# Lattice-Based zk-SNARKs from Square Span Programs

Rosario Gennaro, Michele Minelli, Anca Nitulescu, Michele Orrù

# Lattice-Based zk-SNARKs from Square Span Programs

Rosario Gennaro[1], Michele Minelli[2,3], Anca Nitulescu[2,3], and Michele Orrù[2,3]

[1] City College of New York, USA
[2] DIENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France
[3] Inria

rosario@cs.ccny.cuny.edu
{michele.minelli, anca.nitulescu, michele.orru}@ens.fr

**Abstract.** Zero-knowledge SNARKs (zk-SNARKs) are non-interactive proof systems with short (i.e., independent of the size of the witness) and efficiently verifiable proofs. They elegantly resolve the juxtaposition of individual privacy and public trust, by providing an efficient way of demonstrating knowledge of secret information without actually revealing it. To this day, zk-SNARKs are widely deployed all over the planet and are used to keep alive a system worth billion of euros, namely the cryptocurrency Zcash. However, all current SNARKs implementations rely on so-called *pre-quantum* assumptions and, for this reason, are not expected to withstand cryptanalitic efforts over the next few decades.
In this work, we introduce a new zk-SNARK that can be instantiated from lattice-based assumptions, and which is thus believed to be post-quantum secure. We provide a generalization in the spirit of Gennaro et al. (Eurocrypt'13) to the SNARK of Danezis et al. (Asiacrypt'14) that is based on Square Span Programs (SSP) and relies on weaker computational assumptions. We focus on designated-verifier proofs and propose a protocol in which a proof consists of just 5 LWE encodings. We provide a concrete choice of parameters, showing that our construction is practically instantiable.

**Keywords:** SNARK, zero-knowledge, post-quantum.

## 1 Introduction

In a zero-knowledge proof, a powerful prover $\mathsf{P}$ can prove to a weaker verifier $\mathsf{V}$ that a particular statement $x \in L$ is true, for some NP language $L$ (with corresponding witness relation $\mathcal{R}$), without revealing any additional information about the witness. For NP languages, $\mathsf{P}$ can be a polynomial time machine with input also the witness $w$ that $x \in L$ (the witness is a proof that $x \in L$, i.e. $\mathcal{R}(x, w)$ holds, but is not a zero-knowledge proof, since it reveals more information than just the mere fact that $x \in L$). Since their introduction in [GMR89] zero-knowledge (ZK) proofs have been shown to be a very powerful instrument in the design of secure cryptographic protocols.

For practical applications, researchers immediately recognized two limiting factors in zero-knowledge proofs: the original protocols were interactive and the proof could be as long as (if not longer than) the witness. Non-interactive zero-knowledge (NIZK) proofs [BFM88] and succinct ZK arguments [Kil92, Mic94] were introduced shortly thereafter. Those results were considered mostly theoretical proofs of concept until more recently, when several theoretical and practical breakthroughs have shown that such proofs (renamed zk-SNARGs for Succinct Non-interactive ARGuments or zk-SNARKs if the proofs also guarantee that the Prover knows the witness $w$) can indeed be used in practical applications.

Gennaro, Gentry, Parno and Raykova [GGPR13] proposed a new, influential characterization of the complexity class NP using *Quadratic Span Programs* (QSPs), a natural extension of span programs defined by Karchmer and Wigderson [KW93]. They show there is a very efficient reduction from boolean circuit satisfiability problems to QSPs. Their work has lead to fast progress towards practical verifiable computations. For instance, using Quadratic Arithmetic Programs (QAPs), a generalization of QSPs for arithmetic circuits, Pinocchio [PHGR13] provides evidence that verified remote computation can be faster than local computation. At the same time, their construction is

**Table 1.** Security estimates for different choices of LWE parameters (circuit size fixed to $d = 2^{15}$), together with the corresponding sizes of the proof $\pi$ and of the CRS (when using a seeded PRG for its generation).

| security level | $\lambda$ | $n$ | $\log \alpha$ | $\log q$ | $\lvert\pi\rvert$ | $\lvert\mathsf{crs}\rvert$ | ZK |
|---|---|---|---|---|---|---|---|
| **medium** | 164 | 950 | $-150$ | 800 | 0.45 MB | 9.37 MB | |
| | 172 | 1400 | $-180$ | 800 | 0.67 MB | 9.37 MB | ✓ |
| **high** | 253 | 1200 | $-150$ | 800 | 0.57 MB | 9.37 MB | |
| | 247 | 1700 | $-180$ | 800 | 0.81 MB | 9.37 MB | ✓ |
| **paranoid** | 357 | 1450 | $-150$ | 800 | 0.69 MB | 9.37 MB | |
| | 363 | 2100 | $-180$ | 800 | 1.00 MB | 9.37 MB | ✓ |

zero-knowledge, enabling the server to keep intermediate and additional values used in the computation private. Optimized versions of SNARK protocols based on QSPs approach are used in various practical applications, including cryptocurrencies such as Zcash [BCG+14a], to guarantee anonymity while preventing double-spending (via the ZK property).

The QSP approach was generalized in [BCI+13] under the concept of *Linear PCP* (LPCP) (there is a construction of an LPCP for a QSP satisfiability problem) – these are a form of interactive ZK proofs where security holds under the assumption that the prover is restricted to compute only affine combinations of its inputs. These proofs can then be turned into (designated-verifier) SNARKs by using a *linear-only* encryption, i.e. an encryption scheme where any adversary can output a valid new ciphertext, only if this is an affine combination of some previous encodings that the adversary had as input (intuitively this "limited malleability" of the encryption scheme, will force the prover into the above restriction).

