is perfectly $\delta$-robust, on input $\mathbf{x}'$, all of the uncorrupted parties will output 0. In this case, there are at most $\delta t$ corrupted parties, so there are at least $(1-\delta)t$ honest parties $P_i$ where the output is 0, and correspondingly, $f_i(\mathbf{x}, \mathbf{w}) = 0$ for those parties. Conversely, if there are more than $\delta t$ views that are inconsistent, then for any $\mathsf{view}_i$ that has an inconsistency, $f_i(\mathbf{x}, \mathbf{w}) = 0$.

More formally, for a collection of views $\mathbf{w} = (\mathsf{view}_1, \ldots, \mathsf{view}_t)$, we say that the view of party $P_i$ for $i \in [t]$ is "bad" if $\mathsf{view}_i$ is either inconsistent with an honest evaluation of $\Pi_f$ on input $\mathbf{x}$, or if there is a discrepancy between an outgoing message from $P_i$ in $\mathsf{view}_i$ and an incoming message for $P_j$ (from $P_i$) in $\mathsf{view}_j$. Let $B \subseteq [t]$ be the subset of "bad" parties. We consider two cases:

- If $|B| < \delta t$, then consider a protocol execution of $\Pi_f$ where the adversary corrupts the set of parties $B$ and behaves in a way so that the view of any party $P_j$ for $j \notin B$ is $\mathsf{view}_j$. By construction of the set $B$, this means that the views between any two parties $i, j \in [t] \setminus B$ are mutually consistent with an honest execution of $\Pi_f$ on input $\mathbf{x}'$. Thus, since $\Pi_f$ is perfectly $\delta$-robust, the output of all parties $P_j$, where $j \notin B$ will be 0. Equivalently, for all $j \in [t] \setminus B$, $f_j(\mathbf{x}, \mathbf{w}) = 0$, so there are at most $(1-\delta)t$ indices $i$ where $f_i(\mathbf{x}, \mathbf{w}) = 1$.
- If $|B| \geq \delta t$, then for all $i \in B$, $f_i(\mathbf{x}, \mathbf{w}) = 0$. This means that there are at most $(1-\delta)t$ indices $i \in [t] \setminus B$ where $f_i(\mathbf{x}, \mathbf{w}) = 1$, and the claim follows. □

**Fact B.5** ([DIK10])**.** Let $t \in \mathbb{N}$ be an integer. For any constant $0 < \delta < 1/3$ and arithmetic circuit $C(\mathbf{x}, \mathbf{w}_1, \ldots, \mathbf{w}_t)$ on $t+1$ inputs with width $\Omega(t)$, there exists a $t$-party MPC protocol $\Pi$ which computes $C$ with perfect $\delta$-robustness and where the total computational complexity is

$$|C| \cdot \mathrm{polylog}(t, |C|) + \mathrm{poly}(t, \log |C|) \cdot \mathrm{depth}(C)^2$$

**Corollary B.6.** *Fix an integer $t \in \mathbb{N}$. Then, for any constant $0 < \delta < 1/3$, there exists a quasi-optimal $(t, 1-\delta)$-robust decomposition for any arithmetic circuit $C\colon \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F}^h$ of size $\Omega(t)$.*

*Proof.* We instantiate Construction B.3 using the information-theoretic MPC protocol from Fact B.5. To obtain the required asymptotics, we make the following observations:

- First, we need to construct an arithmetic circuit that implements the functionality in Eq. (B.1). We construct the circuit $C' \colon \mathbb{F}^n \times \mathbb{F}^m \to \{0,1\}$ from $C$ as follows. Circuit $C'$ first evaluates $C$ on $(\mathbf{x}, \mathbf{w})$ and then projects the $h$ outputs of $C$ onto $\{0,1\}$ (mapping 0 to 0 and all non-zero elements in $\mathbb{F}$ to 1). Projecting each element can be done with a circuit of size and depth $O(\log |\mathbb{F}|)$. Finally $C'$ computes the Boolean AND on the negation of each of the projected output bits. When $\mathbb{F}$ is polynomial-sized (e.g., $|\mathbb{F}| = O(|C|)$), this transformation only adds logarithmic overhead to $C$. In particular, $\text{depth}(C') = \text{depth}(C) + O(\log |\mathbb{F}|)$ and $|C'| = |C| + O(h \cdot \log |\mathbb{F}|)$.

- For proof verification, we can also assume without loss of generality that the circuit $C$ has constant depth. In particular, for verifying that $(\mathbf{x}, \mathbf{w})$ is a satisfying input to $C$, we can always construct a new circuit of size $|C|$ which takes as input the statement $\mathbf{x}$ and the value of each wire in $C(\mathbf{x}, \mathbf{w})$. The new circuit then simply checks that every wire is correctly computed, and that the output value of $C(\mathbf{x}, \mathbf{w})$ is $\mathbf{0}^{h'}$. Coupled with the transformation from the previous step, we conclude that checking satisfiability of an arithmetic circuit $C$ can always be reduced to checking satisfiability of a related circuit $C'$ of size $O(|C| \log |C|)$ and depth $O(\log |C|)$. Invoking Fact B.5 on the circuit $C'$, we conclude that the $\mathsf{inp}$ and $\mathsf{wit}$ encoding functions satisfy the efficiency requirements of Definition 4.1.

- Moreover, when simulating an execution of the MPC protocol from Fact B.5, we uniformly distribute the inputs (i.e., the bits of the witness) across the $t$ parties. This ensures that the computational costs are distributed evenly across all $t$ parties, and so the local computational complexity of each party becomes

$$|C|/t \cdot \text{polylog}(t, |C|) + \text{poly}(t, \log |C|).$$

Since each $f_i$ is verifying integrity of $\mathsf{view}_i$, we conclude that each $f_i$ can be computed by a circuit of size $\widetilde{O}(|C|/t) + \text{poly}(t, \log |C|)$. □

**Remark B.7** (Robust Decomposition and Quasilinear PCPs)**.** Our robust decomposition essentially provides a way to convert a circuit satisfiability instance into checking satisfiability of a collection of smaller constraint functions defined over a common set of variables. This is reminiscent of viewing a traditional PCP (for a circuit satisfiability instance) as a constraint satisfaction problem (CSP), where each constraint in the CSP reads a small number of bits of the PCP. Thus, another potential way of obtaining a quasi-optimal robust decomposition is to use quasilinear PCPs [Din06, BS08]. Specifically, we view the PCP as a CSP instance; an encoding of a statement-witness pair corresponds to an assignment to the variables in the CSP, and the constraint functions in the robust decomposition simply implement the constraints of the CSP. However, with traditional PCPs, the variables on which each constraint depends varies with the statement being proved. One of the requirements of our robust decomposition is that each constraint only depends on a *fixed* subset of the bits of the encoded statement and witness, *irrespective* of the statement being proved. Thus, it is not clear how to leverage traditional PCPs to implement our robust decomposition.

In contrast, our MPC-based robust decomposition satisfies this *input-independence* property. Specifically, the components of the encoded statement-witness pair read by the $i^{\text{th}}$ constraint just correspond to the view of the $i^{\text{th}}$ party in the simulated MPC protocol, which is always a fixed subset of the encoded statement-witness pair, and independent of the statement being proved. It is an interesting problem to construct an input-independent quasilinear PCP, which may in turn yield another approach for realizing our robust decomposition primitive.

## B.2 Constructing Randomized Permutation Decompositions

In this section, we show how to instantiate the underlying building blocks we require for performing our consistency checks. First, we construct a regularity-inducing permutation assuming that every value in the replication structure $\mathbf{A}$ appears at most twice. This assumption is satisfied, for instance, by the replication structure of our robust decomposition based on MPC from Appendix B.1.

**Construction B.8** (Regularity-Inducing Permutations)**.** Fix integers $m, t, q \in \mathbb{N}$, with $t$ even, and let $\mathbf{A} \in [m]^{t \times q}$ be a replication structure where every value in $\mathbf{A}$ appears at most twice. We construct permutations $\Pi_1, \Pi_2$ over the entries of $\mathbf{A}$ as follows:

- First, we partition $\mathbf{A}$ into $t' = t/2$ blocks, each containing a pair of rows: for $i \in [t']$, let $\mathbf{A}_i = \mathbf{A}_{[2(i-1)+1,2i]}$.
- We construct matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)} \in [m]^{t \times q}$ as follows. For each block $i \in [t']$, we associate with it two sets of values $S_i^{(1)}, S_i^{(2)} \subseteq [m]$. For $j \in \{1, 2\}$, $S_i^{(j)}$ is the set of values that appear in the $j^{\text{th}}$ row of $\mathbf{A}_i$, but not in any previous set $S_{i'}^{(j)}$ where $i' < i$. Matrix $\mathbf{A}^{(j)}$ is then constructed as follows:
  1. Initialize all values in $\mathbf{A}^{(j)}$ with $\perp$ to denote an "unassigned" position.
  2. Let $\mathbf{A}_i^{(j)} = \mathbf{A}_{[2(i-1)+1,2i]}^{(j)}$ denote the $i^{\text{th}}$ block of $\mathbf{A}^{(j)}$. Let $v_1, \ldots, v_d \in [m]$ be the elements in $S_i^{(j)}$ where $d \leq q$. Then, for $k \in [d]$, let $c_{v_k}$ be the total number of times $v_k$ appears in $\mathbf{A}$. Set the first $c_{v_k}$ entries of column $k$ of $\mathbf{A}_i^{(j)}$ to the value $v_k$.
  3. For all remaining values that appear in $\mathbf{A}$ but not $\mathbf{A}^{(j)}$, assign them arbitrarily to any unassigned position in $\mathbf{A}^{(j)}$.
- By construction, each $\mathbf{A}^{(j)}$ is a permutation of the entries in $\mathbf{A}$. Output the permutations $\Pi_1, \Pi_2$ where $\mathbf{A}^{(1)} = \Pi_1(\mathbf{A})$ and $\mathbf{A}^{(2)} = \Pi_2(\mathbf{A})$.

