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

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

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

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 give the first 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.

*The full version of this paper is available from `https://crypto.stanford.edu/people/dwu4/snargs.html`.

1 Introduction

Interactive proofs systems [49] 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 *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 [53] gave the first succinct four-round interactive argument system for NP based on collision-resistant hash functions and probabilistically-checkable proofs (PCPs). Subsequently, Micali [63] showed how to convert Killian's four-round argument into a single-round argument for NP by applying the Fiat-Shamir heuristic [38]. Micali's "computationally-sound proofs" (CS proofs) is the first candidate construction of a *succinct non-interactive argument* (i.e., a "SNARG" [46]) 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 $\text{negl}(\lambda)$ 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 [15] (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 [72, 73].

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. [12]. 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. [41] gave the first candidate construction of $i\mathcal{O}$ for general circuits based on multilinear maps [24, 39, 33, 43].

Since the breakthrough result of Garg et al., there has been a flurry of works showcasing the power of $i\mathcal{O}$ [41, 69, 25, 40, 19]. However, in spite of the numerous constructions and optimizations that have been developed in the last few years [11, 30, 5, 10, 74, 7], concrete instantiations of program obfuscation remain purely theoretical. Even obfuscating a relatively simple function such as the AES block cipher requires multilinear maps capable of supporting unimaginable levels of multilinearity ($\gg 2^{100}$ [74]). 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[1] 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 [33], 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 [11, 30, 5, 10, 74, 7] rely on the underlying multilinear map as a black-box. Recently, several works [57, 59, 58, 4] gave the first candidate constructions of $i\mathcal{O}$ based on *constant-degree* multilinear maps (by going through the functional encryption route introduced in [3, 20]). Even more impressively, the most recent constructions by Lin [58] as well as Ananth and Sahai [4] only require a degree-5 multilinear map, which is certainly implementable [56]. 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,

[1]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 [50, 6] where the obfuscated primitive needs to be invoked for each gate in the circuit or each step of a Turing machine evaluation.

due to the complexity of existing multilinear map constructions [39, 33], 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 [46] showed that no SNARG (for a sufficiently difficult language) can be proven secure under any "falsifiable" assumption [65]. 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 [35, 14, 64, 51, 61, 42], extractable collision-resistant hashing [15, 36], homomorphic encryption with a homomorphism extraction property [17] and linear-only encryption [18].

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 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. [18] 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 [18] 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.[2] Bitansky et al. give several

[2]Bitansky et al. [18] 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."

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 [66] as well as bilinear maps [52, 22].

Linear PCPs. Like [18], our SNARG constructions rely on linear PCPs (LPCPs). 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.

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. [18].

New compiler for preprocessing SNARGs. The preprocessing SNARGs we construct in this work enjoy several advantages over those of [18]. 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 [18] 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 [18], 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 [9] as well as those based on quadratic-span programs (QSPs) [42], 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 [18] 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 [18], and thus, avoids the soundness penalty $O(1/|\mathbb{F}|)$ incurred by the consistency check.[3] 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 [68] encryption scheme of Peikert, Vaikuntanathan, and Waters [67] 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.[4] Specifically, starting with a QSP-based LPCP [42], we obtain a SNARG with quasi-optimal succinctness. As discussed above, this is the first such SNARG that can plausibly resist quantum attacks. We note here that a direct instantiation of the construction in [18] with a Regev-based candidate for linear-only encryption yields a SNARG that is suboptimal in *both* prover complexity and proof length (Remark 4.13). 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 the full version of this paper, we further extend our techniques to obtain the first instantiation of a *quasi-optimal* SNARG for Boolean circuit satisfiability—that is, a SNARG where 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 $\text{negl}(\lambda)$ 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.

[3]This is the main difference between our approach and that taken in [18]. 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, [18] 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.

[4]While it would be preferable to obtain a construction based on the hardness of standard lattice assumptions like learning with errors (LWE) [68], the separation results of Gentry and Wichs [46] suggest that stronger, non-falsifiable assumptions may be necessary to construct SNARGs.