So far all known practical SNARKs rely on "classical" pre-quantum assumptions[4]. Yet, widely deployed systems relying on SNARKs (such as the Zcash cryptocurrency [BCG+14b]) are expected not to withstand cryptanalitic efforts over the course of the next 10 years [ABL+17, Appendix C]. It is an interesting research question, as well our responsibility as cryptographers, to provide protocols that can guarantee people's privacy over the next decade. We attempt to make a step forward in this direction by building a designated-verifier zk-SNARK that relies on the Learning With Errors (LWE) assumption, initially proposed by Regev in 2005 [Reg05], and right now the most widespread post-quantum cryptosystem supported by a theoretical proof of security.

**SNARKs based on lattices.** Recently, in two companion papers [BISW17, BISW18], Boneh et al. provided the first designated-verifier SNARKs construction based on lattice assumptions.

The first paper has two main results: an improvement on the LPCP construction in [BCI+13] and a construction of linear-only encryption based on LWE. The second paper presents a different approach where the information-theoretic LPCP is replaced by a LPCP with multiple provers, which is then compiled into a SNARK again via linear-only encryption. The main advantage of this approach is that it reduces the overhead on the prover achieving what they call *quasi-optimality*[5].

**Our contributions.** In this paper, we frame the construction of Danezis et al. [DFGK14] for Square Span Programs in the framework of "encodings" introduced by Gennaro et al. [GGPR13].

[4] We note that the original protocol of Kilian [Kil92] is a SNARK which can be instantiated with a post-quantum assumption since it requires only a collision-resistant hash function – however (even in the best optimized version recently proposed in [BSBHR18]) the protocol does not seem to scale well for even moderately complex computations.

[5] This is the first scheme where the prover does not have to compute a cryptographic group operation for each wire of the circuit, which is instead true e.g., in QSP-based protocols.

We slightly modify the definition of encoding to accommodate for the noisy nature of LWE schemes. This allows us to have a more fine-grained control over the error growth, while keeping previous examples of encodings still suitable for our construction. Furthermore, SSPs are similar but simpler than Quadratic Span Programs (QSPs) since they use a single series of polynomials rather than 2 or 3. We use SSPs to build simpler and more efficient designated-verifier SNARKs and Non-Interactive Zero-Knowledge arguments (NIZKs) for circuit satisfiability (CIRC-SAT).

We think our approach is complementary to [BISW17, BISW18]. However, there are several reasons why we believe that our approach is preferable:

- **Zero-Knowledge.** The LPCP-based protocols in [BISW17, BISW18] are not ZK and the works do not explicitly describe ways to make them ZK (except by referring to generic transformations). Considering the LPCP constructed for a QSP satisfiability problem, there is a general transformation to obtain ZK property [BCI+13], but this introduces some overhead. Nevertheless, in the lattice setting, we are not sure this approach still holds. In contrast, our protocol is SSP-based and can thus be made ZK at essentially no cost for either the Prover or the Verifier. Our transformation is different, exploiting special features of SSPs, and yields a zk-SNARK with almost no overhead (if an adapted encoding is used).
- **Weaker Assumptions.** The linear-only property introduced in [BCI+13] implies all the security assumptions needed by a SSP-suitable encoding, but the reverse is not known to hold. Our proof of security therefore relies on weaker assumptions, and by doing so, "distills" the minimal known assumptions needed to prove security for SSP, and instantiates them with an LWE-based approach.
- **Simplicity and Efficiency.** While the result in [BISW18] seems asymptotically more efficient than any SSP-based approach, we suspect that, for many applications, the simplicity and efficiency of the SSP construction will still provide a concrete advantage in practice. To drive this point home we have been implementing our scheme and testing it on real-life applications and the results are encouraging (on the other hand no implementation is offered for [BISW17, BISW18] pointing to the theoretical nature of those results).

**Technical challenges.** Although conceptually similar to the original proof of security for QSP-based SNARKs, our proof encounters some specific technical challenges due to the noise growth of the LWE-based encoding. In particular these impose additional LWE-specific verification checks not needed in a "pure" QSP implementation. Such issues arise from the reduction to the weaker assumptions used in our proofs and are not needed in [BISW17, BISW18] because of the stronger linear-only assumption used there. Additionally, we incorporate some optimizations from SSP-based SNARKS [DFGK14].

Instantiating our encoding scheme with a lattice-based scheme like Regev encryption, differs from [GGPR13] and introduces some technicalities, first in the verification step of the protocol, and second in the proof of security. Our encoding scheme is additively homomorphic and supports affine operations. On the other hand, we are constrained to only allow for a limited number of homomorphic operations because of the bounded error growth in lattice-based encryption schemes. Since in these schemes the error is additive, to compute a linear combination of $N$ encodings (where the coefficients for the linear combination are drawn from a field $\mathbf{F} = \mathbf{Z}_p$), we need to scale some parameters for correctness to hold. However, if the encryption scheme supports modulus switching, it may be possible to work with a smaller modulus during decoding. Anyway, we will consider in this work that we are allowed to perform just a bounded number of "linear" operations on encodings and make sure that this bound is sufficient to perform verification and to make a security reduction.

Furthermore, the operations considered are affine rather than linear. The main reason for this adaptation is that the description is more appropriate for our proposed lattice-based encoding (in which a careful analysis of the noise growth needs to be made).