**Lemma B.9.** *Fix integers $m, t, q \in \mathbb{N}$, with $t$ even, and let $\mathbf{A} \in [m]^{t \times q}$ be a replication structure where every entry appears at most twice. Let $\Pi_1, \Pi_2$ be the permutations of $\mathbf{A}$ from Construction B.8. Then, $(\Pi_1, \Pi_2)$ is a collection of 2-regularity-inducing permutations (Definition 4.12) with respect to $\mathbf{A}$.*

*Proof.* Set $t' = t/2$, and for $i \in [t']$, let $S_i = \{2(i-1)+1, 2i\}$. Let $M$ be the matching in $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ of size $s$. Take any edge $(i_1, i_2) \in M$, and define $i_1', i_2' \in [t']$ so that $i_1 \in S_{i_1'}$ and $i_2 \in S_{i_2'}$. Without loss of generality, suppose $i_1' \leq i_2'$. This means that there exists $j_1, j_2 \in [q]$ where $\mathbf{A}_{i_1,j_1} = \mathbf{A}_{i_2,j_2}$, but $\mathbf{W}_{i_1,j_1} \neq \mathbf{W}_{i_2,j_2}$. Let $v = \mathbf{A}_{i_1,j_1} \in [m]$. Since each entry in $\mathbf{A}$ appears at most twice, the first block in $\mathbf{A}$ that contains $v$ is $i_1'$. We consider two cases:

- Suppose $i_1 = 1 \bmod 2$. This means that $v \in S_{i_1'}^{(1)}$ in Construction B.8. Thus, $\Pi_1$ maps entries in positions $(i_1, j_1)$ and $(i_2, j_2)$ to some column in rows $i_1$ and $i_1 + 1$. Thus, $\mathcal{G}_{\mathbf{W}_1,\mathbf{A}_1}$ contains an inconsistency in rows $(i_1, i_1 + 1) \in S_{i_1'}$.
- Suppose $i_1 = 0 \bmod 2$. By an analogous argument as that used in the previous case, we conclude that $\mathcal{G}_{\mathbf{W}_2,\mathbf{A}_2}$ contains an inconsistency in rows $(i_1 - 1, i_1) \in S_{i_1'}$.

The above analysis shows that each edge $(i_1, i_2) \in M$ either contributes an edge $(i_1, i_1 + 1) \in S_{i'_1}$ to $\mathcal{G}_{\mathbf{W}_1,\mathbf{A}_1}$ or contribute an edge $(i_1 - 1, i_1) \in S_{i'_1}$ to $\mathcal{G}_{\mathbf{W}_2,\mathbf{A}_2}$. In other words, each edge $(i_1, i_2)$ contributes a single edge to a *regular* matching in $\mathcal{G}_{\mathbf{W}_1,\mathbf{A}_1}$ or $\mathcal{G}_{\mathbf{W}_2,\mathbf{A}_2}$ Since $|M| \geq s$, we conclude that at least one of the graphs $\mathcal{G}_{\mathbf{W}_1,\mathbf{A}_1}, \mathcal{G}_{\mathbf{W}_2,\mathbf{A}_2}$ must contain a regular matching of size at least $s/2$. Moreover, the correspondence between the edges in the regular matching for graphs $\mathcal{G}_{\mathbf{W}_1,\mathbf{A}_1}$ and $\mathcal{G}_{\mathbf{W}_2,\mathbf{A}_2}$ and the edges in $M$ is immediate from the above analysis. $\square$

**Randomized permutation decomposition.** Next, we show how to construct a sequence of non-concentrating 2-locally decomposable permutations. Recall that a non-concentrating sequence of permutations $\Gamma$ implementing a permutation $\Pi$ is an ordered set of permutations $(\Pi_1, \ldots, \Pi_\alpha)$ such that if we start with *any* inconsistency matrix $\mathbf{B} \in \{0,1\}^{t \times q}$ where both $\mathbf{B}$ and $\Pi(\mathbf{B})$ have at most one inconsistency in each row, then the positions of the inconsistencies in the intermediate matrices $\mathbf{B}_\ell = \Pi_\ell(\mathbf{B}_{\ell-1})$ and $\mathbf{B}_0 = \mathbf{B}$ do not concentrate in a small number of rows. We begin by giving a high-level outline of how we sample such sequences for a target permutation $\Pi$.

- First, we apply Lemma A.8 to $\Pi$ to obtain three permutations $\Pi_1, \Pi_2, \Pi_3$ where $\Pi_1$ and $\Pi_3$ are row-wise restricted and $\Pi_2$ is column-wise restricted. Moreover, the decomposition satisfies $\Pi = \Pi_3 \circ \Pi_2 \circ \Pi_1$.

- Next, we randomize the first row-rise restricted permutation $\Pi_1$. Our randomization procedure exploits the observation that if two entries $j_1, j_2$ in the same row have the same target column (under $\Pi$), then we can swap the entries $j_1$ and $j_2$ under $\Pi_1$ and undo the swap in $\Pi_3$. We describe this procedure in Construction B.10. Then, we show in Theorem B.12 that as long as there are many entries in each row of $\Pi$ that map to the same column, this randomization procedure distributes the inconsistencies in $\mathbf{B}$ across many columns.

- After randomizing $\Pi_1$, we apply the randomized 2-local decomposition from Construction A.12 based on randomized routing in a Beneš network to $\Pi_2$ to obtain a (randomized) sequence of 2-local permutations implementing $\Pi_2$. We show in Theorem B.13 that over the randomness used to sample the randomized 2-local decomposition, the inconsistencies in $\mathbf{B}$ do not concentrate in a small number of rows. Intuitively, this follows from the fact that the inconsistencies in $\mathbf{B}$ are distributed across many columns (from the row-wise shuffling procedure in the previous step), and the fact that each column is independently randomized in Construction A.12. We can then show that the probability that the inconsistencies across many columns concentrate in a small number of rows is exponentially small.

- Theorems B.12 and B.13 show that for a *fixed* inconsistency pattern $\mathbf{B}$, the probability that the inconsistencies in $\mathbf{B}$ concentrate in a small number of rows is exponentially small. Here, the probability is taken over the randomness used to sample the row-wise and column-wise decompositions. To construct a collection of permutation sequences $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ such that for *all* inconsistency patterns $\mathbf{B}$, there is a non-concentrating sequence, we simply sample many independent sequences $\Gamma^{(j)}$. In Theorem B.18, we show that if we sample $\gamma = O(\log^3 t)$ such sequences, then with probability $1 - 2^{-\Omega(t)}$, for all inconsistency patterns $\mathbf{B}$ that contain at most one inconsistency in each row, at least one of the sequences $\Gamma^{(j)}$ will be non-concentrating. We give the overall construction in Construction B.17.