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 [62] 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 [41, 11, 30], which together with multilinear maps [39, 33, 43], yielded obfuscation for NC^1 (via Barrington's theorem [13]).[5] 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 [5, 10] 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 [74] as well as Applebaum and Brakerski [7] 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.

[5] Garg et al. [41] as well as Brakerski and Rothblum [30] 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 [2] 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 [31, 28, 27, 45, 2, 37] and the verification algorithm of our new lattice-based SNARG. Using a variant of the bootstrapping theorem in [30], VBB obfuscation of these two functionalities suffice for VBB obfuscation of all circuits.

We remark here that Applebaum [6] 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 [55]). 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 [31, 28, 27, 45, 2, 37] 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 [2], 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 the full version. 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.[6] 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 [31, 28, 37]. 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.
– **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 the full version. We note that when considering the public-key notion of linear-only encryption [18], 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

[6]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.

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 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 [33], 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 [2] to improve the concrete efficiency of FHE bootstrapping.

A concrete obfuscation construction. In the full version, 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[7] 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.

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

[7]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 [33], the plaintext ring already supports the requisite number of sub-rings, so using CRT for parallelization does not incur any overhead.

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.

For a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote that x is drawn uniformly at random from S. For a distribution $\mathcal{D}$, 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 $f(\lambda) = \text{negl}(\lambda)$ to denote that f is a negligible function in λ and $f(\lambda) = \text{poly}(\lambda)$ 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 [71, 75]:

Lemma 2.1 (Schwartz-Zippel Lemma [71, 75]). *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,*

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

In the full version, 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 [18].

Definition 3.1 (Linear PCPs [18]). *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 $V_{\mathsf{LPCP}}^{\boldsymbol{\pi}} = (Q_{\mathsf{LPCP}}, D_{\mathsf{LPCP}})$ 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* $V_{\mathsf{LPCP}}^{\boldsymbol{\pi}}(\mathbf{x})$ *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* $V_{\mathsf{LPCP}}^{\boldsymbol{\pi}^*}(\mathbf{x})$ *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 $\boldsymbol{\Pi} = (\boldsymbol{\pi}, \mathbf{b})$ 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* $\boldsymbol{\Pi} = (\boldsymbol{\pi}, \mathbf{b})$*, the verification algorithm* $V_{\mathsf{LPCP}}^{\boldsymbol{\Pi}}$ *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* $\boldsymbol{\Pi}^* = (\boldsymbol{\pi}^*, \mathbf{b}^*)$ *where* $\boldsymbol{\pi}^* \in \mathbb{F}^m$ *and* $\mathbf{b}^* \in \mathbb{F}^\ell$*, the probability that* $V_{\mathsf{LPCP}}^{\boldsymbol{\Pi}^*}(\mathbf{x})$ *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 [18] 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 [18]). *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 $V_{\mathsf{LPCP}}^{\boldsymbol{\pi}^*}(\mathbf{x})$ 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. [18] give two constructions of algebraic LPCPs for Boolean circuit satisfaction problems: one from the Hadamard-based PCP of Arora et al. [9], and another from the quadratic span programs (QSPs) of Gennaro et al. [42]. 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 $\varepsilon^\kappa = \text{negl}(\kappa)$.

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 $\left(P_{\mathsf{LPCP}}^{(\text{weak})}, V_{\mathsf{LPCP}}^{(\text{weak})}\right)$ be an ℓ-query linear PCP for $\mathcal{R}$, where $V_{\mathsf{LPCP}}^{(\text{weak})} = \left(Q_{\mathsf{LPCP}}^{(\text{weak})}, D_{\mathsf{LPCP}}^{(\text{weak})}\right)$. 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 $P_{\mathsf{LPCP}}^{(\text{weak})}(\mathbf{x}, \mathbf{w})$.
– **Verifier's Query Algorithm Q_{LPCP}:** The query algorithm invokes $Q_{\mathsf{LPCP}}^{(\text{weak})}$ 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 $\mathbf{Y} \xleftarrow{\text{R}} \mathbb{F}^{\kappa\ell \times \kappa\ell}$ 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 $\boldsymbol{\Pi}^* = (\boldsymbol{\pi}^*, \mathbf{b}^*)$, 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 [9] as well as the QSP-based LPCP [42]), 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 [18] 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 [18, Lemma C.3]). Bitansky et al. [18, 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. [18], 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. [9] 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. [42], 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. We conclude by describing a candidate instantiation of our linear-only vector encryption scheme using the LWE-based encryption scheme of Peikert, Vaikuntanathan, and Waters [67]. In the full version of this paper, we also 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. [18].

