Lattice-Based SNARGs and Their Application to More Efficient Obfuscation*

Dan Boneh† Yuval Ishai‡ Amit Sahai§ David J. Wu¶

Abstract

Succinct non-interactive arguments (SNARGs) enable verifying NP computations with substantially lower complexity than that required for classical NP verification. In this work, we first construct a lattice-based SNARG candidate with quasi-optimal succinctness (where the argument size is quasilinear in the security parameter). Further extension of our methods yields the first SNARG (from any assumption) that is quasi-optimal in terms of *both* prover overhead (polylogarithmic in the security parameter) as well as succinctness. Moreover, because our constructions are lattice-based, they plausibly resist quantum attacks. Central to our construction is a new notion of *linear-only vector encryption* which is a generalization of the notion of linear-only encryption introduced by Bitansky et al. (TCC 2013). We conjecture that variants of Regev encryption satisfy our new linear-only definition. Then, together with new information-theoretic approaches for building statistically-sound linear PCPs over small finite fields, we obtain the first quasi-optimal SNARGs.

We then show a surprising connection between our new lattice-based SNARGs and the concrete efficiency of program obfuscation. All existing obfuscation candidates currently rely on multilinear maps. Among the constructions that make black-box use of the multilinear map, obfuscating a circuit of even moderate depth (say, 100) requires a multilinear map with multilinearity degree in excess of 2^{100}. In this work, we show that an ideal obfuscation of both the decryption function in a fully homomorphic encryption scheme and a variant of the verification algorithm of our new lattice-based SNARG yields a general-purpose obfuscator for all circuits. Finally, we give some concrete estimates needed to obfuscate this "obfuscation-complete" primitive. We estimate that at 80-bits of security, a (black-box) multilinear map with $\approx 2^{12}$ levels of multilinearity suffices. This is over 2^{80} times more efficient than existing candidates, and thus, represents an important milestone towards implementable program obfuscation for all circuits.

1 Introduction

Interactive proofs systems [GMR85] are fundamental to modern cryptography and complexity theory. In this work, we consider computationally sound proof systems for NP languages, also known as *argument systems*. An argument system is *succinct* if its communication complexity is

*This is a preliminary full version of [BISW17].

†Stanford and Center for Encrypted Functionalities. Email: dabo@cs.stanford.edu.

‡UCLA, Technion, 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.

polylogarithmic in the running time of the NP verifier for the language. Notably, the size of the argument is polylogarithmic in the size of the NP witness.

Kilian [Kil92] gave the first succinct four-round interactive argument system for 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 NP by applying the Fiat-Shamir heuristic [FS86]. Micali's "computationally-sound proofs" (CS proofs) is the first candidate construction of a *succinct non-interactive argument* (i.e., a "SNARG" [GW11]) in the random oracle model. In the standard model, single-round argument systems are impossible for sufficiently hard languages, so we consider the weaker goal of two-message succinct argument systems where the verifier's initial message is generated independently of the statement being proven. This message is often referred to as the *common reference string* (CRS).

In this work, we are interested in minimizing the prover complexity and proof length of SNARGs. Concretely, for a security parameter λ, we measure the asymptotic cost of achieving soundness against provers of circuit size 2^λ with $\mathrm{negl}(\lambda)$ error. We say that a SNARG has *quasi-optimal succinctness* if its proof length is $\widetilde{O}(\lambda)$ and that it is *quasi-optimal* if in addition, the SNARG prover's running time is larger than that of a classical prover by only a polylogarithmic factor (in λ and the running time). In this paper, we construct the first SNARG that is quasi-optimal in this sense. The soundness of our SNARG is based on a new plausible intractability assumption, which is in the spirit of assumptions on which previous SNARGs were based (see Section 1.2). Moreover, based on a stronger variant of the assumption, we get a SNARK [BCCT12] (i.e., a SNARG of knowledge) with similar complexity (see Remark 4.9). All previous SNARGs, including heuristic ones, were suboptimal in at least one of the two measures by a factor of $\Omega(\lambda)$. For a detailed comparison with previous approaches, see Table 1.

We give two SNARG constructions: one with quasi-optimal succinctness based on standard lattices, and another that is quasi-optimal based on ideal lattices over polynomial rings. Because all of our SNARGs are lattice-based, they plausibly resist known quantum attacks. All existing SNARGs with quasi-optimal succinctness rely, at the minimum, on number-theoretic assumptions such as the hardness of discrete log. Thus, they are vulnerable to quantum attacks [Sho94, Sim97].[1]

Application to efficient obfuscation. Independently of their asymptotic efficiency, our SNARGs can also be used to significantly improve the *concrete* efficiency of program obfuscation. Program obfuscation is the task of making code unintelligible such that the obfuscated program reveals nothing more about the implementation details beyond its functionality. The theory of program obfuscation was first formalized by Barak et al. [BGI+01]. In their work, they introduced the natural notion of virtual black-box (VBB) obfuscation, and moreover, showed that VBB obfuscation for all circuits is impossible in the standard model. In the same work, Barak et al. also introduced the weaker notion of *indistinguishability obfuscation* ($i\mathcal{O}$); subsequently, Garg et al. [GGH+13b] gave the first candidate construction of $i\mathcal{O}$ for general circuits based on multilinear maps [BS03, GGH13a, CLT13, GGH15].

Since the breakthrough result of Garg et al., there has been a flurry of works showcasing the power of $i\mathcal{O}$ [GGH+13b, SW14, BZ14, GGHR14, BPW16]. However, in spite of the numerous constructions and optimizations that have been developed in the last few years [BGK+14, BR14, AGIS14, BMSZ16, Zim15, AB15], concrete instantiations of program obfuscation remain purely theoretical. Even obfuscating a relatively simple function such as the AES block cipher requires

[1] Hash-based SNARGs [Mic00, BCCT12, BCC+14] plausibly remain secure against quantum algorithms, but these constructions do not achieve quasi-optimal succinctness (Remark 4.16).

multilinear maps capable of supporting unimaginable levels of multilinearity ($\gg 2^{100}$ [Zim15]). In this work, we show that our new lattice-based SNARG constructions can be combined with existing lattice-based fully homomorphic encryption schemes (FHE) to obtain an "obfuscation-complete" primitive[2] with significantly better concrete efficiency. Targeting 80 bits of security, we show that we can instantiate our obfuscation-complete primitive over a composite-order multilinear map supporting $\approx 2^{12}$ levels of multilinearity. The number of multilinear map encodings in the description of the obfuscated program is $\approx 2^{44}$. While the levels of multilinearity required is still beyond what we can efficiently realize using existing composite-order multilinear map candidates [CLT13], future multilinear map candidates with better efficiency as well as further optimizations to the components that underlie our transformation will bring our constructions closer to reality. Concretely, our results are many orders of magnitude more efficient than existing constructions (that make black-box use of the underlying multilinear map), and thus, represent an important stepping stone towards implementable obfuscation.

Non-black-box alternatives. Nearly all obfuscation constructions [BGK+14, BR14, AGIS14, BMSZ16, Zim15, AB15] rely on the underlying multilinear map as a black-box. Recently, several works [Lin16a, LV16, Lin16b, AS17] gave the first candidate constructions of $i\mathcal{O}$ based on *constant-degree* multilinear maps (by going through the functional encryption route introduced in [AJ15, BV15]). Even more impressively, the most recent constructions by Lin [Lin16b] as well as Ananth and Sahai [AS17] only require a degree-5 multilinear map, which is certainly implementable [LMA+16]. However, this reduction in multilinearity comes at the cost of a *non-black-box* construction. Notably, their construction requires a gate-by-gate transformation to be applied to a Boolean circuit description of the encoding function of the underlying multilinear map. While further investigation of non-black-box approaches is certainly warranted, due to the complexity of existing multilinear map constructions [GGH13a, CLT13], this approach faces major hurdles with regards to implementability. In this work, we focus on constructions that use the multilinear map in a black-box manner.

1.1 Background

Constructing SNARGs. Gentry and Wichs [GW11] showed that no SNARG (for a sufficiently difficult language) can be proven secure under any "falsifiable" assumption [Nao03]. Consequently, all existing SNARG constructions for NP in the standard model (with a CRS) have relied on non-falsifiable assumptions such as knowledge-of-exponent assumptions [Dam91, BP04a, Mie08, Gro10, Lip12, GGPR13], extractable collision-resistant hashing [BCCT12, DFH12, BCC+14], homomorphic encryption with a homomorphism extraction property [BC12] and linear-only encryption [BCI+13].

Designated-verifier arguments. Typically, in a non-interactive argument system, the arguments can be verified by anyone. Such systems are said to be "publicly verifiable." In some applications (notably, bootstrapping certain types of obfuscation), it suffices to consider a relaxation where the setup algorithm for the argument system also outputs a *secret* verification state which is needed for

[2] An "obfuscation-complete" primitive is a function whose ideal obfuscation (e.g., using tamper-proof hardware) can be used for obfuscating arbitrary functions. While we do not provide a provably secure instantiation of this primitive using $i\mathcal{O}$, it can be heuristically instantiated using existing $i\mathcal{O}$ candidates. Moreover, our obfuscation-complete primitive has the appealing property that it needs to be invoked exactly *once* regardless of the function being obfuscated. This is in contrast to alternative constructions [GIS+10, App14] where the obfuscated primitive needs to be invoked for each gate in the circuit or each step of a Turing machine evaluation.

proof verification. Soundness holds provided that the prover does not know the secret verification state. These systems are said to be *designated verifier*. A key question that arises in the design and analysis of designated verifier arguments is whether the same common reference string can be reused for multiple proofs. Formally, this "multi-theorem" setting is captured by requiring soundness to hold even against a prover that makes adaptive queries to a proof verification oracle. If the prover can choose its queries in a way that induces noticeable correlations between the outputs of the verification oracle and the secret verification state, then the adversary can potentially compromise the soundness of the scheme. Thus, special care is needed to construct designated-verifier argument systems in the multi-theorem setting.

SNARGs from linear-only encryption. Bitansky et al. [BCI$^+$13] introduced a generic compiler for building SNARGs in the "preprocessing" model based on a notion called "linear-only" encryption. In the preprocessing model, the setup algorithm that constructs the CRS can run in time that depends polynomially on a time bound T of the computations that will be verified. The resulting scheme can then be used to verify computations that run in time at most T. The compiler of [BCI$^+$13] can be decomposed into an information-theoretic transformation and a cryptographic transformation, which we outline here:

- First, they restrict the interactive proof model to only consider "affine-bounded" provers. An affine-bounded prover is only able to compute affine functions (over a ring) of the verifier's queries.[3] Bitansky et al. give several constructions of succinct two-message interactive proofs in this restricted model by applying a generic transformation to existing "linear PCP" constructions.

- Next, they introduce a new cryptographic primitive called linear-only encryption, which is a (public-key) encryption scheme that *only* supports linear homomorphisms on ciphertexts. Bitansky et al. show that combining a linear-only encryption scheme with the affine-restricted interactive proofs from the previous step suffices to construct a designated-verifier SNARG in the preprocessing model. The construction is quite natural: the CRS for the SNARG system is a linear-only encryption of what would be the verifier's first message. The prover then homomorphically computes its response to the verifier's encrypted queries. The linear-only property of the encryption scheme constrains the prover to only using affine strategies. This ensures soundness for the SNARG. To check a proof, the verifier decrypts the prover's responses and applies the decision algorithm for the underlying two-message proof system. Bitansky et al. give several candidate instantiations for their linear-only encryption scheme based on Paillier encryption [Pai99] as well as bilinear maps [Jou00, BF01].

Linear PCPs. Like [BCI$^+$13], our SNARG constructions rely on linear PCPs (LPCPs).[4] A LPCP of length m over a finite field $\mathbb{F}$ is an oracle computing a linear function $\boldsymbol{\pi} : \mathbb{F}^m \to \mathbb{F}$. On any query $\mathbf{q} \in \mathbb{F}^m$, the LPCP oracle responds with $\mathbf{q}^\top \boldsymbol{\pi}$. More generally, if ℓ queries are made to the LPCP oracle, the ℓ queries can be packed into the columns of a query matrix $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$. The response of the LPCP oracle can then be written as $\mathbf{Q}^\top \boldsymbol{\pi}$. We provide more details in Section 3.

[3] Bitansky et al. [BCI$^+$13] refer to this as "linear-only," even though the prover is allowed to compute affine functions. To be consistent with their naming conventions, we will primarily write "linear-only" to refer to "affine-only."

[4] Linear PCPs were first used by Ishai, Kushilevitz, and Ostrovsky [IKO07] to construct efficient *interactive* argument systems from any homomorphic encryption scheme. Subsequently, Bitansky et al. [BCI$^+$13] gave a construction of succinct *non-interactive* arguments from linear PCPs together with a linear-only encryption scheme.

1.2 Our Results: New Constructions of Preprocessing SNARGs

In this section, we summarize our main results on constructing preprocessing SNARGs based on a more advanced form of linear-only encryption. Our results extend the framework introduced by Bitansky et al. [BCI$^+$13].

New compiler for preprocessing SNARGs. The preprocessing SNARGs we construct in this work enjoy several advantages over those of [BCI$^+$13]. We enumerate some of them below:

- **Direct construction of SNARGs from linear PCPs.** Our compiler gives a *direct* compilation from linear PCPs over a finite field $\mathbb{F}$ into a preprocessing SNARG. In contrast, the compiler in [BCI$^+$13] first constructs a two-message linear interactive proof from a linear PCP by introducing an additional linear consistency check. The additional consistency check not only increases the communication complexity of their construction, but also introduces a soundness error $O(1/|\mathbb{F}|)$. As a result, their construction only provides soundness when working over a large field (that is, when $|\mathbb{F}|$ is super-polynomial in the security parameter). By using a direct compilation of linear PCPs into SNARGs, we avoid both of these problems. Our construction does not require any additional consistency checks and moreover, it preserves the soundness of the underlying linear PCP. Thus, as long as the underlying linear PCP is statistically sound, applying our compiler yields a computationally sound argument (even if $|\mathbb{F}|$ is small).
- **Constructing linear PCPs with strong soundness.** As noted in the previous section, constructing multi-theorem designated-verifier SNARGs can be quite challenging. In [BCI$^+$13], this is handled at the information-theoretic level (by constructing interactive proof systems satisfying a notion of "strong" or "reusable" soundness) and at the cryptographic level (by introducing strengthened definitions of linear-only encryption). A key limitation in their approach is that the information-theoretic construction of two-round interactive proof systems again requires LPCPs over super-polynomial-sized fields. This is a significant barrier to applying their compiler to natural LPCP constructions over small finite fields (which are critical to our approach for bootstrapping obfuscation). In this work, we show how to apply soundness amplification to standard LPCPs with constant soundness error against linearly-bounded provers (and which do not necessarily satisfy strong soundness) to obtain strong, statistically-sound LPCPs against affine-bounded provers. Coupled with our direct compilation of LPCPs to preprocessing SNARGs, we obtain multi-theorem designated-verifier SNARGs.

We describe our construction of strong statistically sound LPCPs against affine provers from LPCPs with constant soundness error against linear provers in Section 3. Applying our transformation to linear PCPs based on the Walsh-Hadamard code [ALM$^+$92] as well as those based on quadratic-span programs (QSPs) [GGPR13], we obtain two LPCPs with strong statistical soundness against affine provers over polynomial-size fields.

From linear PCPs to preprocessing SNARGs. The primary tool we use construction of preprocessing SNARGs from linear PCPs is a new cryptographic primitive we call linear-only *vector encryption*. A vector encryption scheme is an encryption scheme where the plaintexts are vectors of ring (or field) elements. Next, we extend the notion of linear-only encryption [BCI$^+$13] to the

context of vector encryption. We say that a vector encryption scheme is linear-only if the only homomorphisms it supports is addition (and scalar multiplication) of vectors.

Our new notion of linear-only vector encryption gives an immediate method of compiling an ℓ-query linear PCP (over a finite field $\mathbb{F}$) into a designated-verifier SNARG. The construction works as follows. In a ℓ-query linear PCP over $\mathbb{F}$, the verifier's query can be written as a matrix $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$ where m is the query length of the LPCP. The LPCP oracle's response is $\mathbf{Q}^\top \boldsymbol{\pi}$ where $\boldsymbol{\pi} \in \mathbb{F}^m$ is the proof. To compile this LPCP into a preprocessing SNARG, we use a linear-only vector encryption scheme with plaintext space $\mathbb{F}^\ell$. The setup algorithm takes the verifier's query matrix $\mathbf{Q}$ (which is *independent* of the statement being proved) and encrypts each row of $\mathbf{Q}$ using the vector encryption scheme. The key observation is that the product $\mathbf{Q}^\top \boldsymbol{\pi}$ is a linear combination of the rows of $\mathbf{Q}$. Thus, the prover can homomorphically compute an encryption of $\mathbf{Q}^\top \boldsymbol{\pi}$. To check the proof, the verifier decrypts to obtain the prover's responses and then invokes the decision algorithm for the underlying LPCP. Soundness is ensured by the linear-only property of the underlying vector encryption scheme. The advantage of linear-only vector encryption (as opposed to standard linear-only encryption) is that the prover is constrained to evaluating a single linear function on *all* of the query vectors simultaneously. This insight enables us to remove the extra consistency check introduced in [BCI$^+$13], and thus, avoids the soundness penalty $O(1/|\mathbb{F}|)$ incurred by the consistency check.[5] Consequently, we can instantiate our transformation with statistically-sound linear PCPs over *any* finite field $\mathbb{F}$. We describe our construction in Section 4.

New lattice-based SNARG candidates. We then conjecture that the Regev-based [Reg05] encryption scheme of Peikert, Vaikuntanathan, and Waters [PVW08] is a secret-key linear-only vector encryption scheme over $\mathbb{Z}_p^\ell$ where p is a prime whose bit-length is polynomial in the security parameter λ. Then, applying our generic compiler from LPCPs to SNARGs (Construction 4.5) to our new LPCP constructions over polynomial-size fields $\mathbb{Z}_p$, we obtain a lattice-based construction of a designated-verifier SNARG (for Boolean circuit satisfiability) in the preprocessing model.[6] Specifically, starting with a QSP-based LPCP [GGPR13], we obtain a SNARG with quasi-optimal succinctness. In fact, our SNARG is the first lattice-based candidate with quasi-optimal succinctness and soundness error $2^{-\lambda}$ against 2^λ-size provers. When considering the weaker notion of $\mathrm{negl}(\lambda)$ soundness against 2^λ-size provers, the construction in [BCI$^+$13] instantiated with a Regev-based candidate for linear-only encryption also gives a SNARG with quasi-optimal succinctness. But under the stronger notion of achieving soundness error $2^{-\lambda}$ against 2^λ-size provers, the corresponding lattice-based instantiation of [BCI$^+$13] is suboptimal in *both* prover complexity and proof length (Remark 4.15). Thus, for Boolean circuit satisfiability, using lattice-based linear-only *vector encryption* provides some concrete advantages over vanilla linear-only encryption.

Quasi-optimal SNARGs. In Section 4.4, we further extend our techniques to obtain the first instantiation of a *quasi-optimal* SNARG for Boolean circuit satisfiability—that is, a SNARG where

[5] This is the main difference between our approach and that taken in [BCI$^+$13]. By making the *stronger* assumption of linear-only *vector encryption*, we avoid the need for an extra consistency check, thus allowing for a *direct* compilation from linear PCPs to SNARGs. In contrast, [BCI$^+$13] relies on the weaker assumption of linear-only encryption, but requires an extra step of first constructing a two-message linear interactive proof (incorporating the consistency check) from the linear PCP.

[6] While it would be preferable to obtain a construction based on the hardness of standard lattice assumptions like learning with errors (LWE) [Reg05], the separation results of Gentry and Wichs [GW11] suggest that stronger, non-falsifiable assumptions may be necessary to construct SNARGs.

the prover complexity is $\widetilde{O}(s)$ and the argument size is $\widetilde{O}(\lambda)$, where s is the size of the Boolean circuit and λ is a security parameter guaranteeing soundness against 2^λ-size provers with $\mathrm{negl}(\lambda)$ error. All previous constructions with quasi-optimal succinctness (including our lattice-based candidate described above) achieved at best prover complexity $\widetilde{O}(s\lambda)$. We refer to Table 1 for a detailed comparison. Our construction relies on a new information-theoretic construction of a linear PCP operating over rings. In conjunction with a linear-only vector encryption scheme where the underlying message space is a ring, we can apply our compiler to obtain a SNARG. To achieve quasi-optimality, we require that the ciphertext expansion factor of the underlying vector encryption scheme be polylogarithmic. Using Regev-based vector encryption based on the ring learning with errors (RLWE) problem [LPR10] and conjecturing that it satisfies our linear-only requirements, we obtain the first quasi-optimal SNARG construction. We leave open the question of realizing a stronger notion of quasi-optimality, where the soundness error (against 2^λ-size provers) is $2^{-\lambda}$ rather than $\mathrm{negl}(\lambda)$.

1.3 Our Results: Concrete Efficiency of Bootstrapping Obfuscation

In spite of the numerous optimizations and simplifications that have been proposed for indistinguishability obfuscation ($i\mathcal{O}$) and VBB obfuscation (in a generic model), obfuscating even relatively simple functions like AES remains prohibitively expensive. In this section, we describe how the combination of our new lattice-based SNARG candidate and fully homomorphic encryption (FHE) allows us to obtain VBB obfuscation for all circuits (in a generic model) with concrete parameters that are significantly closer to being implementable. Our construction is over 2^{80} times more efficient than existing constructions.

Background. The earliest candidates of $i\mathcal{O}$ and VBB obfuscation operated on matrix branching programs [GGH$^+$13b, BGK$^+$14, BR14], which together with multilinear maps [GGH13a, CLT13, GGH15], yielded obfuscation for NC^1 (via Barrington's theorem [Bar86]).[7] The primary source of inefficiency in these branching-program-based obfuscation candidates is the enormous overhead incurred when converting NC^1 circuits to an equivalent branching program representation. While subsequent work [AGIS14, BMSZ16] has provided significant asymptotic improvements for representing NC^1 circuits as matrix branching programs, the levels of multilinearity required to obfuscate a computation of depth d still grows *exponentially* in d. Thus, obfuscating even a simple function like AES, which has a circuit of relatively low depth (≈ 100), still requires a multilinear map capable of supporting $\gg 2^{100}$ levels of multilinearity and a similarly astronomical number of encodings. This is completely infeasible.

Zimmerman [Zim15] as well as Applebaum and Brakerski [AB15] showed how to directly obfuscate circuits. While their constructions do not incur the exponential overhead of converting NC^1 circuits to matrix branching programs, due to the noise growth in existing multilinear map candidates, the level of multilinearity required again grows exponentially in the depth of the circuit d. However, the number of multilinear map encodings is substantially smaller with these candidates. In the case of VBB obfuscation of AES, Zimmerman estimates that the obfuscation would contain $\approx 2^{17}$ encodings of a multilinear map capable of supporting $\gg 2^{100}$ levels of multilinearity. Despite the more modest number of encodings required, the degree of multilinearity required remains prohibitively large.

[7] Garg et al. [GGH$^+$13b] as well as Brakerski and Rothblum [BR14] show how to combine obfuscation for NC^1 together with fully homomorphic encryption (FHE) and low-depth checkable proofs to bootstrap $i\mathcal{O}$ and VBB obfuscation from NC^1 to $\mathsf{P/poly}$.

Revisiting the branching-program based obfuscation. In this work, we revisit the branching-program-based constructions of obfuscation. However, rather than follow the traditional paradigm of taking a Boolean circuit, converting it to a matrix branching program via Barrington's theorem, and then obfuscating the resulting branching program, we take the more direct approach of using the matrix branching program to compute simple functions over $\mathbb{Z}_q$ (for polynomial-sized q). The key observation is that the additive group $\mathbb{Z}_q$ embeds into the symmetric group S_q of $q \times q$ permutation matrices. This technique was previously used by Alperin-Sheriff and Peikert [AP14] for improving the efficiency of bootstrapping for FHE. While the functions that can be evaluated in this way are limited, they are expressive enough to include both the decryption function for lattice-based FHE [BV11, BGV12, Bra12, GSW13, AP14, DM15] and the verification algorithm of our new lattice-based SNARG. Using a variant of the bootstrapping theorem in [BR14], VBB obfuscation of these two functionalities suffice for VBB obfuscation of all circuits.

We remark here that Applebaum [App14] described a simpler approach for bootstrapping VBB obfuscation of all circuits based on obfuscating a pseudorandom function (PRF) in conjunction with randomized encodings. While this approach is conceptually simpler, it is unclear whether this yields a scheme with concrete efficiency. One problem is that we currently do not have any candidate PRFs that are amenable to existing obfuscation candidates. Constructing an "obfuscation-friendly" PRF remains an important open problem. Perhaps more significantly, this approach requires invoking the obfuscated program multiple times (a constant number of times per gate in the circuit, or per step of the computation in the case of Turing machines [KLW15]). In contrast, in this work, we focus on building an "obfuscation-complete" primitive such that a *single* call to the obfuscated program suffices for program evaluation.

Computing in $\mathbb{Z}_q$ via matrix branching programs. By leveraging the power of bootstrapping, it suffices to obfuscate a program that performs FHE decryption and SNARG verification. Using FHE schemes based on standard lattices [BV11, BGV12, Bra12, GSW13, AP14, DM15] and our new lattice-based SNARG, both computations effectively reduce to computing rounded inner products over $\mathbb{Z}_q$—that is, functions where we first compute the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ of two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{Z}_q^\ell$ and then reduce the result modulo a smaller value p. In our setting, one of the vectors $\mathbf{y}$ is embedded within the obfuscated program. We briefly describe the technique here. Our presentation is adapted from [AP14], who use this technique to improve the efficiency of FHE bootstrapping.

The key idea is to embed the group $\mathbb{Z}_q$ in the symmetric group S_q. The embedding is quite straightforward. A group element $y \in \mathbb{Z}_q$ is represented by the basis vector $\mathbf{e}_y \in \{0,1\}^q$ (i.e., the vector with a single 1 in the y^{th} position). Addition by an element $x \in \mathbb{Z}_q$ corresponds to multiplying by a permutation matrix that implements a cyclic rotation by x positions. Specifically, to implement the function $f_x(y) = x + y$ where $x, y \in \mathbb{Z}_q$, we define the permutation matrix $\mathbf{B}_x \in \{0,1\}^{q \times q}$ where $\mathbf{B}_x \mathbf{e}_y = \mathbf{e}_{x+y \bmod q}$ for all $y \in [q]$. Then, to compute f_x on an input y, we simply take the q-by-q permutation matrix $\mathbf{B}_x$ and multiply it with the basis vector $\mathbf{e}_y$ representing the input. Scalar multiplication can be implemented by repeated additions. Finally, modular reduction with respect to p can be implemented via multiplication by a p-by-q matrix where the i^{th} row sums the entries of the q-dimensional indicator vector corresponding to those values in $\mathbb{Z}_q$ that reduce to i modulo p. As long as q is small, this method gives an efficient way to compute simple functions over $\mathbb{Z}_q$.

Optimizing the SNARG construction. While computing a single rounded inner product suffices for FHE decryption, it is not sufficient for SNARG verification. We introduce a series of

additional optimizations to make our SNARG verification algorithm more branching-program-friendly and minimize the concrete parameters needed to obfuscate the functionality. These optimizations are described in detail in Sections 5.3 through 5.5. We highlight the most significant ones here:

- **Modulus switching.** Recall that the SNARG verifier has to first decrypt a proof (encrypted under the linear-only vector encryption scheme) before applying the underlying LPCP decision procedure. While decryption in this case does consist of evaluating a rounded inner product, the size of the underlying field scales *quadratically* in the running time of the computation being verified.[8] As a result, the width of the branching programs needed to implement the SNARG verification scales quadratically in the running time of the computation, which can quickly grow out of hand. However, since the ciphertexts in question are essentially LWE ciphertexts, we can apply the modulus switching trick that has featured in many FHE constructions [BV11, BGV12, DM15]. With modulus switching, after the prover homomorphically computes its response (a ciphertext vector over a large ring), the prover rescales each component of the ciphertext to be defined with respect to a much smaller modulus (one that grows *polylogarithmically* with the running time of the computation). The actual decryption then operates on the rescaled ciphertext, which can be implemented as a (relatively) small branching program. We describe this in Section 5.4.