## 2 Prerequisites

**Notation.** We denote the real numbers by $\mathbf{R}$, the natural numbers by $\mathbf{N}$, the integers by $\mathbf{Z}$ and the integers modulo some $q$ by $\mathbf{Z}_q$. Let $\lambda \in \mathbf{N}$ be the computational security parameter, and $\kappa \in \mathbf{N}$ the statistical security parameter. For two integers $a, b \in \mathbf{Z}$, we denote with $a//b$ the quotient of the Euclidean division between $a$ and $b$. We say that a function is *negligible* in $\lambda$, and we denote it by $\mathsf{negl}(\lambda)$, if it is a $f(\lambda) = o(\lambda^{-c})$ for every fixed constant $c$. We also say that a probability is *overwhelming* in $\lambda$ if it is $1 - \mathsf{negl}(\lambda)$. We let $\mathsf{M.rl}(\lambda)$ be a *length function* (i.e. a function $\mathbf{N} \to \mathbf{N}$ polynomially bounded) in $\lambda$ defining the length of the randomness for a probabilistic interactive Turing Machine $\mathsf{M}$. When sampling uniformly at random the value $a$ from the set $S$, we employ the notation $a \leftarrow_{\$} S$. When sampling the value $a$ from the probabilistic algorithm $\mathsf{M}$, we employ the notation $a \leftarrow \mathsf{M}$. We use $:=$ to denote assignment. For an $n$-dimensional column vector $\vec{a}$, we denote its $i$-th entry by $a_i$. In the same way, given a polynomial $f$, we denote its $i$-th coefficient by $f_i$. Unless otherwise stated, the norm $\|\cdot\|$ considered in this work is the $\ell_2$ norm. We denote by $\vec{a} \cdot \vec{b}$ the dot product between vectors $\vec{a}$ and $\vec{b}$. For a $\mathsf{NP}$ relation $\mathcal{R}$ between a set of statements denoted by $u$ and witnesses denoted by $w$: we use $\mathcal{L}(\mathcal{R})$ to denote the language associated to $\mathcal{R}$.

Unless otherwise specified, all the algorithms defined throughout this work are assumed to be probabilistic Turing machines that run in time $\mathsf{poly}(\lambda)$ - i.e. $\mathsf{PPT}$. An adversary is denoted by $\mathcal{A}$; when it is interacting with an oracle $\mathcal{O}$ we write $\mathcal{A}^{\mathcal{O}}$. For two $\mathsf{PPT}$ machines $\mathsf{A}, \mathsf{B}$, with the writing $(\mathsf{A} \| \mathsf{B})(x)$ we denote the execution of $\mathsf{A}$ followed by the execution of $\mathsf{B}$ on the same input $x$ and with the same random coins. The output of the two machines is concatenated and separated with a semicolon, e.g., $(out_{\mathsf{A}}; out_{\mathsf{B}}) \leftarrow (\mathsf{A} \| \mathsf{B})(x)$.

### 2.1 Square Span Programs

We characterize $\mathsf{NP}$ as Square Span Programs (SSPs) over some field $\mathbf{F}$ of order $p$. SSPs were introduced first by Danezis et al. [DFGK14].

**Definition 1 (SSP).** *A Square Span Program (SSP) over the field $\mathbf{F}$ is a tuple consisting of $m+1$ polynomials $v_0(x), \ldots, v_m(x) \in \mathbf{F}[x]$ and a target polynomial $t(x)$ such that $\deg(v_i(x)) \leqslant \deg(t(x))$ for all $i = 0, \ldots, m$. We say that the square span program $\mathsf{ssp}$ has size $m$ and degree $d = \deg(t(x))$. We say that $\mathsf{ssp}$ accepts an input $a_1, \ldots, a_\ell \in \{0,1\}$ if and only if there exist $a_{\ell+1}, \ldots, a_m \in \{0,1\}$ satisfying:*

$$t(x) \text{ divides } \left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x)\right)^2 - 1.$$

We say that $\mathsf{ssp}$ verifies a boolean circuit $\mathsf{C} : \{0,1\}^\ell \to \{0,1\}$ if it accepts exactly those inputs $(a_1, \ldots, a_\ell) \in \{0,1\}^\ell$ that satisfy $\mathsf{C}(a_1, \ldots, a_\ell) = 1$.

**Universal circuit.** In the definition, we may see $\mathsf{C}$ as a logical specification of a satisfiability problem. In our zk-SNARK we will split the $\ell$ inputs into $\ell_u$ public and $\ell_w$ private inputs to make it compatible with universal circuits $C_{\mathcal{U}} : \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$, that take as input an $\ell_u$-bit description of a freely chosen circuit $\mathsf{C}$ and an $\ell_w$-bit value $w$ and return 1 if and only if $\mathsf{C}(w) = 1$. Along the lines of [DFGK14], we consider the "public" inputs from the point of view of the prover. For an outsourced computation, they might include both the inputs sent by the clients and the outputs returned by the server performing the computation. For CIRC-SAT, they may provide a partial instantiation of the problem or parts of its solution. This treatment is more general than CIRC-SAT, for which $\ell_u = 0$ - since the SSP is satisfied if the witness $w$ satisfies $\mathsf{C}(w) = 1$.

**Theorem 2 ([DFGK14, Theorem 2]).** *For any boolean circuit $\mathsf{C} : \{0,1\}^\ell \to \{0,1\}$ of $m$ wires and $n$ fan-in 2 gates and for any prime $p \geqslant \max(n, 8)$, there exist polynomials $v_0(x), \ldots, v_m(x)$ and distinct roots $r_1, \ldots, r_d \in \mathbf{F}$ such that $\mathsf{C}$ is satisfiable if and only if:*

$$\prod_{i=1}^{d}(x - r_i) \text{ divides } \left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x)\right)^2 - 1,$$

*where* $a_1, \ldots, a_m \in \{0,1\}$ *correspond to the values on the wires in a satisfying assignment for the circuit.*

*Define* $t(x) := \prod_{i=1}^{d}(x - r_i)$*, then for any circuit* $\mathsf{C} : \{0,1\}^\ell \to \{0,1\}$ *of* $m$ *wires and* $n$ *gates, there exists a degree* $d = m + n$ *square span program* $\mathsf{ssp} = (v_0(x), \ldots, v_m(x), t(x))$ *over a field* $\mathbf{F}$ *of order* $p$ *that verifies* $\mathsf{C}$*.*