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 the full version, 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 [48] 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[8] 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* [23] 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. [18]. This notion already suffices for constructing a designated-verifier SNARG.

Definition 4.2 (Linear Targeted Malleability [23, 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:*

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 [23, 18], we can also define a variant of linear targeted malleability where the adversary is allowed to output

[8] 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.

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 [16]. 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), 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 the full version, 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 $\mathbb{Z}_p$.[9] 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 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.

[9] 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 [18, Claim A.2].

– $\mathsf{Verify}(\tau, \mathbf{x}, \pi)$: 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 ([18, Lemma 6.3]). *Let $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ be a linear PCP that is statistically sound against affine provers, and let $\mathit{\Pi}_{\mathsf{venc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a vector encryption scheme with linear targeted malleability. Then, applying Construction 4.5 to $(P_{\mathsf{LPCP}}, V_{\mathsf{LPCP}})$ and $\mathit{\Pi}_{\mathsf{venc}}$ yields a (non-adaptive) designated-verifier SNARG in the preprocessing model.*

Proof. Our proof is similar to the proof of [18, 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 $\mathit{\Pi}_{\mathsf{enc}}$ 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 [18, 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 the full version, we present an analog of the strengthened version of linear-only encryption from [18, 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,[10] 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 [18, Lemma 6.3, Remark 6.4].

4.2 A Candidate 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

[10]Roughly, the knowledge property states that there exists an extractor such that for every affine strategy $\boldsymbol{\Pi}^*$ 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 from [9] also satisfies this stronger knowledge property.

(Definition 4.2). In particular, we conjecture that the Regev-based encryption scheme [68] due to Peikert, Vaikuntanathan, and Waters [67, §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 $\exp(-\pi \left\|\mathbf{x}\right\|^2 / \sigma^2)$ to each element $\mathbf{x} \in \Lambda$.

Learning with errors. The learning with errors problem [68] 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 $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{m \times n}$ and error $\mathbf{e} \leftarrow \chi^n$ and outputting the pair $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. The learning with errors assumption $\mathsf{LWE}_{n,q,\chi}$ (parameterized by parameters n, q, χ) states that for all $m = \text{poly}(n)$, the LWE distribution $A_{\mathbf{s},m,\chi}$ for a randomly sampled $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ 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 [67, §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 [1, 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 [67] is defined as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Choose $\bar{\mathbf{A}} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\bar{\mathbf{S}} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell}$, 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 $\mathbf{r} \xleftarrow{\text{R}} \{0,1\}^m$ and output the ciphertext $\mathbf{c} \in \mathbb{Z}_q^{n+\ell}$ where
$$\mathbf{c} = \mathbf{A}\mathbf{r} + \begin{bmatrix} \mathbf{0}^n \\ \mathbf{v} \end{bmatrix}.$$

– $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$: 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. [8] and Brakerski et al. [29] 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 [44, 37].

Correctness. Correctness of the encryption scheme follows as in [67]. In the full version of this paper, we provide the concrete bounds on the parameters under which correctness holds. This analysis will prove useful for estimating the concrete parameters needed to instantiate our candidate obfuscation scheme in Section 5.

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. In the full version, 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 [67, Section 7.2.1] for the formal analysis.

Theorem 4.11 (Semantic Security [67]). *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 4.12. *The PVW vector encryption scheme Π_{venc} from Section 4.2 satisfies linear targeted malleability (Definition 4.2).*

Under Conjecture 4.12, 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 $q = \text{poly}(\lambda)$.[11] 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. [9] 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. [42], 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 $\widetilde{O}(\lambda^2 + \lambda n)$.