- **Strengthening the linear-only assumption.** To further reduce the overhead of the SNARG verification, we also consider strengthened definitions of (secret-key) linear-only vector encryption. In particular, we conjecture that our candidate lattice-based vector encryption scheme only supports a *restricted* set of affine homomorphisms. This allows us to use LPCPs with simpler and more branching-program-friendly verification procedures. We introduce the definition and state our conjecture in Section 5.5. We note that when considering the public-key notion of linear-only encryption [BCI$^+$13], one *cannot* restrict the set of affine homomorphisms available to the adversary. By definition, the adversary can compute arbitrary linear functions on the ciphertexts, and moreover, it can also encrypt values of its choosing and linearly combine those values with the ciphertexts. This allows the adversary to realize arbitrary affine functions in the public-key setting. However, in the secret-key setting, the adversary does *not* have the flexibility of constructing arbitrary ciphertexts of its own, and so, it is plausible that the encryption scheme only permits more limited homomorphisms. Our techniques here are not specific to our particular SNARG instantiation, and thus, may be useful in optimizing other SNARG constructions (at the expense of making stronger linear-only assumptions).

- **Parallelization via CRT.** Unlike FHE decryption, the SNARG verification algorithm requires computing a matrix-vector product of the form $\mathbf{Ax}$, where the matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times \ell}$ is embedded inside the program and $\mathbf{x} \in \mathbb{Z}_q^{\ell}$ is part of the input. The verification algorithm then applies an (independent) test to each of the components of $\mathbf{Ax}$. Verification succeeds if and only if each of the underlying tests pass. While a matrix-vector product can be computed by iterating the algorithm for computing an inner product m times and performing the m

[8] This is fine from the SNARG perspective since the number of *bits* in the proof is still growing logarithmically in the running time of the computation. The quadratic overhead is due to our reliance on a linear PCP based on the Walsh-Hadamard code [ALM$^+$92]. While it is possible to reduce the prover overhead to quasilinear in the running time of the computation by using a linear PCP based on QSPs [GGPR13], the resulting SNARG construction is not branching-program-friendly. We refer to Remark 5.7 for more details.

checks sequentially, this increases the length of the branching program by a factor of m. A key observation here is that since the components of $\mathbf{Ax}$ are processed independently of one another, this computation can be performed in parallel if we consider matrix branching programs over composite-order rings. Then, each of the rows of $\mathbf{A}$ can be embedded in the different sub-rings according to the Chinese Remainder Theorem (CRT). Assuming the underlying multilinear map is composite-order, this method can potentially yield a factor m reduction in the length of the branching program. Indeed, using the CLT multilinear map [CLT13], the plaintext space naturally decomposes into sufficiently many sub-rings, thus allowing us to take advantage of parallelism with essentially no extra cost. A similar technique of leveraging CRT to parallelize computations was also used in [AP14] to improve the concrete efficiency of FHE bootstrapping.

A concrete obfuscation construction. In Section 5.6, we describe our methodology for instantiating the building blocks for our obfuscation-complete primitive (for VBB obfuscation). Our parameter estimates show that targeting $\lambda = 80$ bits of security, implementing FHE decryption together with SNARG verification can be done with a branching program (over composite-order rings[9] of length 4150 and size $\approx 2^{44}$. While publishing 2^{44} encodings of a multilinear map capable of supporting 4150 levels of multilinearity is likely beyond the scope of existing candidates, further optimizations to the underlying multilinear map as well as to the different components of our pipeline can lead to a realizable construction. Compared to previous candidates which require $\gg 2^{100}$ levels of multilinearity, our construction is over 2^{80} times more efficient.

Concurrent work. In a concurrent work [HHSS17], Halevi et al. describe a concrete implementation of a branching-program-based obfuscator using the Garg et al. [GGH15] multilinear map. Their implementation focuses on obfuscating (oblivious) read-once branching programs (equivalently, nondeterministic finite automata). They describe a series of concrete optimizations tailored for their specific implementation over the [GGH15] multilinear map. In this work, we focus on building an obfuscation candidate that suffices for general circuits. Our construction relies only on black-box use of an underlying multilinear map. Correspondingly, the simplifications and heuristics we introduce in our concrete obfuscation candidate are not specific to the internal workings of any particular multilinear map construction.

2 Preliminaries

We begin by defining the notation that we use throughout this paper. For an integer n, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. For a positive integer p, we write $\mathbb{Z}_p$ to denote the ring of integers modulo p. We typically use bold uppercase letters (e.g., $\mathbf{A}$, $\mathbf{B}$) to denote matrices and bold lowercase letters (e.g., $\mathbf{u}, \mathbf{v}$) to denote vectors. Given two vectors $\mathbf{u} \in \mathbb{Z}^m$ and $\mathbf{v} \in \mathbb{Z}^n$, we write $\mathbf{u} \otimes \mathbf{v} \in \mathbb{Z}^{mn}$ to denote the tensor product of $\mathbf{u}$ with $\mathbf{v}$, or equivalently, the vector of pairwise products $u_i v_j$ for $i \in [m]$ and $j \in [n]$ of the entries in $\mathbf{u}$ and $\mathbf{v}$, respectively.

[9] To minimize the degree of multilinearity required, we require a composite-order ring that splits into ≈ 200 sub-rings. Instantiating our construction with the composite-order CLT multilinear map [CLT13], the plaintext ring already supports the requisite number of sub-rings, so using CRT for parallelization does not incur any overhead. But this may not be the case in general. We discuss this in greater detail in Remark 5.6.

For a finite set S, we write $x \xleftarrow{R} S$ to denote that x is drawn uniformly at random from S. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote a sample from $\mathcal{D}$. Unless otherwise noted, we write λ to denote a computational security parameter and κ to denote a statistical security parameter. We say a function $f(\lambda)$ is negligible in λ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We write $f(\lambda) = \mathrm{negl}(\lambda)$ to denote that f is a negligible function in λ and $f(\lambda) = \mathrm{poly}(\lambda)$ to denote that f is a polynomial in λ. We say an algorithm is efficient if it runs in probabilistic polynomial time. For two families of distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, we write $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$ if the two distributions are computationally indistinguishable (that is, if no efficient algorithm is able to distinguish $\mathcal{D}_1$ from $\mathcal{D}_2$, except with negligible probability). We will also use the Schwartz-Zippel lemma [Sch80, Zip79]:

Lemma 2.1 (Schwartz-Zippel Lemma [Sch80, Zip79])**.** *Let p be a prime and let $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree d, not identically zero. Then,*

$$\Pr[\alpha_1, \ldots, \alpha_n \xleftarrow{R} \mathbb{Z}_p : f(\alpha_1, \ldots, \alpha_n) = 0] \leq \frac{d}{p}.$$

In Appendix A, we also review the standard definitions of succinct non-interactive arguments (SNARGs).

3 Linear PCPs

We begin by reviewing the definition of linear probabilistically checkable proofs (LPCPs). In an LPCP system for a binary relation $\mathcal{R}$ over a finite field $\mathbb{F}$, the proof consists of a vector $\boldsymbol{\pi} \in \mathbb{F}^m$ and the PCP oracle is restricted to computing a linear function on the verifier's query vector. Specifically, on input a query matrix $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$, the PCP oracle responds with $\mathbf{y} = \mathbf{Q}^\top \boldsymbol{\pi} \in \mathbb{F}^\ell$. We now give a formal definition adapted from [BCI+13].

Definition 3.1 (Linear PCPs [BCI+13])**.** Let $\mathcal{R}$ be a binary relation, $\mathbb{F}$ be a finite field, P_{LPCP} be a deterministic prover algorithm, and V_{LPCP} be a probabilistic oracle verification algorithm. Then, $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is a ℓ-query linear PCP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ε and query length m if it satisfies the following requirements:

- **Syntax:** For a vector $\boldsymbol{\pi} \in \mathbb{F}^m$, the verification algorithm $V^{\boldsymbol{\pi}}_{\mathsf{LPCP}} = (Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}})$ consists of an input-oblivious probabilistic query algorithm Q_{LPCP} and a deterministic decision algorithm D_{LPCP}. The query algorithm Q_{LPCP} generates a query matrix $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$ (independently of the statement $\mathbf{x}$) and some state information st. The decision algorithm D_{LPCP} takes the statement $\mathbf{x}$, the state st, and the response vector $\mathbf{y} = \mathbf{Q}^\top \boldsymbol{\pi} \in \mathbb{F}^\ell$ and either "accepts" or "rejects."
- **Completeness:** For every $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, the output of $P_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$ is a vector $\boldsymbol{\pi} \in \mathbb{F}^m$ such that $V^{\boldsymbol{\pi}}_{\mathsf{LPCP}}(\mathbf{x})$ accepts with probability 1.
- **Soundness:** For all $\mathbf{x}$ where $(\mathbf{x}, \mathbf{w}) \notin \mathcal{R}$ for all $\mathbf{w}$ and for all vectors $\boldsymbol{\pi}^* \in \mathbb{F}^m$, the probability that $V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(\mathbf{x})$ accepts is at most ε.

We say that $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is statistically sound if $\varepsilon(\kappa) = \mathrm{negl}(\kappa)$, where κ is a statistical security parameter.

Soundness against affine provers. In Definition 3.1, we have only required soundness to hold against provers that employ a *linear* strategy, and not an *affine* strategy. Our construction of SNARGs (Section 4), will require the stronger property that soundness holds against provers using an affine strategy—that is, a strategy which can be described by a tuple $\mathbf{\Pi} = (\boldsymbol{\pi}, \mathbf{b})$ where $\boldsymbol{\pi} \in \mathbb{F}^m$ represents a linear function and $\mathbf{b} \in \mathbb{F}^\ell$ represents an affine shift. Then, on input a query matrix $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$, the response vector is constructed by evaluating the affine relation $\mathbf{y} = \mathbf{Q}^\top \boldsymbol{\pi} + \mathbf{b}$. We now define this stronger notion of soundness against an affine prover.

Definition 3.2 (Soundness Against Affine Provers)**.** Let $\mathcal{R}$ be a relation and $\mathbb{F}$ be a finite field. A linear PCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is a ℓ-query linear PCP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ε against affine provers if it satisfies the requirements in Definition 3.1 with the following modifications:

- **Syntax:** For any affine function $\mathbf{\Pi} = (\boldsymbol{\pi}, \mathbf{b})$, the verification algorithm $V^{\mathbf{\Pi}}_{\mathsf{LPCP}}$ is still specified by a tuple $(Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}})$. Algorithms $Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}}$ are the same as in Definition 3.1, except that the response vector $\mathbf{y}$ computed by the PCP oracle is an affine function $\mathbf{y} = \mathbf{Q}^\top \boldsymbol{\pi} + \mathbf{b} \in \mathbb{F}^\ell$ of the query matrix $\mathbf{Q}$ rather than a linear function.
- **Soundness against affine provers:** For all $\mathbf{x}$ where $(\mathbf{x}, \mathbf{w}) \notin \mathcal{R}$ for all $\mathbf{w}$, and for all affine functions $\mathbf{\Pi}^* = (\boldsymbol{\pi}^*, \mathbf{b}^*)$ where $\boldsymbol{\pi}^* \in \mathbb{F}^m$ and $\mathbf{b}^* \in \mathbb{F}^\ell$, the probability that $V^{\mathbf{\Pi}^*}_{\mathsf{LPCP}}(\mathbf{x})$ accepts is at most ε.

Algebraic complexity. There are many ways one can measure the complexity of a linear PCP system such as the number of queries or the number of field elements in the verifier's queries. Another important metric also considered in [BCI+13] is the *algebraic complexity* of the verifier. In particular, the verifier's query algorithm Q_{LPCP} and decision algorithm D_{LPCP} can both be viewed as multivariate polynomials (equivalently, arithmetic circuits) over the finite field $\mathbb{F}$. We say that the query algorithm Q_{LPCP} has degree d_Q if the output of Q_{LPCP} can be computed by a collection of multivariate polynomials of maximum degree d_Q in the verifier's choice of randomness. Similarly, we say that the decision algorithm D_{LPCP} has degree d_D if the output of D_{LPCP} can be computed by a multivariate polynomial of maximum degree d_D in the prover's response and the verification state.

Strong soundness. In this work, we focus on constructing designated-verifier SNARGs. An important consideration that arises in the design of designated-verifier SNARGs is whether the same reference string σ can be reused across many proofs. This notion is formally captured by stipulating that the SNARG system remains sound even if the prover has access to a proof-verification oracle. While this property naturally follows from soundness if the SNARG system is publicly-verifiable, the same is not true in the designated-verifier setting. Specifically, in the designated-verifier setting, soundness is potentially compromised if the responses of the proof-verification oracle is correlated with the verifier's secrets. Thus, to construct a *multi-theorem* designated-verifier SNARG, we require linear PCPs with a stronger soundness property, which we state below.

Definition 3.3 (Strong Soundness [BCI+13])**.** A ℓ-query LPCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ with soundness error ε satisfies strong soundness if for every input $\mathbf{x}$ and every proof $\boldsymbol{\pi}^* \in \mathbb{F}^m$, either $V^{\boldsymbol{\pi}^*}_{\mathsf{LPCP}}(\mathbf{x})$ accepts with probability 1 or with probability at most ε.

Roughly speaking, in an LPCP that satisfies strong soundness, *every* LPCP prover either causes the LPCP verifier to accept with probability 1 or with bounded probability. This prevents correlation

attacks where a malicious prover is able to submit (potentially malformed) proofs to the verifier and seeing responses that are correlated with the verifier's secrets. We can define an analogous notion of strong soundness against affine provers.

3.1 Constructing Linear PCPs with Strong Soundness

A natural first question is whether linear PCPs with strong soundness against affine provers exist. Bitansky et al. [BCI+13] give two constructions of algebraic LPCPs for Boolean circuit satisfaction problems: one from the Hadamard-based PCP of Arora et al. [ALM+92], and another from the quadratic span programs (QSPs) of Gennaro et al. [GGPR13]. In both cases, the linear PCP is defined over a finite field $\mathbb{F}$ and the soundness error scales inversely with $|\mathbb{F}|$. Thus, the LPCP is statistically sound only if $|\mathbb{F}|$ is *superpolynomial* in the (statistical) security parameter. However, when we apply our LPCP-based SNARGs to bootstrap obfuscation, the size of the obfuscated program grows polynomially in $|\mathbb{F}|$, and so we require LPCPs with statistical soundness over small (polynomially-sized) fields.

In this section, we show that starting from any LPCP with constant soundness error against linear provers, we can generically obtain an LPCP that is statistically sound against affine provers. Our generic transformation consists of two steps. The first is a standard soundness amplification step where the verifier makes κ sets of independently generated queries (of the underlying LPCP scheme) to the PCP oracle, where κ is a statistical security parameter. The verifier accepts only if the prover's responses to all κ sets of queries are valid. Since the queries are independently generated, each of the κ sets of responses (for a false statement) is accepted with probability at most ε (where ε is proportional to $1/|\mathbb{F}|$). Thus, an honest verifier only accepts with probability at most $\varepsilon^\kappa = \mathrm{negl}(\kappa)$.

However, this basic construction does not achieve *strong* soundness against *affine* provers. For instance, a malicious LPCP prover using an affine strategy could selectively corrupt the responses to exactly one set of queries (by applying an affine shift to its response for a single set of queries). When this selective corruption is applied to a well-formed proof and the verifier's decision algorithm has low algebraic complexity, then the verifier will accept with some noticeable probability less than 1, which is sufficient to break strong soundness. To address this problem, the verifier first applies a (secret) random linear shift to its queries before submitting them to the PCP oracle. This ensures that any prover using an affine strategy with a non-zero offset will corrupt its responses to *every* set of queries, and the proof will be rejected with overwhelming probability. We now describe our generic construction in more detail.

Construction 3.4 (Statistically Sound Linear PCPs over Small Fields)**.** Fix a statistical security parameter κ. Let $\mathcal{R}$ be a binary relation, $\mathbb{F}$ be a finite field, and $\left(P^{(\mathrm{weak})}_{\mathsf{LPCP}}, V^{(\mathrm{weak})}_{\mathsf{LPCP}}\right)$ be an ℓ-query linear PCP for $\mathcal{R}$, where $V^{(\mathrm{weak})}_{\mathsf{LPCP}} = \left(Q^{(\mathrm{weak})}_{\mathsf{LPCP}}, D^{(\mathrm{weak})}_{\mathsf{LPCP}}\right)$. Define the $(\kappa\ell)$-query linear PCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ where $V_{\mathsf{LPCP}} = (Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}})$ as follows:

- **Prover's Algorithm P_{LPCP}:** On input $(\mathbf{x}, \mathbf{w})$, output $P^{(\mathrm{weak})}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$.
- **Verifier's Query Algorithm Q_{LPCP}:** The query algorithm invokes $Q^{(\mathrm{weak})}_{\mathsf{LPCP}}$ a total of κ times to obtain (independent) query matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_\kappa \in \mathbb{F}^{m \times \ell}$ and state information $\mathsf{st}_1, \ldots, \mathsf{st}_\kappa$. It constructs the concatenated matrix $\mathbf{Q} = [\mathbf{Q}_1 | \mathbf{Q}_2 | \cdots | \mathbf{Q}_\kappa] \in \mathbb{F}^{m \times \kappa\ell}$. Finally, it chooses a

random matrix $\mathbf{Y} \xleftarrow{R} \mathbb{F}^{\kappa\ell \times \kappa\ell}$ and outputs the queries $\mathbf{Q}' = \mathbf{Q}\mathbf{Y}$ and state $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_\kappa, \mathbf{Y}')$ where $\mathbf{Y}' = (\mathbf{Y}^\top)^{-1}$.

- **Verifier's Decision Algorithm D_{LPCP}:** On input the statement $\mathbf{x}$, the prover's response vector $\mathbf{a}' \in \mathbb{F}^{\kappa\ell}$ and the state $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_\kappa, \mathbf{Y}')$, the verifier's decision algorithm computes $\mathbf{a} = \mathbf{Y}'\mathbf{a}' \in \mathbb{F}^{\kappa\ell}$. Next, it writes $\mathbf{a}^\top = [\mathbf{a}_1^\top | \mathbf{a}_2^\top | \cdots | \mathbf{a}_\kappa^\top]$ where each $\mathbf{a}_i \in \mathbb{F}^\ell$ for $i \in [\kappa]$. Then, for each $i \in [\kappa]$, the verifier runs $D_{\mathsf{LPCP}}^{(\text{weak})}(\mathbf{x}, \mathbf{a}_i, \mathsf{st}_i)$ and accepts if $D_{\mathsf{LPCP}}^{(\text{weak})}$ accepts for all κ instances. It rejects otherwise.

Theorem 3.5. *Fix a statistical security parameter κ. Let $\mathcal{R}$ be a binary relation, $\mathbb{F}$ be a finite field, and $(P_{\mathsf{LPCP}}^{(\text{weak})}, V_{\mathsf{LPCP}}^{(\text{weak})})$ be a strongly-sound ℓ-query linear PCP for $\mathcal{R}$ with constant soundness error $\varepsilon \in [0, 1)$ against linear provers. If $|\mathbb{F}| > d_D$, where d_D is the degree of the verifier's decision algorithm $D_{\mathsf{LPCP}}^{(\text{weak})}$, then the linear PCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ from Construction 3.4 is a $(\kappa\ell)$-query linear PCP for $\mathcal{R}$ with strong statistical soundness against affine provers.*

Proof. Completeness follows immediately from completeness of the underlying LPCP system, so it suffices to check that the linear PCP is statistically sound against affine provers. Take any statement $\mathbf{x}$, and consider an affine prover strategy $\mathbf{\Pi}^* = (\boldsymbol{\pi}^*, \mathbf{b}^*)$, where $\boldsymbol{\pi}^* \in \mathbb{F}^m$ and $\mathbf{b}^* \in \mathbb{F}^{\kappa\ell}$. We consider two cases:

- Suppose $\mathbf{b}^* \neq 0^{\kappa\ell}$. Then, the decision algorithm D_{LPCP} starts by computing

$$\mathbf{a} = \mathbf{Y}'\mathbf{a}' = \mathbf{Y}'(\mathbf{Y}^\top\mathbf{Q}^\top\boldsymbol{\pi}^* + \mathbf{b}^*) = \mathbf{Q}^\top\boldsymbol{\pi}^* + \mathbf{Y}'\mathbf{b}^* \in \mathbb{F}^{\kappa\ell}.$$

  Next, the verifier invokes the decision algorithm $D_{\mathsf{LPCP}}^{(\text{weak})}$ for the underlying LPCP on the components of $\mathbf{a}$. By assumption, $D_{\mathsf{LPCP}}^{(\text{weak})}$ is a polynomial of maximum degree d_D in the components of the prover's response $\mathbf{a}$, and by extension, in the components of the matrix $\mathbf{Y}'$. Since $\mathbf{b}^*$ is non-zero, this is a non-zero polynomial in the $\mathbf{Y}'$. Since $\mathbf{Y}'$ is sampled uniformly at random (and independently of $\mathbf{Q}, \boldsymbol{\pi}^*, \mathbf{b}^*$), by the Schwartz-Zippel lemma, $D_{\mathsf{LPCP}}^{(\text{weak})}(\mathbf{x}, \mathbf{a}_i, \mathsf{st}_i)$ accepts with probability at most $d_D/|\mathbb{F}|$ for each $i \in [\kappa]$. Thus, the verifier rejects with probability at least $1 - (d_D/|\mathbb{F}|)^\kappa = 1 - \text{negl}(\kappa)$ since $|\mathbb{F}| > d_D$.

- Suppose $\mathbf{b}^* = 0^{\kappa\ell}$. Then, the prover's strategy is a linear function $\boldsymbol{\pi}^*$. Since the underlying PCP satisfies strong soundness against linear provers, it follows that $D_{\mathsf{LPCP}}^{(\text{weak})}(\mathbf{a}_i, \mathsf{st}_i)$ either accepts with probability 1 or with probability at most ε. In the former case, D_{LPCP} also accepts with probability 1. In the latter case, because the verifier constructs the κ queries to the underlying LPCP independently, D_{LPCP} accepts with probability at most $\varepsilon^\kappa = \text{negl}(\kappa)$. We conclude that the proof system $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ satisfies strong soundness against affine provers. □

Remark 3.6 (Efficiency of Transformation)**.** Construction 3.4 incurs a κ overhead in the number of queries made to the PCP oracle and a quadratic overhead in the algebraic complexity of the verifier's decision algorithm. Specifically, the degree of the verifier's decision algorithm in Construction 3.4 is d_D^2, where d_D is the degree of the verifier's decision algorithm in the underlying LPCP. The quadratic factor arises from undoing the linear shift in the prover's responses before applying the decision algorithm of the underlying LPCP. In many existing LPCP systems, the verifier's decision algorithm has low algebraic complexity (e.g., $d_D = 2$ for both the Hadamard-based LPCP [ALM+92] as well

as the QSP-based LPCP [GGPR13]), so the verifier's algebraic complexity only increases modestly. However, the increase in degree means that we can no longer leverage pairing-based linear-only one-way encodings [BCI+13] to construct publicly-verifiable SNARGs (since these techniques only apply when the algebraic complexity of the verifier's decision algorithm is exactly 2). No such limitations apply in the designated-verifier setting.

Remark 3.7 (Comparison with [BCI+13, Lemma C.3])**.** Bitansky et al. [BCI+13, Lemma C.3] previously showed that any algebraic LPCP over a finite field $\mathbb{F}$ with soundness error ε is also strongly sound with soundness error $\varepsilon' = \max\left\{\varepsilon, \frac{d_Q d_D}{|\mathbb{F}|}\right\}$. For sufficiently large fields $\mathbb{F}$ (e.g., when $|\mathbb{F}|$ is superpolynomial), statistical soundness implies strong statistical soundness. However, when $|\mathbb{F}|$ is polynomial, then their lemma is insufficient to argue strong statistical soundness of the underlying LPCP. In contrast, using our construction (Construction 3.4), any LPCP with just constant soundness against linear provers can be used to construct an algebraic LPCP with strong statistical soundness against affine provers (at the cost of increasing the query complexity and the verifier's algebraic complexity).

Concrete instantiations. Applying Construction 3.4 to the algebraic LPCPs for Boolean circuit satisfaction of Bitansky et al. [BCI+13], we obtain statistically sound LPCPs for Boolean circuit satisfaction over small finite fields. In the following, fix a (statistical) security parameter κ and let C be a Boolean circuit of size s.

- Starting from the Hadamard-based PCP of Arora et al. [ALM+92] over a finite field $\mathbb{F}$, there exists a 3-query LPCP with strong soundness error $2/|\mathbb{F}|$. The algebraic complexity of the decision algorithm for this PCP is $d_D = 2$. Applying Construction 3.4 and working over any finite field where $|\mathbb{F}| > 2$, we obtain a (3κ)-query LPCP with strong statistical soundness against affine provers and where queries have length $O(s^2)$.
- Starting from the quadratic span programs of Gennaro et al. [GGPR13], there exists a 3-query LPCP over any (sufficiently large) finite field $\mathbb{F}$ with strong soundness error $O(s/|\mathbb{F}|)$. The algebraic complexity of the decision algorithm for this PCP is $d_D = 2$. Applying Construction 3.4 and working over a sufficiently large finite field of size $|\mathbb{F}| = \widetilde{O}(s)$, we obtain a (3κ)-query LPCP with strong statistical soundness against affine provers where queries have length $O(s)$.

4 SNARGs from Linear-Only Vector Encryption

In this section, we introduce the notion of a linear-only vector encryption scheme. We then show how linear-only vector encryption can be directly combined with the linear PCPs from Section 3 to obtain multi-theorem designated-verifier preprocessing SNARGs in the standard model. Then, we describe a candidate instantiation of our linear-only vector encryption scheme using the LWE-based encryption scheme of Peikert, Vaikuntanathan, and Waters [PVW08]. Finally, we show how using linear-only vector encryption over polynomial rings, our techniques can be further extended to obtain the first quasi-optimal SNARG from any assumption (namely, a SNARG that is quasi-optimal in *both* the prover complexity and the proof length). Our notion of linear-only vector encryption is a direct generalization of the notion of linear-only encryption first introduced by Bitansky et al. [BCI+13].

4.1 Vector Encryption and Linear Targeted Malleability

A vector encryption scheme is an encryption scheme where the message space is a vector of ring elements. In this section, we take $\mathbb{Z}_p$ as the underlying ring and $\mathbb{Z}_p^\ell$ as the message space (for some dimension ℓ). In Section 4.4, we also consider vector encryption schemes where the ring R is a polynomial ring and the message space is R^ℓ. We introduce the basic schema below:

Definition 4.1 (Vector Encryption Scheme over $\mathbb{Z}_p^\ell$)**.** A secret-key vector encryption scheme over $\mathbb{Z}_p^\ell$ consists of a tuple of algorithms $\Pi_{\mathsf{enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^\ell) \to \mathsf{sk}$: The setup algorithm takes as input the security parameter λ and the dimension ℓ of the message space and outputs the secret key sk.
- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}) \to \mathsf{ct}$: The encryption algorithm takes as input the secret key sk and a message vector $\mathbf{v} \in \mathbb{Z}_p^\ell$ and outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to \mathbb{Z}_p^\ell \cup \{\perp\}$: The decryption algorithm takes as input the secret key sk and a ciphertext ct and either outputs a message vector $\mathbf{v} \in \mathbb{Z}_p^\ell$ 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. Next, we say that a vector encryption scheme over $\mathbb{Z}_p^\ell$ is additively homomorphic if given encryptions $\mathsf{ct}_1, \mathsf{ct}_2$ of two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}_p^\ell$, respectively, there is a public operation[10] that allows one to compute an encryption ct_{12} of the (component-wise) sum $\mathbf{v}_1 + \mathbf{v}_2 \in \mathbb{Z}_p^\ell$. Note that additively homomorphic vector encryption can be constructed directly from any additively homomorphic encryption scheme by simply encrypting each component of the vector separately. However, when leveraging vector encryption to build efficient SNARGs, we require that our encryption scheme satisfies a more restrictive homomorphism property. We define this now.

A vector encryption scheme satisfies *linear targeted malleability* [BSW12] if the only homomorphic operations the adversary can perform on ciphertexts is evaluate affine functions on the underlying plaintext vectors. We now state our definition more precisely. Note that our definition is a vector generalization of the "weaker" notion of linear-only encryption introduced by Bitansky et al. [BCI+13]. This notion already suffices for constructing a designated-verifier SNARG.

Definition 4.2 (Linear Targeted Malleability [BSW12, adapted])**.** Fix a security parameter λ. A (secret-key) vector encryption scheme $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ for a message space $\mathbb{Z}_p^\ell$ satisfies *linear targeted malleability* if for all efficient adversaries $\mathcal{A}$ and plaintext generation algorithms $\mathcal{M}$ (on input 1^ℓ, algorithm $\mathcal{M}$ outputs vectors in $\mathbb{Z}_p^\ell$), 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:

[10] In principle, homomorphic evaluation might require additional *public* parameters to be published by the setup algorithm. For simplicity of presentation, we will assume that no additional parameters are required, but all of our notions extend to the setting where the setup algorithm outputs a public evaluation key.