**SSP generation.** We consider the uniform probabilistic algorithm $\mathsf{SSP}$, that on input a boolean circuit $\mathsf{C} : \{0,1\}^\ell \to \{0,1\}$ of $m$ wires and $d$ gates, chooses a field $\mathbf{F}$, with $|\mathbf{F}| \geqslant \max(n, 8)$, and samples $d = m+n$ random elements $r_1, \ldots, r_d \in \mathbf{F}$ to define the target polynomial $t(x) = \prod_{i=1}^{d}(x - r_i)$, together with the set of polynomials $\{v_0(x), \ldots, v_m(x)\}$ composing the SSP corresponding to $\mathsf{C}$.

$$(v_0(x), \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathsf{C})$$

### 2.2 Succinct Non-Interactive Arguments

In this section we provide formal definitions for the notion of succinct non-interactive arguments of knowledge (SNARKs).

**Definition 3.** *A* designated-verifier non-interactive proof system *for a relation* $\mathcal{R}$ *is a triple of algorithms* $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ *as follows:*

$(\mathsf{vrs}, \mathsf{crs}) \leftarrow \mathsf{G}(1^\lambda, \mathcal{R})$ *takes as input some complexity* $1^\lambda$ *and outputs a common reference string* $\mathsf{crs}$ *that will be given publicly, and* $\mathsf{vrs}$*, a trapdoor key that will be used for verification. For simplicity, we will assume in the future that* $\mathsf{crs}$ *can be extracted from* $\mathsf{vrs}$*, and that the unary complexity* $1^\lambda$ *can be derived as well from* $\mathsf{crs}$*.*

$\pi \leftarrow \mathsf{P}(\mathsf{crs}, u, w)$ *takes as input the* $\mathsf{crs}$*, a statement* $u$ *and a witness* $w$*, and outputs some proof of knowledge* $\pi$*.*

$\boldsymbol{bool} \leftarrow \mathsf{V}(\mathsf{vrs}, u, \pi)$ *takes as input a statement* $u$ *together with a proof* $\pi$*, and the trapdoor key* $\mathsf{vrs}$*, and outputs* ***true*** *if the proof was accepted,* ***false*** *otherwise.*

If the verification algorithm $\mathsf{V}$ takes as input the CRS instead of $\mathsf{vrs}$, then the NI proof system is called *publicly verifiable.*

**Definition 4 (SNARK).** *A* succinct non-interactive argument of knowledge *(SNARK) is a non-interactive proof system that satisfies the additional properties of completeness, succinctness, and knowledge soundness.*

Roughly speaking, *completeness* means that all correctly generated proofs verify; *succinctness* that the size of the proof is linear in the security parameter $\lambda$; *knowledge soundness* [BG93] that for any prover able to produce a valid proof for a statement in the language, there exists an efficient algorithm capable of extracting a witness for the given statement. More formally:

**Definition 5 (Completeness).** *A non-interactive proof system* $\Pi$ *for the relation* $\mathcal{R}$ *is (computationally)* complete *if for any* $\mathsf{PPT}$ *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathrm{compl}}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{COMPL}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)\right] = 1 - \mathsf{negl}(\lambda)\,,$$

*where* $\mathrm{COMPL}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$ *is the game depicted in Fig. 1.*

**Definition 6 (Knowledge Soundness).** *A non-interactive proof system* $\Pi$ *for the relation* $\mathcal{R}$ *is* knowledge-sound *if for any* $\mathsf{PPT}$ *adversary* $\mathcal{A}$ *there exists an extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that:*

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\Pi,\mathcal{R},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr\left[\mathrm{KSND}_{\Pi,\mathcal{R},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)\right] = \mathsf{negl}(\lambda)\,,$$

*where* $\mathrm{KSND}_{\Pi,\mathcal{R},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is defined in Figure 1.*

Game $\mathrm{KSND}_{\Pi,\mathcal{R},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$

$(\mathsf{crs}, \mathsf{vrs}) \leftarrow \Pi.\mathsf{G}(1^\lambda)$
$(u, \pi; w) \leftarrow (\mathcal{A} \| \mathsf{Ext}_{\mathcal{A}})(\mathsf{crs})$
**return** $(\mathcal{R}(u, w) = \textbf{false} \ \wedge \ \Pi.\mathsf{V}(\mathsf{vrs}, u, \pi))$

Game $\mathrm{COMPL}_{\Pi,\mathcal{R},\mathcal{A}}(\lambda)$

$\mathsf{crs} \leftarrow \Pi.\mathsf{G}(1^\lambda)$
$(u, w) \leftarrow \mathcal{A}(\mathsf{crs})$
$\pi \leftarrow \Pi.\mathsf{P}(\mathsf{crs}, u, w)$
**return** $\Pi.\mathsf{V}(\mathsf{crs}, u, \pi)$ **and** $\mathcal{R}(u, w) = \textbf{true}$

Game $\mathrm{ZK}_{\Pi,\mathcal{R},\mathsf{Sim},\mathcal{A}}(\lambda)$

$(\mathsf{crs}, \mathsf{vk}) \leftarrow \Pi.\mathsf{G}(1^\lambda)$
$b \leftarrow\!\!\$ \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\textsc{Prove}}(\mathsf{vrs})$
**return** $(b = b')$

Oracle $\textsc{Prove}(u, w)$

**if** $\mathcal{R}(u, w) = \textbf{false}$ **return** $\perp$
**if** $b = 1$ $\pi \leftarrow \Pi.\mathsf{P}(\mathsf{crs}, u, w)$
**else** $\pi \leftarrow \mathsf{Sim}(\mathsf{vrs}, u)$
**return** $\pi$

**Fig. 1.** Games for completeness (COMPL), knowledge soundness (KSND), and zero-knowledge (ZK).

An *argument of knowledge* is a *knowledge-sound* proof system. If the adversary is computationally unbounded, we speak of *proofs* rather than arguments.