**Construction B.10** (Row-Wise Random Permutation Decomposition)**.** Fix positive integers $t, q \in \mathbb{N}$. Let $\Pi$ be a permutation over the entries of a $t$-by-$q$ matrix. The row-wise random permutation decomposition $(\Pi_1, \Pi_2, \Pi_3)$ is then defined as follows:

- First let $(\hat{\Pi}_1, \hat{\Pi}_2, \hat{\Pi}_3)$ be the decomposition of $\Pi$ from Lemma A.8. In particular, $\hat{\Pi}_1$ and $\hat{\Pi}_3$ are row-wise restricted permutations and $\hat{\Pi}_2$ is a column-wise restricted permutation.
- Since $\hat{\Pi}_1$ is a row-wise restricted permutation, we can decompose it into a product of $t$ independent permutations $\hat{\Pi}_1^{(1)}, \ldots, \hat{\Pi}_1^{(t)}$, where the $i$th permutation $\hat{\Pi}_1^{(i)} \colon [q] \to [q]$ is applied to the $i$th row of the matrix. Similarly, we can express $\hat{\Pi}_2$ as a product of $q$ independent permutations $\hat{\Pi}_2^{(1)}, \ldots, \hat{\Pi}_2^{(q)}$, where each $\hat{\Pi}_2^{(j)} \colon [t] \to [t]$ is applied to the $j$th column of the matrix.
- For each $i \in [t]$, define the vector $\hat{\mathbf{c}}^{(i)} \in [t]^q$, where for all $j \in [q]$, $\hat{\mathbf{c}}_j^{(i)}$ is the permuted row-index of the $(i,j)^{\text{th}}$ entry of the matrix under $\hat{\Pi}_2 \circ \hat{\Pi}_1$. Specifically, $\hat{\mathbf{c}}_j^{(i)} = \hat{\Pi}_2^{(\hat{\Pi}_1^{(i)}(j))}(i)$. Then, define sets $\hat{T}_1^{(i)}, \ldots, \hat{T}_t^{(i)} \subseteq [q]$ where $\hat{T}_\beta^{(i)}$ is the set of indices $j \in [q]$ where $\hat{\mathbf{c}}_j^{(i)} = \beta$. Clearly, $\hat{T}_1^{(i)}, \ldots, \hat{T}_t^{(i)}$ form a partition for $[q]$. For each $\beta \in [t]$, sample a random permutation $\pi_\beta^{(i)} \colon [q] \to [q]$ that *randomly* permutes the indices in $\hat{T}_\beta^{(i)}$ and leaves the other indices unchanged. Define $\Pi_1^{(i)} = \pi_q^{(i)} \circ \cdots \circ \pi_1^{(i)} \circ \hat{\Pi}_1^{(i)}$ and $\Pi_1$ to be the row-wise restricted permutation where its action on row $i$ is given by $\Pi_1^{(i)}$.
- Let $\Pi_2 = \hat{\Pi}_2$ and choose $\Pi_3$ such that $\Pi = \Pi_3 \circ \Pi_2 \circ \Pi_1$.

**Lemma B.11.** *Fix positive integers $t, q \in \mathbb{N}$ and let $\Pi$ be a permutation over the entries of a $t$-by-$q$ matrix. Let $(\Pi_1, \Pi_2, \Pi_3)$ be the row-wise random permutation decomposition of $\Pi$ from Construction B.10. Then, $\Pi = \Pi_3 \circ \Pi_2 \circ \Pi_1$, both $\Pi_1$ and $\Pi_3$ are row-wise restricted, and $\Pi_2$ is column-wise restricted.*

*Proof.* By construction, $\Pi = \Pi_3 \circ \Pi_2 \circ \Pi_1$. Let $\hat{\Pi}_1, \hat{\Pi}_2, \hat{\Pi}_3$ be the permutations used to construct $\Pi_1, \Pi_2, \Pi_3$ in Construction B.10. Since $\hat{\Pi}_1$ is row-wise restricted and permutation $\hat{\Pi}_2$ is column-wise restricted, $\Pi_1$ and $\Pi_2$ are by construction row-wise restricted and column-wise restricted, respectively. It suffices to show that $\Pi_3$ is row-wise restricted.

Take any entry $(i,j) \in [t] \times [q]$, and let $(i_t, j_t)$ be its image under permutation $\Pi$. Let $(i_1, j_1) = \Pi_1(i,j)$ and $(i_2, j_2) = \Pi_2(i_1, j_1)$. We show that $i_2 = i_t$, or equivalently, after applying $\Pi_2 \circ \Pi_1$, every entry $(i,j)$ ends up in the same row as $\Pi(i,j)$. In this case, the permutation $\Pi_3$ only needs to changes the column of each entry, and the claim follows. We show this in a sequence of steps.

- Let $(i_1', j_1') = \hat{\Pi}_1(i,j)$ and $(i_2', j_2') = \hat{\Pi}_2(i_1', j_1') = \Pi_2(i_1', j_1')$, since $\Pi_2 = \hat{\Pi}_2$.
- For $j \in [q]$, let $\Pi_2^{(j)} \colon [t] \to [t]$ be the permutation $\Pi_2$ implements on the $j$th column. In addition, let $\Pi_1^{(i)}, \hat{\Pi}_1^{(i)} \colon [q] \times [q]$ be the permutations $\Pi_1$ and $\hat{\Pi}_1$ implement on $i$th row, respectively.
- Since $\Pi_1$, $\hat{\Pi}_1$, and $\hat{\Pi}_3$ are row-wise restricted, $i_1' = i = i_1$ and $i_2' = i_t$. Moreover, by definition, $i_2' = \Pi_2^{(j_1')}(i)$, $i_2 = \Pi_2^{(j_1)}(i)$, $j_1' = \hat{\Pi}_1^{(i)}(j)$, and $j_1 = \Pi_1^{(i)}(j)$.
- By construction,

$$j_1 = \Pi_1^{(i)}(j) = \big(\pi_q^{(i)} \circ \cdots \circ \pi_1^{(i)}\big)\big(\hat{\Pi}_1^{(i)}(j)\big) = \big(\pi_q^{(i)} \circ \cdots \circ \pi_1^{(i)}\big)(j_1').$$

By design, each $\pi_\beta^{(i)}$ for $\beta \in [q]$ only permutes indices $j \in [q]$ where $\hat{\Pi}_2^{(\hat{\Pi}_1^{(i)}(j))}(i) = \beta$. In particular, this means that if $j_1 = \left(\pi_q^{(i)} \circ \cdots \circ \pi_1^{(i)}\right)(j_1')$, it must be the case that $i_2' = \hat{\Pi}_2^{(j_1')}(i) = \hat{\Pi}_2^{(j_1)}(i) = i_2$. Since $i_2' = i_t$, this means that $i_2 = i_t$, which complete the proof. □

**Theorem B.12** (No-Concentration in Columns)**.** *Fix integers $t, q \in \mathbb{N}$, where $q = \mathrm{poly}(t)$, and let $\Pi$ be a permutation over the entries of a $t$-by-$q$ matrix. Define the following quantities:*

- *Let $(\Pi_1, \Pi_2, \Pi_3)$ be the row-wise random permutation decomposition of $\Pi$ from Construction B.10.*
- *Let $\mathbf{B} \in \{0,1\}^{t \times q}$ be an inconsistency matrix with $z$ inconsistencies at indices $(i_1, j_1), \ldots, (i_z, j_z) \in [t] \times [q]$. In particular, $\mathbf{B}_{i_\beta, j_\beta} = 1$ for all $\beta \in [z]$.*
- *Let $(i_1', j_1'), \ldots, (i_z', j_z')$ be the positions of the inconsistencies in $\Pi_1(\mathbf{B})$.*

*Suppose that the following condition holds:*

- ***Condition 1:** All of the $i_1, \ldots, i_z$ are distinct. Namely, each row of $\mathbf{B}$ contains at most one inconsistency.*
- ***Condition 2:** For every pair of rows $i_1, i_2 \in [t]$, there are at least $t$ indices $j \in [q]$ such that $\Pi(i_1, j)$ is in row $i_2$.*

*For a set $S \subseteq [q]$ of column indices, let $n_S$ denote the number of indices $\beta \in [z]$ where $j_\beta' \in S$. Then, for every constant $c_{\mathsf{col}} > 0$, there exists a constant $s_{\mathsf{col}} > 0$ such that with probability $1 - 2^{-\Omega(t/\log t)}$ (taken over the randomness used to sample $\Pi$), the following condition holds: for all sets $S \subseteq [q]$ where $|S| \leq s_{\mathsf{col}} \cdot t/\log^2 t$, we have that $n_S < c_{\mathsf{col}} \cdot t/\log t$.*

*Proof.* Let $c_{\mathsf{col}} > 0$ be any constant, and fix a constant $s_{\mathsf{col}} > 0$. Let $S \subseteq [q]$ be a set where $|S| \leq s_{\mathsf{col}} \cdot t/\log^2 t$. Suppose Conditions 1 and 2 hold and consider the event where $n_S \geq c_{\mathsf{col}} \cdot t/\log t$. We first show that this event happens with probability $2^{-k \cdot c_{\mathsf{col}} \cdot t/\log t}$ for some constant $k > 0$ (independent of $s_{\mathsf{col}}$), where the probability is taken over the randomness used to sample $\Pi_1$ in Construction B.10.