Real Distribution:

1. $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
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 $\boldsymbol{\pi} \in \mathbb{Z}_p^m$, $\mathbf{b} \in \mathbb{Z}_p^\ell$
3. $\mathbf{v}' \leftarrow [\mathbf{v}_1|\mathbf{v}_2|\cdots|\mathbf{v}_m] \cdot \boldsymbol{\pi} + \mathbf{b}$
4. Output $\left(\{\mathbf{v}_i\}_{i\in[m]}, s, \mathbf{v}'_i\right)$

Remark 4.3 (Multiple Ciphertexts)**.** Similar to [BSW12, BCI+13], we can also define a variant of linear targeted malleability where the adversary is allowed to output multiple ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_m$. In this case, the simulator should output an affine function $(\boldsymbol{\Pi}, \mathbf{B})$ where $\boldsymbol{\Pi} \in \mathbb{Z}_p^{m\times m}$ and $\mathbf{B} \in \mathbb{Z}_p^{\ell\times m}$ that "explains" the ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_m$. However, the simple variant we have defined above where the adversary just outputs a single ciphertext is sufficient for our construction.

Remark 4.4 (Auxiliary Input Distributions)**.** In Definition 4.2, the simulator is required to succeed for all auxiliary inputs $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$. This requirement is quite strong since z can be used to encode difficult cryptographic problems that the simulator needs to solve in order to correctly simulate the output distribution [BCPR14]. However, many of these pathological auxiliary input distributions are not problematic for Definition 4.2, since the simulator is allowed to be *computationally unbounded*. In other cases where we require the simulator to be efficient (e.g., to obtain succinct arguments of knowledge via Remark 4.9 or Definition C.2), we note that Definition 4.2 can be relaxed to only consider "benign" auxiliary input distributions for which the definition plausibly holds. For instance, for the multi-theorem SNARK construction described in Appendix C, it suffices that the auxiliary information is a uniformly random string.

Construction 4.5 (SNARG from Linear-Only Vector Encryption)**.** Fix a prime p (so the ring $\mathbb{Z}_p$ is a field), and let $\mathcal{C} = \{C_k\}_{k\in\mathbb{N}}$ be a family of arithmetic circuits over $\mathbb{Z}_p$.[11] Let $\mathcal{R}_\mathcal{C}$ be the relation associated with $\mathcal{C}$. Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be an ℓ-query input-oblivious linear PCP for $\mathcal{C}$. Let $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a secret-key vector encryption scheme for $\mathbb{Z}_p^\ell$. 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 defined as follows:

- $\mathsf{Setup}(1^\lambda, 1^k) \to (\sigma, \tau)$: On input the security parameter λ and the circuit family parameter k, the setup algorithm first invokes the query algorithm Q_{LPCP} for the LPCP to obtain a query matrix $\mathbf{Q} \in \mathbb{Z}_p^{m\times\ell}$ and some state information st. Next, it generates a secret key for the vector encryption scheme $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$. Then, it encrypts each row (an element of $\mathbb{Z}_p^\ell$) of the query matrix $\mathbf{Q}$. More specifically, for $i \in [m]$, let $\mathbf{q}_i \in \mathbb{Z}_p^\ell$ be the i^{th} row of $\mathbf{Q}$. Then, the setup algorithm computes ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{q}_i)$. Finally, the setup algorithm outputs the common reference string $\sigma = (\mathsf{ct}_1, \ldots, \mathsf{ct}_m)$ and the verification state $\tau = (\mathsf{sk}, \mathsf{st})$.
- $\mathsf{Prove}(\sigma, \mathbf{x}, \mathbf{w})$: On input a common reference string $\sigma = (\mathsf{ct}_1, \ldots, \mathsf{ct}_m)$, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prover invokes the prover algorithm P_{LPCP} for the LPCP to obtain a vector $\boldsymbol{\pi} \in \mathbb{Z}_p^m$. Viewing $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ as vector encryptions of the rows of a query matrix $\mathbf{Q} \in \mathbb{Z}_p^{m\times\ell}$, the prover uses the linear homomorphic properties of Π_{venc} to homomorphically compute an

[11] While we describe a SNARG for arithmetic circuit satisfiability (over $\mathbb{Z}_p$), the problem of Boolean circuit satisfiability easily reduces to arithmetic circuit satisfiability with only constant overhead [BCI+13, Claim A.2].

encryption of the matrix vector product $\mathbf{Q}^\top \boldsymbol{\pi}$. In particular, the prover homomorphically computes the sum $\mathsf{ct}' = \sum_{i \in [m]} \pi_i \cdot \mathsf{ct}_i$. The prover outputs the ciphertext ct' as its proof.

- $\mathsf{Verify}(\tau, \mathbf{x}, \pi)$: On input the (secret) verification state $\tau = (\mathsf{sk}, \mathsf{st})$, the statement $\mathbf{x}$, and the proof $\pi = \mathsf{ct}'$, the verifier decrypts the proof ct' using the secret key sk to obtain the prover's responses $\mathbf{a} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}')$. If $\mathbf{a} = \perp$, the verifier stops and outputs 0. Otherwise, it invokes the verification decision algorithm D_{LPCP} on the statement $\mathbf{x}$, the responses $\mathbf{a}$, and the LPCP verification state st to decide whether the proof is valid or not. The verification algorithm echoes the output of the decision algorithm.

Theorem 4.6 ([BCI+13, Lemma 6.3])**.** *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be a linear PCP that is statistically sound against affine provers, and let $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a vector encryption scheme with linear targeted malleability. Then, applying Construction 4.5 to $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ and Π_{venc} yields a (non-adaptive) designated-verifier SNARG in the preprocessing model.*

Proof. Our proof is similar to the proof of [BCI+13, Lemma 6.3]. Let P^* be a malicious prover that convinces the verifier of some false statement $\mathbf{x} \notin \mathcal{L}_{\mathcal{C}}$ with non-negligible probability $\varepsilon(\lambda)$, where $\mathcal{L}_{\mathcal{C}}$ is the language associated with $\mathcal{C}$. Since Π_{enc} satisfies linear targeted malleability (Definition 4.2), there exists a simulator $\mathcal{S}$ such that the following distributions are computationally indistinguishable:

Real Distribution:

1. $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
2. $(\mathsf{st}, \mathbf{Q}) \leftarrow Q_{\mathsf{LPCP}}$ where $\mathbf{Q} \in \mathbb{Z}_p^{m \times \ell}$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{q}_i)$ where $\mathbf{q}_i$ is the i^{th} row of $\mathbf{Q}$ for $i \in [m]$
4. $\mathsf{ct}' \leftarrow P^*(\mathsf{ct}_1, \ldots, \mathsf{ct}_q; \mathbf{x})$ such that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \neq \perp$
5. $\mathbf{a} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \in \mathbb{Z}_p^\ell$
6. Output $(\mathbf{Q}, \mathsf{st}, \mathbf{a})$

Ideal Distribution:

1. $(\mathsf{st}, \mathbf{Q}) \leftarrow Q_{\mathsf{LPCP}}$ where $\mathbf{Q} \in \mathbb{Z}_p^{m \times \ell}$
2. $(\boldsymbol{\pi}, \mathbf{b}) \leftarrow \mathcal{S}(\mathbf{x})$ where $\boldsymbol{\pi} \in \mathbb{Z}_p^m$ and $\mathbf{b} \in \mathbb{Z}_p^\ell$
3. $\hat{\mathbf{a}} \leftarrow \mathbf{Q}^\top \boldsymbol{\pi} + \mathbf{b}$
4. Output $(\mathbf{Q}, \mathsf{st}, \hat{\mathbf{a}})$

By assumption, P^* convinces an honest verifier with probability $\varepsilon = \varepsilon(\lambda)$, or equivalently, in the real distribution, $D_{\mathsf{LPCP}}(\mathbf{x}, \mathsf{st}, \mathbf{a}) = 1$ with probability at least ε. Since D_{LPCP} is efficiently computable, computational indistinguishability of the real and ideal experiments means that $D_{\mathsf{LPCP}}(\mathbf{x}, \mathsf{st}, \hat{\mathbf{a}}) = 1$ with probability at least $\varepsilon - \mathrm{negl}(\lambda)$. However, in the ideal distribution, the affine function $(\boldsymbol{\pi}, \mathbf{b})$ is generated *independently* of the verifier's queries $\mathbf{Q}$ and state st. By an averaging argument, this means that there must exist some affine function $(\boldsymbol{\pi}^*, \mathbf{b}^*)$ such that with probability at least $\varepsilon - \mathrm{negl}(\lambda)$ taken over the randomness of Q_{LPCP}, the verifier's decision algorithm D_{LPCP} on input $\mathbf{x} \notin \mathcal{L}_{\mathcal{C}}$, st, and $\mathbf{Q}^\top \boldsymbol{\pi}^* + \mathbf{b}^*$ accepts. But this contradicts statistical soundness (against affine provers) of the underlying linear PCP. □

Remark 4.7 (Adaptivity)**.** In Theorem 4.6, we showed that instantiating Construction 4.5 with a vector encryption scheme with linear targeted malleability and a linear PCP yields a non-adaptive SNARG in the preprocessing model. The same construction can be shown to satisfy *adaptive* soundness for proving efficiently decidable statements. As noted in [BCI+13, Remark 6.5], we can relax Definition 4.2 and allow the adversary to additionally output an arbitrary string in the real distribution which the simulator must produce in the ideal distribution. Invoking Construction 4.5 with an encryption scheme that satisfies this strengthened linear targeted malleability definition

yields a SNARG with adaptive soundness for the case of verifying deterministic polynomial-time computations. Note that the proof system necessary to bootstrap obfuscation is used to verify correctness of a polynomial-time computation (i.e., FHE evaluation), so adaptivity for this restricted class of statements is sufficient for our primary application.

Remark 4.8 (Multi-Theorem SNARGs)**.** Our basic notion of linear targeted malleability for vector encryption only suffices to construct a single-theorem SNARG. While the same construction can be shown secure for an adversary that is allowed to make any constant number of queries to a proof verification oracle, we are not able to prove that the construction is secure against a prover who makes polynomially many queries to the proof verification oracle. In Appendix C, we present an analog of the strengthened version of linear-only encryption from [BCI$^+$13, Appendix C] that suffices for constructing a multi-theorem SNARG. Combined with a linear PCP that is *strongly sound* against affine provers, Construction 4.5 can then be applied to obtain a multi-theorem, designated-verifier SNARG. This raises the question of whether the same construction using the weaker notion of linear targeted malleability also suffices when the underlying linear PCP satisfies strong soundness. While we do not know how to prove security from this weaker definition, we also do not know of any attacks. This is especially interesting because at the information-theoretic level, the underlying linear PCP satisfies *strong* soundness, which intuitively would suggest that the responses the malicious prover obtains from querying the proof verification oracle are *uncorrelated* with the verifier's state (strong soundness states that for any proof, either the verifier accepts with probability 1 or with negligible probability).

Remark 4.9 (Arguments of Knowledge)**.** Theorem 4.6 shows that instantiating Construction 4.5 with a linear PCP with soundness against affine provers and a vector encryption scheme with linear targeted malleability suffices for a SNARG. In fact, the same construction yields a SNARK (that is, a succinct non-interactive argument *of knowledge*) if the soundness property of the underlying LPCP is replaced with a corresponding knowledge property,[12] and the vector encryption scheme satisfies a variant of linear targeted malleability (Definition 4.2) where the simulator is required to be *efficient* (i.e., polynomially-sized). For more details, we refer to [BCI$^+$13, Lemma 6.3, Remark 6.4].

4.2 A Candidate Lattice-Based Linear-Only Vector Encryption Scheme

The core building block in our new SNARG construction is a *vector encryption* scheme for $\mathbb{Z}_p^\ell$ that plausible satisfies our notion of linear targeted malleability (Definition 4.2). In particular, we conjecture that the Regev-based encryption scheme [Reg05] due to Peikert, Vaikuntanathan, and Waters [PVW08, §7.2] satisfies our required properties. Before describing the scheme, we review some notation as well as the learning with errors (LWE) assumption which is essential (though not sufficient) for arguing security of the vector encryption scheme.

Notation. For $x \in \mathbb{Z}$ and a positive odd integer q, we write $[x]_q$ to denote the value $x \bmod q$, with values in the interval $(-q/2, q/2]$. For a lattice Λ and a positive real value $\sigma > 0$, we write $D_{\Lambda,\sigma}$ to denote the discrete Gaussian distribution over Λ with standard deviation σ. In particular, $D_{\Lambda,\sigma}$ assigns a probability proportional to $\exp(-\pi \|\mathbf{x}\|^2 / \sigma^2)$ to each element $\mathbf{x} \in \Lambda$.

[12] Roughly, the knowledge property states that there exists an extractor such that for every affine strategy $\mathbf{\Pi}^*$ that convinces the verifier of some statement $\mathbf{x}$ with high probability, the extractor outputs a witness $\mathbf{w}$ such that $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$. The Hadamard LPCP (Appendix B) also satisfies this stronger knowledge property.

Learning with errors. The learning with errors problem [Reg05] is parameterized by a dimension $n \geq 1$, an integer modulus $q \geq 2$ and an error distribution χ over the integers $\mathbb{Z}$. In this work, the noise distribution is always the discrete Gaussian distribution $\chi = D_{\mathbb{Z},\sigma}$. For $\mathbf{s} \in \mathbb{Z}_q^n$, the LWE distribution $A_{\mathbf{s},m,\chi}$ over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^n$ is specified by choosing a uniformly random matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{m \times n}$ and error $\mathbf{e} \leftarrow \chi^n$ and outputting the pair $(\mathbf{A}, \mathbf{As} + \mathbf{e}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. The learning with errors assumption $\mathsf{LWE}_{n,q,\chi}$ (parameterized by parameters n, q, χ) states that for all $m = \mathrm{poly}(n)$, the LWE distribution $A_{\mathbf{s},m,\chi}$ for a randomly sampled $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$ is computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.

The PVW encryption scheme. We now review the encryption scheme due to Peikert, Vaikuntanathan, and Waters [PVW08, §7.2]. To slightly simplify the notation, we describe the scheme where the message is embedded in the least significant bits of the plaintext. Note that when the modulus q is odd, this choice of "most significant bit" and "least significant bit" encoding makes no difference and the encodings are completely interchangeable [AP13, Appendix A]. In our setting, it suffices to just consider the secret-key setting. Let $\mathbb{Z}_p^\ell$ be the plaintext space. The vector encryption scheme $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ in [PVW08] is defined as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Choose $\bar{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $\bar{\mathbf{S}} \xleftarrow{R} \mathbb{Z}_q^{n \times \ell}$, and $\bar{\mathbf{E}} \leftarrow \chi^{\ell \times m}$, where $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$ are polynomials in the security parameter. Define the matrices $\mathbf{A} \in \mathbb{Z}_q^{(n+\ell) \times m}$ and $\mathbf{S} \in \mathbb{Z}_q^{(n+\ell) \times \ell}$ as follows:

$$\mathbf{A} = \begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{S}}^\top \bar{\mathbf{A}} + p\bar{\mathbf{E}} \end{bmatrix} \qquad \mathbf{S} = \begin{bmatrix} -\bar{\mathbf{S}} \\ \mathbf{I}_\ell \end{bmatrix},$$

  where $\mathbf{I}_\ell \in \mathbb{Z}_q^{\ell \times \ell}$ is the ℓ-by-ℓ identity matrix. Output the secret key $\mathsf{sk} = (\mathbf{A}, \mathbf{S})$.

- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$: To encrypt a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$, choose $\mathbf{r} \xleftarrow{R} \{0,1\}^m$ and output the ciphertext $\mathbf{c} \in \mathbb{Z}_q^{n+\ell}$ where

$$\mathbf{c} = \mathbf{Ar} + \begin{bmatrix} \mathbf{0}^n \\ \mathbf{v} \end{bmatrix}.$$

- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: Compute and output $[[\mathbf{S}^\top \mathbf{c}]_q]_p$.

Remark 4.10 (Low-Norm Secret Keys)**.** For some of our applications (namely, those that leverage modulus switching), it is advantageous to sample the LWE secret $\mathbf{s} \in \mathbb{Z}_q^n$ from a low-norm distribution. Previously, Applebaum et al. [ACPS09] and Brakerski et al. [BLP+13] showed that the LWE variant where the secret key $\mathbf{s} \leftarrow \chi^n$ is sampled from the error distribution is still hard under the standard LWE assumption. In the same work, Brakerski et al. also showed that LWE instances with binary secrets (i.e., $\mathbf{s} \in \{0,1\}^n$) is as hard as standard LWE (with slightly larger parameters). Sampling the secret keys from a binary distribution has been used to achieve significant concrete performance gains in several implementations of lattice-based cryptosystems [GHS12, DM15].

Correctness. Correctness of the encryption scheme follows as in [PVW08]. Since we require concrete parameter settings for our obfuscation analysis in Section 5, we state the concrete requirements needed for correctness. Our analysis uses the following tail bound on discrete Gaussian distributions due to Banaszczyk [Ban95].

Lemma 4.11 ([Ban95, Lemma 2.4])**.** *For any real $\sigma > 0$ and $T > 0$, and $\mathbf{x} \in \mathbb{R}^n$,*

$$\Pr[|\langle \mathbf{x}, D_{\mathbb{Z}^n,\sigma}\rangle| \geq T \cdot \sigma \|\mathbf{x}\|] < 2 \cdot \exp(-\pi \cdot T^2).$$

Lemma 4.12. *Fix a statistical security parameter κ and a computational security parameter λ. Fix parameters $n, m, q = \mathrm{poly}(\lambda)$. Let $\chi = D_{\mathbb{Z},\sigma}$ be the discrete Gaussian distribution with standard deviation $\sigma = \sigma(\lambda)$. Suppose that $q \geq ps\sqrt{m\kappa}$. Then, for $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and any vector $\mathbf{v} \in \mathbb{Z}_p^\ell$, letting $\mathbf{c} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$, it follows that*

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

Proof. By construction, we have that

$$\begin{aligned}[\mathbf{S}^\top \mathbf{c}]_q &= \begin{bmatrix} -\bar{\mathbf{S}}^\top & \mathbf{I}_\ell \end{bmatrix} \left(\begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{S}}^\top \bar{\mathbf{A}} + p\bar{\mathbf{E}} \end{bmatrix} \mathbf{r} + \begin{bmatrix} \mathbf{0}^n \\ \mathbf{v} \end{bmatrix} \right) \\ &= p\bar{\mathbf{E}}\mathbf{r} + \mathbf{v} \pmod q. \end{aligned} \tag{4.1}$$

Correctness follows as long as $\bar{\mathbf{E}}\mathbf{r}$ is small. In particular, it suffices to argue that with overwhelming probability, each component $(\bar{\mathbf{E}}\mathbf{r})_j$ for $j \in [\ell]$ is at most $q/2p$ in magnitude. First, since $\mathbf{r} \in \{0,1\}^m$, $\|\mathbf{r}\| \leq \sqrt{m}$. Invoking Lemma 4.11,

$$\Pr\left[\left|(\bar{\mathbf{E}}\mathbf{r})_j\right| \geq T \cdot \sigma\sqrt{m}\right] < 2 \cdot \exp(-\pi \cdot T^2).$$

Since $q \geq p\sigma\sqrt{m\kappa}$,

$$\Pr\left[\left|(\bar{\mathbf{E}}\mathbf{r})_j\right| \geq q/2p\right] \leq \Pr\left[\left|(\bar{\mathbf{E}}\mathbf{r})_j\right| \geq \sqrt{\kappa}/2 \cdot \sigma\sqrt{m}\right] < 2 \cdot \exp(-\kappa \cdot \pi/4) < 2^{-\kappa} = \mathrm{negl}(\kappa).$$

The claim then follows by a union bound over the ℓ components of $\bar{\mathbf{E}}\mathbf{r}$. □

Additive homomorphism. Like Regev encryption, the scheme is additively homomorphic and supports scalar multiplication. Since the error is additive, to compute a linear combination of ξ ciphertexts (where the coefficients for the linear combination are drawn from $\mathbb{Z}_p$), we need to scale the modulus q by a factor ξp for correctness to hold. This means we require that $q \geq \xi p^2 \sigma\sqrt{m\kappa}$. In Section 5.4, we show that this encryption scheme supports modulus switching, and thus, it is possible to work with a smaller modulus during decryption. However, this optimization is not necessary when using the vector encryption scheme to construct a SNARG (via Construction 4.5). It becomes important when we combine the SNARG with other tools to obtain more efficient bootstrapping of obfuscation for all circuits (Section 5).

Semantic security. Security of this construction follows fairly naturally from the LWE assumption. We state the main theorem here, but refer readers to [PVW08, Section 7.2.1] for the formal analysis.

Theorem 4.13 (Semantic Security [PVW08])**.** *Fix a security parameter λ and let $n, q = \mathrm{poly}(\lambda)$. Let $\chi = D_{\mathbb{Z},\sigma}$ be a discrete Gaussian distribution with standard deviation $\sigma = \sigma(\lambda)$. Then, if $m \geq 3(n+\ell)\log q$, and assuming the $\mathsf{LWE}_{n,q,\chi}$ assumption holds, then the vector encryption scheme Π_{venc} is semantically secure.*

4.3 Our Lattice-Based SNARG Candidate

We now state our concrete conjecture on the vector encryption scheme Π_{venc} from Section 4.2 that yields the first lattice-based candidate of a designated-verifier, preprocessing SNARG with quasi-optimal succinctness.

Conjecture 4.14. The PVW vector encryption scheme Π_{venc} from Section 4.2 satisfies linear targeted malleability (Definition 4.2).

Under Conjecture 4.14, we can apply Construction 4.5 in conjunction with algebraic LPCPs to obtain designated-verifier SNARGs in the preprocessing model (Theorem 4.6). To conclude, we give an asymptotic characterization of the complexity of our lattice-based SNARG system, and compare against existing SNARG candidates for Boolean circuit satisfiability. Let λ be a security parameter, and let C be a Boolean circuit of size $s = s(\lambda)$. We describe the parameters needed to achieve $2^{-\lambda}$ soundness against provers of size 2^{λ}.

- **Prover complexity.** In Construction 4.5, the prover performs m homomorphic operations on the encrypted vectors, where m is the length of the underlying linear PCP. When instantiating the vector encryption scheme Π_{venc} over the plaintext space $\mathbb{Z}_p^{\ell}$ where $p = \text{poly}(\lambda)$, the ciphertexts consist of vectors of dimension $O(\lambda + \ell)$ over a ring of size $q = \text{poly}(\lambda)$.[13] Homomorphic operations on ciphertexts corresponds to scalar multiplication (by values from $\mathbb{Z}_p$) and vector additions. Since all operations are performed over a *polynomial-sized* domain, all of the basic arithmetic operations can be performed in polylog(λ) time. Thus, as long as the underlying LPCP operates over a polynomial-sized field, the prover's overhead is $\widetilde{O}(m(\lambda + \ell))$.

  If the underlying LPCP is instantiated with the Arora et al. [ALM+92] PCP based on the Walsh-Hadamard code, then $m = O(s^2)$ and $\ell = O(\lambda)$. The overall prover complexity in this case is thus $\widetilde{O}(\lambda s^2)$. If the underlying LPCP is instead instantiated with one based on the QSPs of Gennaro et al. [GGPR13], then $m = \widetilde{O}(s)$ and $\ell = O(\lambda)$. The overall prover complexity in this case is $\widetilde{O}(\lambda s)$.

- **Proof length.** Proofs in Construction 4.5 consist of a single ciphertext of the vector encryption scheme, which has length $\widetilde{O}(\lambda + \ell)$. Thus, both of our candidate instantiations of the LPCP (based on the Hadamard code and on QSPs) yield proofs of size $\widetilde{O}(\lambda)$.

- **Verifier complexity.** In Construction 4.5, the verifier first invokes the decryption algorithm of the underlying vector encryption scheme and then applies the verification procedure for the underlying linear PCP. Decryption consists of a rounded matrix-vector product over a polynomial-sized ring, which requires $\widetilde{O}(\lambda(\lambda + \ell))$ operations. In both of our candidate LPCP constructions, the verifier's decision algorithm runs in time $O(n)$, where n is the length of the statement. Moreover, the decision algorithm for the underlying LPCP is applied $O(\lambda)$ times for soundness amplification. Thus, the overall complexity of the verifier for both of our candidate instantiations is $\widetilde{O}(\lambda^2 + \lambda n)$.

[13] More precisely, the ciphertexts are actually vectors of dimension $n + \ell$, where n is the dimension of the lattice in the LWE problem. Currently, the most effective algorithms for solving LWE rely either on BKW-style [BKW00, KF15] or BKZ-based attacks [SE94, CN11]. Based on our current understanding [LP11, CN11, KF15, BCD+16], the best-known algorithms for LWE all require time $2^{\Omega(n/\log^c n)}$ for some constant c. Thus, in terms of a concrete security parameter λ, we set the lattice dimension to be $n = \widetilde{O}(\lambda)$.

Note that we can generically reduce the verifier complexity to $\widetilde{O}(\lambda^2 + n)$ by first applying a collision-resistant hash function to the statement and having the prover argue that it knows a preimage to the hash function and that the preimage is in the language. After applying this transformation, the length of the statement is simple the output length of of a collision-resistant hash function, namely $O(\lambda)$.

Remark 4.15 (Comparison with [BCI+13])**.** An alternative route to obtaining a lattice-based SNARG is to directly instantiate [BCI+13] with Regev-based encryption. However, to achieve soundness error $2^{-\lambda}$, Bitansky et al. [BCI+13] require a LPCP (and consequently, an additively homomorphic encryption) over a field of size 2^{λ}. Instantiating the construction in [BCI+13] with Regev-based encryption over a plaintext space of size 2^{λ}, the resulting SNARGs have length $\widetilde{O}(\lambda^2)$ and the prover complexity is $\widetilde{O}(s\lambda^2)$. Another possibility is to instantiate [BCI+13] with Regev-based encryption over a polynomial-size field (thus incurring $1/\mathrm{poly}(\lambda)$-soundness error) and perform parallel repetition at the SNARG level to amplify the soundness. But this method suffers from the same drawback as above. While each individual SNARG instance (over a polynomial-size field) is quasi-optimally succinct, the size of the overall proof is still $\widetilde{O}(\lambda^2)$ and the prover's complexity remains at $\widetilde{O}(s\lambda^2)$. This is a factor λ worse than using linear-only vector encryption over a polynomial-size field.

For the weaker notion of achieving soundness error $\mathrm{negl}(\lambda)$ (rather than $2^{-\lambda}$) against provers of size 2^{λ}, it suffices to use LPCPs over a field of size $2^{\log^c \lambda}$ for some $c > 1$. In this case, the prover complexity of the [BCI+13] construction instantiated with Regev-based linear-only encryption becomes $\widetilde{O}(s\lambda)$ and the proof size is $\widetilde{O}(\lambda)$. Using linear-only vector encryption over a polynomial-size field, our SNARG construction achieves the same performance but with the stronger soundness guarantee of providing $2^{-\lambda}$ soundness error against provers of size 2^{λ}.

Remark 4.16 (Comparison with Hash-Based SNARGs)**.** An alternative approach for constructing SNARGs is to start with Kilian's succinct *interactive* argument [Kil92] and apply the Fiat-Shamir heuristic [FS86] to obtain a SNARG in the random-oracle model (i.e., a CS proof [Mic00]). With a suitable heuristic instantiation of the random oracle, this method yields a (quasi)-optimal SNARG in terms of the prover's complexity (i.e., the prover overhead is additive in $\mathrm{poly}(\lambda)$ rather than multiplicative, where λ is a concrete security parameter). However, due to their reliance on Fiat-Shamir, CS proofs inherently *cannot* satisfy (quasi)-optimal succinctness. Recall that in Kilian's protocol, the prover first uses a Merkle hash tree to commit to a (standard) PCP for the statement being proved. The verifier then challenges the prover to open bits in the committed PCP, and checks that the revealed bits satisfy the PCP verification algorithm. Using the quasilinear PCPs of [Din06, BS08], Kilian's protocol achieves constant soundness error $\varepsilon < 1$ against provers of size 2^{λ} with $\widetilde{O}(\lambda)$ communication. Parallel repetition (using a polylogarithmic number of instances) can be used to amplify the soundness to $\varepsilon^{\omega(\log \lambda)} = \mathrm{negl}(\lambda)$ while maintaining proofs of size $\widetilde{O}(\lambda)$.

However, when applying the Fiat-Shamir heuristic to Kilian's protocol to obtain a succinct *non-interactive* argument, the resulting argument does *not* provides $\mathrm{negl}(\lambda)$ soundness against provers of size 2^{λ}. In particular, since the Fiat-Shamir heuristic yields a non-interactive, *publicly-verifiable* argument system, the prover can leverage the public verifiability to sample a valid proof for each individual Kilian instance in time $1/\varepsilon$. If there are only polylogarithmic instances, then in time, $(1/\varepsilon)^{\log^c \lambda} = o(2^{\lambda})$, the prover is able to break soundness of the system (with constant probability). To defend against this rejection-sampling attack, it is necessary to use $\Omega(\lambda)$ independent instances of Kilian's protocol, resulting in proofs of size $\widetilde{O}(\lambda^2)$. Thus, CS proofs *cannot* be quasi-optimally succinct. We provide a concrete comparison in Table 1.