*Remark 7.* An important consideration that arises when defining knowledge soundness in the designated-verifier scenario is whether the adversary should be granted access to a proof-verification oracle. Pragmatically, allowing a verification oracle captures whether or not a CRS can be reused $\mathsf{poly}(\lambda)$ times. While this property follows immediately in the public-verifier setting, the same is not true for the designated-verifier setting. In the specific case of our construction, we formulate and prove our protocol with the stronger notion (which has been addressed to as *strong soundness* in the past [BISW17]), and quickly discuss which optimizations can take place when using the weaker notion of soundness.

We distinguish two types of arguments of knowledge: *publicly verifiable* ones, where the security holds against adversaries that have access to $\mathsf{vrs}$; and those *with designated verifier*, where the verification step needs access to $\mathsf{vrs}$. It is straightforward to note that, with the help of an encryption scheme, any publicly-verifiable proof system can be transformed into an analogous designated-verifier one. It is nonetheless important to note that in the standard model, all constructions we are aware of so far somehow imply the existence of an encryption scheme.

A proof system $\Pi$ for $\mathcal{R}$ is zero-knowledge if no information about the witness is leaked by the proof. More precisely:

**Definition 8 (Zero-Knowledge).** *A non-interactive proof system* $\Pi$ *is* zero-knowledge *if there exists a simulator* $\mathsf{Sim}$ *such that for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{R},\mathsf{Sim},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{ZK}_{\Pi,\mathcal{R},\mathsf{Sim},\mathcal{A}}(\lambda)\right] = \mathsf{negl}(\lambda)\,,$$

*where* $\mathrm{ZK}_{\Pi,\mathcal{R},\mathsf{Sim},\mathcal{A}}(\lambda)$ *is defined in Figure 1. Zero-knowledge SNARKs are informally called* zk-SNARKs*.*

### 2.3 Encoding Schemes

**Definition 9 (Encoding Scheme).** *An encoding scheme* $\mathsf{Enc}$ *over a field* $\mathbf{F}$ *is composed of the following algorithms:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$*, a key generation algorithm that takes as input some complexity* $1^\lambda$ *and outputs some secret state* $\mathsf{sk}$ *together with some public information* $\mathsf{pk}$*. To ease notation, we are going to assume the message space is always part of the public information and that* $\mathsf{pk}$ *can be derived from* $\mathsf{sk}$*.*

– $S \leftarrow \mathsf{E}(a)$, *a non-deterministic encoding algorithm mapping a field element* $a$ *to some encoding space* $S$*, such that* $\{\{\mathsf{E}(a)\} : a \in \mathbf{F}\}$ *partitions* $S$*, where* $\{\mathsf{E}(a)\}$ *denotes the set of the possible evaluations of the algorithm* $\mathsf{E}$ *on* $a$*, that is* $\{\mathsf{E}(a; r) : r \in \mathsf{E.rl}(\lambda)\}$*. In other words, we require the decoding algorithm* $\mathsf{D}$ *to be a function.*
*Depending on the encoding algorithm,* $\mathsf{E}$ *will require either the public information* $\mathsf{pk}$ *generated from* $\mathsf{K}$ *or the secret state. For our application targeted at* designated-verifier proofs *it will be the case of* $\mathsf{sk}$*. To ease notation, we will omit this additional argument.*

*The above algorithms must satisfy the following properties:*

**$d$-affinely homomorphic:** *there exists a* $\mathsf{poly}(\lambda)$ *algorithm* $\mathsf{Eval}$ *that, given as input the public parameters* $\mathsf{pk}$*, a vector of encodings* $(\mathsf{E}(a_i))_i^d$*, coefficients* $\vec{c} = (c_i)_i^d \in \mathbf{F}$ *and constant factor* $b \in \mathbf{F}$*, outputs a valid encoding of* $\vec{a} \cdot \vec{c} + b$ *with probability overwhelming in* $\lambda$*. If the constant factor is omitted, it is assumed to be* $0$*.*

**quadratic root detection:** *there exists an efficiently computable algorithm* $Q(\delta, \mathsf{pp})$ *that, given as input some parameter* $\delta$ *(either the public information* $\mathsf{pk}$ *or the verification key* $\mathsf{sk}$*, depending on the kind of verifier), can test if the evaluation of quadratic polynomial* $\mathsf{pp}$ *with coefficients in the field is zero.*

**image verification:** *there exists an efficiently computable algorithm* $\in$ *that, given as input some parameter* $\delta$ *(again, either* $\mathsf{pk}$ *or* $\mathsf{sk}$*), can distinguish if an element* $c$ *is a correct encoding of a field element.*

Sometimes, in order to ease notation, we will employ the writing $\mathsf{ct} := \mathsf{Eval}((\mathsf{E}(a_i)), (c_i)_i) = \mathsf{E}(c)$ actually meaning that $\mathsf{ct}$ is a valid encoding of $c = \sum a_i c_i$, that is $\mathsf{ct} \in \{\mathsf{E}(c)\}$. It will be clear from the context (and the use of symbol for assignment instead of that for sampling) that the randomized encoding algorithm is not actually invoked.

**Decoding algorithm.** When using an encryption scheme in order to instantiate an encoding scheme, we can naturally define the *decoding algorithm* $\mathsf{D}$ that simply takes advantage of the decryption procedure. Encoding schemes that only need the public parameters $\mathsf{pk}$ to perform quadratic root detection and image verification lead to a SNARK that is *publicly verifiable*. Encoding schemes that rely on the secret state $\mathsf{sk}$ - as those we focus on in this work - lead instead to *designated-verifier* proofs. More specifically, since we study encoding schemes derived from encryption functions, quadratic root detection for designated-verifiers is trivially obtained by using the decoding algorithm $\mathsf{D}$.