- First, let $\hat{\Pi}_2$ be the column-wise restricted permutation from Construction B.10, and for $j \in [q]$, let $\hat{\Pi}_2^{(j)} \colon [t] \to [t]$ be the permutation $\hat{\Pi}_2$ implements on column $j$. By Condition 2 and the fact that $\hat{\Pi}_1$ and $\hat{\Pi}_3$ are both row-wise restricted, this means that for all pairs of rows $i_1, i_2 \in [t]$, there are at least $t$ indices $j \in [q]$ where $\hat{\Pi}_2^{(j)}(i_1) = i_2$. This means that for each row $i \in [t]$, the sets $\hat{T}_1^{(i)}, \ldots, \hat{T}_t^{(i)}$ in Construction B.10 all contain at least $t$ elements. Thus, over the randomness used to sample $\Pi_1$, for any inconsistency $(i_\beta, j_\beta)$, the value of $j_\beta'$ is uniformly random over a set of size at least $t$. This means that for all $\beta \in [z]$, $\Pr[j_\beta' \in S] \leq |S|/t = O(1/\log^2 t)$ since $|S| = O(t/\log^2 t)$. In particular, for all $\beta \in [z]$ (and sufficiently large $t$), $\Pr[j_\beta' \in S] \leq c_{\mathsf{col}}/(2\log t)$
- For $\beta \in [z]$, let $X_\beta$ be an indicator random variable for the event $j_\beta' \in S$. From the above analysis, we have that $\Pr[X_\beta = 1] \leq c_{\mathsf{col}}/(2\log t)$. Moreover, $\Pi_1$ is composed of $t$ independent row permutations, and there is only one inconsistency in each row of $\mathbf{B}$ (Condition 1), so the

variables $X_1, \dots, X_z$ are all independent. By definition, $n_S = \sum_{\beta \in [z]} X_\beta$, so by a Chernoff bound,

$$\Pr[n_S \geq c_{\mathsf{col}} \cdot t/\log t] \leq 2^{-k \cdot c_{\mathsf{col}} \cdot (t/\log t)},$$

where $k > 0$ is a constant.

To conclude the proof, we apply a union over all sets $S \subseteq [q]$ of size $s_{\mathsf{col}} \cdot t/\log^2 t$. The number of such sets is bounded by

$$\binom{q}{s_{\mathsf{col}} \cdot t/\log^2 t} \leq q^{s_{\mathsf{col}} \cdot t/\log^2 t} \leq 2^{k' \cdot s_{\mathsf{col}} \cdot (t/\log t)},$$

for some constant $k' > 0$ (independent of $s_{\mathsf{col}}$) since $q = \mathrm{poly}(t)$. The claim follows by taking $s_{\mathsf{col}} < k/k' \cdot c_{\mathsf{col}}$. ☐

**Theorem B.13** (No-Concentration in Rows)**.** *Fix positive integers $m, t, q \in \mathbb{N}$ where $t = 2^d$ for some $d \in \mathbb{N}$, and $q = \mathrm{poly}(t)$. Then, define the following quantities:*

- *Let $\Pi$ be a column-wise restricted permutation over the entries of a $t$-by-$q$ matrix, and $\Pi_1, \dots, \Pi_{2d}$ be a randomized 2-local decomposition of $\Pi$ from Construction A.16.*
- *Let $\mathbf{B} \in \{0,1\}^{t \times q}$ be an inconsistency matrix with $z = \Omega(t)$ inconsistencies at locations $(i_1, j_1), \dots, (i_z, j_z) \in [t] \times [q]$. In other words, $\mathbf{B}_{i_1,j_1} = \cdots = \mathbf{B}_{i_z,j_z} = 1$.*
- *For $\ell \in [2d]$, let $(i_1^{(\ell)}, j_1^{(\ell)}), \dots, (i_z^{(\ell)}, j_z^{(\ell)})$ be the positions $(i_1, j_1), \dots, (i_z, j_z)$ permuted according to the permutation $\Pi_\ell \circ \cdots \circ \Pi_1$.*
- *For a subset $S \subseteq [t]$ of row indices, let $n_S^{(\ell)} = \left|\left\{\beta \in [z] : i_\beta^{(\ell)} \in S\right\}\right|$. In words, $n_S^{(\ell)}$ is the number of inconsistencies in $(\Pi_\ell \circ \cdots \circ \Pi_1)(\mathbf{B})$ that fall into the rows identified by $S$.*

*Suppose that the following conditions hold:*

- ***Condition 1:** All of the $i_1, \dots, i_z$ are distinct. Similarly, $i_1^{(2d)}, \dots, i_z^{(2d)}$ are also all distinct. Namely, each row of $\mathbf{B}$ and $\Pi(\mathbf{B})$ contain at most one inconsistency. In particular, $z \leq t$.*
- ***Condition 2:** For every constant $c_{\mathsf{col}} > 0$, there exists a constant $s_{\mathsf{col}} > 0$ such that the following holds: for all sets $S \subseteq [q]$ of column indices where $|S| \leq s_{\mathsf{col}} \cdot t/\log^2 t$, the number of indices $\beta \in [z]$ where $j_\beta \in S$ is less than $c_{\mathsf{col}} \cdot t/\log t$.*

*Then, for all constants $c_{\mathsf{row}} > 0$ and $s_{\mathsf{row}} > 0$, with probability $1 - 2^{-\Omega(t/\log^2 t)}$ (taken over the choice of randomness in sampling the 2-local decomposition of $\Pi$), the following holds: for all indices $\ell \in [2d]$ and sets $S \subseteq [t]$ where $|S| \leq s_{\mathsf{row}} \cdot t/\log^5 t$, it holds that $n_S^{(\ell)} < c_{\mathsf{row}} \cdot t/\log t$.*

*Proof.* Fix constants $c_{\mathsf{row}}, s_{\mathsf{row}} > 0$, a set $S \subseteq [t]$ where $|S| \leq s_{\mathsf{row}} \cdot t/\log^5 t$ and an index $\ell \in [2d]$. Suppose that Conditions 1 and 2 hold, and consider the event where $n_S^{(\ell)} \geq c_{\mathsf{row}} \cdot t/\log t$. We now show that this event occurs with probability $2^{-\Omega(t/\log^2 t)}$. We first analyze the case where $\ell \leq d$. The $\ell > d$ case is analogous. First, by appealing to Condition 2, we argue that the inconsistencies in $\mathbf{B}$ must be distributed across many columns.

**Claim B.14.** *Let $s_{\mathsf{col}} > 0$ be the constant from Condition 2 for the case where $c_{\mathsf{col}} = c_{\mathsf{row}}$. Then, there exists a set $T \subseteq [q]$ of column indices where $|T| \geq (s_{\mathsf{col}}/2) \cdot t/\log^2 t$ satisfying the following properties:*

- *For all $j \in T$, the number of indices $\beta \in [z]$ where $j_\beta = j$ is at most $2/s_{\mathsf{col}} \cdot \log^2 t$. That is, the $j^{\text{th}}$ column of $\mathbf{B}$ contains at most $(2/s_{\mathsf{col}}) \cdot \log^2 t$ inconsistencies.*
- *For all $j \in T$, there is at least one $\beta \in [z]$ where $i_\beta^{(\ell)} \in S$ and $j_\beta^{(\ell)} = j$. That is, after applying $\Pi_1, \ldots, \Pi_\ell$ to $\mathbf{B}$, there is an inconsistency in row $i$ and column $j$ of the resulting matrix.*

*Proof.* Let $\hat{T} = \left\{ j_\beta^{(\ell)} : \beta \in [z] \text{ and } i_\beta^{(\ell)} \in S \right\}$ be the set of column indices of the inconsistencies whose row indices fall into set $S$ in layer $\ell$. Our argument proceeds in two steps:

- Since $\Pi_1, \ldots, \Pi_\ell$ are column-wise restricted, it follows that $j_\beta^{(\ell)} = j_\beta$ for all $\beta \in [z]$. By assumption, $n_S^{(\ell)} \geq c_{\mathsf{row}} \cdot t/\log t$. This means that $\left|\hat{T}\right| > s_{\mathsf{col}} \cdot t/\log^2 t$, since otherwise, $\hat{T}$ is a set with at most $s_{\mathsf{col}} \cdot t/\log^2 t$ indices that contains $c_{\mathsf{row}} \cdot t/\log t = c_{\mathsf{col}} \cdot t/\log t$ indices from the multiset $\{j_1, \ldots, j_z\}$. This violates Condition 2.
- Define $T \subseteq \hat{T}$ to be the set of indices $j \in \hat{T}$ where the $j^{\text{th}}$ column of $\mathbf{B}$ contains fewer than $(2/s_{\mathsf{col}}) \cdot \log^2 t$ inconsistencies. By construction, the set $T$ satisfies both of the required properties. It suffices to argue that $|T| \geq (s_{\mathsf{col}}/2) \cdot t/\log^2 t$. Suppose otherwise. From above, we know that $\left|\hat{T}\right| > s_{\mathsf{col}} \cdot t/\log^2 t$, and so if $|T| < (s_{\mathsf{col}}/2) \cdot t/\log^2 t$, there are more than $(s_{\mathsf{col}}/2) \cdot t/\log^2 t$ columns in $\mathbf{B}$ that contain $2/s_{\mathsf{col}} \cdot \log^2 t$ inconsistencies, which means that $\mathbf{B}$ contains more than $t$ inconsistencies, which is a contradiction. Thus, $|T| \geq (s_{\mathsf{col}}/2) \cdot t/\log^2 t$, and the claim follows. ☐

In the following, we take $s_{\mathsf{col}} > 0$ to be the constant from Claim B.14. Now, since the permutations $\Pi_1, \ldots, \Pi_{2d}$ are generated according to Construction A.16, the entries in each column of the inconsistency matrix are routed using independent $\textsc{beneš}_d$ networks (each of which implements the permutation $\Pi$ on its respective column). This means that in layer $\ell$, we can partition the $t$ rows of $\mathbf{W}$ into $r = 2^{d-\ell}$ disjoint subsets $R_1, \ldots, R_r$, each containing $2^\ell$ rows, such that the permutation $\Pi_\ell \circ \cdots \circ \Pi_1$ factors into the product of $r$ independent permutations, each operating on one of the subsets $R_1, \ldots, R_r$ (Fact A.10). Let $w = |S|$ and $w_1, \ldots, w_r$ be the number of rows of $S$ that fall into each of the subsets $R_1, \ldots, R_r$, respectively. We now show that there are not many blocks in $\mathbf{B}$ where a large fraction of the rows within those blocks contain an inconsistency.