An alternative way of making Kilian's protocol non-interactive is to use extractable collision-resistant hash functions [BCCT12, BCC$^+$14] and a single-server private information retrieval (PIR) protocol with polylogarithmic communication complexity [CMS99]. While the [BCCT12, BCC$^+$14] construction is designated-verifier, and thus, not vulnerable to the proof rejection-sampling attack described above, their construction is not (quasi)-optimal for the prover's complexity (in contrast to CS proofs). In the Bitansky et al. construction, the prover runs several instances of a PIR protocol (playing the role of the server) over a database of size $\widetilde{\Omega}(s\lambda)$. Since the computational complexity of the server is necessarily linear in the size of the database size in any single-server PIR protocol, the prover's overhead is *at least* linear in the security parameter.

In Table 1, we compare our new lattice-based SNARG constructions to existing constructions for Boolean circuit satisfiability (the same results apply for arithmetic circuit satisfiability over polynomial-size fields). Among SNARGs with quasi-optimal succinctness (proof size $\widetilde{O}(\lambda)$), Construction 4.5 instantiated with a QSP-based LPCP achieves the same prover efficiency as the current state-of-the-art (GGPR [GGPR13] and BCIOP [BCI$^+$13]). However, in contrast to current schemes, our construction is lattice-based, and thus, plausibly resists quantum attacks. One limitation is that our new constructions are designated-verifier, while existing constructions are publicly verifiable. We stress here though that a common limitation of designated-verifier SNARGs—that the common reference string cannot be reused for multiple proofs [CL08, GLR11, BCCT12, BCC$^+$14]—does not apply to our construction. As noted by [BCI$^+$13], this limitation can be circumvented by SNARG constructions relying on algebraic PCPs such as ours. We show in Appendix C that a variant of our construction (with the same asymptotic complexity) gives a multi-theorem designated-verifier SNARG in the preprocessing model.

Remark 4.17 (Arithmetic Circuit Satisfiability over Large Fields)**.** Construction 4.5 also applies to arithmetic circuit satisfiability over large finite fields (say, $\mathbb{Z}_p$ where $p = 2^\lambda$). However, if the size of the plaintext space for the vector encryption scheme Π_{venc} from Section 4.2 is 2^λ, then the bit-length of the ciphertexts becomes $\widetilde{O}(\lambda^2)$ bits. Consequently, the proof system is no longer quasi-optimally succinct. In contrast, the QSP-based constructions [GGPR13, BCI$^+$13] remain quasi-optimally succinct for arithmetic circuit satisfiability over large fields.

4.4 Quasi-Optimal SNARGs from Ideal Lattices

Our lattice-based SNARG construction from Section 4.3 achieves quasi-optimal succinctness, but like existing constructions of SNARGs with quasi-optimal succinctness, the prover overhead is quasilinear in the security parameter. A natural question to ask is whether we can construct quasi-optimal SNARGs where the prover overhead is only polylogarithmic in the security parameter (i.e., a SNARG where the overall prover complexity is $\widetilde{O}(s)$), while maintaining quasi-optimal succinctness. In this section, we show that using Regev encryption based on the ring learning with errors (RLWE) problem [LPR10], we obtain the first quasi-optimal designated-verifier SNARG in the preprocessing model.

Linear-only vector encryption over rings. Previously, we have only considered linear-only encryption schemes where the message space is a finite field (or a vector space over a finite field). It is equally valid to also consider a notion of linear-only vector encryption where the underlying message space is a ring R that splits (via the Chinese Remainder Theorem) into multiple sub-rings

Construction	Type*	Prover Complexity	Proof Size	Assumption
CS Proofs [Mic00]	PV	$\widetilde{O}(s+\lambda^2)$	$\widetilde{O}(\lambda^2)^\dagger$	Random Oracle
Groth [Gro10]	PV	$\widetilde{O}(s^2\lambda + s\lambda^2)$	$\widetilde{O}(\lambda)$	Knowledge of Exponent
GGPR [GGPR13]	PV	$\widetilde{O}(s\lambda)$	$\widetilde{O}(\lambda)$	
BCIOP [BCI+13]‡ (Paillier)	DV	$\widetilde{O}(s\lambda^3)$	$\widetilde{O}(\lambda^3)$	Linear-Only Encryption
BCIOP [BCI+13]‡ (Pairing)	PV	$\widetilde{O}(s\lambda)$	$\widetilde{O}(\lambda)$	
BCIOP [BCI+13]‡ (Regev)§	DV	$\widetilde{O}(s\lambda)$	$\widetilde{O}(\lambda)$	
Const. 4.5¶ (Hadamard LPCP)	DV	$\widetilde{O}(s^2\lambda)$	$\widetilde{O}(\lambda)$	Linear-Only Vector Enc.
Const. 4.5¶ (QSP-based LPCP)	DV	$\widetilde{O}(s\lambda)$	$\widetilde{O}(\lambda)$	
Const. 4.5 (Section 4.4)‖	DV	$\widetilde{O}(s)$	$\widetilde{O}(\lambda)$	Linear-Only Vector Enc.

*We write "PV" to denote public verifiability and "DV" for designated verifiability.
†Even though we only require $\mathrm{negl}(\lambda)$ soundness error, the length of a CS proof still scales with λ^2 (Remark 4.16).
‡Instantiated using a LPCP based on QSPs.
§Based on a direct instantiation of [BCI+13] using Regev-based encryption. This construction achieves $\mathrm{negl}(\lambda)$ soundness error (as opposed to $2^{-\lambda}$ soundness error) against provers of size 2^λ (Remark 4.15).
¶Instantiated with the PVW [PVW08] encryption scheme from Section 4.2.
‖Instantiated with the RLWE-based vector encryption scheme from Section 4.4. This construction is the first which is quasi-optimal with respect to *both* prover complexity and proof size. This construction achieves $\mathrm{negl}(\lambda)$ soundness error (as opposed to $2^{-\lambda}$ soundness error) against provers of size 2^λ.

Table 1: Asymptotic performance of different SNARG systems for Boolean circuit satisfiability. Here, s is the size of the circuit and λ is a security parameter guaranteeing $\mathrm{negl}(\lambda)$ soundness error against provers of size 2^λ. Most of these schemes achieve $2^{-\lambda}$ soundness error with the same complexity (the exceptions being the direct instantiation of [BCI+13] with Regev-based encryption and the quasi-optimal SNARG construction from Section 4.4). All of the schemes can be converted into an *argument of knowledge* (i.e., a SNARK)—in some cases, this requires a stronger cryptographic assumption.

$R_1 \times \cdots \times R_n$. For instance, the ring might be $\mathbb{Z}_N$ where N is composite or a polynomial ring $\mathbb{Z}_p[x]/\Phi_m(x)$ for some modulus p and $\Phi_m(x)$ is the m^{th} cyclotomic polynomial. In this case, an affine combination of R-elements can be viewed as computing n *independent* affine combinations over the CRT-components of R. One of the key advantages of considering linear-only vector encryption over rings is that there exist candidate constructions (based on RLWE [GHS12]) where the encryption overhead is only *polylogarithmic* in the security parameter. This in turn yields a viable route for constructing SNARGs with polylogarithmic prover overhead.

PCP decomposition. Our construction relies on the following "robust" PCP-based decomposition for Boolean circuit satisfiability:

Lemma 4.18. *Let $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. Then, there exists a 3-CNF formula φ of length $\widetilde{O}(s)$ over a set of variables $x_1, \ldots, x_m$ where $m = \widetilde{O}(s)$ such that the following holds:*

- ***Completeness:*** *For any* $\mathbf{x} \in \{0,1\}^n$ *such that there exists* $\mathbf{w} \in \{0,1\}^h$ *where* $C(\mathbf{x}, \mathbf{w}) = 1$*, there exists a satisfying assignment* $(x_1^*, \ldots, x_m^*)$ *to* φ*. Moreover, given* $\mathbf{x}$ *and* $\mathbf{w}$*, the satisfying assignment* $(x_1^*, \ldots, x_m^*)$ *can be computed in time* $\widetilde{O}(s)$*.*
- ***Robustness:*** *For any* $\mathbf{x} \in \{0,1\}^n$ *where for all* $\mathbf{w} \in \{0,1\}^h$*,* $C(\mathbf{x}, \mathbf{w}) = 0$*, the formula* φ *is unsatisfiable. Moreover, there exists a constant* $\varepsilon \in (0,1]$ *such that for any such* $\mathbf{x}$*, all assignments* $(x_1^*, \ldots, x_m^*)$ *to* φ *can satisfy at most a* $(1-\varepsilon)$*-fraction of the clauses in* φ*.*

Proof. From [Din06, BS08], we have that any language decidable in non-deterministic time t has a PCP verifier (with perfect completeness and constant soundness error) that makes a constant number of queries to the underlying PCP and which tosses $\log(t \cdot \operatorname{polylog} t)$ random coins. This means that the length of the PCP can be taken to be $\widetilde{O}(t)$, or in other words, *quasilinear* in the time bound t. We now describe how to build the formula φ. The formula is defined over $\widetilde{O}(t)$ variables, one for each bit of the PCP. Next, each of the $\widetilde{O}(t)$ possible settings of the verifier's randomness yields a constant set of positions in the PCP that the verifier inspects. Each of these tests can be expressed as a constant-size 3-CNF in the bit positions of the PCP. Thus, we obtain a formula φ of size $\widetilde{O}(t)$. The first condition (that φ is satisfiable for statements contained in the language) follows from perfect completeness of the PCP while the second condition (that at most an $(1-\varepsilon)$-fraction of the clauses of φ are satisfiable for statements not contained in the language) follows from soundness of the PCP. The lemma follows immediately if we consider the language corresponding to Boolean circuit satisfiability. □

Construction overview. We give a high-level description of our quasi-optimal SNARG construction for Boolean circuit satisfiability. Let κ be a statistical security parameter and let $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. We first describe how to construct a linear PCP (over a ring R that splits into at least κ copies of some field $\mathbb{Z}_p$—this is possible, for instance, when R is a polynomial ring $R = \mathbb{Z}_p[x]/\Phi_m(x)$ and $p = 1 \pmod m$). We can then invoke our compiler (Construction 4.5) together with an appropriate linear-only vector encryption scheme over R to obtain a SNARG.

To construct our LPCP, we begin by applying Lemma 4.18 to the circuit C to obtain a 3-CNF formula φ of size $\widetilde{O}(s)$ over $\widetilde{O}(s)$ variables. Next, we arbitrarily split φ into κ different 3-CNF formulas $\varphi_1, \ldots, \varphi_\kappa$, each of size $\widetilde{O}(s/\kappa)$. The first observation is that we can check satisfiability of each of these 3-CNF formulas using an LPCP of size $\widetilde{O}(s/\kappa)$, for instance, by using the 3-query QSP-based LPCP from [BCI$^+$13]. Note that if $\mathbb{Z}_p$ is small, then this LPCP has *constant* soundness error. We embed the i^{th} LPCP instance (for checking φ_i) in the i^{th} copy of $\mathbb{Z}_p$ of R. To argue soundness, we use the robustness guarantee from Lemma 4.18, which states that for a false statement, a constant ε-fraction of the clauses are unsatisfiable. Equivalently, this means that an ε-fraction of the formulas $\varphi_1, \ldots, \varphi_\kappa$ are unsatisfiable. Since each of the underlying LPCP instances has a constant soundness error of at most $\rho < 1$, the verifier accepts a false proof with probability at most $\rho^{-\varepsilon\kappa} = \operatorname{negl}(\kappa)$. However, this analysis only works if we assume the prover makes a *consistent* assignment to the variables appearing in the formulas $\varphi_1, \ldots, \varphi_\kappa$. We now describe how to perform consistency checks so as to bind the prover to making a consistent assignment.

Enforcing consistency. As noted above, for soundness to hold, we need to enforce consistency in the prover's assignments to variables across the 3-CNF formulas $\varphi_1, \ldots, \varphi_\kappa$. Abstractly speaking, we can view a proof $\boldsymbol{\pi} \in R^m$ as a κ-by-m matrix where the i^{th} row contains the proof that formula

φ_i is satisfied. We can define a variable's replication pattern to denote the positions in $\boldsymbol{\pi}$ where the same variable appears. The goal is to check whether $\boldsymbol{\pi}$ conforms to a specific replication pattern of variables. This can be done using a technique from [Gro09] and which was also used in [IPS09] for performing a similar kind of consistency check. The high level idea is as follows. To test whether a set of M values $(v_1, \ldots, v_M)$ satisfy a particular replication pattern, we choose M random values $r_1, \ldots, r_M \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and compare the value $\sum_{i \in [M]} v_i r_i \in \mathbb{Z}_p$ to the value $\sum_{i \in [M]} v_i r'_i \in \mathbb{Z}_p$ where $r'_1, \ldots, r'_M$ are obtained by permuting the values $r_1, \ldots, r_M$ along the "cycles" (the set of positions corresponding to the same variable) induced by the replication pattern. The key is that each check simply corresponds to evaluating a linear function on the proof entries, and thus, can be performed by making additional queries to the LPCP oracle. By the Schwartz-Zippel lemma, an inconsistent assignment will be discovered with at least constant probability (over the randomness of the r_i's). By repeating this consistency check $\omega(\log \kappa)$ times, the verifier will catch an inconsistent assignment with overwhelming probability (though not necessarily probability $2^{-\kappa}$). We present our LPCP construction below:

Construction 4.19 (Linear PCP with Polylogarithmic Query Complexity)**.** Fix a statistical security parameter κ. Let $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s, R be a polynomial ring that splits into (at least) κ isomorphic copies of a finite field $\mathbb{Z}_p$ where $p = \widetilde{O}(s)$. Let φ be the 3-CNF formula obtained when applying Lemma 4.18 to the circuit C, and let $\varphi_1, \ldots, \varphi_\kappa$ be an arbitrary partitioning of the clauses of φ into smaller 3-CNF formulas, each with $\widetilde{O}(s/\kappa)$ clauses (over the same underlying set of variables). Take any Q where $Q = \log^c \kappa$ for some $c > 1$. Define the $\widetilde{O}(1)$-query linear PCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ over R as follows:

- **Prover's algorithm** P_{LPCP}: On input the statement $\mathbf{x} \in \{0,1\}^n$ and witness $\mathbf{w} \in \{0,1\}^h$, the prover computes a satisfying assignment to φ. For each $i \in [\kappa]$, the prover constructs a proof $\boldsymbol{\pi}_i \in \mathbb{F}_p^m$ that φ_i is satisfiable using the QSP-based LPCP of [BCI+13]. Here, $m = \widetilde{O}(s/\kappa)$ since each φ_i is a formula of size $\widetilde{O}(s/\kappa)$. It outputs the proof $\boldsymbol{\pi} \in R^m$ where the CRT decomposition of $\boldsymbol{\pi}$ is given by the vector $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\kappa)$.

- **Verifier's query algorithm** Q_{LPCP}: For each $i \in [\kappa]$, the verifier invokes the query algorithm for the 3-query QSP-based LPCP (for proving the statement defined by φ_i) to obtain queries $\mathbf{q}_1^{(i)}, \mathbf{q}_2^{(i)}, \mathbf{q}_3^{(i)} \in \mathbb{F}_p^m$. It sets $\mathbf{q}_1 \in R^m$ to be the vector whose CRT decomposition is given by $(\mathbf{q}_1^{(1)}, \ldots, \mathbf{q}_1^{(\kappa)})$. It constructs $\mathbf{q}_2$ and $\mathbf{q}_3$ in a similar manner. Then, for each $j \in [Q]$, take $\mathbf{s}_1^{(j)} \xleftarrow{\mathrm{R}} R^m$, and let $\mathbf{s}_2^{(j)}$ be $\mathbf{s}_1^{(j)}$ permuted according to the replication pattern of the variables in $\varphi_1, \ldots, \varphi_\kappa$. The verifier outputs the queries $\left(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \left\{\mathbf{s}_1^{(j)}, \mathbf{s}_2^{(j)}\right\}_{j \in [Q]}\right)$. The verifier's secret verification state consists of the verification state for each of the underlying QSP-based LPCP queries.

- **Verifier's decision algorithm** D_{LPCP}: Let $a_1, a_2, a_3, \left\{b_1^{(j)}, b_2^{(j)}\right\}_{j \in [Q]} \in R$ be the prover's responses. The verifier first decomposes a_1, a_2, a_3 into their CRT components and verifies the κ instances of the underlying QSP-based LPCP. If any of the κ proofs fail to verify, the verifier rejects. Next, for each $j \in [Q]$, the verifier takes $b_1^{(j)}$ and $b_2^{(j)}$, decomposes them into their CRT components and computes the sum of the CRT components (over $\mathbb{Z}_p$). The verifier rejects if the sums do not match for any $j \in [Q]$. If all checks pass, the verifier accepts.

By inspection, the verifier in Construction 4.19 makes a total of $2Q + 3$ queries to the LPCP oracle. Since $Q = \log^c \kappa = \widetilde{O}(1)$, Construction 4.19 is a $\widetilde{O}(1)$-query LPCP. Before showing that

Construction 4.19 has negligible soundness error, we briefly characterize the prover's complexity. First, by Lemma 4.18, the prover can compute the satisfying assignment to φ in $\widetilde{O}(s)$ time. Constructing a LPCP proof for φ_i requires $\widetilde{O}(s/\kappa)$ time since φ_i is a formula of size $\widetilde{O}(s/\kappa)$. Thus, constructing $\boldsymbol{\pi}$ can be done in time $\widetilde{O}(s)$. Next, we formally show that Construction 4.19 is a statistically-sound LPCP.

Theorem 4.20. *Construction 4.19 is a statistically-sound $\widetilde{O}(1)$-query linear PCP.*

Proof. Completeness is immediate. For soundness, take any $\mathbf{x} \in \{0,1\}^n$ such that $C(\mathbf{x}, \mathbf{w}) = 0$ for all $\mathbf{w} \in \{0,1\}^h$. Let $\boldsymbol{\pi}^* \in R$ be a proof. We consider two cases:

- **The proof $\boldsymbol{\pi}^*$ contains a consistent assignment to all variables.** By the robustness property of φ (Lemma 4.18), no consistent assignment for the false statement $\mathbf{x}$ can satisfy more than a constant $(1-\varepsilon)$-fraction of the formulas $\varphi_1, \ldots, \varphi_\kappa$. For each unsatisfiable φ_i, the verifier will reject with at least constant probability ρ. Since the κ instances of the LPCP are independent, we conclude by soundness of the QSP-based LPCP that the verifier accepts with probability $(1-\rho)^{-\kappa\varepsilon} = \mathrm{negl}(\kappa)$.
- **The proof $\boldsymbol{\pi}^*$ contains an inconsistent assignment to the variables.** In this case, the difference in the sums over the CRT components of $b_1^{(j)}$ and $b_2^{(j)}$ can be viewed as a linear polynomial over the CRT components of $\mathbf{s}_1^{(j)}$. Since $\boldsymbol{\pi}$ contains an inconsistent assignment, by construction of $\mathbf{s}_1^{(j)}$ and $\mathbf{s}_2^{(j)}$, this polynomial is not identically zero. Since $\mathbf{s}_1^{(j)}$ is sampled uniformly at random, by the Schwartz-Zippel lemma, the verifier rejects with probability $1/p$. Since the verifier performs $Q = \log^c \kappa$ independent checks, the verifier accepts with probability $p^{-\log^c \kappa} = \mathrm{negl}(\kappa)$.

Soundness against affine prover strategies follows in a similar manner as long as the underlying linear PCP provides (constant) soundness against affine provers. □

Vector encryption over polynomial rings. To obtain a SNARG based on the linear PCP from Construction 4.19, it suffices to construct a candidate vector encryption scheme over a ring R that satisfies linear targeted malleability. Invoking Theorem 4.6 then yields the desired result. One such candidate is the natural generalization of the cryptosystem in Section 4.2 to the ring LWE setting. Ring LWE analogs of Regev encryption have been used previously for optimizing FHE [BGV12, GHS12, GHS12]. We describe the construction for encrypting vectors below. Fix a cyclotomic polynomial ring $R = \mathbb{Z}[x]/\Phi_m(x)$ and let χ be an error distribution. The plaintext ring will be $R_p = \mathbb{Z}_p[x]/\Phi_m(x)$ and the ciphertext ring will be $R_q = \mathbb{Z}_q[x]/\Phi_m(x)$ where m, q are polynomials in the security parameter.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Choose $\bar{\mathbf{a}} \xleftarrow{R} R_q$, $\mathbf{s} \xleftarrow{R} R_q^\ell$, and $\bar{\mathbf{e}} \leftarrow \chi^\ell$. Define $\mathbf{a} \in R_q^{(1+\ell)}$ as follows:

$$\mathbf{a} = \begin{bmatrix} \bar{\mathbf{a}} \\ \bar{\mathbf{a}}\mathbf{s} + p\bar{\mathbf{e}} \end{bmatrix},$$

  Output the secret key $\mathsf{sk} = (\mathbf{a}, \mathbf{s})$.

- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$: To encrypt a vector $\mathbf{v} \in R_p^\ell$, choose $\mathbf{r} \xleftarrow{R} R_2$ and output the ciphertext $\mathbf{c} \in R_q^{(1+\ell)}$ where

$$\mathbf{c} = \mathbf{a}\mathbf{r} + \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix}.$$

- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: Let c_1 denote the first component of $\mathbf{c}$ and let $\bar{\mathbf{c}} \in R_q^\ell$ be the last ℓ components of $\mathbf{c}$. Compute and output $[[\bar{\mathbf{c}} - c_1\mathbf{s}]_q]_p$.

If we conjecture that the above vector encryption scheme satisfies linear targeted malleability, then combining Construction 4.5 with the linear PCP from Construction 4.19, we obtain a quasi-optimal SNARG (with respect to both the prover complexity as well as succinctness). This is because a single ciphertext under this vector encryption scheme is $\widetilde{O}(\lambda)$ bits long (since $\ell = \widetilde{O}(1)$), and queries in the LPCP of Construction 4.19 have length $\widetilde{O}(s/\lambda)$. Thus, the total number of operations the prover has to perform is $\widetilde{O}(s)$. There are a couple caveats that we note below.

Remark 4.21 (Quasi-Optimality and Soundness Gap)**.** The SNARG obtained by applying Construction 4.5 to the LPCP from Construction 4.19 achieves $2^{-\log^c \lambda}$ (for $c > 1$) soundness error against 2^λ-bounded provers. Thus, it is the first quasi-optimal SNARG in terms of both prover complexity as well as succinctness. An interesting open problem is to construct a quasi-optimal SNARG that achieves the stronger notion of $2^{-\lambda}$ soundness error against provers of size 2^λ.

Remark 4.22 (Strong Soundness and Multi-Theorem SNARGs)**.** One limitation of the LPCP from Construction 4.19 is that it does not satisfy strong soundness (while retaining $\widetilde{O}(1)$ query complexity). Thus, we are not able to leverage Theorem C.3 (in conjunction with a suitable vector encryption scheme) to argue that the resulting SNARG construction is a multi-theorem SNARGs. We leave the construction of a quasi-optimal multi-theorem SNARG as another open problem.

5 Concrete Efficiency of Bootstrapping VBB Obfuscation

In this section, we describe how to leverage our new lattice-based SNARG candidates to improve the concrete efficiency of bootstrapping obfuscation. In Sections 5.1 and 5.2, we review some standard definitions as well as the VBB bootstrapping theorem of [BR14]. Next, in Section 5.3, we describe how matrix branching programs can be used to perform simple computations over $\mathbb{Z}_q$. In particular, we show how we can implement FHE decryption and SNARG verification as a matrix branching program. Then, in Sections 5.4 and 5.5, we introduce a series of algorithmic as well as heuristic optimizations to improve the concrete efficiency of the candidate obfuscator. We conclude by giving an estimate of the parameters needed to instantiate our obfuscation candidate in Section 5.6.

To summarize, after applying our optimizations, implementing FHE decryption together with SNARG verification can be done with a branching program (over composite-order rings) of length 4150 and size $\approx 2^{44}$ (at a security level of $\lambda = 80$). While publishing 2^{44} encodings of a multilinear map capable of supporting 4150 levels of multilinearity is likely beyond the scope of existing candidates, further optimizations to the underlying multilinear map as well as to the different components of our pipeline can plausibly lead to a realizable construction. Thus, our construction represents an important milestone towards the ultimate goal of implementable program obfuscation.

5.1 Preliminaries

We begin by reviewing some standard definitions for obfuscation and matrix branching programs.

Definition 5.1 (Virtual Black-Box Obfuscation in an $\mathcal{M}$-Idealized Model [BGI$^+$01, BGK$^+$14])**.** Let $\mathcal{M}$ be a stateful (possibly randomized) oracle and $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of Boolean circuits. We say that an efficient machine $\mathcal{O}$ is a virtual black-box (VBB) obfuscator for $\mathcal{C}$ in the $\mathcal{M}$-idealized model if the following conditions are satisfied:

- **Correctness:** For all $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$, every input x to C, and all possible random coins for $\mathcal{M}$, it follows that
$$\Pr[(\mathcal{O}^{\mathcal{M}}(1^\lambda, C))(x) \neq C(x)] = \mathrm{negl}(\lambda),$$
where the probability is taken over the random coins of $\mathcal{O}$.
- **Virtual Black-Box:** For every efficient adversary $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for all efficient distinguishers D, all $\lambda \in \mathbb{N}$ and all $C \in \mathcal{C}_\lambda$, it follows that
$$\left|\Pr[D(\mathcal{A}^{\mathcal{M}}(\mathcal{O}^{\mathcal{M}}(1^\lambda, C))) = 1] - \Pr[D(\mathcal{S}^{\mathcal{C}}(1^\lambda, 1^{|C|})) = 1]\right| = \mathrm{negl}(\lambda),$$
where the probability is taken over the random coins of D, $\mathcal{A}$, $\mathcal{S}$, $\mathcal{O}$, and $\mathcal{M}$.

Since the obfuscator $\mathcal{O}$ is required to be efficient, its output has size at most $\mathrm{poly}(\lambda)$. Thus, the additional *polynomial slowdown* requirement in [BGI$^+$01, BGK$^+$14] is implicitly captured by this definition.

Branching programs. In this work, we show how an VBB obfuscator for branching programs (equivalently, the class NC^1) can be efficiently bootstrapped to an VBB obfuscator for $\mathsf{P/poly}$. We first review the definition of a branching program. While previous works [BGK$^+$14, AGIS14, BMSZ16] focused exclusively on branching programs over inputs drawn from a binary field $\{0,1\}^\ell$, in our applications, it is more conducive to consider branching programs where the inputs are vectors in a larger finite field $\mathbb{Z}_q^\ell$ (see Remark 5.4). In our setting, we use the notion of a generalized matrix branching program where the matrices can be non-square and singular [BMSZ16].

Definition 5.2 (Generalized Matrix Branching Program [BMSZ16, adapted])**.** Let R be a finite ring. A generalized matrix branching program (MBP) over R of length n and shape $(d_0, \ldots, d_n)$ for inputs over $\mathbb{Z}_q^\ell$ is given by a sequence
$$\mathsf{BP} = \left(\mathsf{inp}, \{\mathbf{B}_{i,j}\}_{i\in[n], j\in[q]}\right),$$
where for all $i \in [n]$ and $j \in [q]$, $\mathbf{B}_{i,j} \in R^{d_{i-1}\times d_i}$ and $\mathsf{inp} : [n] \to [\ell]$ is the input bit position examined in step i of the computation. When all of the matrices $\mathbf{B}_{i,j}$ are square matrices with dimension d, we refer to d as the width of the branching program. The output of the branching program BP on an input $\mathbf{x} \in \mathbb{Z}_q^\ell$ is an element of $\{0,1\}^{d_0\times d_n}$ where for $i \in [d_0]$ and $j \in [d_1]$, we have
$$[\mathsf{BP}(\mathbf{x})]_{i,j} = 0 \text{ if and only if } \mathbf{e}_i^\top \left(\prod_{i\in[n]} \mathbf{B}_{i,x_{\mathsf{inp}(i)}}\right) \mathbf{e}_j \neq 0,$$
where $\mathbf{e}_i, \mathbf{e}_j$ denote the standard basis vectors of the appropriate dimension. We say that the branching program is oblivious if $\mathsf{inp} : [n] \to [\ell]$ is a fixed function that is independent of the function being evaluated.