*Remark 10.* Our specific instantiation of the encoding scheme presents some slight differences with [GGPR13]. First, we allow only for a limited number of homomorphic operations because of the error growth in lattice-based encoding schemes. Furthermore, these operations are affine rather than linear. The main reason for this adaptation is that the description is more apt for our proposed lattice-based encoding (in which a careful analysis of the noise growth needs to be made), and that at the same time it does not exclude previous constructions.

The reason for allowing affine operations rather limiting ourselves to only linear is a mere technicality. The inhomogeneous part can always be constructed for linear-only schemes by adding $\mathsf{E}(1)$ to the public information $\mathsf{pk}$, which, as a matter of fact, happens to be already present in all previous encoding schemes. For example, in pairing-based encodings this is just the group generator, and it is usually included already in the pairing group description. The converse cannot be said about Regev encryption where, given $\mathsf{E}(m)$, it is always possible to compute a valid encoding of $m+1$ without any additional information. Furthermore, the bounds on the number of allowed linear operations, those can simply be considered $\infty$ for the encodings provided in the past [GGPR13].

In order to guarantee a security reduction of our construction of Section 4, we will have to guarantee that some encoding provided by the adversary is not "too noisy" and that it is still possible to perform homomorphic operations on it. Let us consider a function $\mathsf{test\text{-}error}$ which, given as input the secret state $\mathsf{sk}$ together with some encoding $\mathsf{ct}$, returns **true** or **false** depending

on whether it is still possible to compute a certain linear operation known in advance. Since the function takes as input the secret key itself, it is easy to build such a function relying just on the $\mathsf{Eval}$ and $\in$ - image verification - algorithms.

*Example 11.* We present the classical example of encoding scheme using symmetric pairings on elliptic curves. The asymmetric variant of this encoding scheme is the most classical example of zk-SNARKs. Consider the cyclic groups $\mathbb{G}, \mathbb{G}_T$ of the same prime order $p$ equipped with the bilinear non-degenerate map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The groups $\mathbb{G}$, $\mathbb{G}_T$ are generated respectively by $G \in \mathbb{G}$ and by $e(G, G) \in \mathbb{G}_T$. For instance, the family of elliptic curves $\mathbb{G} := E(\mathbf{F}_q)$. described in [BF01] satisfies the above description. The encoding scheme simply computes $\mathsf{E} : x \mapsto xG$. The public information $\mathsf{pk}$ consists of the pairing group description $\Gamma := (p, \mathbb{G}, \mathbb{G}_T, e, G)$; the secret state $\mathsf{sk}$ is set to $\perp$. This encoding satisfies the three requirements as follows:

- $d$-affine homomorphic evaluations between a vector of encodings $(\mathsf{E}(a_i))_i^d$ with the coefficients $(c_i)_i^d$ and constant term $b$ is done as follows:

$$\mathsf{Eval}\big((\mathsf{E}(a_i))_i^d, (c_i)_i^d, b\big) := \sum_i^d \mathsf{E}(a_i c_i) + b\,\mathsf{E}(1),$$

  In other words, the $\mathsf{Eval}$ algorithm simply outputs the group element $\left(\sum_i^d a_i c_i + b\right) G$.
- The efficiently-computable quadratic root detection algorithm $Q$ simply consists of the pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and the quadratic test takes place in the target group $\mathbb{G}_T$. More concretely, given encodings $(\mathsf{E}(a_i))_i^d$, use the bilinear map to compute $e(G, G)^{\mathsf{pp}(a_1,\ldots,a_d)}$ where $\mathsf{pp}$ is a quadratic polynomial, and check whether it equals the identity element in $\mathbb{G}_T$.
- Image verification is straightforward. A group element $P$ is an encoding of an element $s$ in $\mathbb{G}$ iff $P = sG = \mathsf{E}(s)$.

A more concrete encoding scheme will be discussed in Section 3. In particular, we conjecture that it satisfies the assumptions of the following section.

### 2.4 Assumptions

Throughout this paper we rely on a number of computational assumptions. All of them have been introduced in the past (e.g., [GGPR13]): we report them here for completeness and in order to explore the relations between them.

The $q$-power knowledge of exponent assumption ($q$-PKE) is a generalization of the knowledge of exponent assumption (KEA) introduced by Damgard [Dam92]. It says that given $\mathsf{E}(s), \ldots, \mathsf{E}(s^q)$ and $\mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^q)$ for some coefficient $\alpha$, it is difficult to generate $\mathsf{ct}, \widehat{\mathsf{ct}}$ encodings of $c, \alpha c$ without knowing the linear combination of the powers of $s$ that produces $\mathsf{ct}$.

**Assumption 1 ($q$-PKE).** *The $q$-Power Knowledge of Exponent ($q$-PKE) assumption holds relative to an encoding scheme $\mathsf{Enc}$ and for the class $\mathcal{Z}$ of auxiliary input generators if, for every non-uniform polynomial time auxiliary input generator $z \in \mathcal{Z}$ and non-uniform $\mathsf{PPT}$ adversary $\mathcal{A}$, there exists a non-uniform extractor $\mathsf{Ext}$ such that:*

$$\mathsf{Adv}^{\mathrm{pke}}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr\left[\text{q-PKE}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)\right] = \mathsf{negl}(\lambda)\,,$$

*where* $\text{q-PKE}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is the game depicted in Figure 2.*

The $q$-PDH assumption has been a long-standing, standard $q$-type assumption [Gro10, BBG05], Basically it states that given $(\mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q}))$, it is hard to compute an encoding of the missing power $\mathsf{E}(s^{q+1})$.