  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

[11]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 [21, 54] or BKZ-based attacks [70, 32]. Based on our current understanding [60, 32, 54, 26], 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 $n = \widetilde{O}(\lambda)$.

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 4.13 (Comparison with [18]). An alternative route to obtaining a lattice-based SNARG is to directly instantiate [18] with Regev-based encryption. However, to achieve soundness error $2^{-\lambda}$, Bitansky et al. [18] require a LPCP (and consequently, an additively homomorphic encryption) over a field of size 2^{λ}. Instantiating the construction in [18] 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 [18] with Regev-based encryption over a polynomial-size field (thus incurring $1/\text{poly}(\lambda)$-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. We provide a concrete comparison in Table 1.

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). Amongst 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 [42] and BCIOP [18]). 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 [34, 47, 15]—does not apply to our construction. As noted by [18], this limitation can be circumvented by SNARG constructions relying on algebraic PCPs such as ours. We show in the full version that a variant of our construction (with the same asymptotic complexity) gives a multi-theorem designated-verifier SNARG in the preprocessing model.

Remark 4.14 (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 [42, 18] remain quasi-optimally succinct for arithmetic circuit satisfiability over large fields.

Quasi-optimal SNARG. In the full version of this paper, we also show how vector encryption over polynomial rings that satisfy linear targeted malleability can be

Construction	**Type**[*]	**Prover Complexity**	**Proof Size**	**Assumption**
CS Proofs [63]	PV	$\widetilde{O}(s+\lambda^2)$	$\widetilde{O}(\lambda^2)$	Random Oracle
Groth [51]	PV	$\widetilde{O}(s^2\lambda+s\lambda^2)$	$\widetilde{O}(\lambda)$	Knowledge of Exponent
GGPR [42]	PV	$\widetilde{O}(s\lambda)$	$\widetilde{O}(\lambda)$	
BCIOP [18][†] (Paillier)	DV	$\widetilde{O}(s\lambda^3)$	$\widetilde{O}(\lambda^3)$	Linear-Only Encryption
BCIOP [18][†] (Pairing)	PV	$\widetilde{O}(s\lambda)$	$\widetilde{O}(\lambda)$	
BCIOP [18][†] (Regev)[‡]	DV	$\widetilde{O}(s\lambda^2)$	$\widetilde{O}(\lambda^2)$	
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 (RLWE-based)[¶]	DV	$\widetilde{O}(s)$	$\widetilde{O}(\lambda)$	Linear-Only Vector Enc.

[*]We write "PV" to denote public verifiability and "DV" for designated verifiability.
[†]Instantiated using a LPCP based on QSPs.
[‡]Based on a direct instantiation of [18] using Regev-based encryption (Remark 4.13).
[§]Instantiated with the PVW [67] encryption scheme from Section 4.2.
[¶]Instantiated with the RLWE-based vector encryption scheme described in the full version. This construction is the first which is quasi-optimal with respect to *both* prover complexity and proof size.

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 negl(λ) soundness error against provers of size 2^λ. (Some of the schemes can achieve $2^{-\lambda}$ soundness error with the same complexity.) 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.

leveraged to obtain the first SNARG construction that achieves quasi-optimal prover complexity as well as quasi-optimal succinctness. Our construction makes use of a new information-theoretic construction of LPCPs over rings.

5 Concrete Efficiency of Bootstrapping VBB Obfuscation

Due to space limitations, we defer our results on the concrete efficiency of bootstrapping obfuscation to the full version, and give an outline of our main results here. We start by describing 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, 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.

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.