Remark 5.3 (Dual-Input Matrix Branching Programs)**.** Most existing branching-program based constructions of VBB obfuscation [GGH+13b, BGK+14, AGIS14, BMSZ16] operate on *dual-input matrix branching programs*. In a dual-input matrix branching program, each step of the computation depends on two fixed bits of the input, and correspondingly, there are two input bit selector functions $\mathsf{inp}_0, \mathsf{inp}_1 : [n] \to [\ell]$. Likewise, the description of a dual-input branching program consists of a sequence

$$\mathsf{BP} = \left(\mathsf{inp}_0, \mathsf{inp}_1, \{\mathbf{B}_{i,j_0,j_1}\}_{i\in[n],j_0,j_1\in[q]}\right).$$

As noted in previous works, any single-input MBP can be converted to a dual-input MBP by simply setting $\mathsf{inp}_0 = \mathsf{inp}_1 = \mathsf{inp}$, $\mathbf{B}_{i,j_0,j_0} = \mathbf{B}_{i,j_0}$, and $\mathbf{B}_{i,j_0,j_1}$ arbitrarily for all $j_1 \neq j_0$. However, this incurs a quadratic overhead (in q) on the size of the branching program. Since the extra matrices $\mathbf{B}_{i,j_0,j_1}$ for $j_1 \neq j_0$ are *never* used in the normal functioning of the scheme (and only show up in the context of the security proof), we can consider the analogous scheme where these additional matrices are not available (this effectively brings us back to the single-input setting). Although dropping the extraneous matrices means that we are no longer able to formally prove VBB security, Lewi et al. [LMA+16] show that any attack on the simplified obfuscator (where the obfuscated program does not include the unused matrices) translates to an attack on the original VBB obfuscator. Thus, when estimating the concrete size of an obfuscated program, it suffices to just consider the case where the extra matrices are not included.

Remark 5.4 (Binary Inputs vs. $\mathbb{Z}_q$ Inputs)**.** The matrix branching programs in Definition 5.2 are defined for inputs drawn from $\mathbb{Z}_q^\ell$ rather than the more traditional choice of $\{0,1\}^\ell$. All of the functionalities we consider in this section can be modified to work when the inputs are represented as binary strings of length $\ell \cdot \lceil \log q \rceil$. This increases the length of the branching program by a factor $\log q$, but reduces the number of matrices in each step of the computation by a factor q. Asymptotically, for large values of q, this allows for substantially smaller branching programs. However, in existing multilinear map candidates [GGH13a, CLT13], the size of each encoding scales quadratically in the multilinearity. Thus, if the length of the branching program is increased by a factor $\log q$, then the size of the obfuscated program increases by a factor $\log^3 q$. For concrete values of q, it is more efficient to obfuscate programs over $\mathbb{Z}_q$ rather than $\mathbb{F}_2$. This optimization of using a larger base field for the branching program representation to achieve better concrete performance was also leveraged in [LMA+16].

5.2 Bootstrapping for VBB Obfuscation via FHE and SNARGs

In this section, we review how FHE and NC^1-checkable proofs can be combined with VBB obfuscation for matrix branching programs (of polynomial length and width) to obtain VBB obfuscation for general circuits. Several variants of bootstrapping theorems have been proposed in the context of both indistinguishability obfuscation [GGH+13b] as well as VBB obfuscation [BR14, App14]. For completeness, we present our specific construction which is adapted from that of Brakerski and Rothblum [BR14]. Traditionally, bootstrapping theorems have relied on statistically sound proofs checkable in NC^1. One of the observations we make in this work is that for VBB obfuscation, it suffices to relax the statistical soundness requirement and use computationally sound arguments, which in particular, enables the use of *succinct* arguments. However, substituting SNARGs for the statistically sound proofs does not seem to suffice when considering bootstrapping of indistinguishability obfuscation.

Fully homomorphic encryption. The first primitive we require for bootstrapping VBB obfuscation is fully homomorphic encryption [Gen09]. We briefly review the syntax here. Let $\mathcal{C}$ be a class of polynomially-sized circuits. A public-key fully homomorphic encryption scheme $\Pi_{\mathsf{FHE}} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Encrypt}, \mathsf{FHE.Decrypt}, \mathsf{FHE.Eval})$ over a binary message space $\{0,1\}$ is a tuple of algorithms with the following properties:

- $\mathsf{FHE.KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ, the key generation algorithm outputs a public key pk and a secret key sk.
- $\mathsf{FHE.Encrypt}(\mathsf{pk}, m) \to \mathsf{ct}$: On input the public key pk and a message $m \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{FHE.Decrypt}(\mathsf{sk}, \mathsf{ct}) \to m$: On input the secret key sk and a ciphertext ct, the decryption algorithm outputs a message $m \in \{0,1\}$.
- $\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \to \mathsf{ct}'$: On input the public key pk, a circuit $C \in \mathcal{C}$ on $\ell = \mathrm{poly}(\lambda)$ inputs, and a collection of ℓ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the evaluation algorithm is a deterministic algorithm that outputs a ciphertext ct'.

The correctness requirement for an FHE scheme is that for any circuit $C \in \mathcal{C}$ and any sequence of inputs $m_1, \ldots, m_\ell \in \{0,1\}$, if $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and $\mathsf{ct}_i \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, m_i)$ for all $i \in [\ell]$, the following holds:

$$\Pr[\mathsf{FHE.Decrypt}(\mathsf{sk}, \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)) \neq C(m_1, \ldots, m_\ell)] = \mathrm{negl}(\lambda).$$

Security is the usual notion of semantic security.

We now review how FHE encryption and SNARGs can be combined to bootstrap VBB obfuscation from NC^1 to $\mathsf{P/poly}$

Construction 5.5 (VBB Bootstrapping [BR14, adapted])**.** Fix a security parameter λ. Let $\mathcal{C} = \{C_\ell : \{0,1\}^n \to \{0,1\}\}_{\ell \in \mathbb{N}}$ be a collection of Boolean circuits of polynomial size $s = s(\ell)$ on $n = n(\ell)$ bit inputs. Let U_ℓ be the universal circuit for evaluating Boolean circuits of size $s(\ell)$ on $n(\ell)$-bit inputs, and let $\Pi_{\mathsf{FHE}} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Encrypt}, \mathsf{FHE.Decrypt}, \mathsf{FHE.Eval})$ be an FHE scheme where the decryption algorithm $\mathsf{FHE.Decrypt}$ can be implemented by a matrix branching program. Let $\mathcal{C}' = \{C'_\ell\}_{\ell \in \mathbb{N}}$ be a family of Boolean circuits for the following NP language (parameterized by ℓ):

$$\mathcal{L} = \left\{ (\mathbf{x}, \mathsf{ct}) : \mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{pk}, U_\ell(\cdot, \mathbf{x}), \hat{C}_\ell) \right\},$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^\ell)$ and $\hat{C}_\ell \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, C_\ell)$. Next, let $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a SNARG system for the family of circuits $\mathcal{C}'$. Let $\mathcal{O}_{\mathsf{MBP}}$ be a VBB obfuscator for MBPs of polynomial length and width. The obfuscator $\mathcal{O}$ for $\mathcal{C}$ works as follows:

- **Obfuscation.** On input the security parameter λ and the circuit parameter ℓ, the obfuscator $\mathcal{O}$ does the following:
  1. Run the setup algorithm $(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ for the SNARG system to obtain a CRS σ and a verification state τ.
  2. Generate a public/private key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ for the FHE scheme.

3. Encrypt the circuit C_ℓ to obtain an encrypted circuit $\hat{C}_\ell \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, C_\ell)$.
4. Construct an obfuscation $\hat{P}_\ell$ of the following program using $\mathcal{O}_{\mathsf{MBP}}$:

   > **Constants:** The program has the FHE secret key sk and the SNARG verification state τ hard-coded inside it.
   >
   > On input a tuple $(\mathbf{x}, \mathsf{ct}, \pi)$, the program does the following:
   >
   > (a) Run the SNARG verifier $\mathsf{Verify}(\tau, (\mathbf{x}, \mathsf{ct}), \pi)$. If the SNARG fails to verify, output 0.
   >
   > (b) If the SNARG verification succeeds, output $\mathsf{FHE.Decrypt}(\mathsf{sk}, \mathsf{ct})$.

5. Output $(\hat{P}_\ell, \sigma, \mathsf{pk}, \hat{C}_\ell)$

- **Evaluation.** Evaluation of an obfuscated program $(\hat{P}_\ell, \sigma, \mathsf{pk}, \hat{C}_\ell)$ on an input $\mathbf{x} \in \{0,1\}^n$ works as follows:

  1. Homomorphically evaluate the function $\mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{pk}, U_\ell(\cdot, \mathbf{x}), \hat{C}_\ell)$.
  2. Let s be the statement that $\mathsf{ct} = \mathsf{FHE.Eval}(\mathsf{pk}, U_\ell(\cdot, \mathbf{x}), \hat{C}_\ell)$. Let $\mathbf{w}$ be a witness that $(\mathbf{x}, \mathsf{ct}) \in \mathcal{L}$ (i.e., the steps of the computation). Construct a proof $\pi \leftarrow \mathsf{Prove}(\sigma, (\mathbf{x}, \mathsf{ct}), w)$.
  3. Run the obfuscated decryption function $\hat{P}_\ell$ on input $(\mathbf{x}, \mathsf{ct}, \pi)$ and output the result.

Correctness. Correctness of the scheme follows from correctness of the underlying building blocks (the VBB obfuscation for MBPs and the FHE scheme) as well as completeness of the SNARG. Moreover, assuming that the FHE decryption circuit and the SNARG verification circuit can be expressed as a matrix branching program (of polynomial length and width), then the decryption program $\hat{P}_\ell$ that is obfuscated can also be expressed as a matrix branching program (of polynomial length and width). Correctness follows immediately.

Security. The proof of security proceeds analogously to the proof given in [BR14, Lemma 4.3].

5.3 Simple Functions over $\mathbb{Z}_q$ via Branching Programs

In Section 5.2, we demonstrated that FHE and SNARGs suffice to bootstrap a VBB obfuscator for matrix branching programs into one that works for general circuits, provided that both FHE decryption and SNARG verification can themselves be expressed as a simple matrix branching program. In many existing FHE candidates [BV11, BGV12, Bra12, GSW13, AP14, DM15], decryption corresponds to evaluating a rounded inner product over a polynomial-size ring. Similarly, using the SNARG formed by applying Construction 4.5 to our candidate linear-only vector encryption scheme based on Regev encryption (Section 4.2) along with the Hadamard PCP, we obtain a SNARG where the verification algorithm almost reduces to evaluating a rounded matrix-vector product over $\mathbb{Z}_q$ (followed by a simple quadratic test). Thus, the basic building block we require is evaluation of rounded inner products over a polynomial-size ring $\mathbb{Z}_q$.

In this section, we show that computing rounded inner products over finite fields of polynomial size can be efficiently realized using (generalized) matrix branching programs. In light of this, we refer to schemes where the basic operations correspond to computing rounded inner products

as "branching-program-friendly." This paper gives the first construction of a SNARG with a branching-program-friendly verification procedure. Combined with existing FHE schemes with branching-program-friendly decryption and VBB obfuscation for branching programs, we obtain practical VBB obfuscation for general circuits.

Simple functions over $\mathbb{Z}_q$ via branching programs. Many existing obfuscation candidates are tailored for obfuscating branching programs. In these constructions, obfuscating a branching program of length ℓ and width w typically requires a multilinear map that supports approximately ℓ levels of multilinearity and the resulting obfuscated program contains approximately w^2 encodings. Thus, existing constructions are best-suited for computations that are captured by *simple* branching programs. In the more traditional approach of obfuscating Boolean circuits by first converting the circuit into a branching program (via Barrington's theorem [Bar86]), the length of the resulting branching program grows *exponentially* in the depth of the circuit. Thus, the parameters needed to obfuscate even a simple computation of moderate depth quickly becomes astronomical.

Instead of obfuscating branching programs obtained via Barrington's theorem, in this work, we take the more direct approach of using the matrix branching program to compute simple functions over $\mathbb{Z}_q$ (for polynomial-sized q). The key observation we use is that the additive group $\mathbb{Z}_q$ embeds into the symmetric group S_q of $q \times q$ permutation matrices. This idea was previously used by Alperin-Sheriff and Peikert [AP14] for improving the efficiency of bootstrapping for FHE. While the functions that can be evaluated in this way are limited, they are expressive enough to include both the decryption function for lattice-based FHE [BV11, BGV12, Bra12, GSW13, AP14, DM15] and a (simplified) variant of the verification algorithm of our new lattice-based SNARG.

Computing inner products over $\mathbb{Z}_q$. As described in Section 1.3, embedding $\mathbb{Z}_q$ into S_q is quite straightforward. An element $x \in \mathbb{Z}_q$ is represented by the x^{th} basis vector in $\{0,1\}^q$. The group operation in $\mathbb{Z}_q$ (i.e., addition of group elements) corresponds to performing cyclic rotations (i.e., permutations) of the indicator vector. For example, the univariate function $f_x(y) = x + y \in \mathbb{Z}_q$, can be represented by a q-by-q permutation matrix $\mathbf{B} \in \{0,1\}^{q \times q}$ that performs a cyclic rotation by x positions. In other words, given the embedding of $y \in \mathbb{Z}_q$ (i.e., the y^{th} basis vector $\mathbf{e}_y \in \{0,1\}^q$), the product $\mathbf{B}\mathbf{e}_y$ yields an embedding of $x + y \in \mathbb{Z}_q$ (i.e., the $(x+y)^{\text{th}}$ basis vector $\mathbf{e}_{x+y}$).

It is easy to extend this method to compute inner products. Consider the function $f_{\mathbf{x}}(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}_q$ where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^\ell$. The inner product can be decomposed into ℓ steps, where each step $i \in [\ell]$ computes the product of the input value x_i with the fixed value y_i and adds it to an "accumulator" that stores the value from first $i-1$ operations (the accumulator is initialized to 0). Each of these intermediate steps can be encoded by a permutation matrix in S_q. Specifically, for all $i \in [\ell]$ and $j \in [q]$, let $\mathbf{B}_{i,j} \in \{0,1\}^{q \times q}$ be the permutation matrix that performs a cyclic rotation by $x_i \cdot j \pmod q$ positions. In other words, for all $t \in [q]$, we have that $\mathbf{B}_{i,j}\mathbf{e}_t = \mathbf{e}_{t + x_i \cdot j \pmod q}$. In particular this means that $\left(\prod_{i \in [\ell]} \mathbf{B}_{i,y_i}\right)\mathbf{e}_q = \mathbf{e}_{[\langle \mathbf{x}, \mathbf{y} \rangle]_q}$.

The description of the (generalized) matrix branching program that computes the inner product functionality $f_{\mathbf{x}}$ then consists of the matrices $\{\mathbf{B}_{i,j}\}_{i \in [\ell], j \in [q]}$. The input encoding function inp is simply the identity function over $[\ell]$ (irrespective of $\mathbf{x}$). Thus, computing an inner product over $\mathbb{Z}_q^\ell$ can be done using an oblivious generalized branching program of length ℓ and width q.

Computing rounded inner products over $\mathbb{Z}_q$. Modular reduction are handled in a straightforward manner. Consider the function $f'_{\mathbf{x}}(\mathbf{y}) = [[\langle \mathbf{x}, \mathbf{y} \rangle]_q]_p$ where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^\ell$. Previously, we described

a sequence of matrices $\mathbf{B}_{i,j}$ such that $\left(\prod_{i\in[n]} \mathbf{B}_{i,y_i}\right)\mathbf{e}_q = \mathbf{e}_{[\langle \mathbf{x},\mathbf{y}\rangle]_q}$. Additionally, we now define the matrix $\mathbf{R}_p \in \{0,1\}^{p\times q}$ where $\mathbf{R}_p[i,j] = 1$ if and only if $i = j \pmod p$. By construction, for any $t \in [q]$, $\mathbf{R}_p\mathbf{e}_t = \mathbf{e}_{[t]_p} \in \{0,1\}^p$. Thus, left-multiplying an embedded-$\mathbb{Z}_q$ element by $\mathbf{R}_p$ corresponds to rounding modulo p. Thus, it is straightforward to evaluate a rounded inner product over $\mathbb{Z}_q^\ell$ as a matrix branching program. Moreover, since $\mathbf{R}_p$ is *input-independent*, it can be pre-multiplied into the other input-dependent matrices, so the length of the resulting branching program (i.e., the number of matrices that need to be multiplied) is unchanged.

Parallelizing computation via CRT. Thus far, we have only considered the case of evaluating functions involving a single inner product. While computing a single rounded inner product suffices for decryption in many FHE schemes, this is not the case for the verification algorithm in our candidate SNARG construction from linear-only vector encryption. The verification algorithm from Construction 4.5 (when instantiated with the PVW encryption scheme described in Section 4.2) requires first evaluating a matrix-vector product followed by a simple decision procedure (based on the verifier's decision algorithm for the underlying linear PCP). The core operation is to compute functions of the form $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ where $\mathbf{A} \in \mathbb{Z}_q^{m\times\ell}$. Evaluating $f_{\mathbf{A}}$ on an input $\mathbf{x}$ corresponds to evaluating m independent inner products between the rows of $\mathbf{A}$ and the input vector $\mathbf{x}$. The direct way of doing this is to evaluate the m inner products sequentially. However, this will increase the length of the branching program by a factor of m. When obfuscating branching programs using multilinear maps, the degree of multilinearity required is linear in the length of the branching program. Thus, sequential evaluation of the inner products will increase the multilinearity by a multiplicative factor, which significantly reduces the concrete efficiency of the scheme.

We now describe an optimization that enables computation of the m inner products without needing to increase the degree of multilinearity. The key observation that underlies our optimization is the fact that evaluating $f_{\mathbf{A}}(\mathbf{x})$ on some input $\mathbf{x}$ corresponds to taking the inner product of the rows of $\mathbf{A}$ with the *same* vector $\mathbf{x}$. We take advantage of this by considering branching programs over a composite-order ring $\mathbb{Z}_N$ rather than a finite field. Moreover, we choose the order N of the ring such that it splits into (at least) m independent sub-rings $\mathbb{Z}_{q_1}, \ldots, \mathbb{Z}_{q_m}$ via the Chinese Remainder Theorem (CRT): $\mathbb{Z}_N \cong \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_m}$. Then, to evaluate m inner products with an input vector $\mathbf{x} \in \mathbb{Z}_q^\ell$ in parallel, we embed the branching program matrices for each of the m inner products in the m sub-rings of $\mathbb{Z}_N$.

More concretely, for $k \in [m]$, let $\mathbf{A}_k$ denote the k^{th} row of $\mathbf{A}$. Let $\left\{\mathbf{B}_{i,j}^{(k)}\right\}_{i\in[\ell],j\in[q]}$ be a collection of matrices where each $\mathbf{B}_{i,j}^{(k)} \in \{0,1\}^{q\times q}$ such that $\left(\prod_{i\in[\ell]} \mathbf{B}_{i,x_i}^{(k)}\right)\mathbf{e}_q = \mathbf{e}_{[\mathbf{A}_k\mathbf{x}]_q}$. Now, define matrices $\{\mathbf{B}_{i,j}\}_{i\in[\ell],j\in[q]}$ where $\mathbf{B}_{i,j} \in \mathbb{Z}_N^{q\times q}$, and for all $i \in [\ell]$, $j \in [q]$, and $k \in [m]$,

$$\mathbf{B}_{i,j} = \mathbf{B}_{i,j}^{(k)} \pmod{q_k}.$$

By construction, for all $k \in [m]$

$$\left(\prod_{i\in[\ell]} \mathbf{B}_{i,x_i}\right)\mathbf{e}_q = \left(\prod_{i\in[\ell]} \mathbf{B}_{i,x_i}^{(k)}\right)\mathbf{e}_q = \mathbf{e}_{[\mathbf{A}_k\mathbf{x}]_q} \pmod{q_k}.$$

Thus, by considering matrix branching programs over composite-order rings, it is possible to evaluate matrix-vector products without needing to increase the length of the branching program. However,

note that the different slots in the CRT representation are independent and non-interacting. Thus, after computing a matrix-vector product in parallel, we are not able to evaluate additional functions that depend on more than one component in the resulting vector. Nonetheless, we are able to leverage this optimization to significantly reduce the length of the necessary branching program needed for SNARG verification.

An important question to ask is whether considering matrix branching programs over large composite-order rings introduces additional overhead. We discuss this in the following remark.

Remark 5.6 (Viability of CRT Embedding)**.** It is possible that obfuscating matrix branching programs over a ring that splits into m sub-rings incurs an overhead comparable to the cost of obfuscating a matrix branching program over a smaller prime-order field that performs m sequential evaluations. If this is the case, there is no benefit to this optimization. However, in existing multilinear map candidates over composite-order rings [CLT13], security of the scheme already necessitates working over a composite-order plaintext ring where the order is a product of $\omega(\lambda)$ primes, where λ is a security parameter. Thus, when the number of parallel computations is bounded by $O(\lambda)$, the CLT multilinear map candidate naturally supports the parallelism (with no extra cost). We note that this optimization is not possible in multilinear map candidates that only support encodings over prime-order rings, such as [GGH13a].

5.4 Modulus Switching for Faster SNARG Verification

In this section as well as the next section, we describe optimizations and heuristics we apply to make the SNARG verification procedure from Section 4 more branching-program friendly. We begin by describing the concrete SNARG instantiation we use for bootstrapping obfuscation.

Instantiating the linear PCP. First, we instantiate Construction 3.4 with the linear PCP based on the Walsh-Hadamard code [ALM+92] to obtain a (3κ)-query linear PCP with strong soundness against affine provers. Here, κ is a statistical security parameter. A limitation of the Hadamard LPCP is that the prover overhead is quadratic in the size of the circuit. While alternative LPCP constructions exist where the prover overhead is only quasilinear, they seem less suitable for bootstrapping obfuscation. We discuss one alternative and its limitation in the following remark.

Remark 5.7 (Other LPCP Candidates)**.** We can also instantiate Construction 3.4 with an LPCP based on QSPs [GGPR13]. While the prover overhead in the QSP-based construction is only quasilinear in the size of the circuit, it is necessary to work over a field $\mathbb{F}$ of size $\widetilde{O}(s)$ (where s is the size of the Boolean circuit). Using our construction, working over a field of this size would increase the size of the branching program needed for the verifier's decision algorithm by a multiplicative factor $\widetilde{O}(s^2)$. This quadratic factor in the field size arises from the fact that to implement the verifier's decision procedure in the QSP-based LPCP construction (as well as the Hadarmard LPCP construction), the branching program needs to compute a quadratic relation over more than one proof element. In other words, the branching program needs to "remember" one of the decrypted proof components to run the verification algorithm. A simple (but inefficient) way of achieving this is to increase the number of states in the branching program (by a factor equal to the field size). But this incurs a $\widetilde{O}(s^2)$ cost in the size of the branching program needed for SNARG verification. Thus, under our design, using the Hadamard PCP results in greater prover inefficiency, but allows a much more compact (and thus, more likely to be implementable) branching program representation. An interesting problem is to design new LPCPs with quasilinear (or even quasi-optimal) prover

overhead over smaller fields. Alternatively, constructing more efficient branching programs for the SNARG verification step would also improve the concrete efficiency of our construction.

Instantiating the vector encryption scheme. Next, we instantiate the vector encryption scheme with the one by Peikert, Vaikuntanathan, and Waters [PVW08] described in Section 4.2. For estimating concrete efficiency, we rely on the single-theorem SNARG construction from Section 4.3 rather than the slightly less efficient construction from Appendix C. This seems like a reasonable heuristic to make when the underlying linear PCP satisfies strong soundness against affine prover strategies (Remark 4.8). An alternative and more conservative approach is to instantiate the encryption scheme with the "double-encryption" variant described in Remark C.4. Assuming the modified encryption scheme satisfies the stronger notion of linear-only encryption (Definition C.2), it can be combined with Construction 4.5 to obtain a multi-theorem SNARG (Theorem C.3). Applying this transformation incurs a (multiplicative) factor of 2 overhead in the ciphertext size (which translates roughly to a factor of 2 increase in the degree of multilinearity required).

In the remainder of this section, we describe how modulus switching can be used to reduce the width of the branching program needed to implement the verification algorithm. Then, in Section 5.5, we describe how to simplify the decision algorithm itself in the underlying linear PCPs that we use (Construction 3.4) to further reduce the length of the branching program.

Modulus switching. As described in Section 5.3, computing an inner product between two n-dimensional vectors over $\mathbb{Z}_q$ requires a branching program of dimension n and width q. When we instantiate Construction 3.4 with the Hadamard LPCP, the length of the verifier's query vector is $O(s^2)$, where s is the size of the Boolean circuit representing the computation being verified. Thus, the underlying vector encryption scheme must be able to support $O(s^2)$ homomorphic operations. Moreover, as described in Section 4.2, the ciphertext modulus q scales linearly with the number of homomorphic operations the encryption scheme must support. In our concrete estimates, we target $s \approx 2^{20}$, in which case the ciphertext modulus is at least $q \geq 2^{40}$. As a result, it is infeasible to implement the matrix-vector product needed for decryption within a branching program. Thus, for the SNARG verification procedure to be practically viable when implemented as a matrix branching program, it is critical that the ciphertext modulus is much smaller.

Fortunately, as was previously observed in the context of fully homomorphic encryption [BV11, BGV12, CNT12, AP14, DM15], it is possible to perform the homomorphic operations with respect to a large modulus q, and then reduce the resulting ciphertexts to be with respect to a smaller modulus q' (that scales sublinearly in the number of homomorphic operations ξ the scheme supports). In this work, we use the rescaling approach of [BGV12]. Modulus switching yields substantial improvements when we combine our new SNARG constructions with fully homomorphic encryption to bootstrap obfuscation (Section 5).

To apply modulus switching, we first define a rescaling operation $\mathsf{Scale}_{q,p,r}$ from [BGV12]. The rescaling operation is parameterized by three values p, q, r where $q > p > r$.

- $\mathsf{Scale}_{q,p,r}(\mathbf{x})$: On input a vector $\mathbf{x} \in \mathbb{Z}_q^n$, the $\mathsf{Scale}_{q,p,r}$ operator outputs the vector $\mathbf{x}' \in \mathbb{Z}_p^n$ that is closest to $(p/q) \cdot \mathbf{x}$ such that $\mathbf{x}' = \mathbf{x} \pmod{r}$. In other words, the $\mathsf{Scale}_{q,p,r}$ operator rescales each component of $\mathbf{x}$ by p/q (over the rationals) and then rounds the result to the nearest value in $\mathbb{Z}_p$ such that $\mathbf{x}' = \mathbf{x} \pmod{r}$.

To take advantage of modulus switching, we modify the decryption algorithm of the vector encryption scheme from Section 4.2 as follows. First, fix some modulus $q' < q$ such that $\gcd(q', q) = 1$. Then, we redefine the $\mathsf{Decrypt}$ function as follows:

- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: Compute and output $[[\mathbf{S}^\top \cdot \mathsf{Scale}_{q,q',p}(\mathbf{c})]_{q'}]_p$.

The key observation is that the matrix-vector product between the secret key and the ciphertext is now performed with respect to the modulus q' rather than q. First, we show that modulus switching cannot affect the correctness of our scheme (certainly, it cannot affect security, since the only modification we have made to the original scheme is in the decryption function).

Lemma 5.8 ([BGV12, Lemma 4], adapted)**.** *Let $q = q' = 1 \pmod p$. Take any vector $\mathbf{c} \in \mathbb{Z}_q^n$ and let $\mathbf{c}' = \mathsf{Scale}_{q,q',p}(\mathbf{c})$. Then, for any $\mathbf{s} \in \mathbb{Z}^n$ where $|[\langle \mathbf{s}, \mathbf{c} \rangle]_q| < q/2 - (q/q') \cdot (p/2) \cdot \|\mathbf{s}\|_1$, we have that*

$$[[\langle \mathbf{s}, \mathbf{c}' \rangle]_{q'}]_p = [[\langle \mathbf{s}, \mathbf{c} \rangle]_q]_p.$$

Proof. First, we write $[\langle \mathbf{s}, \mathbf{c} \rangle]_q = \langle \mathbf{s}, \mathbf{c} \rangle - kq$ for some $k \in \mathbb{Z}$. Let $\varepsilon = \langle \mathbf{s}, \mathbf{c}' \rangle - kq'$. Then,

$$\begin{aligned}
|\varepsilon| &= \left| -kq' + \langle \mathbf{s}, (q'/q) \cdot \mathbf{c} \rangle + \langle \mathbf{s}, \mathbf{c}' - (q'/q) \cdot \mathbf{c} \rangle \right| \\
&\leq \left| -kq' + \langle \mathbf{s}, (q'/q) \cdot \mathbf{c} \rangle \right| + \left| \langle \mathbf{s}, \mathbf{c}' - (q'/q) \cdot \mathbf{c} \rangle \right| \\
&\leq (q'/q) \, |[\langle \mathbf{s}, \mathbf{c} \rangle]_q| + p/2 \cdot \|\mathbf{s}\|_1 \\
&< q'/2.
\end{aligned}$$

We conclude that $\varepsilon = [\langle \mathbf{s}, \mathbf{c}' \rangle]_{q'}$. Working modulo p, we thus have that

$$[\langle \mathbf{s}, \mathbf{c}' \rangle]_{q'} = \varepsilon = \langle \mathbf{s}, \mathbf{c}' \rangle - kq' = \langle \mathbf{s}, \mathbf{c} \rangle - kq = [\langle \mathbf{s}, \mathbf{c} \rangle]_q \pmod p,$$

where we have used that fact that $\mathbf{c} = \mathbf{c}' \pmod p$ by definition of the $\mathsf{Scale}_{q,q',p}$ function and the assumption that $q = q' \pmod p$. The claim holds. □

Bounding the modulus size. We now derive lower bounds on the size of the decryption modulus q' (for correctness to hold). Let h be a bound on the ℓ_1 norm of the columns of the secret key $\mathbf{S}$. Since the amount of noise introduced by the rescaling operation (Lemma 5.8) scales with h, it is advantageous to work with low-norm secret keys. For example, if $\bar{\mathbf{S}}$ is sampled from a binary distribution, then $h \leq n + 1 \leq m$.[14] Then, we obtain the following corollary to Lemma 5.8.