Game $\text{q-PKE}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}},z}(\lambda)$

$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$\alpha, s \leftarrow_\$ \mathbf{F}^*$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^q))$
$z \leftarrow \mathcal{Z}(\mathsf{pk}, \sigma)$
$(\mathsf{ct}, \hat{\mathsf{ct}}; a_1, \ldots, a_q) \leftarrow (\mathcal{A} \| \mathsf{Ext}_{\mathcal{A}})(\sigma, z)$
**return** $\mathsf{ct} \in \left\{\mathsf{E}(\sum_k^d a_k s^k)\right\} \ \wedge \ \hat{\mathsf{ct}} \in \left\{\mathsf{E}(\alpha \sum_k^d a_k s^k)\right\}$

Game $\text{q-PKEQ}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$

$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$s \leftarrow_\$ \mathbf{F}$
$\mathsf{crs} \leftarrow (\mathsf{pk}, \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q}))$
$(\mathsf{E}(c), e; b) \leftarrow (\mathcal{A} \| \mathsf{Ext}_{\mathcal{A}})(\mathsf{crs})$
**if** $b = 0$ **return** $e \in \{\mathsf{E}(c)\}$
**else return** $e \notin \{\mathsf{E}(c)\}$

Game $\text{q-PDH}_{\mathsf{Enc},\mathcal{A}}(\lambda)$

$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$s \leftarrow_\$ \mathbf{F}$
$\sigma \leftarrow (\mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q}))$
$y \leftarrow \mathcal{A}(\sigma)$
**return** $y \in \{\mathsf{E}(s^{q+1})\}$

**Fig. 2.** Games for $q$-PKE, $q$-PKEQ, $q$-PDH assumptions.

**Assumption 2 ($q$-PDH).** *The $q$-Power Diffie-Hellman ($q$-PDH) assumption holds for encoding* Enc *if for all* PPT *adversaries $\mathcal{A}$ we have:*

$$\mathsf{Adv}^{\text{q-pdh}}_{\mathsf{Enc},\mathcal{A}}(\lambda) := \Pr\left[\text{q-PDH}_{\mathsf{Enc},\mathcal{A}}(\lambda)\right] - 1/2 = \mathsf{negl}(\lambda)\,,$$

*where* $\text{q-PDH}_{\mathsf{Enc},\mathcal{A}}(\lambda)$ *is defined as in Figure 2.*

Finally, we need another assumption to be able to "compare" encoded messages. The $q$-PKEQ assumption boils down to the question of whether $\mathcal{A}$ can output $(\mathsf{E}(c), e)$ without $\mathsf{Ext}_{\mathcal{A}}$ being able to tell whether $e$ is also an encoding of $c$.

**Assumption 3 ($q$-PKEQ).** *The $q$-Power Knowledge of Equality ($q$-PKEQ) assumption holds for the encoding scheme* Enc *if for every* PPT *adversary $\mathcal{A}$ there exists an extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that:*

$$\mathsf{Adv}^{\text{q-pkeq}}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr\left[\text{q-PKEQ}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)\right] = \mathsf{negl}(\lambda)\,,$$

*where* $\text{q-PKEQ}_{\mathsf{Enc},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is the game depicted in Figure 2.*

This last assumption is needed solely in the case where the attacker has access to a verification oracle (see Remark 7). Since the encoding could be non-deterministic, the simulator in the security reduction ofSection 5.2 needs to rely on $q$-PKEQ to simulate the verification oracle. Pragmatically, this assumption allows us to test for equality of two encoded messages even without having access to the secret key.

## 3 Lattice-based encodings

In this section we give a brief introduction to lattices and we describe a possible encoding scheme based on lattice assumptions.

**Lattices.** A $m$-dimensional lattice $\Lambda$ is a discrete additive subgroup of $\mathbf{R}^m$. For an integer $k < m$ and a rank $k$ matrix $B \in \mathbf{R}^{m \times k}$, $\Lambda(B) = \left\{B\vec{x} \in \mathbf{R}^m \mid \vec{x} \in \mathbf{Z}^k\right\}$ is the lattice generated by the columns of $B$.

| Game $\mathrm{dLWE}_{\mathsf{Pg},\mathcal{A}}(\lambda)$ | Oracle ENCODE |
|---|---|
| $\Gamma := (p, q, n, \alpha) := \mathsf{Pg}(1^\lambda)$ | $\vec{a} \leftarrow_\$ \mathbf{Z}_q^n$ |
| $\vec{s} \leftarrow_\$ \mathbf{Z}_q^n$ | $e \leftarrow \chi_{q\alpha}$ |
| $b \leftarrow_\$ \{0,1\}$ | **if** $b = 1$ $c := \vec{s} \cdot \vec{a} + e$ |
| $b' \leftarrow \mathcal{A}^{\text{ENCODE}}(\Gamma)$ | **else** $c \leftarrow_\$ \mathbf{Z}_q$ |
| **return** $(b = b')$ | **return** $(\vec{a}, c)$ |

**Fig. 3.** The decisional LWE problem for parameters $\Gamma$.

**Gaussian distribution.** For any $\sigma \in \mathbf{R}^+$, let $\rho_\sigma(\vec{x}) := e^{-\pi\|\vec{x}\|^2/\sigma^2}$ be the Gaussian function over $\mathbf{R}^n$ with mean 0 and parameter $\sigma$. For any discrete subset $D \subseteq \mathbf{R}^n$ we define $\rho_\sigma(D) := \sum_{\vec{x}\in D} \rho_\sigma(\vec{x})$, the discrete integral of $\rho_\sigma$ over $D$. We then define $\chi_\sigma$, the discrete Gaussian distribution over $D$ with mean 0 and parameter $\sigma$ as:

$$\chi_\sigma : D \to \mathbf{R}^+ : \vec{y} \mapsto \frac{\rho_\sigma(\vec{y})}{\rho_\sigma(D)}.$$

We denote by $\chi_\sigma^n$ the discrete Gaussian distribution over $\mathbf{R}^n$ where each entry is independently sampled from $\chi_\sigma$.