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

1. Alperin-Sheriff, J., Peikert, C.: Practical bootstrapping in quasilinear time. In: CRYPTO (2013)
2. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: CRYPTO (2014)
3. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: CRYPTO (2015)
4. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: EUROCRYPT (2017)
5. Ananth, P.V., Gupta, D., Ishai, Y., Sahai, A.: Optimizing obfuscation: Avoiding barrington's theorem. In: ACM CCS (2014)
6. Applebaum, B.: Bootstrapping obfuscators via fast pseudorandom functions. In: ASIACRYPT (2014)
7. Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: TCC (2015)
8. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: CRYPTO (2009)
9. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. In: FOCS (1992)

10. Badrinarayanan, S., Miles, E., Sahai, A., Zhandry, M.: Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In: EUROCRYPT (2016)
11. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: EUROCRYPT (2014)
12. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO (2001)
13. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In: STOC (1986)
14. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: CRYPTO (2004)
15. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS (2012)
16. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: STOC (2014)
17. Bitansky, N., Chiesa, A.: Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In: CRYPTO (2012)
18. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: TCC (2013)
19. Bitansky, N., Paneth, O., Wichs, D.: Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In: TCC (2016)
20. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: FOCS (2015)
21. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: STOC (2000)
22. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: CRYPTO (2001)
23. Boneh, D., Segev, G., Waters, B.: Targeted malleability: homomorphic encryption for restricted computations. In: ITCS (2012)
24. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics 324(1) (2003)
25. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: CRYPTO (2014)
26. Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. IACR Cryptology ePrint Archive 2016 (2016)
27. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: CRYPTO (2012)
28. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS (2012)
29. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: STOC (2013)
30. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: TCC (2014)

31. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS (2011)
32. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: ASIACRYPT (2011)
33. Coron, J., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: CRYPTO (2013)
34. Crescenzo, G.D., Lipmaa, H.: Succinct NP proofs from an extractability assumption. In: 4th Conference on Computability (2008)
35. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: CRYPTO (1991)
36. Damgård, I., Faust, S., Hazay, C.: Secure two-party computation with low communication. In: TCC (2012)
37. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT (2015)
38. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO (1986)
39. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: EUROCRYPT (2013)
40. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: TCC (2014)
41. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
42. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: EUROCRYPT (2013)
43. Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: TCC (2015)
44. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: EUROCRYPT (2012)
45. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO (2013)
46. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: STOC (2011)
47. Goldwasser, S., Lin, H., Rubinstein, A.: Delegation of computation without rejection problem from designated verifier cs-proofs. IACR Cryptology ePrint Archive 2011 (2011)
48. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: STOC (1982)
49. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: STOC (1985)
50. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: TCC (2010)
51. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT (2010)
52. Joux, A.: A one round protocol for tripartite diffie-hellman. In: ANTS (2000)

53. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC (1992)
54. Kirchner, P., Fouque, P.: An improved BKW algorithm for LWE with applications to cryptography and lattices. In: CRYPTO (2015)
55. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: STOC (2015)
56. Lewi, K., Malozemoff, A.J., Apon, D., Carmer, B., Foltzer, A., Wagner, D., Archer, D.W., Boneh, D., Katz, J., Raykova, M.: 5gen: A framework for prototyping applications using multilinear maps and matrix branching programs. In: ACM CCS (2016)
57. Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: EUROCRYPT (2016)
58. Lin, H.: Indistinguishability obfuscation from DDH on 5-linear maps and locality-5 prgs. IACR Cryptology ePrint Archive 2016 (2016)
59. Lin, R., Vaikuntanathan, V.: Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In: FOCS (2016)
60. Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. In: CT-RSA (2011)
61. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC (2012)
62. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT (2010)
63. Micali, S.: Computationally sound proofs. SIAM J. Comput. 30(4) (2000)
64. Mie, T.: Polylogarithmic two-round argument systems. J. Mathematical Cryptology 2(4), 343–363 (2008)
65. Naor, M.: On cryptographic assumptions and challenges. In: CRYPTO (2003)
66. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT (1999)
67. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: CRYPTO (2008)
68. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC (2005)
69. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC (2014)
70. Schnorr, C., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. 66 (1994)
71. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27(4) (1980)
72. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: FOCS (1994)
73. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. 26(5) (1997)
74. Zimmerman, J.: How to obfuscate programs directly. In: EUROCRYPT (2015)
75. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: EUROSAM (1979)