Corollary 5.9. *Fix a statistical security parameter κ and a computational security parameter λ. Define parameters n, m, q, χ, σ as in Lemma 4.12. Let $\xi = \xi(\lambda)$ be a bound on the number of homomorphic operations the scheme supports and let $h = h(\lambda)$ be a bound on the ℓ_1 norm of the columns of $\mathbf{S}$. Suppose that $q \geq 2 \cdot \xi p^2 \sigma \sqrt{\kappa m}$ and $h < m$. Then, setting $q' = 2 \cdot p\sqrt{mh}$, the vector encryption scheme Π_{venc} with rescaled decryption is correct with overwhelming probability in κ.*

[14] If the secret keys are sampled from a uniform distribution, then $\|\mathbf{s}\|$ is large, and the rescaling operation significantly increases the noise in the ciphertext. Nonetheless, it is still possible to perform modulus switching by first applying a binary decomposition to the components of the secret key (and a corresponding "powers-of-two" transformation to the ciphertext components) before rescaling. This has the effect of increasing the dimension by a factor $\lceil \log q \rceil$. Similar techniques have been widely applied in the context of FHE and other lattice-based primitives [BV11, BGV12].

Proof. Let $\mathbf{c}$ be the ciphertext formed after performing ξ homomorphic operations (addition and scalar multiplications) on fresh ciphertexts. Since $q \geq 2 \cdot \xi p^2 \sigma \sqrt{m\kappa}$, we conclude by Lemma 4.12 that the magnitude of $[\mathbf{S}^\top \mathbf{c}]_q$ is bounded by $q/4$ with probability $1 - \mathsf{negl}(\kappa)$. Next, $q/q' = \xi p \sigma \sqrt{\kappa/h}$ and

$$\begin{aligned} q/2 - (q/q') \cdot (p/2) \cdot h &= q/2 - \xi p \sigma \sqrt{\kappa/h} \cdot p/2 \cdot h \\ &= q/2 - \frac{1}{2} \cdot \xi p^2 \sigma \sqrt{\kappa h} \\ &\geq q/2 - \frac{1}{2} \cdot \xi p^2 \sigma \sqrt{m\kappa} \geq \frac{q}{4}. \end{aligned}$$

The claim then follows by Lemma 5.8 (applied to the inner products between the ciphertext $\mathbf{c}$ and the columns of the secret key $\mathbf{S}$). □

Putting the pieces together. By Corollary 5.9, the modulus q' needed for decryption is now $q' = O(p\sqrt{mh})$. By Theorem 4.13, we need to set $m = \Theta(n \log q)$ for the underlying scheme to be semantically secure. Assuming the secret keys are drawn from a binary distribution (Remark 4.10), $h \leq (n + \ell) = \mathrm{poly}(\lambda)$, where λ is a security parameter. In our construction, the plaintext space is a polynomial-sized field $\mathbb{Z}_p$ (e.g., $p = 3$). Thus, the size of the modulus q' needed scales with $O(\sqrt{\log q} \cdot \mathrm{poly}(\lambda))$. Since q depends linearly on ξ, modulus switching allows us to use a modulus that scales *sub-logarithmically* in the number of homomorphic operations rather than linearly. This means decryption can be performed using branching programs of a much smaller width q', even if the computation that needs to be verified is complicated and requires a very large modulus q. Moreover, the vector encryption scheme in [PVW08] naturally supports modulus switching, so taking advantage of this optimization does not introduce any new security assumptions.

5.5 Simplifying the SNARG Decision Algorithm

In this section, we describe an optimization to reduce the length of the branching program needed to implement the SNARG verification procedure. Our optimization requires making a stronger linear-only assumption about the vector encryption scheme from Section 4.2. We note that the relaxations we describe here are not specific to our particular instantiation and can be used more generally to obtain SNARG candidates with better concrete efficiency (at the expense of needing a stronger cryptographic assumption).

The linear PCP construction from Section 3 (Construction 3.4) works by running $O(\kappa)$ independent copies of some underlying PCP with constant soundness error. To defend against malicious provers using affine strategies, the verifier first applies a random linear shift to its queries before sending them to the prover. During verification, the verifier takes the prover's response vector, undoes the linear shift, and then verifies each of the underlying LPCPs. While performing this linear shift does not significantly increase the verifier's work in the standard model, the extra overhead is quite considerable in the branching program model.

Suppose for sake of argument that the linear shift is unnecessary—that is, the verifier simply decrypts the prover's response vector and applies the decision procedure of the underlying LPCP to each of the $O(\kappa)$ instances. Since each of these $O(\kappa)$ instances of the underlying LPCP are *independent*, we can use the CRT embedding described in Section 5.3 to decrypt and verify the $O(\kappa)$ proofs *in parallel*. However, if the verification procedure requires applying a random linear

transformation to the decrypted responses, the verifier can no longer leverage the CRT embedding to decrypt and verify the $O(\kappa)$ responses in parallel. Verifying the $O(\kappa)$ proofs sequentially incurs an extra $O(\kappa)$ overhead in the length of the branching program, which significantly reduces the concrete efficiency of our candidate obfuscation construction.

Soundness against restricted affine provers. The random linear shift introduced in Construction 3.4 is essential for showing robustness against provers that use an *arbitrary* affine strategy. However, if we impose additional restrictions on the strategies available to the prover, we can obtain a more lightweight LPCP. In conjunction with a stronger notion of linear-only encryption that constrains the prover to a restricted set of affine strategies, we can still apply Construction 4.5 to obtain a designated-verifier SNARG.

One such relaxation of soundness against affine provers is to only require soundness against *linear* provers. In this case, Construction 3.4 without the random linear shift suffices for obtaining a linear PCP with soundness against linear provers. But to compile a proof system with soundness against linear provers to a SNARG requires a very strict linear-only encryption scheme that our candidate construction does not seem to satisfy. We now introduce an intermediate notion of "soundness against restricted affine provers" that interpolates between requiring soundness to hold against only linear strategies and requiring it to hold against arbitrary affine strategies. We then prove (in Theorem 5.12) an analog of Theorem 3.5 and show that Construction 3.4 without the random linear shift (i.e., Construction 3.4 where $\mathbf{Y} = \mathbf{I}_{\kappa\ell}$ is the identity matrix) suffices to give a linear PCP with strong soundness against certain classes of restricted affine provers. Since the random linear shift is no longer applied, Theorem 5.12 imposes a stronger "robustness" requirement on the underlying LPCP against affine strategies. Note that existing LPCP constructions such as the Hadamard LPCP (Lemma B.2, Remark B.3) satisfy this robustness property.

Definition 5.10 (Soundness Against Restricted Affine Provers)**.** Let $\mathcal{R}$ be a relation and $\mathbb{F}$ be a finite field. Fix some parameter $t \in \mathbb{N}$ and let $\mathcal{D}$ be a distribution over $\mathbb{F}^{t\times k}$. A linear PCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ is a k-query linear PCP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ε against $\mathcal{D}$-restricted affine provers if it satisfies the requirements in Definition 3.1 with the following modifications:

- **Syntax:** For vectors $\boldsymbol{\pi} \in \mathbb{F}^m$ and $\mathbf{v} \in \mathbb{F}^t$, the verification algorithm $V_{\mathsf{LPCP}}^{(\boldsymbol{\pi},\mathbf{v})}$ is a tuple of two algorithms $(Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}})$ with the properties specified in Definition 3.1. The only difference is that the response vector $\mathbf{y}$ computed by the PCP oracle is an affine function $\mathbf{y} = \mathbf{Q}^\top\boldsymbol{\pi} + \mathbf{S}^\top\mathbf{v}$ where $\mathbf{S} \leftarrow \mathcal{D}$.
- **Completeness:** For every $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, the output of $P_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$ is a vector $\boldsymbol{\pi} \in \mathbb{F}^m$ such that $V_{\mathsf{LPCP}}^{(\boldsymbol{\pi},\mathbf{0})}(\mathbf{x})$ accepts with probability 1.
- **Soundness against $\mathcal{D}$-restricted provers:** For all $\mathbf{x}$ where $(\mathbf{x}, \mathbf{w}) \notin \mathcal{R}$ for all $\mathbf{w}$, and for all vectors $\boldsymbol{\pi}^* \in \mathbb{F}^m$ and $\mathbf{v}^* \in \mathbb{F}^t$, the probability that $V_{\mathsf{LPCP}}^{(\boldsymbol{\pi}^*,\mathbf{v}^*)}(\mathbf{x})$ accepts is at most ε, where the probability is taken over the randomness of the verifier as well as the response from the PCP oracle.

Following Definition 3.3, we can similarly define a notion of strong soundness against $\mathcal{D}$-restricted provers.

Remark 5.11 (Interpreting Definition 5.10)**.** Definition 5.10 can be viewed as an interpolation between requiring soundness against provers who can only use linear strategies (no affine term at

all) and provers that can use arbitrary affine strategies (fully specify the affine term). Both of these extremes are captured by Definition 5.10. Soundness against linear provers corresponds to setting $t = 1$ and letting $\mathcal{D}$ be a point distribution over the all-zeroes vector $\mathbf{0}^k$, while soundness against affine provers corresponds to setting $t = k$ and letting $\mathcal{D}$ be a point distribution over the identity matrix $\mathbf{I}_k$.

Theorem 5.12. *Fix a statistical security parameter κ. Let $\mathcal{R}$ be a binary relation, $\mathbb{F}$ be a finite field, and $(P^{(\text{weak})}_{\mathsf{LPCP}}, V^{(\text{weak})}_{\mathsf{LPCP}})$ be a strongly sound ℓ-query linear PCP for $\mathcal{R}$ with constant soundness error $\varepsilon \in [0, 1)$ against linear provers. Suppose moreover that $(P^{(\text{weak})}_{\mathsf{LPCP}}, V^{(\text{weak})}_{\mathsf{LPCP}})$ satisfies the following robustness property against affine provers: for any prover P^*_{LPCP} employing an affine strategy $\boldsymbol{\Pi}^* = (\boldsymbol{\pi}^*, \mathbf{b}^*)$ where $\mathbf{b}^* \neq \mathbf{0}^\ell$, the verifier $V^{(\text{weak})}_{\mathsf{LPCP}}$ rejects with constant probability. Fix some parameter $t \in \mathbb{N}$ and let $\mathcal{D}$ be a distribution over $\mathbb{F}^{t \times \kappa\ell}$. Suppose there exists some constant $c \in (0, 1]$ such that for all non-zero $\mathbf{v} \in \mathbb{F}^t$,*

$$\Pr_{\mathbf{S} \leftarrow \mathcal{D}}\left[\text{nonzero}(\mathbf{S}^\top \mathbf{v}) \geq c \cdot \kappa\ell\right] = 1 - \text{negl}(\kappa), \tag{5.1}$$

where on input a vector $\mathbf{x} \in \mathbb{F}^k$, $\text{nonzero}(\mathbf{x})$ outputs the number of non-zero entries in $\mathbf{x}$. Then, the linear PCP $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ from Construction 3.4 without the random linear shift (i.e., setting $\mathbf{Y} = \mathbf{I}_{\kappa\ell}$) is a $(\kappa\ell)$-query linear PCP for $\mathcal{R}$ with strong statistical soundness against $\mathcal{D}$-restricted affine provers.

Proof. Completeness follows as in the proof of Theorem 3.5. It suffices to check that the linear PCP is statistically sound with strong soundness against $\mathcal{D}$-restricted affine provers. Take any statement $\mathbf{x}$ and consider a prover strategy $(\boldsymbol{\pi}^*, \mathbf{v}^*)$ where $\boldsymbol{\pi}^* \in \mathbb{F}^m$ and $\mathbf{v}^* \in \mathbb{F}^t$. We consider two cases:

- Suppose $\mathbf{v}^* \neq 0^t$. Then, the verifier applies its decision algorithm to the quantity $\mathbf{a} = \mathbf{Q}^\top \boldsymbol{\pi}^* + \mathbf{S}^\top \mathbf{v}^*$ where $\mathbf{S} \leftarrow \mathcal{D}$. Specifically, the verifier V_{LPCP} writes $\mathbf{a} = [\mathbf{a}_1 | \mathbf{a}_2 | \cdots | \mathbf{a}_\kappa]$ and applies $D^{(\text{weak})}_{\mathsf{LPCP}}$ to each of $\mathbf{a}_1, \ldots, \mathbf{a}_\kappa$. Next, by assumption, with overwhelming probability, $\text{nonzero}(\mathbf{S}^\top \mathbf{v}^*) \geq c \cdot \kappa\ell$. In particular, this means that there exists some collection of $c \cdot \kappa$ indices $i \in [\kappa]$ such that $\mathbf{a}_i = \mathbf{Q}_i^\top \boldsymbol{\pi}^* + \mathbf{z}_i$ where $\mathbf{Q}_i \in \mathbb{F}^{m \times \ell}$ is the i^{th} query and $\mathbf{z}_i \neq \mathbf{0}^\ell$ is a non-zero affine term. But by the robustness property on the underlying linear PCP, $D^{(\text{weak})}_{\mathsf{LPCP}}$ rejects $\mathbf{a}_i$ with constant probability. Thus, in this case, V_{LPCP} rejects with probability $1 - \text{negl}(\kappa)$.
- Suppose $\mathbf{v}^* = 0^t$. In this case, the prover's strategy is a linear function $\boldsymbol{\pi}^*$. The claim then follows by the corresponding argument in the proof of Theorem 3.5. □

The main advantage of Theorem 5.12 over Theorem 3.5 is that it is possible to achieve strong soundness against certain algebraically-bounded provers without needing to apply a random linear shift to the queries. Combined with a stronger notion of of linear-only vector encryption, it is possible to construct a SNARG where the verification algorithm can be efficiently implemented using a matrix branching program (over a composite-order ring).

Restricted linear targeted malleability. We now introduce our strengthened notion of $\mathcal{D}$-restricted targeted malleability for a (secret-key) vector encryption scheme.

Definition 5.13 ($\mathcal{D}$-Restricted Targeted Malleability)**.** Fix a security parameter λ and a finite field $\mathbb{F}$. Let $\mathcal{D}$ be a distribution over $\mathbb{F}^{t\times k}$ for some $t \in \mathbb{N}$. A secret-key vector encryption scheme for a vector space over a finite field $\mathbb{F}$ satisfies $\mathcal{D}$-restricted linear targeted malleability if for all efficient adversaries $\mathcal{A}$ and plaintext generation algorithms $\mathcal{M}$ (on input 1^k, algorithm $\mathcal{M}$ outputs vectors in $\mathbb{F}^k$), 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. $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$
2. $(s, \mathbf{v}_1, \ldots, \mathbf{v}_q) \leftarrow \mathcal{M}(1^k)$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for all $i \in [q]$
4. $\mathsf{ct}' \leftarrow \mathcal{A}(\{\mathsf{ct}_i\}_{i\in[q]}; z)$ where $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') \neq \perp$
5. Output $\left(\{\mathbf{v}_i\}_{i\in[q]}, s, \mathsf{Decrypt}_{\mathsf{sk}}(\mathsf{ct}')\right)$

Ideal Distribution:

1. $(s, \mathbf{v}_1, \ldots, \mathbf{v}_q) \leftarrow \mathcal{M}(1^k)$
2. $(\boldsymbol{\pi}, \mathbf{b}) \leftarrow \mathcal{S}(z)$ where $\boldsymbol{\pi} \in \mathbb{F}^q$, $\mathbf{b} \in \mathbb{F}^t$
3. $\mathbf{v}' \leftarrow [\mathbf{v}_1|\mathbf{v}_2|\cdots|\mathbf{v}_q] \cdot \boldsymbol{\pi} + \mathbf{S}^\top \mathbf{b}$ where $\mathbf{S} \leftarrow \mathcal{D}$
4. Output $\left(\{\mathbf{v}_i\}_{i\in[q]}, s, \mathbf{v}'_i\right)$

At a high level, a vector encryption scheme satisfying Definition 5.13 is (at the minimum) linearly homomorphic, but also supports a limited number of affine transformations (determined by the distribution $\mathcal{D}$). Note that this notion only makes sense in the secret-key setting. In the public-key setting, any linearly homomorphic encryption scheme necessarily supports affine transformations since the adversary can simply encrypt vectors of its choosing. In the secret-key setting where the adversary is unable to generate encryptions of arbitrary vectors, the vector encryption scheme can plausibly only support a limited set of affine transformations.

Next, we combine a vector encryption scheme that only allows a restricted set of affine homomorphisms with a linear PCP with soundness against prover strategies belonging to the same restricted set of affine functions to obtain a SNARG. We state the analog of Theorem 4.6 in the context of $\mathcal{D}$-restricted linear targeted malleability and LPCPs with soundness against $\mathcal{D}$-restricted affine provers.

Theorem 5.14. *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be a linear PCP that is sound against $\mathcal{D}$-restricted affine provers for some distribution $\mathcal{D}$, and let $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a vector encryption scheme satisfying $\mathcal{D}$-restricted linear targeted malleability. Then, applying Construction 4.5 to $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ and Π_{venc} yields a single-theorem, designated-verifier SNARG in the preprocessing model.*

Proof. Identical to the proof of Theorem 4.6, except using Definitions 5.10 and 5.13 in place of Definitions 3.2 and 4.2, respectively. □

Tweaking the vector encryption scheme. Consider the vector encryption scheme from [PVW08] in Section 4.2. Decryption in that scheme consists of taking a matrix-vector product between the secret key $\mathbf{S}$ and the ciphertext vector $\mathbf{c}$ and then reducing each component modulo p. Notably, modifying any *single* component of the ciphertext will affect the decrypted value in *multiple* components of the underlying vector (provided that the rows of the secret key $\mathbf{S}$ do not have low-weight). More concretely, take a ciphertext $\mathbf{c} \in \mathbb{Z}_q^{n+\ell}$ and suppose we add a fixed vector $\mathbf{v} \in \mathbb{Z}_q^{n+\ell}$ offset to $\mathbf{c}$. The decryption function then computes $[[\mathbf{S}^\top \mathbf{c} + \mathbf{S}^\top \mathbf{v}]_q]_p$. This has the semblance of applying an affine shift to the underlying message, where the affine offset is given by a linear combination of the rows of $\mathbf{S}$ with coefficients specified by the components of $\mathbf{v}$. In light of this observation,

we conjecture that the vector encryption scheme from Section 4.2 plausibly satisfies $\mathcal{D}$-restricted targeted malleability where $\mathcal{D}$ is the distribution from which the secret key $\mathbf{S}$ is sampled.

The question then becomes whether there exists a linear PCP with strong soundness against $\mathcal{D}$-restricted affine provers, where $\mathcal{D}$ is the secret-key distribution for the encryption scheme from Section 4.2. The best candidate is to apply Theorem 5.12 to a suitable constant-error LPCP such as the Hadamard PCP. However, to invoke Theorem 5.12, it is necessary that for all $\mathbf{v} \in \mathbb{Z}_q^{n \times \ell}$, the product $\mathbf{S}^\top \mathbf{v}$ where $\mathbf{S} \leftarrow \mathcal{D}$ has a constant fraction of non-zero elements. This is not true when $\mathcal{D}$ is the secret-key distribution for the Peikert-Vaikuntanathan-Waters encryption scheme from Section 4.2. Specifically, when $\mathbf{S}^\top = [-\bar{\mathbf{S}}^\top | \, \mathbf{I}_\ell]$, where $\bar{\mathbf{S}}$ is uniformly random, there are many vectors $\mathbf{v} \in \mathbb{Z}_q^{n+\ell}$ where $\mathbf{S}^\top \mathbf{v}$ has a small (constant) number of non-zero entries (for instance, the basis vectors $\mathbf{e}_{n+1}, \ldots, \mathbf{e}_{n+\ell}$). The problem arises from the fact that $\mathbf{S}$ is sampled from a distribution where some rows have low Hamming weight (i.e., the rows of $\mathbf{I}_\ell$). This can be addressed by substituting a random invertible matrix $\mathbf{U}$ for the identity matrix in the secret key. We describe the modified vector encryption $\Pi'_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ scheme below:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Choose $\bar{\mathbf{A}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\bar{\mathbf{S}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times \ell}$, $\bar{\mathbf{U}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell \times \ell}$, and $\bar{\mathbf{E}} \leftarrow \chi^{\ell \times m}$, where $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$ are polynomials in the security parameter. Define matrices $\mathbf{A} \in \mathbb{Z}_q^{(n+\ell) \times m}$ and $\mathbf{S} \in \mathbb{Z}_q^{(n+\ell) \times \ell}$ as follows:

$$\mathbf{A} = \begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{U}}^{-1}(\bar{\mathbf{S}}^\top \bar{\mathbf{A}} + p\bar{\mathbf{E}}) \end{bmatrix} \qquad \mathbf{S} = \begin{bmatrix} -\bar{\mathbf{S}} \\ \bar{\mathbf{U}}^\top \end{bmatrix},$$

  where $\mathbf{I}_\ell \in \mathbb{Z}_q^{\ell \times \ell}$ is the ℓ-by-ℓ identity matrix. Output the secret key $\mathsf{sk} = (\mathbf{A}, \mathbf{S}, \mathbf{U})$.

- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$: To encrypt a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$, choose $\mathbf{r} \xleftarrow{\mathrm{R}} \{0,1\}^m$ and output the ciphertext $\mathbf{c} \in \mathbb{Z}_q^{n+\ell}$ where

$$\mathbf{c} = \mathbf{A}\mathbf{r} + \begin{bmatrix} \mathbf{0}^n \\ \mathbf{U}^{-1}\mathbf{v} \end{bmatrix}.$$

- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: Compute and output $[[\mathbf{S}^\top \mathbf{c}]_q]_p$.

By inspection, applying the secret linear shift on the messages does not affect the correctness or the security of the scheme. Additionally, this scheme remains compatible with the modulus switching optimization described in Section 5.4.

In addition, in this variant of the vector encryption scheme, the secret key is uniformly random over $\mathbb{Z}_q^{(n+\ell) \times \ell}$. Thus, for all $\mathbf{v} \in \mathbb{Z}_q^{n+\ell}$, it follows that for any $i \in [\ell]$, $(\mathbf{S}^\top \mathbf{v})_i = 0$ with probability $1/q$ over the randomness used to sample $\mathbf{S}$. Since each row of $\mathbf{S}^\top$ is sampled independently, it holds that $\mathbf{S}^\top \mathbf{v}$ is zero in any constant fraction of the components with probability that is negligible in the dimension ℓ. We remark that this remains true even if $\bar{\mathbf{S}}$ and $\bar{\mathbf{U}}$ are sampled from a binary distribution (where each entry is independently 0/1 with equal probability).

Finally, putting all the pieces together, we obtain a new candidate SNARG construction with a more lightweight verification procedure:

Corollary 5.15. *Let Π'_{venc} be the vector encryption scheme described above and suppose that Π'_{venc} satisfies $\mathcal{D}$-restricted linear targeted malleability where $\mathcal{D}$ is the distribution from which the secret key of Π'_{venc} is sampled. Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be the linear PCP be the output of applying Construction 3.4 without the random affine shift to the 3-query Hadamard LPCP. Then applying Construction 4.5 to*

$(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ *and* Π'_{venc} *yields a single-theorem, designated-verifier SNARG in the preprocessing model.*

Proof. By construction of Π'_{venc}, the distribution $\mathcal{D}$ is the uniform distribution over $\mathbb{Z}_q^{(n+\ell)\times\ell}$, and thus, trivially satisfies Equation (5.1). The 3-query Hadamard LPCP (described in detail in Appendix B) satisfies the stronger notion of "robustness" against affine queries needed to invoke Theorem 5.12. The corollary then follows from Theorems 5.12 and 5.14. □

The verification procedure for the SNARG given by Corollary 5.15 has the appealing property that it can be efficiently implemented using a matrix branching program. Specifically, verification consists of decrypting the encrypted response vector and then applying the Hadamard decision algorithm to each of the decrypted responses. Verification succeeds only if all of the underlying $O(\kappa)$ Hadamard instances verify. Notably, decrypting and verifying the prover's responses to each of of the underlying Hadamard LPCP instances can be performed in parallel via the CRT embedding trick described in Section 5.3. In fact, the verification algorithm for the Hadamard LPCP[15] also decomposes into two independent checks on the prover's responses, which can also be verified in parallel.

5.6 Concrete Parameters for Bootstrapping VBB Obfuscation

In this section, we give some concrete parameter estimates for the length and width of the branching programs needed to implement the VBB bootstrapping construction in Section 5.2 (after applying the optimizations of Sections 5.4 and 5.5).

Our concrete parameter estimates are based on choosing parameters for both the FHE scheme as well as our optimized vector encryption scheme from Section 5.5. For all of our estimates, we set the computational security parameter to $\lambda = 80$ bits and the statistical security parameter to $\kappa = 40$ bits. Similar to other practical works on lattice-based cryptography [GHS12, DM15], we choose our parameters based on the runtime of the best-known attack (to our knowledge) rather than the (more pessimistic) bounds given by the worst-case reduction to hard lattice problems.

Concrete parameters for LWE. The security of our core building blocks for bootstrapping VBB obfuscation relies on the hardness of LWE. Our parameter estimates for LWE instances follow the analysis of Ducas and Micciancio [DM15], which is itself derived from the analysis of Lindner and Peikert [LP11]. This type of analysis has also been used as the basis for setting parameters in other lattice-based cryptosystems [GHS12].

To achieve a distinguishing advantage of ε for the LWE problem with dimension n, modulus q, and an error distribution $\chi = D_{\mathbb{Z},\sigma}$ with standard deviation σ, Lindner and Peikert [LP11] estimate that the best known attack via lattice reduction requires computing a basis with root Hermite factor

$$\delta = \delta\text{-}\mathsf{LWE}(n, q, \sigma, \varepsilon) = 2^{(\log^2 \beta)/(4n \log q)},$$

where $\beta = (q/\sigma)\sqrt{\ln(1/\varepsilon)/\pi}$. Based on an empirical study, Lindner and Peikert give the following estimate on the runtime $t(\delta)$ of the BKZ lattice reduction algorithm needed to obtain a basis with a root Hermite factor δ:

$$t(\delta) \geq 1.8/\log(\delta) - 110.$$

[15]The details of the verification algorithm are given in Appendix B.

We use this estimate to set our parameters. We caution readers here that our analysis is only intended to serve as a rough estimate for the parameters under which our scheme should be secure. For a more precise analysis, one should also take into consideration the more involved analysis of [CN11, LN13, APS15] as well as any recent advances in lattice cryptanalysis.

The analysis of Lindner and Peikert pertains to the standard LWE setting where the secret keys are sampled from a uniform distribution. For better control over the noise growth, it is preferable to sample the secret key from a low norm distribution (Remark 4.10). In deriving our concrete parameter estimates, we assume that the secret keys are sampled from a binary distribution. For a lattice dimension n, modulus q, and error distribution $\chi = D_{\mathbb{Z},\sigma}$, we write $\mathsf{binLWE}_{n,q,\chi}$ to denote the LWE assumption where the secret keys are instead sampled uniformly at random from the binary distribution $\{0,1\}^n$.