**Claim B.15.** *For any constant $\varepsilon > 0$ (and sufficiently large $t$), there are at most $(s_{\mathsf{col}}/4) \cdot t/\log^2 t$ rows $i \in [t]$ where $i \in R_k$ for some $k \in [r]$ where $w_k/2^\ell > \varepsilon/\log^2 t$.*

*Proof.* Suppose there are $(s_{\mathsf{col}}/4) \cdot t/\log^2 t = (s_{\mathsf{col}}/4)(r \cdot 2^\ell)/\log^2 t$ rows $i$ where $i \in R_k$ and $w_k/2^\ell > \varepsilon/\log^2 t$. Since each block contains $2^\ell$ rows, this means there are at least $(s_{\mathsf{col}}/4) \cdot r/\log^2 t$ blocks $k \in [r]$ where $w_k/2^\ell > \varepsilon/\log^2 t$. But now,

$$w = \sum_{k \in [r]} w_k > (s_{\mathsf{col}}/4)(r/\log^2 t)(2^\ell \cdot \varepsilon/\log^2 t) = \varepsilon \cdot (s_{\mathsf{col}}/4) \cdot t/\log^4 t = \Theta(t/\log^4 t).$$

This is a contradiction since $w = |S| = O(t/\log^5 t)$. ☐

Let $T$ be the set of column indices from Claim B.14 where for all $j \in T$, the $j^{\text{th}}$ column in $\mathbf{B}$ contains at most $2/s_{\mathsf{col}} \cdot \log^2 t$ inconsistencies. To complete the proof, we first say that a column $j \in T$ is "good" if for all $(i_\beta, j_\beta)$ where $j_\beta = j$, then $i_\beta \in R_k$ for some $k$ where $w_k/2^\ell < s_{\mathsf{col}}/(4\log^2 t)$. Otherwise, we say the column is "bad." By Condition 1, each row has at most one inconsistency, and by Claim B.15, the number of rows where $i_\beta \in R_k$ for some $k$ where $w_k/2^\ell > s_{\mathsf{col}}/(4\log^2 t)$ is at most $(s_{\mathsf{col}}/4) \cdot t/\log^2 t$. Thus, there can be at most $(s_{\mathsf{col}}/4) \cdot t/\log^2 t$ "bad" columns. Since $|T| \geq (s_{\mathsf{col}}/2) \cdot t/\log^2 t$, we conclude that there are at least $(s_{\mathsf{col}}/4) \cdot t/\log^2 t$ "good" columns in $T$. We now show the following claim:

**Claim B.16.** *If $j \in T$ is a "good" column, then the probability that there exists $\beta \in [z]$ where $j_\beta^{(\ell)} = j$ and $i_\beta^{(\ell)} \in S$ is at most $1/2$. Here, the probability is taken over the randomness used to sample the routing configuration for column $j$.*

*Proof.* Take any inconsistency $(i_\beta, j_\beta)$ where $j_\beta = j$. Set $k \in [r]$ so that $i_\beta \in R_k$. By Lemma A.14, over the choice of the randomness used to sample the routing configuration for column $j$, the distribution of $i_\beta^{(\ell)}$ is uniform over $R_k$. Therefore, $\Pr[i_\beta^{(\ell)} \in S] = w_k/2^\ell < s_{\mathsf{col}}/(4\log^2 t)$, since $j$ is a "good" column. Finally, since $j \in T$, column $j$ contains at most $(2/s_{\mathsf{col}}) \cdot \log^2 t$ inconsistencies. By a union bound, the probability that there exists $\beta \in [z]$ where $j_\beta = j$ and $i_\beta^{(\ell)} \in S$ is bounded by $1/2$. ☐

From above, there are at least $(s_{\mathsf{col}}/4) \cdot t/\log^2 t$ "good" columns in $T$. Moreover, for every $j \in T$, there is a $\beta \in [z]$ where $i_\beta^{(\ell)} \in S$ and $j_\beta^{(\ell)} = j$. Since Construction A.16 samples the routing configuration for each of the columns of $\Pi$ randomly and independently, we conclude from Claim B.16 that

$$\Pr[\forall j \in T : i_\beta^{(\ell)} \in S \text{ and } j_\beta^{(\ell)} = j] \leq 2^{-(s_{\mathsf{col}}/4)\cdot t/\log^2 t} = 2^{-\Omega(t/\log^2 t)},$$

where the probability is taken over the randomness used to sample the decomposition $\Pi$. Correspondingly, this implies that the probability that $n_S^{(\ell)} \geq c_{\mathsf{row}} \cdot t/\log t$ is bounded by $2^{-\Omega(t/\log^2 t)}$.

To conclude the proof, we apply a union over all sets $S \subseteq [q]$ of size $s = O(t/\log^5 t)$ and all indices $\ell \in [2d]$. The number of such sets is bounded by $q^s \leq 2^{O(t/\log^4 t)}$ since $q = \mathrm{poly}(t)$. Moreover, $d = \log t$, so by a union bound, the probability that there exists $S \subseteq [t]$ of size $|S| \leq s$ and $\ell \in [2d]$ where $n_S^{(\ell)} \geq c_{\mathsf{row}} \cdot t/\log t$ is $2^{-\Omega(t/\log^2 t)}$. The claim follows. ☐

**Construction B.17** (Randomized Permutation Decomposition)**.** Fix positive integers $t, q, \gamma \in \mathbb{N}$ where $t = 2^d$ for some $d \in \mathbb{N}$, and $q, \gamma = \mathrm{poly}(t)$. Let $\Pi$ be a permutation over the entries of a $t$-by-$q$ matrix. The randomized permutation decomposition of $\Pi$ is a collection of $\gamma$ sequences of permutations $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ where each $\Gamma^{(i)} = \left(\Pi_1^{(1)}, \Pi_{2,1}^{(1)}, \ldots, \Pi_{2,2d}^{(1)}, \Pi_3^{(1)}\right)$ is a sequence of $\alpha = 2d + 2$ 2-locally decomposable permutations on the entries of a $t$-by-$q$ matrix. We construct each sequence $\Gamma^{(i)}$ as follows:

- For each $i \in [\gamma]$, apply the row-wise random permutation decomposition from Construction B.10 to $\Pi$ to obtain permutations $(\Pi_1^{(i)}, \Pi_2^{(i)}, \Pi_3^{(i)})$.
- For each $i \in [\gamma]$, apply the randomized 2-local decomposition from Construction A.12 to $\Pi_2^{(i)}$ to obtain $\Pi_{2,1}^{(i)}, \ldots, \Pi_{2,2d}^{(i)}$.

By construction, each permutation sequence $\Gamma^{(i)}$ implements $\Pi$ in the following sense: $\Pi = \Pi_3^{(i)} \circ \Pi_{2,2d}^{(i)} \circ \cdots \circ \Pi_{2,1}^{(i)} \circ \Pi_1^{(i)}$.

**Theorem B.18** (Randomized Permutation Decomposition)**.** *Fix positive integers $t, q, \gamma \in \mathbb{N}$ where $t = 2^d$ for some $d \in \mathbb{N}$ and $q = \text{poly}(t)$. Then, define the following quantities:*

- *Let $\Pi$ be a permutation over $t$-by-$q$ matrices where for every pair of rows $i_1, i_2 \in [t]$, there are at least $t$ indices $j \in [q]$ such that $\Pi(i_1, j)$ is in row $i_2$.*
- *Let $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ be the collection of permutation sequences obtained by applying Construction B.17 to $\Pi$. In particular, $\Gamma^{(j)} = \left(\Pi_1^{(j)}, \Pi_{2,1}^{(j)}, \ldots, \Pi_{2,2d}^{(j)}, \Pi_3^{(j)}\right)$ for all $j \in [\gamma]$.*
- *Let $\mathcal{B} \subseteq \{0,1\}^{t \times q}$ be the set of inconsistency patterns $\mathbf{B}$ where $\mathbf{B}$ and $\Pi(\mathbf{B})$ have at most one inconsistency in each row.*

*Let $c_1, c_2 > 0$ be arbitrary constants, and let $\kappa_1 = c_1 \cdot t/\log^5 t$ and $\kappa_2 = c_2 \cdot t/\log t$. Then, there exists $\gamma = O(\log^3 t)$ such that with probability $1 - 2^{-\Omega(t)}$ over the choice of randomness in Construction B.17, the collection of permutation sequences $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ implementing $\Pi$ is $(\kappa_1, \kappa_2)$-non-concentrating for $\mathcal{B}$ (Definition 4.13).*

*Proof.* We use a union bound. Let $\mathbf{B} \in \mathcal{B}$ be an inconsistency matrix with $z$ inconsistencies at indices $(i_1, k_1), \ldots, (i_z, k_z)$. Consider the probability (over the randomness used by Construction B.17) that the sequence $\Gamma^{(j)} = \left(\Pi_1^{(j)}, \Pi_{2,1}^{(j)}, \ldots, \Pi_{2,2d}^{(j)}, \Pi_3^{(j)}\right)$ is $(\kappa_1, \kappa_2)$-concentrating for $\mathbf{B}$.

- For $\beta \in [z]$, let $k_{\beta,1}$ denote the column index of $(i_\beta, k_\beta)$ after applying the first permutation $\Pi_1^{(j)}$. By Theorem B.12, for all constants $c_{\mathsf{col}} > 0$, there exists a constant $s_{\mathsf{col}} > 0$ such that with probability $1 - 2^{-\Omega(t/\log t)}$, the following holds: for all sets $S \subseteq [q]$ where $|S| \le s_{\mathsf{col}} \cdot t/\log^2 t$, the number of indices $\beta \in [z]$ where $k_{\beta,1} \in S$ is less than $c_{\mathsf{col}} \cdot t/\log t$.
- For $\beta \in [z]$ and $\ell \in [2d]$, let $i_{\beta,2,\ell}$ denote the row index of $(i_\beta, k_\beta)$ after applying the sequence of permutations $\Pi_{2,\ell}^{(j)} \circ \cdots \circ \Pi_{2,1}^{(j)} \circ \Pi_1^{(j)}$. By Theorem B.13, with probability $1 - 2^{-\Omega(t/\log^2 t)}$, the following holds: for all indices $\ell \in [2d]$ and all sets $S \subseteq [t]$ where $|S| \le c_1 \cdot t/\log^5 t$, the number of indices $\beta \in [z]$ where $i_{\beta,2,\ell} \in S$ is less than $c_2 \cdot t/\log t$. In other words, with probability $1 - 2^{-\Omega(t/\log^2 t)}$, no subset of $\kappa_1$ rows in $\Pi_{2,\ell}^{(j)} \circ \cdots \circ \Pi_{2,1}^{(j)} \circ \Pi_1^{(j)}(\mathbf{B})$ contain more than $\kappa_2$ inconsistencies.
- By assumption, $\mathbf{B}$ has at most 1 inconsistency in each row. Thus, no subset of $\kappa_1$ rows can contain $\kappa_2$ inconsistencies (for sufficiently large $t$). Since $\Pi_1^{(j)}$ is a row-wise restricted permutation, $\Pi_1^{(j)}(\mathbf{B})$ also contains at most 1 inconsistency in each row. Finally, by assumption $\Pi(\mathbf{B})$ contains at most 1 inconsistency in each row, so no subset of $\kappa_1$ rows of $\Pi(\mathbf{B})$ can contain $\kappa_2$ inconsistencies.

By the above analysis, the probability that $\Gamma^{(j)}$ is *not* $(\kappa_1, \kappa_2)$-non-concentrating for $\mathbf{B}$ is bounded by $2^{-\Omega(t/\log^2 t)}$. Since Construction B.17 samples all of the permutation sequences $\Gamma^{(j)}$ independently, the probability that $\Gamma^{(j)}$ is *not* $(\kappa_1, \kappa_2)$-non-concentrating for $\mathbf{B}$ for all $j \in [\gamma]$ is bounded by $2^{-\Omega(\gamma t/\log^2 t)}$. Concretely, let $2^{-c_1' \cdot \gamma t/\log^2 t}$ where $c_1' > 0$ is a constant be a bound on the probability that $\Gamma^{(j)}$ is not $(\kappa_1, \kappa_2)$-non-concentrating for $\mathbf{B}$ for all $j \in [\gamma]$

To complete the analysis, we use a union bound to bound the probability that $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ is not $(\kappa_1, \kappa_2)$-non-concentrating for the set $\mathcal{B}$. First, we have $|\mathcal{B}| \leq (q+1)^t$, since $\mathcal{B}$ only contains inconsistency matrices with at most one inconsistency in each row. Since $q = \mathrm{poly}(t)$, this means that $|\mathcal{B}| \leq 2^{c_2' \cdot t \log t}$ for some constant $c_2' > 0$. By the union bound, the probability that there exists $\mathbf{B} \in \mathcal{B}$ for which $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ is not $(\kappa_1, \kappa_2)$-non-concentrating for $\mathbf{B}$ is at most $2^{c_2' \cdot t \log t - c_1' \cdot \gamma t / \log^2 t}$. Setting $\gamma = 2 \cdot c_2'/c_1' \log^3 t = O(\log^3 t)$, this probability becomes $2^{-c_2' \cdot t \log t} = 2^{-\Omega(t)}$. Thus, with probability $1 - 2^{-\Omega(t)}$, the collection of permutation sequences $\Gamma^{(1)}, \ldots, \Gamma^{(\gamma)}$ is $(\kappa_1, \kappa_2)$-non-concentrating for $\mathcal{B}$. $\square$

**Remark B.19** (Padding the Linear PCPs)**.** Theorem B.18 requires that the permutation $\Pi$ over $t$-by-$q$ matrices has the property that for every pair of rows $i_1, i_2 \in [t]$, there are at least $\Omega(t)$ indices $j \in [q]$, where $\Pi(i_1, j)$ is in row $i_2$. In the quasi-optimal linear MIP construction (Construction 4.14), the permutations $\Pi$ on which we apply the randomized permutation decomposition may not natively satisfy this property. This problem can be addressed by defining a new permutation $\Pi'$ that operates on $t$-by-$(q + t^2)$ matrices as follows:

- For all $i \in [t]$ and $j \leq q$, $\Pi'(i, j) := \Pi(i, j)$.
- For all $i \in [t]$ and $j > q$, $\Pi'(i, j) := (i + j \bmod t, j)$.

This construction ensures that for all pairs of rows $i_1, i_2 \in [t]$, there are at least $t$ indices $j \in [q]$ such that $\Pi'(i_1, j)$ is in row $i_2$, while preserving the operation of $\Pi$ in the leftmost $t$-by-$q$ block of the matrix. In Construction 4.14, we would thus pad the assignment matrices $\mathbf{W}$ accordingly (with a $q$-by-$t^2$ block of 0's that are used only for the consistency checks). Padding the assignment matrices in this way increases the query dimension of the linear MIP in Construction 4.14 from $\widetilde{O}(s/t)$ to $\widetilde{O}(s/t) + t^2$, thus increasing the overall prover overhead by an *additive* factor of $\mathrm{poly}(t) = \mathrm{poly}(\lambda)$.

## B.3 Quasi-Optimal Linear MIP Analysis

In this section, we give the formal security analysis of our linear MIP from Section 4.3 (Construction 4.14). First, we state and prove the main theorem on the properties satisfied by Construction 4.14. We conclude by giving the proof of Theorem 4.15.

**Theorem B.20.** *Let $\lambda$ be a security parameter, and let $C$ be an arithmetic circuit over $\mathbb{F}$. Then, define parameters $t, \delta, k, \varepsilon, d, \kappa_1, \kappa_2, \alpha, \gamma, z$ as in Construction 4.14. Suppose $\delta > 0$ is a constant, $\varepsilon \leq 1/\mathrm{poly}(\lambda)$, and $\alpha \cdot \kappa_2 < (1-\delta)/(8\rho) \cdot t$. Then, Construction 4.14 is an input-oblivious $(k + \alpha\gamma z)$-query linear MIP for the language $\mathcal{L}_C$ of arithmetic circuit satisfiability with $t \cdot (1 + \alpha\gamma z)$ provers and soundness error $2^{-\Omega(t/\rho)} + \alpha \cdot 2^{-\Omega(\kappa_1)}$.*

*Proof.* We show completeness and soundness of the linear MIP separately.

**Completeness.** Completeness follows from completeness of the underlying linear PCP systems and completeness of the consistency check procedure (Construction 4.8).

**Soundness.** Take any $\mathbf{x} \notin \mathcal{L}_C$, and consider the probability (taken over the randomness of query generation) that there exists a proof that the verifier accepts. For $i \in [t]$, let $\mathbf{y}_i = \mathbf{Q}_i^\top \boldsymbol{\pi}_i$ denote the responses the verifier obtains from prover $P_i$ on its linear PCP query $\mathbf{Q}_i$. We appeal to the soundness of the underlying linear PCP instances to argue that with probability $1 - 2^{-\Omega(t)}$, if the verifier accepts all of the linear PCP queries responses, then the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching of size $\Omega(t)$. Recall that $\mathbf{W}$ is the matrix with rows $\mathbf{w}_1, \ldots, \mathbf{w}_t$.