While the hardness of $\mathsf{binLWE}_{n,q,\chi}$ reduces to the standard LWE assumption (with larger parameters) [BLP+13], choosing the parameters according to the security reduction is overly pessimistic. Instead, we adopt the approaches of [GHS12, DM15] and base our parameters on the known attacks on LWE instances with a binary secret. As noted in [DM15], LWE with binary secrets does not enjoy the same level of concrete security as standard LWE. In particular, binary secrets allow an attacker to switch the LWE ciphertexts to a smaller modulus q' without significantly affecting the modulus-to-noise ratio (the parameter β). As shown in [DM15], given a binary LWE instance $\mathsf{binLWE}_{n,q,\chi}$ where $\chi = D_{\mathbb{Z},\sigma}$, one can apply modulus switching to obtain a new binary LWE instance $\mathsf{binLWE}_{n,q',\chi'}$ of the same dimension, but with smaller modulus $q' < q$ and scaled noise distribution $\chi' = D_{\mathbb{Z},\sigma'}$ where $\sigma' = \sqrt{(q'/q)^2\sigma^2 + \|\mathbf{s}\|_1/12} \approx \sigma q'/q$. In light of this, Ducas and Micciancio estimate the necessary root Hermite factor needed to break an $\mathsf{binLWE}_{n,q,\chi}$ instance to be

$$\delta\text{-}\mathsf{binLWE}(n,q,\sigma,\varepsilon) = \min_{q' \leq q} \delta\text{-}\mathsf{LWE}(n,q',\sigma' = \sqrt{(q'/q)^2\sigma^2 + \|\mathbf{s}\|_1/12},\varepsilon), \tag{5.2}$$

where $\mathbf{s} \in \mathbb{Z}_q^n$ is the LWE secret. When $\mathbf{s}$ is sampled from a binary distribution (i.e., $\mathbf{s} \xleftarrow{R} \{0,1\}^n$), we can use $n/2$ as a bound on $\mathbf{s}$ (concretely, one can sample the keys as usual, but only take the ones where the norm is at most $n/2$). Computing the minimum can be done using standard numeric analysis tools. This is the heuristic we use when estimating our concrete parameters.

Fully homomorphic encryption. The first building block we require is a FHE scheme where the decryption function can be efficiently computed by a matrix branching program, or more concretely, a rounded inner product over a polynomial-sized ring (Section 5.3). Most existing FHE constructions based on LWE have this property. To facilitate concrete parameter estimates, we use the lightweight scheme by Ducas and Micciancio [DM15]. In the Ducas-Micciancio FHE scheme, ciphertexts are plain LWE ciphertexts and decryption consists of a single rounded inner product between the secret key and the ciphertext. Ducas and Micciancio recommend using an LWE instance with dimension $n = 500$ and modulus $q = 2^9$. These parameter settings[16] should correspond to approximately 80 bits of security (the requisite root Hermite factor needed to obtain a 2^{-40} advantage in breaking the binary LWE assumption for their proposed set of parameters is $\delta = 1.008$, which under the Lindner-Peikert run-time estimates for lattice reduction algorithms, requires time $\approx 2^{40}$ to compute).

[16] Due to specific details of their construction, they require security for a binary LWE instance with a much larger modulus $Q = 2^{32}$. This is the modulus used to estimate the concrete security of the scheme. Note that because of modulus switching, the final LWE ciphertext that is decrypted is still with respect to the *small* modulus q.

Remark 5.16 (Alternative FHE Schemes)**.** When performing long computations, it is oftentimes more efficient (both concretely and asymptotically) to use FHE schemes where the overhead of FHE evaluation is only polylogarithmic [GHS12] or ones that supports parallel evaluation of vectors of plaintexts [BGV12, SV14]. Many of these alternative constructions are based on the ring-LWE problem, where decryption corresponds to evaluating an inner product in a polynomial ring. While it does not seem straightforward to represent these more involved decryption functions as simple branching programs, we note that using a combination of a "branching-program-friendly" FHE scheme (i.e. FHE scheme where decryption can be implemented by a rounded inner product over a polynomial-size ring) with a more efficient FHE scheme gives a construction that achieves the "best of both worlds". The hybrid scheme works as follows:

- **Setup:** The public key for the hybrid FHE scheme consists of the public keys for both underlying FHE schemes as well as an encryption of the secret key for the efficient FHE scheme under the public key of the branching-program-friendly FHE scheme. The secret key is the decryption key for the branching-program-friendly FHE scheme.
- **Encryption:** Encryption is just encryption under the public key for the efficient FHE scheme.
- **Evaluation:** Computing on ciphertexts is handled using the homomorphic properties of the efficient FHE scheme.
- **Decryption:** Recall that ciphertexts in the scheme are all encrypted under the efficient FHE scheme. Decryption works by first converting the ciphertext into a ciphertext that encrypts the same message, but under the key for the branching-program-friendly FHE scheme. This is done by first re-encrypting the ciphertext using the branching-program-friendly FHE scheme and then homomorphically evaluating the decryption function for the efficient FHE scheme (using the encrypted secret key provided in the public parameters). This yields an encryption of the original message under the secret key for the branching-program-friendly FHE scheme, which can now be decrypted.

By using this kind of composition, the computational overhead of the FHE evaluation is determined by the asymptotic (or concrete) efficiency of the efficient FHE scheme. The possibly less efficient, but branching-program-friendly FHE scheme is only used to homomorphically evaluate the decryption function of another FHE scheme, which is usually a simple circuit (of size that depends only on the security parameter) and independent of the complexity of the obfuscated program.

SNARG system. The second building block we require is a SNARG system where the SNARG verification procedure can be expressed as a simple branching program. As noted in Section 5.4, while the basic SNARG obtained from directly applying Constructions 3.4 and 4.5 can be computed using an MBP, it is unlikely to yield much concrete efficiency. To obtain a more efficient construction (at the cost of making more aggressive assumptions about our underlying primitives), we use the optimized variant of the SNARG construction from Section 5.5 (Corollary 5.15). We now describe how we choose the concrete parameters for our SNARG system.

- **The plaintext modulus p.** Since we are instantiating our SNARG construction (Construction 4.5) using the Hadamard-based linear PCP, we require $p > 2$. Using larger values of p will allow for using a smaller plaintext dimension ℓ (described below), but in exchange, requires a larger ciphertext modulus q to accommodate the greater noise introduced by the homomorphic operations. In our setting, using the smallest value of $p = 3$ yields the best parameters.

- **The plaintext dimension ℓ.** The plaintext dimension ℓ corresponds to the number of LPCP oracle queries. For the linear PCP resulting from applying Construction 3.4 to the 3-query Hadamard LPCP, we have $\ell = 3\ell'$ where ℓ' is the number of instances of the constant-query LPCP we perform to obtain the desired security level. Targeting a statistical soundness error of $2^{-\kappa}$, and applying Construction 3.4 to the Hadamard PCP, we require that $(2/p)^{\ell'} < 2^{-\kappa}$. For $p = 3$, we require $\ell' \geq 68$ independent instances of the underlying LPCP. For our estimates, we use a larger ℓ' than necessary so as to be able to use a sparser secret key distribution (discussed below). We set $\ell' = 120$, which corresponds to $\ell = 3\ell' = 360$.

- **The ciphertext modulus q.** The ciphertext modulus q is chosen based on the number of homomorphic operations ξ the scheme needs to support (additions and scalar multiplications). When we apply Construction 4.5 to construct a SNARG from the Hadamard PCP, the underlying linear-only vector encryption scheme must be able to support a total of $s^2 + s$ homomorphic additions, where s is the size of the Boolean circuit for the computation being verified. In the case of bootstrapping for VBB obfuscation, the statement being verified is correctness of FHE evaluation. Due to the complicated arithmetic operations involved in FHE evaluation, the circuit needed to verify correctness of such a computation can be very large. In this case, the quadratic overhead of the Hadamard PCP can render the scheme infeasible. In Remark 5.17, we show that using SNARK composition techniques, it suffices that our branching-program-friendly SNARG is able to verify computations of modest size. For our estimates, we consider verifying computations containing up to 2^{20} gates. Thus, we choose the ciphertext modulus q so that the linear-only encryption scheme is able to support $\xi \geq 2^{40}$ homomorphic operations (with some slack to allow for modulus switching). This corresponds to $q \approx 2^{59}$.

- **The secret key distribution.** The first step in SNARG verification is linear-only decryption which requires computing a series of (rounded) inner products over $\mathbb{Z}_q$. Using our approach for inner product evaluation (Section 5.3), this requires branching programs of width $q \approx 2^{62}$, which is prohibitively expensive. Thus, it is critical that we apply the modulus switching optimization (Section 5.4) to shrink the ciphertext modulus before decryption. Since the size of the branching program scales cubically in the magnitude of the reduced modulus q' (the MBPs are defined over an alphabet of size q), to maximize the concrete efficiency of the scheme, it is important to minimize the value of q'. According to Lemma 5.8 (and Corollary 5.9), the value of q' scales with the norm of the rows of the secret key matrix $\mathbf{S}^\top = \left[-\bar{\mathbf{S}}^\top \middle| \bar{\mathbf{U}}\right] \in \mathbb{Z}_q^{\ell \times (n+\ell)}$. One possibility is to choose $\mathbf{S}^\top$ uniformly from the binary distribution $\{0,1\}^{\ell \times (n+\ell)}$. An even more aggressive option is to sample the rows from a binary distribution with a *fixed* Hamming weight. This kind of secret key distribution has previously been used in [GHS12] to improve the concrete performance of FHE.

  When choosing a secret key distribution, it is important to also keep in mind the requirements of Corollary 5.15, namely, that the distribution from which we sample the secret key should satisfy Equation (5.1). Our goal is to minimize the weight of the rows of $\mathbf{S}^\top$ while ensuring that each column of $\mathbf{S}^\top$ have approximately $\approx \kappa = 40$ non-zero entries. For our estimates, we sample each row of $\bar{\mathbf{S}}^\top$ (each of the ℓ LWE secret keys) from a binary distribution with exactly $h_{\bar{\mathbf{S}}} = 130$ non-zero values, each set to ± 1 with equal probability. Up to small constant factors, the number of non-zero entries in each column of $\bar{\mathbf{S}}$ can be approximated by $h_{\bar{\mathbf{S}}}/n \cdot \ell$. Similarly,

we sample each column of $\bar{\mathbf{U}}$ to have $h_{\bar{\mathbf{U}}} = 40$ non-zero entries. Thus, each row of $\mathbf{S}^\top$ has exactly $h = h_{\bar{\mathbf{S}}} + h_{\bar{\mathbf{U}}} = 170$ non-zero entries. For this setting of parameters, the columns of $\mathbf{S}^\top$ should have approximately 40 non-zero entries. We estimate the concrete security of this LWE instance using the Ducas-Micciancio heuristic for LWE instances with binary secrets (Equation (5.2)).

- **Other parameters.** The other LWE parameters (the lattice dimension n and the standard deviation σ of the error distribution $\chi = D_{\mathbb{Z},\sigma}$) are chosen to provide $\lambda = 80$ bits of security. The number of samples m is set to $m = 3(n+\ell)\log q$, as required by Theorem 4.13. The reduced modulus q' is chosen to be the smallest value such that the resulting scheme remains correct assuming the rows of the secret key matrix $\mathbf{S}^\top$ have fixed Hamming weight $h = 170$. We compute q' based on the concrete error bounds in Lemma 5.8.[17] This yields concrete estimates $n = 1175$, $\sigma = 8$, and $q' \approx 2^{10}$.

Based on our analysis, to achieve a distinguishing advantage of 2^{-40} for a $\mathsf{binLWE}_{n,q,\sigma}$ instance (with secret key drawn from a fixed Hamming weight distribution) with this set of parameters ($n = 1175$, $q \approx 2^{59}$, and $\sigma \approx 9$) requires computing a reduced basis with root Hermite factor $\delta = 1.008$, which under current estimates, requires computation time $t(\delta) \approx 2^{40}$.

Remark 5.17 (Reducing Overhead via SNARK Composition)**.** While our Hadamard-based SNARG construction has a branching-program-friendly verification procedure, there is a significant amount of prover overhead (quadratic in the size of the computation being verified). However, if we use a SNARK in place of the SNARG (recall that the difference between a SNARG and a SNARK is that the soundness requirement in the SNARG is replaced by a knowledge requirement in the SNARK), it is possible to compose several independent SNARK construction to achieve better asymptotic efficiency. We give a high-level description of our construction:

- **Minimizing prover overhead:** On input a statement-witness pair $(\mathbf{x}, \mathbf{w})$, the prover first constructs a proof of the statement using a publicly-verifiable SNARK system that minimizes the prover overhead. One candidate is the pairing-based scheme of [GGPR13] which has quasilinear prover overhead.

- **Minimizing circuit complexity of SNARK verification:** Next, the prover proves that it knows a proof for the statement $\mathbf{x}$ that verifies under the public verification key of the previous SNARK scheme. The prover's run-time in this step scales with the size of the verification circuit of the previous (pairing-based) SNARK. While the size of this circuit depends only polylogarithmically on the size of the original computation, the verification circuit is still quite large in concrete terms (due to needing to encapsulate a pairing computation). Thus, for better concrete efficiency at the obfuscation layer, we apply an intermediate SNARK that reduces (concretely) the size of the verification circuit. Here, we use the publicly-verifiable hash-based constructions of computationally-sound (CS) arguments of knowledge [Mic00, Val08]. The prover overhead of this step is only quasilinear in the size of the verification circuit if we use the quasilinear PCPs of [BBGR16, BBC+17] to construct the CS arguments.

[17] Using the concrete error bounds of Lemma 5.8, we can use a smaller value for q' than that recommended by Corollary 5.9. This is because there is a lot of slack in the bounds provided by Corollary 5.9 when the LWE secret keys are sampled from a low-norm distribution.

- **Minimizing branching-program complexity of verification:** As in the previous step, the prover again proves that it knows a proof for $\mathbf{x}$ that verifies under the previous SNARK scheme. This SNARK should be branching-program friendly, so we use Construction 4.5 instantiated with the Regev-based vector-encryption scheme. Note that Construction 4.5 also yields a SNARK if we make a slightly stronger assumption about the linear-only properties of the underlying vector encryption scheme (Remark 4.9). Due to the use of the Hadamard LPCP, the prover's complexity is now quadratic in the size of the verification circuit for the previous SNARG. Since verifying CS arguments require (many) evaluation of hash functions, we conjecture (optimistically) that using circuits of size $\approx 2^{20}$ suffice.

Similar flavors of SNARK composition have been previously explored in the context of "incremental program verification," and decomposition of SNARKs for a complex computation into SNARKs for simpler computations [Val08, BCCT13]. In this setting, we compose several *different* SNARK constructions to optimize for different factors, thus allowing the resulting construction to inherit the best of many worlds.

Remark 5.18 (More Efficient Linear PCPs)**.** As noted in Remark 5.7, the simplicity of the Hadamard LPCP verification algorithm makes it a natural building block for constructing branching-program-friendly SNARGs. However, this simplicity comes at a high cost: a quadratic blowup in the prover's complexity. Suppose instead that there exists a linear PCP where the verification algorithm is of comparable efficiency as the Hadamard LPCP, but the prover overhead is *linear* (i.e., interpolating between the simplicity of the Hadamard LPCP and the (quasi)-efficiency of the QSP-based LPCP). This can potentially give a considerable reduction to our parameter estimates from above:

- **The ciphertext modulus q.** Instead of choosing q to support $\approx 2^{40}$ homomorphic operations, we can instead choose q to support $\approx 2^{20}$ homomorphic operations. This means that we can take $q \approx 2^{39}$.
- **Other parameters.** Since we are able to use a much smaller ciphertext modulus q in this setting, we can use a smaller lattice dimension $n = 760$ to obtain similar levels of security. Since the dimension of the lattice is smaller, we can also use slightly sparser secret keys and still satisfy the requirements of Equation (5.1). For instance, we can set $h_{\mathbf{S}} = 90$ and $q' \geq 780$.

If we substitute this set of parameters into the concrete estimates at the end of this section, we obtain a 20% reduction in the length of the branching program and a 68% reduction in the overall size of the branching program, which is a nontrivial concrete improvement. Thus, we leave the problem of constructing an LPCP with simple verification and small prover overhead as an open problem with important concrete implications for designing practical obfuscation.

Combining FHE decryption with SNARG verification. We now describe how to concretely implement the program in Construction 5.5 as a matrix branching program, assuming that the underlying components (FHE decryption and SNARG verification) can be represented as matrix branching programs. Let $\left(\mathsf{inp}_f, \{\mathbf{B}_{i,j}^{(\mathsf{FHE})}\}_{i\in[n_f],j\in[q_f]}\right)$ be a matrix branching program of length n_f and width q_f that implements the FHE decryption function with respect to a particular secret key

sk. In particular, on input a ciphertext $\mathbf{c}$ encrypting a bit $b \in \{0,1\}$ under sk,

$$\left(\prod_{i\in[n_f]} \mathbf{B}^{(\mathsf{FHE})}_{i,c_{\mathsf{inp}_f(i)}}\right)\mathbf{e}_{q_f} = \begin{cases}\mathbf{e}_1 & b=1\\ \mathbf{e}_2 & b=0.\end{cases}$$

Similarly, let $\left(\mathsf{inp}_s, \{\mathbf{B}^{(\mathsf{SNARG})}_{i,j}\}_{i\in[n_s],j\in[q_s]}\right)$ be a matrix branching program with length n_s and width q_s that implements the SNARG verification procedure. Specifically, on input a tuple $\mathbf{x} = (\mathbf{c}, \boldsymbol{\pi})$ consisting of a ciphertext $\mathbf{c}$ and a proof $\boldsymbol{\pi}$,

$$\left(\prod_{i\in[n_s]} \mathbf{B}^{(\mathsf{SNARG})}_{i,x_{\mathsf{inp}_s(i)}}\right)\mathbf{e}_{q_s} = \begin{cases}\mathbf{e}_1 & \boldsymbol{\pi} \text{ is a valid proof of } \mathbf{c}\\ \mathbf{e}_2 & \text{otherwise.}\end{cases}$$

For simplicity, suppose that the output dimension of both of these matrix branching programs is 2 (i.e., the outputs are standard basis vectors of a 2-dimensional space). The program in Construction 5.5 outputs 1 if and only if the SNARG verifies and the FHE ciphertext decrypts to 1. This algorithm can be implemented as a matrix branching program in the following way:

- We construct a branching program that first performs the FHE decryption and then verifies the SNARG. To combine the two computations, we first embed the output of the FHE decryption in a (q_s+1)-dimensional space. Specifically, let $\mathbf{Y} \in \{0,1\}^{(q_s+1)\times 2}$ be the matrix where $\mathbf{Y}\mathbf{e}_1 = \mathbf{e}_{q_s}$ and $\mathbf{Y}\mathbf{e}_2 = \mathbf{e}_{q_s+1}$. Then the product $\mathbf{Y}\left(\prod_{i\in[n_f]} \mathbf{B}^{(\mathsf{FHE})}_{i,c_{\mathsf{inp}_f(i)}}\right)\mathbf{e}_{q_f}$ equals $\mathbf{e}_{q_s}$ if the FHE ciphertext decrypts to 1 and $\mathbf{e}_{q_s+1}$ if the FHE ciphertext decrypts to 0.

- Next, for each $i \in [n_s]$ and $j \in [q_s]$, define matrices $\mathbf{C}^{(\mathsf{SNARG})}_{i,j}$ as a block diagonal matrix with $\mathbf{B}^{(\mathsf{SNARG})}_{i,j}$ in the upper left-hand corner and 1 in the lower right-hand:

  $$\mathbf{C}^{(\mathsf{SNARG})}_{i,j} = \begin{pmatrix}\mathbf{B}^{(\mathsf{SNARG})}_{i,j} & 0\\ 0 & 1\end{pmatrix}.$$

  Consider the product

  $$\mathbf{z} = \left(\prod_{i\in[n_s]} \mathbf{C}^{(\mathsf{SNARG})}_{i,x_{\mathsf{inp}_s(i)}}\right)\cdot\mathbf{Y}\cdot\left(\prod_{i\in[n_f]} \mathbf{B}^{(\mathsf{FHE})}_{i,c_{\mathsf{inp}_f(i)}}\right)\mathbf{e}_{q_f}.$$

  By construction, if the FHE ciphertext decrypts to 0, then $\mathbf{z} = \mathbf{e}_3$ since the matrices $\mathbf{B}^{(\mathsf{SNARG})}_{i,j}$ are applied to the all-zeroes vector $\mathbf{0}^{q_s}$. Alternatively, if the FHE ciphertext decrypts to 1, then $\mathbf{z} = \mathbf{e}_1$ if $\boldsymbol{\pi}$ is a valid proof of the well-formedness of $\mathbf{c}$ and $\mathbf{z} = \mathbf{e}_2$ otherwise. Computing the output can be done by introducing an additional bookend vector $(\mathbf{e}_2 + \mathbf{e}_3)$ and outputting 1 if $(\mathbf{e}_2+\mathbf{e}_3)^\top\mathbf{z} = 0$, and 0 otherwise. In other words, the output is 1 if and only if $\mathbf{z} = \mathbf{e}_1$.

- As an optimization, we will construct our matrix branching programs over a composite-order ring ($\mathbf{Z}_N \cong \mathbf{Z}_{q_1}\times\cdots\times\mathbf{Z}_{q_m}$) and leverage the CRT embedding described in Section 5.3 to parallelize the SNARG verification. In our case, we decompose the SNARG verification into many independent sub-verifications and verification succeeds if and only if all of the sub-verifications pass: $\mathbf{z} = \mathbf{e}_1 \pmod{p_i}$ for all $i \in [m]$). If any of the sub-computations fails, then for some $i \in [m]$, $\mathbf{z} = \mathbf{e}_2 \pmod{q_i}$ for some i, in which case $(\mathbf{e}_2+\mathbf{e}_3)^\top\mathbf{z} \neq 0 \pmod{q_i}$. Thus, $(\mathbf{e}_2+\mathbf{e}_3)^\top\mathbf{z} \neq 0 \in \mathbb{Z}_N$, and the branching program outputs 0.

Given matrix branching programs that implement FHE decryption and the SNARG verification, we can essentially concatenate them to obtain a matrix branching program for the bootstrapping program in Construction 5.5. The length of the resulting program is the sum of the length of the underlying programs while the dimension of the underlying matrices increases slightly.

The overall complexity. Having estimated the parameters needed for FHE decryption and our SNARG construction, we can now derive estimates on the length and size of the (generalized) matrix branching programs needed to implement the bootstrapping program in Construction 5.5. For our estimates, we assume that the inputs to the obfuscated program is $\lambda = 80$ bits (e.g., an input to a PRF).

- **FHE Decryption:** FHE decryption requires computing a rounded inner product between two vectors of dimension $n_{\mathsf{FHE}} = 500$ over a modulus $q_{\mathsf{FHE}} = 2^9$ (based on concrete estimates in [DM15]). Using the construction described in Section 5.3, this can be done using a matrix branching program of length n_{FHE} and width q_{FHE}.
- **SNARG Verification:** SNARG verification consists of two steps: decrypting the prover's responses and applying the Hadamard verification procedure to each of the $\ell/3$ independent instances of the Hadamard LPCP. Verification succeeds if and only if each of the underlying verifications succeed. Both of these steps can be parallelized using the CRT trick described in Section 5.3. Moreover, the decision algorithm for the Hadamard PCP (Equation B.1) consists of two independent checks (Construction B.1), so these operations can also be partially parallelized. The concrete estimates can be determined as follows:
  - Verifying the first relation of Equation B.1 requires decrypting one entry of the encrypted vector followed by computing an inner product with the ciphertext vector. Decryption consists of a rounded inner product between two vectors of dimension $n_{\mathsf{SNARG}} + \ell = 1175 + 360 = 1535$ over a modulus of size $q_{\mathsf{SNARG}} = 2^{10}$. Computing the inner product between the bits of the statement (the input to the program and the FHE ciphertext) with the secret verification state can be done using a branching program of length $\lambda + n_{\mathsf{FHE}} = 580$ over a much smaller field (the plaintext field $\mathbb{Z}_p$ for the SNARG system).
  - Verifying the second relation of Equation B.1 requires decrypting two entries of the encrypted vector and checking that one is the square of the other. This can be done using a branching program of length $2(n_{\mathsf{SNARG}} + \ell) = 3070$. It is necessary to store the first decrypted value in order to compare against the second. Thus, for the first decryption (the first 1535 matrices), we use matrices of dimension $q_{\mathsf{SNARG}} \times q_{\mathsf{SNARG}}$. For the second decryption, we use matrices of dimension $(p \cdot q_{\mathsf{SNARG}}) \times (p \cdot q_{\mathsf{SNARG}})$ where the extra factor of p corresponds to the possible values from the first decryption.
  - These checks can be implemented in parallel, provided that on any particular step, the matrix branching program inspects the same input value. Thus, the decryption in the first relation can be evaluated in parallel with the first decryption in the second relation. The length of the overall branching program that implements the SNARG verification is given by $2(n_{\mathsf{SNARG}} + \ell) + \lambda + n_{\mathsf{FHE}} = 3650$.

Putting everything together, the overall length of the branching program is 500 for the FHE decryption and 3650 for the SNARG verification. Thus, the overall length of the branching program

needed to implement the bootstrapping functionality of Construction 4.5 is roughly 4150. The number of components in the matrix branching program can be estimated as follows:

- The FHE decryption consists of a matrix branching program of length n_{FHE} and uniform width q_{FHE}, so the total number of components is $n_{\mathsf{FHE}} \cdot (q_{\mathsf{FHE}})^3 \approx 2^{36.0}$. The cubic factor in q_{FHE} arises from the fact that each matrix in the branching program is $q_{\mathsf{FHE}} \times q_{\mathsf{FHE}}$, and there are q_{FHE} possible matrices in each input position.

- Verifying the first SNARG relation requires decrypting an entry in the encrypted proof vector followed by computing an inner product modulo $p = 3$ with the components of the ciphertext vector. The number of encodings required is $(n_{\mathsf{SNARG}} + \ell)(q_{\mathsf{SNARG}})^3$ for the decryption and at most $(\lambda + n_{\mathsf{FHE}}) \cdot q_{\mathsf{FHE}} \cdot p^2$ for the inner product computation. The total number of entries comes out to be $\approx 2^{40.6}$.

- Verifying the second SNARG relation requires two decryption operations. The first one is performed in parallel with the decryption for the first check. The remaining decryption query requires an additional $(n_{\mathsf{SNARG}} + \ell) \cdot q_{\mathsf{SNARG}} \cdot (p \cdot q_{\mathsf{SNARG}})^2$ encodings (the extra factor of p^2 arises from the fact that the branching program needs to "remember" the output of the first decryption). The number of entries is then $\approx 2^{43.8}$.

Summing together the different contributions and taking into account the padding needed to combine the FHE decryption with the SNARG verification, the size of the final matrix branching program needed to implement the bootstrapping functionality of Construction 5.5 is about 2^{44}. Thus, if we now apply the obfuscation candidate of [BMSZ16] (instantiated appropriately with a composite-order multilinear map [CLT13]), we can obfuscate the bootstrapping functionality in Construction 5.5. The underlying multilinear map needs to support approximately 4150 levels of multilinearity and the description of the obfuscated program contains roughly 2^{44} multilinear map encodings.

Overhead of other building blocks. While the program in Construction 5.5 is an "obfuscation-complete" primitive, the concrete efficiency of the resulting obfuscation scheme also depends critically on the concrete efficiency of the other building blocks, namely that of the FHE scheme as well as the intermediate SNARG schemes we rely on (Remark 5.17). We briefly discuss the overhead of these additional components.

- **FHE.** The concrete performance of fully homomorphic encryption has improved significantly in the last couple years. Using the scheme of Ducas and Micciancio [DM15], homomorphic evaluation of an NAND gate requires about half a second of computation. With more recent optimizations [CGGI16], the cost of FHE has dropped to under 0.1 seconds per gate. The main limiting factor in this scheme seems to be the large ciphertext expansion factor ($\approx 2^{18}$). Alternatively, one can also use FHE schemes that support batching (e.g., encrypting multiple messages in a single ciphertext block [SV14, BGV12, GHS12]) to obtain schemes with better (amortized) space complexity. For instance, using [BGV12], Gentry, Halevi, and Smart [GHS12] show that evaluating a PRF (i.e., AES) on 120 blocks completes in just 4 minutes on a typical desktop. Note that even though the FHE scheme used in [GHS12] does not have branching-program-friendly decryption, we can still leverage it for function evaluation via the technique described in Remark 5.16.

- **SNARG Constructions.** Our bootstrapping construction requires using a SNARG to verify correctness of an FHE evaluation, which can be a very large circuit in practice. As described in Remark 5.17, we can compose several independent SNARK constructions to avoid the quadratic overhead of the Hadamard-based SNARG construction from Section 4 we use in our bootstrapping construction (since it is the only SNARG with a branching-program-friendly verification procedure). We now examine the overhead of these SNARK constructions. The first SNARK is intended to optimize for prover efficiency, so a reasonable candidate is the QSP-based SNARK in [GGPR13] or [BCI+13]. Both candidates have been implemented [PHGR13, BCG+13, BCTV14] and have been shown to be effective in verifying a number of real-world scenarios. For instance, Ben-Sasson et al. [BCG+13] show that generating a proof for a computation with 10^7 gates requires just a couple hours of computation. With further advancements in computing power as well as new optimizations, this step is unlikely to be the bottleneck in our candidate obfuscation construction.

  The second SNARK construction needed for the composition in Remark 5.17 is designed to minimize the circuit complexity of the verifier. For this, one reasonable candidate is to use Micali's CS proofs [Mic00]. The primary bottleneck here is the overhead needed to encode the computation as a PCP. While we are not aware of any existing implementations that employ general-purpose PCPs (since these types of solutions are typically less efficient compared to alternative approaches), recent work by Ben-Sasson et al. [BBGR16] show that for certain parameter settings, PCPs can become practically viable and even competitive with other techniques. We remark here that our particular goal of minimizing the *circuit complexity* of SNARK verification is not one of the standard objectives in the design of SNARKs. In any case, the recent improvements in the concrete efficiency of PCPs suggest that this step too is not entirely out of the question.

Since existing work suggest that the other components in our candidate obfuscation construction are implementable, the primary bottleneck in our construction is likely to be the 4150-degree multilinear map. Since the study of multilinear maps is still in its infancy, we are optimistic that future candidates of multilinear maps with improved efficiency will provide a viable route towards general-purpose program obfuscation.

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, the Simons Foundation, and an NSF Graduate Research Fellowship. 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. 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

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *TCC*, 2015.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, 2009.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *ACM CCS*, 2014.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. In *FOCS*, 1992.

[AP13] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In *CRYPTO*, 2013.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, 2014.

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In *ASIACRYPT*, 2014.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3), 2015.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT*, 2017.

[Ban95] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in r^n. *Discrete & Computational Geometry*, 13(2), 1995.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *STOC*, 1986.

[BBC+17] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In *EUROCRYPT*, 2017.

[BBGR16] Eli Ben-Sasson, Iddo Bentov, Ariel Gabizon, and Michael Riabzev. Improved concrete efficiency and security analysis of Reed-Solomon PCPPs. *Electronic Colloquium on Computational Complexity (ECCC)*, 23, 2016.

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

[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*, 2014:580, 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.

[BCD+16] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. *IACR Cryptology ePrint Archive*, 2016, 2016.

[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*, pages 781–796, 2014.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, 2001.

[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.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *EUROCRYPT*, 2014.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

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

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *STOC*, 2000.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *EUROCRYPT*, 2016.

[BP04a] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, 2004.

[BP04b] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, 2004.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *TCC*, 2016.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, 2014.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, 2012.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1), 2003.

[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.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO*, 2014.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachéne. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT*, 2016.

[CL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP proofs from an extractability assumption. In *4th Conference on Computability*, 2008.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, 2013.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, 2011.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT*, 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*, pages 532–550, 2014.

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

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

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, 2015.

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

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[GGH+13b] 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.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, 2015.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, 2014.

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

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

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, 2010.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier cs-proofs. *IACR Cryptology ePrint Archive*, 2011, 2011.

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

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

[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.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

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

[HHSS17] Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing bp-obfuscation using graph-induced encoding. *IACR Cryptology ePrint Archive*, 2017:104, 2017.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *CCC*, pages 278–291, 2007.

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

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS*, 2000.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In *CRYPTO*, 2015.

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

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, 2015.

[Lin16a] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *EUROCRYPT*, 2016.

[Lin16b] Huijia Lin. Indistinguishability obfuscation from DDH on 5-linear maps and locality-5 prgs. *IACR Cryptology ePrint Archive*, 2016, 2016.

[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*, pages 41–60, 2013.

[Lip16] Helger Lipmaa. Prover-efficient commit-and-prove zero-knowledge SNARKs. In *AFRICACRYPT*, pages 185–206, 2016.

[LMA+16] Kevin Lewi, Alex J. Malozemoff, Daniel Apon, Brent Carmer, Adam Foltzer, Daniel Wagner, David W. Archer, Dan Boneh, Jonathan Katz, and Mariana Raykova. 5Gen: A framework for prototyping applications using multilinear maps and matrix branching programs. In *ACM CCS*, 2016.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In *CT-RSA*, 2013.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

[LV16] Rachel Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In *FOCS*, 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):343–363, 2008.

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

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

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

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

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

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66, 1994.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, 1994.

[Sim97] Daniel R. Simon. On the power of quantum computation. *SIAM J. Comput.*, 26(5), 1997.

[SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1), 2014.

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

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

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *EUROCRYPT*, 2015.

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

A Succinct Non-Interactive Arguments

We review the definition of succinct non-interactive argument (SNARG) systems. For clarity of exposition, we present our definitions for Boolean circuit satisfaction problems. Given a Boolean circuit $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, the Boolean circuit satisfaction problem is defined by the relation $\mathcal{R}_C = \big\{(x,w) \in \{0,1\}^n \times \{0,1\}^h : C(x,w) = 1\big\}$. We often refer to $x \in \{0,1\}^n$ as the statement and $w \in \{0,1\}^h$ as the witness. We write $\mathcal{L}_C$ to denote the language (the set of statements $x \in \{0,1\}^n$ where $(x,w) \in \mathcal{R}_C$ for some $w \in \{0,1\}^h$). For a family of Boolean circuits $\mathcal{C} = \big\{C_\ell : \{0,1\}^{n(\ell)} \times \{0,1\}^{h(\ell)} \to \{0,1\}\big\}_{\ell \in \mathbb{N}}$ indexed by a parameter ℓ, we write $\mathcal{R}_{\mathcal{C}} = \bigcup_{\ell \in \mathbb{N}} \mathcal{R}_{C_\ell}$ and $\mathcal{L}_{\mathcal{C}} = \bigcup_{\ell \in \mathbb{N}} \mathcal{L}_{C_\ell}$ for the corresponding (infinite) relation and language, respectively.

Definition A.1 (Succinct Non-Interactive Arguments)**.** Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of circuits indexed by a parameter ℓ. A succinct non-interactive argument (SNARG) for the relation $\mathcal{R}_{\mathcal{C}}$ is a tuple $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ of three algorithms defined as follows:

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

Moreover, Π_{SNARG} satisfies the following properties:

- **Completeness:** For all $(x,w) \in \mathcal{R}_{\mathcal{C}}$,

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

- **Soundness:** Depending on the notion of soundness:
  - **Adaptive Soundness:** For every polynomial-size prover P^*,

$$\Pr[(\sigma,\tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); (x,\pi) \leftarrow P^*(\sigma) : \mathsf{Verify}(\tau, x, \pi) = 1 \text{ and } (x,w) \notin \mathcal{R}_{\mathcal{C}} \text{ for all } w] = \mathrm{negl}(\lambda).$$

– **Non-adaptive Soundness:** For every polynomial-size prover P^*, and all statements $x \notin \mathcal{L}_{\mathcal{C}}$,

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

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

Before proceeding, we give some intuition on the different soundness and succinctness notions. A SNARG is *adaptive* if the prover can choose the statement after seeing the reference string σ; otherwise, it is *non-adaptive*. Next, a SNARG is *fully-succinct* if the setup algorithm is efficient (runs in time polylogarithmic in the size of the circuit); otherwise, the SNARG is a *preprocessing* SNARG. For both fully-succinct as well as preprocessing SNARGs, the verifier's runtime and the length of the proof grow polylogarithmically in the size of the underlying circuit.

Public vs. designated verifiability. A SNARG is *publicly verifiable* if the verification state τ is allowed to be public. Alternatively, a *designated-verifier* SNARG is one where security only holds if τ remains secret. In this work, we focus exclusively on constructing designated-verifier SNARGs. In Section 5.2, we show that designated-verifier SNARGs suffice for our primary application of obtaining an efficient general-purpose program obfuscator.

Multi-theorem SNARGs. A useful property for SNARGs to have is the ability to *reuse* the same reference string σ for multiple proofs. This is particularly important in the case of preprocessing SNARGs where an expensive precomputation stage is needed to construct the reference string. This multi-theorem setting can be modeled by imposing a stronger requirement where soundness should hold even if the adversary has access to a proof verification oracle. While this stronger soundness requirement follows immediately if the SNARG system is publicly verifiable, the same is not true in the designated-verifier setting. In fact, by issuing carefully crafted queries to the proof verification oracle, a dishonest prover can potentially learn information about the secret verification state, and thus, compromise the soundness of the SNARG system.

In this work, we construct a compiler that combines an information-theoretic primitive (linear PCPs) with a cryptographic primitive (linear-only vector encryption) to obtain a designated-verifier SNARG. Correspondingly, we address this core issue of reusability at both the information-theoretic level (via a stronger soundness definition) as well as at the cryptographic level (via a stronger notion of linear-only encryption). We give more detail sin Section 3 and Appendix C.

Other properties. In addition to the basic properties outlined above, there are numerous additional notions that pertain to SNARGs. In some applications, the soundness requirement is further strengthened to an extractability property—that is, whenever a prover is able to convince the verifier that a statement x is in the language, there is also an (efficient) extraction algorithm

that is able to extract a witness w for x such that $(x, w) \in \mathcal{R}$. This gives rise to succinct arguments of knowledge (SNARKs).

Another commonly considered notion is zero-knowledge, which roughly states that an honest prover can generate a valid proof for any true statement without revealing any information about the witness or how the proof is generated. As shown by Bitansky et al. [BCCT12], preprocessing SNARKs can be combined with (possibly non-succinct) non-interactive arguments of knowledge (NIZK arguments) to obtain zero-knowledge SNARKs (i.e., "zkSNARKs") in the preprocessing model. We also refer to [Gro10, Lip12, PHGR13, BCG+13, GGPR13, BCI+13, Lip13, BCTV14, DFGK14, Lip16] for constructions and implementations of zero-knowledge SNARKs. Since we do not require these additional notions in this work, we do not discuss them further in this paper.

B A Linear PCP from the Walsh-Hadamard Code

In this section, we review a simple linear PCP due to Arora et al. [ALM+92] which is based on the Walsh-Hadamard code. Our presentation here is largely taken from [BCI+13, Appendix A.1]. Similar to [BCI+13], we describe our variant of the ALMSS construction for a relation $\mathcal{R}(\mathbf{x}, \mathbf{w})$ in terms of an arithmetic circuit over a field $\mathbb{Z}_p$ (for some $p > 2$). We say an arithmetic circuit $C : \mathbb{Z}_p^n \times \mathbb{Z}_p^h \to \mathbb{Z}_p^\ell$ is satisfied on an input $(\mathbf{x}, \mathbf{w}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^h$ if $C(\mathbf{x}, \mathbf{w}) = 0^\ell$. The problem of Boolean circuit satisfiability is reducible to arithmetic circuit satisfiability with only constant overhead. We refer to [BCI+13, Claim A.2] for additional details.

Construction. We now describe the LPCP construction for an arithmetic circuit $C : \mathbb{Z}_p^n \times \mathbb{Z}_p^h \to \mathbb{Z}_p^\ell$ of size s. Let z_i denote the value of the i^{th} wire of C on an input $\mathbf{x} \in \mathbb{Z}_p^n$ and a witness $\mathbf{w} \in \mathbb{Z}_p^h$. The wires z_i for $i \in [n]$ correspond to the values of the input wires, and the wires $z_{s-i+1} = 0$ for $i \in [\ell]$ correspond to the values of the output wires. An honestly generated proof consists of a vector $\boldsymbol{\pi} \in \mathbb{Z}_p^{s+s^2}$ whose components are the values $z_1, \ldots, z_s \in \mathbb{Z}_p$ and the products $z_i z_j \in \mathbb{Z}_p$ for all $i, j \in [s]$. If we define $\mathbf{z} = [z_1, \ldots, z_s]$, then we can write $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$. To verify the proof, the verifier performs the following four consistency checks:

1. **Consistency of $\boldsymbol{\pi}$:** each entry $z_i z_j$ is the product of the corresponding z_i and z_j.
2. **Input consistency:** $z_i = x_i$ for all $i \in [n]$.
3. **Output consistency:** $z_{s-i} = 0$ for all $i \in [\ell]$.
4. **Gate consistency:** For each gate, the value of the output wire is consistent with the value of its input wires. In particular, for addition gates with input wires $i_1, \ldots, i_m$ and output wire k, this corresponds to checking that $\left(\sum_{j=1}^m z_{i_j}\right) - z_k = 0$. For the multiplication gates with input wires i, j and output wire k, this corresponds to checking that $z_i z_j - z_k = 0$.

To verify the first condition, the verifier requests a random linear combination of the z_i's and a corresponding linear combination of the product terms $z_i z_j$'s whose sum corresponds to the square of the first query. The verifier then checks that the square of the oracle's response to the first query matches the oracle's response to the second query. By the Schwartz-Zippel lemma, an inconsistent proof $\boldsymbol{\pi}$ satisfies this quadratic relation with probability at most $2/p$.

The remaining three consistency checks can be performed by taking a random linear combination of the left-hand side of the relations (a function in the z_i's) and comparing against the corresponding

combination of the right-hand side of the relations (a function of the x_i's). If $\boldsymbol{\pi}$ is consistent, we conclude via the Schwartz-Zippel lemma that the verifier will reject a inconsistent wire assignment with probability at least $1/p$.

Construction B.1 (Linear PCP from the Walsh-Hadamard Code)**.** Let $C : \mathbb{Z}_p^n \times \mathbb{Z}_p^h \to \mathbb{Z}_p^\ell$ be an arithmetic circuit of size s over $\mathbb{Z}_p$. The prover and verifier algorithms are then defined as follows:

- **Prover's Algorithm P_{LPCP}:** On input $(\mathbf{x}, \mathbf{w}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^h$ where $C(\mathbf{x}, \mathbf{w}) = 1$, the prover computes the values $z_1, \ldots, z_s$ for the wires of C. It sets $\mathbf{z} = [z_1, \ldots, z_s] \in \mathbb{Z}_p^s$ and outputs the proof $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}] \in \mathbb{Z}_p^{s+s^2}$.

- **Verifier's Query Algorithm Q_{LPCP}:** The query algorithm Q_{LPCP} has hard-coded inside it a matrix $\mathbf{A}_C \in \mathbb{Z}_p^{s \times (s+s^2)}$ and a vector $\mathbf{b}_C \in \mathbb{Z}_p^{s-n}$. Both $\mathbf{A}_C$ and $\mathbf{b}_C$ can be efficiently computed (in linear time) from the circuit C. These correspond precisely to the consistency checks enumerated above. Then, the query algorithm outputs a tuple of three query vectors $(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$, where $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3 \in \mathbb{Z}_p^{s+s^2}$, are constructed as follows:

  1. Sample $\mathbf{u} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^s$. Let $\mathbf{u}_\mathbf{x} \in \mathbb{Z}_p^n$ be the first n components of $\mathbf{u}$ and let $\mathbf{u}_C \in \mathbb{Z}_p^{s-n}$ be the last $s - n$ components of $\mathbf{u}_i$. Let $\mathbf{q}_1 = \mathbf{u} \cdot \mathbf{A}_C$.
  2. Sample $\mathbf{v} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^s$. Set $\mathbf{q}_2 = [\mathbf{v}, 0^{s^2}]$ and $\mathbf{q}_3 = [0^s, \mathbf{v} \otimes \mathbf{v}]$.

  The query algorithm constructs the query matrix $\mathbf{Q} \in \mathbb{Z}_p^{3 \times (s+s^2)}$ whose columns are the query vectors $(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$. It outputs the query matrix $\mathbf{Q}$ and the state $\mathsf{st} = (u_C, \mathbf{u}_\mathbf{x})$, where $u_C = \langle \mathbf{u}_C, \mathbf{b}_C \rangle$.

- **Verifier's Decision Algorithm D_{LPCP}:** Let $a_1, a_2, a_3 \in \mathbb{Z}_p$ be the prover's responses to the verifier's query. The verifier accepts if

$$a_1 - u_C + \langle \mathbf{u}_\mathbf{x}, \mathbf{x} \rangle = 0 \qquad \text{and} \qquad a_2^2 - a_3 = 0. \tag{B.1}$$

Lemma B.2. *Let $C : \mathbb{Z}_p^n \times \mathbb{Z}_p^h \to \mathbb{Z}_p^\ell$ be an arithmetic circuit. Then, Construction B.1 is a 3-query LPCP for $\mathcal{R}_C$ with strong soundness error $2/p$ against affine provers and query length $s + s^2$.*

Proof. First, we show that Construction B.1 satisfies strong soundness against linear provers. Take any input $\mathbf{x} \in \mathbb{Z}_p^s$ and any proof $\boldsymbol{\pi}^* \in \mathbb{Z}_p^{s+s^2}$. Let $\mathbf{z} \in \mathbb{Z}_p^s$ denote the first s components of $\boldsymbol{\pi}^*$. We consider two possibilities:

- The proof $\boldsymbol{\pi}^*$ is inconsistent, namely $\boldsymbol{\pi}^* \neq [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$. In this case, recall the second verification condition

$$\langle \boldsymbol{\pi}^*, \mathbf{q}_2 \rangle^2 - \langle \boldsymbol{\pi}^*, \mathbf{q}_3 \rangle = a_2^2 - a_3 \stackrel{?}{=} 0.$$

  By construction of $\mathbf{q}_2$ and $\mathbf{q}_3$, the left-hand side of this expression can be viewed as a quadratic test polynomial $\mathbf{t}_{\boldsymbol{\pi}^*}(\mathbf{v})$ in the random test vector $\mathbf{v} \in \mathbb{Z}_p^s$ chosen by the verifier. Since $\boldsymbol{\pi}^*$ does not correspond to a consistent assignment, the polynomial $\mathbf{t}_{\boldsymbol{\pi}^*}$ is not the identically zero polynomial in $\mathbf{v}$. By the Schwartz-Zippel lemma, for $\mathbf{v} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^s$, the value $\mathbf{t}_{\boldsymbol{\pi}^*}(\mathbf{v})$ is zero with probability at most $2/p$. In this case, the verifier accepts with probability at most $2/p$.

- The proof $\boldsymbol{\pi}^*$ is consistent, namely $\boldsymbol{\pi}^* = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$ for some assignment $\mathbf{z} \in \mathbb{Z}_p^s$. In this case, the second verification condition is satisfied. We just consider the first verification condition. We express the first verification condition as a linear polynomial $\mathbf{t}'_{\boldsymbol{\pi}^*}(\mathbf{u})$ in the random test vector $\mathbf{u} \in \mathbb{Z}_p^s$ chosen by the verifier. There are two cases to consider. Suppose $\mathbf{z}$ corresponds to a valid assignment to all of the wires of the arithmetic circuit C on input $\mathbf{x} \in \mathbb{Z}_p^n$ and some witness $\mathbf{w} \in \mathbb{Z}_p^h$ where $C(\mathbf{x}, \mathbf{w}) = 0^\ell$. In this case, $\mathbf{t}_{\boldsymbol{\pi}^*}$ is the identically zero polynomial over $\mathbb{Z}_p$, and so, the first verification condition is satisfied for all $i \in [\kappa]$, and the verifier accepts with probability 1.

  Otherwise, suppose $\mathbf{z}$ corresponds to an inconsistent labeling of the wires of C or to a consistent labeling for an input $(\mathbf{x}', \mathbf{w}')$ where either $\mathbf{x} \neq \mathbf{x}'$ or $C(\mathbf{x}, \mathbf{w}') \neq 0^\ell$. By construction, $\mathbf{t}'_{\boldsymbol{\pi}^*}$ is not the identically zero polynomial over its input $\mathbf{u}$, so by the Schwartz-Zippel lemma, $\mathbf{t}'_{\boldsymbol{\pi}^*}(\mathbf{u}) = 0$ with probability at most $1/p$. Thus, the verifier accepts with probability at most $1/p$.

This suffices to show that Construction B.1 satisfies strong soundness against linear provers with soundness error $2/p$. To conclude, we show that the same construction actually enjoys strong soundness against affine provers. Consider an affine strategy $\boldsymbol{\Pi}^* = (\boldsymbol{\pi}^*, \mathbf{b}^*)$ where $\boldsymbol{\pi}^* \in \mathbb{Z}_p^{s^2+s}$ and $\mathbf{b}^* \in \mathbb{Z}_p^3$. We just consider the case where $\mathbf{b}^* \neq \mathbf{0}$.

- Suppose $b_2^* \neq 0$ or $b_3^* \neq 0$. Then the second verification condition becomes

$$(\langle \boldsymbol{\pi}^*, \mathbf{q}_2 \rangle + b_2)^2 - \langle \boldsymbol{\pi}^*, \mathbf{q}_3 \rangle - b_3 \stackrel{?}{=} 0.$$

  Writing $\mathbf{q}_2$ and $\mathbf{q}_3$ in terms of $\mathbf{v}$, and assuming that at least one of b_2^*, b_3^* is non-zero, this is then a non-zero polynomial in $\mathbf{v}$ for all $\boldsymbol{\pi}^* \in \mathbb{Z}_p^{s^2+s}$. By Schwartz-Zippel, this check passes with probability at most $2/p$.

- Suppose $b_1^* \neq 0$. Then the first verification condition becomes

$$\langle \boldsymbol{\pi}^*, \mathbf{q}_1 \rangle + b_1 - u_C + \langle \mathbf{u}_\mathbf{x}, \mathbf{x} \rangle \stackrel{?}{=} 0.$$

  Now, $\mathbf{q}_1$, $u_\mathbf{C}$ and $\mathbf{u}_\mathbf{x}$ are all linear functions of $\mathbf{u}$ (no affine term), so this is necessarily a non-zero polynomial in $\mathbf{u}$. By Schwartz-Zippel, the verifier accepts with probability at most $1/p$.

We have shown that in each case, either the verifier accepts with probability 1, or it accepts with probability $2/p$. This suffices to show strong soundness against affine provers with soundness error $2/p$. □

Remark B.3 (Hadamard LPCP and Affine Provers)**.** The proof of Lemma B.2 shows that the verifier in the Hadamard LPCP rejects all non-linear strategies with probability at least $1 - 2/p$. This property is useful when we use the Hadamard LPCP as a building block to construct statistically sound LPCPs satisfying relaxed notions of soundness against affine provers (e.g., Definition 5.10 and Theorem 5.12).

C Multi-Theorem Designated Verifier SNARGs

In Theorem 4.6 of Section 4.1, we showed how a vector encryption scheme satisfying linear targeted malleability can be combined with a linear PCP with soundness against affine provers to obtain a single-theorem, designated-verifier SNARG. In this section, we introduce a stronger notion of linear-only encryption that can be combined with linear PCPs satisfying strong soundness against affine provers to obtain multi-theorem SNARGs. However, as noted in Section 4.1, it is interesting to see whether the simpler definition of linear targeted malleability together with strongly sound LPCPs already suffices in the multi-theorem setting. We begin by formally introducing the notion of a multi-theorem SNARG.

Definition C.1 (Adaptive, Multi-Theorem SNARG)**.** Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of circuits indexed by a parameter ℓ. Let $\Pi_{\mathsf{SNARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a SNARG for the relation $\mathcal{R}_{\mathcal{C}}$. Then, Π_{SNARG} is an adaptive, multi-theorem SNARG if for all polynomial-size provers P^*, the following holds:

$$\Pr[(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); (x, \pi) \leftarrow (P^*)^{\mathsf{Verify}(\tau, \cdot, \cdot)}(\sigma) : \\ \mathsf{Verify}(\tau, x, \pi) = 1 \text{ and } (x, w) \notin \mathcal{R}_{\mathcal{C}} \text{ for all } w] = \mathrm{negl}(\lambda).$$

In other words, soundness should hold even if the prover has access to a proof verification oracle $\mathsf{Verify}(\tau, \cdot, \cdot)$.

Next, we introduce a stronger notion of linear-only encryption that can be used to obtain a multi-theorem SNARG (via the same construction as Construction 4.5). In order to show a SNARG system is multi-theorem, we need a way to simulate the prover's queries to the verification oracle. In past works [BP04a, BP04b, BCI+13] this has been handled by defining an interactive extractability assumption. Here, we extend the definition in [BCI+13, Definition C.6] to the vector case.

Definition C.2 (Linear-Only with Interactive Extraction [BCI+13])**.** Fix a security parameter λ. Let $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a secret-key vector encryption scheme where the message space consists of vectors of $\mathbb{Z}_p$-elements. Let $\mathcal{M}$ be a message generation algorithm that on input 1^ℓ, outputs a sequence of vectors in $\mathbb{Z}_p^\ell$. Let $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$ be some auxiliary input. We define the interactive linear-only extraction game between $\mathcal{A}$ and $\mathcal{E}$ as follows:

1. **Setup phase:**
   - $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$
   - $(\mathbf{v}_1, \ldots, \mathbf{v}_m) \leftarrow \mathcal{M}(1^\ell)$
   - $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for all $i \in [m]$
2. **Query phase:** for all $i \in [q]$ where $q = \mathrm{poly}(\lambda)$:
   - $\mathsf{ct}'_i \leftarrow \mathcal{A}(\mathsf{ct}_1, \ldots, \mathsf{ct}_m; e_1, \ldots, e_{i-1}; z)$
   - $\boldsymbol{\Pi}_i \leftarrow \mathcal{E}(\mathsf{ct}_1, \ldots, \mathsf{ct}_m; i; z)$ where $\boldsymbol{\Pi}_i$ is either an affine function $(\boldsymbol{\pi}_i, \mathbf{b}_i)$ or $\perp$ where $\boldsymbol{\pi}_i \in \mathbb{Z}_p^m$ and $\mathbf{b} \in \mathbb{Z}_p^\ell$

We say that $\mathcal{A}$ wins the game if one of the following conditions hold:

- For some $i \in [q]$, $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) = \perp$ but $\boldsymbol{\Pi}_i \neq \perp$.

- For some $i \in [q]$, $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) \neq \perp$ and either $\mathbf{\Pi}_i = \perp$ or $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) \neq \mathbf{a}_i$ where $\mathbf{\Pi}_i = (\boldsymbol{\pi}_i, \mathbf{b}_i)$ and $\mathbf{a}_i = [\mathbf{v}_1 | \mathbf{v}_2 | \cdots \mathbf{v}_m] \cdot \boldsymbol{\pi} + \mathbf{b}$

Finally, we say that Π_{venc} satisfies *linear-only with interactive extraction* if for all polynomial-size interactive adversaries $\mathcal{A}$, there exists a polynomial-size interactive extractor $\mathcal{E}$ such that for any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$, any plaintext generation algorithm $\mathcal{M}$, the probability that $\mathcal{A}$ wins the above interactive linear-only extraction game is negligible.

We note state the corresponding theorem that applying Construction 4.5 to a linear PCP with strong soundness against affine provers and a vector encryption scheme satisfying the stronger notion of linear-only encryption with interactive extraction suffices to construct a multi-theorem SNARG. The proof follows analogously to the proof in [BCI$^+$13, Lemma C.8].

Theorem C.3 (Multi-Theorem SNARG [BCI$^+$13, adapted])**.** *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be a linear PCP that is strongly sound against affine provers. Let $\Pi_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a linear-only encryption scheme that satisfies linear-only with interactive extraction (Definition C.2). Then, applying Construction 4.5 to $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ and Π_{venc} yields an adaptive, multi-theorem SNARG in the preprocessing model. Moreover, if $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ satisfies* strong knowledge *against affine provers, then applying Construction 4.5 to $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ and Π_{venc} yields an adaptive, multi-theorem SNARK in the preprocessing model.*

Remark C.4 (Double-Encryption.)**.** As noted in [BCI$^+$13, Remark 5.6], encryption schemes that allow for "oblivious sampling" of ciphertexts cannot satisfy Definition C.2. To potentially satisfy Definition C.2, it should be the case that the set of strings c where $\mathsf{Decrypt}(\mathsf{sk}, c) \neq \perp$ should be sparse (i.e., an adversary cannot simply sample a random string c that successfully decrypts to a valid vector). Certainly, the vector encryption scheme from Section 4.2 does not satisfy this property since $\mathsf{Decrypt}(\mathsf{sk}, c) \neq \perp$ for all strings c in the ciphertext space.

A heuristic method to prevent oblivious sampling is to "sparsify" the ciphertext space using "double-encryption" [GGPR13, BCI$^+$13]. Specifically, an encryption of a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$ consists of two encryptions $(\mathsf{ct}_1, \mathsf{ct}_2)$ of $\mathbf{v}$ under two independent secret keys sk_1 and sk_2, respectively. The secret key for the vector encryption includes the secret keys of both underlying schemes. During decryption, ct_1 and ct_2 are decrypted using sk_1 and sk_2, respectively. The decryption algorithm outputs $\perp$ if ct_1 and ct_2 do not decrypt to the same value; otherwise, the output is $\mathsf{Decrypt}(\mathsf{sk}_1, \mathsf{ct}_1)$. Intuitively, if the adversary was to sample random vectors from the ciphertext space, with overwhelming probability, the two ciphertexts will not decrypt to the same vector. We conjecture that this modified version of Regev-based vector encryption satisfies the stronger notion of linear-only encryption as defined in Definition C.2. Thus, with a factor of 2 overhead, we can obtain multi-theorem SNARGs (resp., SNARKs) based on any LPCP satisfying strong soundness (resp., knowledge) against affine provers.