**Lemma B.21.** *For any $\mathbf{x} \notin \mathcal{L}_C$, it holds that for all proofs $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_t \in \mathbb{F}^d$ to the underlying linear PCP instances $(\mathcal{P}_1, \mathcal{V}_1), \ldots, (\mathcal{P}_t, \mathcal{V}_t)$, with probability $1 - 2^{-\Omega(t)}$, one of the following conditions hold:*

- *There is an $i \in [t]$ such that $\mathcal{V}_i$ rejects $\boldsymbol{\pi}_i$.*
- *If for all $i \in [t]$, $\mathcal{V}_i$ accepts $\boldsymbol{\pi}_i = [\mathbf{w}_i, \mathbf{p}_i]$, then the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$ contains a matching of size $\left(\frac{1-\delta}{2}\right) \cdot \frac{t}{2} = \Omega(t)$.*

*Here the probability is taken over the random coins used to generate the queries for the underlying linear PCP instances $(\mathcal{P}_1, \mathcal{V}_1), \ldots, (\mathcal{P}_t, \mathcal{V}_t)$.* □

*Proof.* To show the lemma, consider the event where for all $i \in [t]$, $\mathcal{V}_i$ accepts $\boldsymbol{\pi}_i = [\mathbf{w}_i, \mathbf{p}_i]$, but there are fewer than $\left(\frac{1-\delta}{2}\right) \cdot \frac{t}{2}$ mutually disjoint pairs of indices $i, i' \in [t]$ where $\mathbf{w}_i$ and $\mathbf{w}_{i'}$ correspond to inconsistent assignments. We argue that this event occurs with probability $2^{-\Omega(t)}$ over the choice of the verifier's randomness. Let $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_\ell}$ be a subset of $(\mathbf{w}_1, \ldots, \mathbf{w}_t)$ that represents a consistent assignment to the shared inputs in $C_{i_1}, \ldots, C_{i_\ell}$. There are at most $\left(\frac{1-\delta}{2}\right) \cdot \frac{t}{2}$ disjoint pairs of witnesses that are inconsistent, so there must exist a consistent subset of witnesses of size $\ell \geq t - \left(\frac{1-\delta}{2}\right) \cdot t = \left(\frac{1+\delta}{2}\right) \cdot t$. Let $\mathbf{w} \in \mathbb{F}^m$ be an assignment that is consistent with $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_\ell}$.

Now, since $\mathbf{x} \notin \mathcal{L}_C$, by $\delta$-robustness of the decomposition $(f_1, \ldots, f_t, \mathsf{inp}, \mathsf{wit})$, at most $\delta \cdot t$ of the constraints $f_1(\mathbf{x}, \mathbf{w}), \ldots, f_t(\mathbf{x}, \mathbf{w})$ can be satisfied. This means that there are at least $\ell - \delta t \geq \left(\frac{1-\delta}{2}\right) \cdot t$ indices $j \in [\ell]$ where $f_{i_j}(\mathbf{x}, \mathbf{w}) = 0$, or equivalently, $C_{i_j}(\mathbf{x}_{i_j}, \mathbf{w}_{i_j}) = 0$. Since each of the linear PCP systems used to verify $C_i$ are systematic and have knowledge error at most $\varepsilon$, the probability that $\mathcal{V}_i$ accepts $\boldsymbol{\pi}_i$ when $C_i(\mathbf{x}_i, \mathbf{w}_i) = 0$ is at most $\varepsilon$. The linear PCP instances are independent, so

$$\Pr[\forall i \in [t] \colon \mathcal{V}_i^{\boldsymbol{\pi}_i}(\mathbf{x}_i) = 1] \leq \varepsilon^{(1-\delta)/2 \cdot t} = 2^{-\Omega(t)},$$

since there are at least $(1 - \delta)/2 \cdot t$ indices $i$ where $C(\mathbf{x}_i, \mathbf{w}_i) = 0$. Thus, with probability $1 - 2^{-\Omega(t)}$, there are at least $\left(\frac{1-\delta}{2}\right) \cdot \frac{t}{2}$ mutually disjoint pairs of indices where $\mathbf{w}_i$ and $\mathbf{w}_{i'}$ are inconsistent. Each disjoint pairs of indices contributes an edge to a matching in $\mathcal{G}_{\mathbf{W},\mathbf{A}}$, and the claim follows. □

Thus, for a false statement $\mathbf{x} \notin \mathcal{L}_C$, with probability $1 - 2^{-\Omega(t)}$, either the verifier rejects the proof or there is a matching of size $\left(\frac{1-\delta}{4}\right) \cdot t$ in the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$. We now show that if there exists such a matching, then at least one of the consistency checks fails with probability $1 - \alpha \cdot 2^{-\Omega(\kappa_1)} - 2^{-\Omega(t/\rho)}$.

**Lemma B.22.** *Suppose there exists a matching of size $\left(\frac{1-\delta}{4}\right) \cdot t$ in $\mathcal{G}_{\mathbf{W},\mathbf{A}}$. Then, at least one of the consistency checks fails with probability $1 - \alpha \cdot 2^{-\Omega(\kappa_1)} - 2^{-\Omega(t/\rho)}$.*

*Proof.* Suppose there exists a matching of size $\left(\frac{1-\delta}{4}\right) \cdot t$ in $\mathcal{G}_{\mathbf{W},\mathbf{A}}$. Since $(\Pi_1, \ldots, \Pi_z)$ is a collection of $\rho$-regularity-inducing permutations, there exists some $\beta \in [z]$ where the inconsistency graph of $\mathbf{W}_\beta = \Pi_\beta(\mathbf{W})$ with respect to $\mathbf{A}_\beta = \Pi_\beta(\mathbf{A})$ contains a *regular* matching of size $s = \left(\frac{1-\delta}{4\rho}\right) \cdot t$. Let $M$ be the regular matching of size $s$ in $\mathcal{G}_{\mathbf{W}_\beta,\mathbf{A}_\beta}$. For each edge $(i_1, i_2) \in M$, we associate with it two indices $j_1, j_2 \in [q]$ where $\mathbf{W}_\beta[i_1, j_1] \neq \mathbf{W}_\beta[i_2, j_2]$ but $\mathbf{A}_\beta[i_1, j_1] = \mathbf{A}_\beta[i_2, j_2]$. Note that $j_1, j_2$ always exists by definition of $\mathcal{G}_{\mathbf{W}_\beta,\mathbf{A}_\beta}$. Define the inconsistency matrix $\mathbf{B}_\beta \in \{0, 1\}^{t \times q}$ where for each $(i_1, i_2) \in M$, $\mathbf{B}_\beta[i_1, j_1] = 1 = \mathbf{B}_\beta[i_2, j_2]$. All other values in $\mathbf{B}_\beta$ are set to 0. Let $\mathbf{B} = \Pi_\beta^{-1}(\mathbf{B}_\beta)$. By construction, $\mathbf{B}_\beta$ has at most a single 1 in each row, and moreover, since each edge in $M$ corresponds to an edge in a matching of the inconsistency graph $\mathcal{G}_{\mathbf{W},\mathbf{A}}$, matrix $\mathbf{B}$ also has at most a single 1 in each row. Note that even through $\mathbf{W}$ and $\mathbf{W}_\beta$ may have more inconsistent assignments than those indicated in $\mathbf{B}$ and $\mathbf{B}_\beta$, it is not necessary to consider them in the analysis.

By construction, $\mathbf{B}$ and $\Pi_\beta(\mathbf{B})$ have at most one inconsistency in each row (where an inconsistency is an entry with value 1). This means that $\mathbf{B} \in \mathcal{B}_\beta$. Next, since $\Gamma_\beta^{(1)}, \ldots, \Gamma_\beta^{(\gamma)}$ is a collection of permutation sequences that is non-concentrating for $\mathcal{B}_\beta$, there exists some $j \in [\gamma]$ where $\Gamma_\beta^{(j)} = \big(\Pi_{\beta,1}^{(j)}, \ldots, \Pi_{\beta,\alpha}^{(j)}\big)$ is non-concentrating for $\mathbf{B}$. For all $\ell \in [\alpha]$, let $\mathbf{B}_\ell = \Pi_{\beta,\ell}^{(j)}(\mathbf{B}_{\ell-1})$ where $\mathbf{B}_0 = \mathbf{B}$. The non-concentration property states that no subset of $\kappa_1$ rows of $\mathbf{B}_\ell$ contains $\kappa_2$ inconsistencies.

Consider now the sequence of consistency checks the verifier performs for the permutations $\Pi_{\beta,1}^{(j)}, \ldots, \Pi_{\beta,\alpha}^{(j)}$. By construction of the consistency check queries, the verifier's behavior precisely corresponds to performing the approximate consistency check procedure in Construction 4.8 to verify the following relations:

$$\mathbf{W}_{\beta,1}^{(j)} \approx \Pi_{\beta,1}^{(j)}(\mathbf{W}) \quad \text{and} \quad \mathbf{W}_{\beta,\ell}^{(j)} \approx \Pi_{\beta,\ell}^{(j)}(\mathbf{W}_{\beta,\ell-1}^{(j)}) \text{ for all } 1 < \ell \leq \alpha.$$

Consider the first relation. By construction, the inconsistency matrix $\mathbf{B}$ encodes the positions of $s$ pairs of inconsistent assignments in $\mathbf{W}$, and the matrix $\mathbf{B}_1$ encodes the (permuted) positions of the same $s$ pairs of inconsistent assignments in $\Pi_{\beta,1}^{(j)}(\mathbf{W})$. We now argue that $\mathbf{W}_{\beta,1}^{(j)}$ contains at least $s - \kappa_2$ pairs of inconsistent assignments, except with probability $2^{-\Omega(\kappa_1)}$. This follows immediately from the assumption that $\Gamma_\beta^{(j)}$ is $(\kappa_1, \kappa_2)$-non-concentrating (Definition 4.13) and soundness of the consistency check (Lemma 4.9). In particular, by soundness of the consistency check, with probability $1 - 2^{-\Omega(\kappa_1)}$ (over the randomness of the query generation algorithm), matrices $\mathbf{W}_{\beta,1}^{(j)}$ and $\Pi_{\beta,1}^{(j)}(\mathbf{W})$ can differ on at most $\kappa_1$ rows. But since $\Gamma_\beta^{(j)}$ is $(\kappa_1, \kappa_2)$-non-concentrating, no subset of $\kappa_1$ rows of $\mathbf{B}_1$ contains $\kappa_2$ inconsistencies. We conclude that $\mathbf{W}_{\beta,1}^{(j)}$ must contain at least $s - \kappa_2$ pairs of inconsistent rows. Applying this argument $\alpha$ times, once for each permutation $\Pi_{\beta,\ell}^{(j)}$ for $\ell \in [\alpha]$, we conclude that with probability at least $1 - \alpha \cdot 2^{-\Omega(\kappa_1)}$, the number of pairs of inconsistent rows in the final matrix $\mathbf{W}_{\beta,\alpha}^{(j)}$ is at least

$$s - \alpha \cdot \kappa_2 = \left(\frac{1-\delta}{4\rho}\right) \cdot t - \alpha \cdot \kappa_2 \leq \left(\frac{1-\delta}{8\rho}\right) \cdot t.$$

Finally, $\mathbf{B}_\alpha = \Pi_\beta(\mathbf{B})$ is the inconsistency matrix derived from a *regular* matching. Thus, if the final matrix $\mathbf{W}_{\beta,\alpha}^{(j)}$ contains at least $\left(\frac{1-\delta}{8\rho}\right) \cdot t$ pairs of inconsistent rows according to the inconsistency pattern $\mathbf{B}_\alpha$, then $\Pi_{\beta'}(\mathbf{W}_{\beta,\alpha}^{(j)})$ and $\mathbf{W}_{\beta,\alpha}^{(j)}$ differs on at least $\left(\frac{1-\delta}{8\rho}\right) \cdot t$ pairs of *adjacent* rows. But then, by Corollary 4.11, the probability that the verifier accepts is at most $2^{-\Omega(t/\rho)}$. Putting everything together, the probability that the verifier accepts is bounded by $\alpha \cdot 2^{-\Omega(\kappa_1)} + 2^{-\Omega(t/\rho)}$, and the claim follows. □

Combining Lemmas B.21 and B.22, we conclude that the verifier accepts a proof of a false statement $\mathbf{x} \notin \mathcal{L}_C$ with probability at most $\alpha \cdot 2^{-\Omega(\kappa_1)} + 2^{-\Omega(t/\rho)}$. □

**Proof of Theorem 4.15.** First, we describe how we instantiate each of the primitives in Construction 4.14:

- We instantiate the $(t,\delta)$-robust decomposition using the construction from Corollary B.6, where $t=\Theta(\lambda)$, and $\delta>0$ is a constant. Let $(f_1,\ldots,f_t,\mathsf{inp},\mathsf{wit})$ be the robust decomposition of $C$, and let $C_1,\ldots,C_t$ be the arithmetic circuits that compute $f_1,\ldots,f_t$, respectively. Each of the circuits $C_i$ can be computed by an arithmetic circuit of size $\widetilde{O}(s/t)+\mathrm{poly}(t,\log s)$.

- We use the $k$-query linear PCP from Fact 4.3 to instantiate each of the linear PCP $(\mathcal{P}_i,\mathcal{V}_i)$ instances for $C_i$ for all $i\in[t]$. In this case, $\varepsilon=1/\mathrm{poly}(\lambda)$, $k=O(1)$, and the query length is $d=\widetilde{O}(s/t)+\mathrm{poly}(t,\log s)$.

- We use Construction B.8 to instantiate the regularity-inducing permutations. In this case, $\rho=O(1)$ and $z=O(1)$.

- We use Construction B.17 to instantiate the non-concentrating sequence of permutations, where we set $\kappa_1=t/\log^5 t$ and $\kappa_2=c\cdot t/\log t$, where the constant $c$ is chosen so that $\kappa_2\cdot(\log t+2)<\frac{1-\delta}{8\rho}\cdot t$. In this case, $\alpha=\log t+2=\Theta(\log t)$ and $\gamma=O(\log^3 t)$.

Note that in order to argue that the sequences of permutations output by Construction B.17 is non-concentrating (Theorem B.18), we may additionally need to pad the query (and proof) vectors with an extra $t^2=O(\lambda^2)$ components (Remark B.19). Putting everything together then, we have the following:

- The number of provers in the linear MIP system is $t\cdot(1+\alpha\gamma z)=t\cdot\mathrm{polylog}(t)=\widetilde{O}(\lambda)$.

- The query length is determined by the query length $d$ of the underlying linear PCP instances (and any extra padding from Remark B.19). Thus, the query length is $\widetilde{O}(s/\lambda)+\mathrm{poly}(\lambda,\log s)$.

- The total number of queries is $k+\alpha\gamma z$. Since $k=O(1)$, $\alpha=\Theta(\log t)$, $\gamma=O(\log^3 t)$, and $z=O(1)$, the total number of queries is $k+O(\log^4 t)=\mathrm{polylog}(\lambda)$.

- The prover's computation can be broken down as follows. First, the robust encoding $\mathbf{x}\leftarrow\mathsf{inp}(\mathbf{x}')$ and $\mathbf{w}\leftarrow\mathsf{wit}(\mathbf{x}',\mathbf{w}')$ can be computed by an arithmetic circuit of size $\widetilde{O}(s)+\mathrm{poly}(t,\log s)$. From Fact 4.3, each of the underlying linear PCP proofs can be computed by a circuit of size $\widetilde{O}(s/t)+\mathrm{poly}(t,\log s)$. Thus, all $t$ proofs for each of the underlying linear PCP instances can be constructed by a circuit of size $\widetilde{O}(s)+\mathrm{poly}(t,\log s)$. Finally, permuting the entries in an assignment matrix (of size $\widetilde{O}(s)+t^2$) can be performed also by a circuit of size $\widetilde{O}(s)+\mathrm{poly}(t)$. There are a total of $\alpha\gamma z=\mathrm{polylog}(t)$ such permutations, which adds another polylogarithmic overhead to the overall prover complexity. Summing together the different contributions and noting that $t=\Theta(\lambda)$, we conclude that the prover's algorithm can be computed by a circuit of size $\widetilde{O}(s)+\mathrm{poly}(\lambda,\log s)$.

- The query-generation procedure can be broken down as follows. From Fact 4.3, generating the queries for the underlying linear PCP instance requires a circuit of size $\widetilde{O}(s/t)$. There are $t$ instances, so generating all of the queries requires a circuit of size $\widetilde{O}(s)$. To perform the consistency checks, the query-generation algorithm additionally generates $\alpha\gamma z=\mathrm{polylog}(t)$ random matrices, each of size $\widetilde{O}(s)+\mathrm{poly}(t)$. Thus, the overall algorithm can be modeled by a circuit of size $\widetilde{O}(s)+\mathrm{poly}(\lambda,\log s)$.

- The verifier's decision algorithm consists of checking $t$ independent linear PCP instances, which can be computed by a circuit of size at most $O(tn)$ (Fact 4.3). In addition, the decision algorithm needs to perform $O(\alpha\gamma z) = \mathrm{polylog}(t)$ consistency checks (Construction 4.8), each of which requires computing $t$ linear relations. This incurs an additive cost of $\widetilde{O}(t)$. Thus, the overall cost is bounded by $\widetilde{O}(\lambda n)$.

- Finally, by Theorem B.20, for this particular choice of parameters, the overall construction achieves soundness error

$$\alpha \cdot 2^{-\Omega(\kappa_1)} + 2^{-\Omega(t/\rho)} = (\log t + 2) \cdot 2^{-\Omega(t/\log^5 t)} + 2^{-\Omega(t)} = 2^{-\Omega(\lambda/\,\mathrm{polylog}(\lambda))}.$$

We can amplify the soundness to $2^{-\lambda}$ by parallel repetition. Since we only require $\mathrm{polylog}(\lambda)$ parallel instances, this introduces an additional $\mathrm{polylog}(\lambda)$ overhead to the prover complexity and the proof size. Thus, the resulting construction remains quasi-optimal. □