**Lattice-based Encoding Scheme.** We propose an encoding scheme $\mathsf{Enc}$ that consists in three algorithms as depicted in Figure 4. This is a slight variation of the classical LWE cryptosystem initially presented by Regev [Reg05], described by parameters $\Gamma := (q, n, p, \alpha)$, with $q, n, p \in \mathbf{N}$ and $0 < \alpha < 1$. This construction is an extension of the one presented in [BV11].

We assume the existence of a deterministic algorithm $\mathsf{Pg}$ that, given as input the security parameter in unary $1^\lambda$, outputs a LWE encoding description $\Gamma$. Similar assumptions have been used in the past by Bellare et al. [BFS16] for bilinear group descriptions. The main advantage in choosing $\mathsf{Pg}$ to be deterministic is that every entity can (re)compute the description for the security parameter, and that no single party needs to be trusted with generating the encoding parameters. Moreover, real-world encodings have fixed parameters for some well-known values of $\lambda$. For the sake of simplicity, we define our encoding scheme with a LWE encoding description $\Gamma$ and assume that the security parameter $\lambda$ can be derived from $\Gamma$.

Roughly speaking, the public information is constituted by the LWE parameters $\Gamma$ and an encoding of $m$ is simply an LWE encryption of $m$. The LWE secret key constitutes the secret state of the encoding scheme. We say that the encoding scheme is (statistically) correct if all valid encodings are decoded successfully (with overwhelming probability).

**Assumption 4 (dLWE).** *The decisional Learning With Errors (dLWE) assumption holds for the parameter generation algorithm* $\mathsf{Pg}$ *if for any* $\mathsf{PPT}$ *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathrm{dlwe}}_{\mathsf{Pg},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{dLWE}_{\mathsf{Pg},\mathcal{A}}(\lambda)\right] - 1/2 = \mathsf{negl}(\lambda)\,,$$

*where* $\mathrm{dLWE}_{\mathsf{Pg},\mathcal{A}}(\lambda)$ *is defined as in Figure 3.*

In [Reg05], Regev showed that solving the decisional LWE problem is as hard as solving some lattice problems in the worst case. We recall here this result:

**Theorem 12 (Hardness of dLWE [Reg05]).** *For any parameter generation algorithm* $\mathsf{Pg}$ *outputting* $p = \mathsf{poly}(\lambda)$*, a modulus* $q \leqslant 2^{\mathsf{poly}(n)}$*, and a (discretized) Gaussian error distribution parameter* $\sigma = \alpha q \geqslant 2\sqrt{n}$ *with* $0 < \alpha < 1$*, solving* $\mathrm{dLWE}_{\mathsf{Pg},\mathcal{A}}(\lambda)$ *is at least as hard as solving* $\mathsf{GapSVP}_{\tilde{O}(n/\alpha)}$*.*

**Definition 13.** *An encoding scheme* $\mathsf{Enc}$ *is* correct *if, for any* $\vec{s} \leftarrow \mathsf{G}(1^\lambda)$ *and* $m \in \mathbf{Z}_p$*:*

$$\Pr[\mathsf{D}(\vec{s}, \mathsf{E}(\vec{s}, m)) \neq m] = \mathsf{negl}(\lambda)\,.$$

*We say that an encoding* $\mathsf{ct}$ *of a message* $m$ *under secret key* $\vec{s}$ *is* valid *if* $\mathsf{D}(\vec{s}, \mathsf{ct}) = m$*. We say that an encoding is* fresh *if it is generated through the* $\mathsf{E}$ *algorithm. We say that an encoding is* stale *if it is not fresh.*

**Lemma 14 (Correctness).** *Let* $\mathsf{ct} = (-\vec{a}, \vec{a} \cdot \vec{s} + pe + m)$ *be an encoding. Then* $\mathsf{ct}$ *is a valid encoding of a message* $m \in \mathbf{Z}_p$ *if* $e < \frac{q}{2p}$*.*

Image verification and quadratic root detection can be implemented using $\mathsf{D}$, providing the secret key as input. The algorithm $\in$ for image verification proceeds as follows: decrypts the encoded element and tests for equality between the two messages. The algorithm $Q$ for quadratic root detection is straightforward: decrypt the message and evaluate the polynomial, testing if it is equal to 0. Given a vector of $d$ encodings $\vec{\mathsf{ct}} \in \mathbf{Z}_q^{d\times(n+1)}$, a vector of coefficients $\vec{c} \in \mathbf{Z}_p^d$ and a constant $b \in \mathbf{Z}_p$, the homomorphic evaluation algorithm is defined as follows: $\mathsf{Eval}\left(\vec{\mathsf{ct}}, \vec{c}, b\right) := \vec{c} \cdot \vec{\mathsf{ct}} + b$. As previously mentioned, whenever $b$ is omitted from the arguments of $\mathsf{Eval}$, we implicitly mean $b = 0$. During the homomorphic evaluation the noise grows as a result of the operations which are performed on the encodings. Consequently, in order to ensure that the output of $\mathsf{Eval}$ is still a valid encoding, we need to start with a sufficiently small noise in each of the initial encodings.

In order to bound the size of the noise, we first need a basic theorem on the tail bound of discrete Gaussian distributions due to Banaszczyk [Ban95]:

**Lemma 15 ([Ban95, Lemma 2.4]).** *For any* $\sigma, T \in \mathbf{R}^+$ *and* $\vec{a} \in \mathbf{R}^n$*:*

$$\Pr[\vec{x} \leftarrow \chi_\sigma^n \;:\; |\vec{x} \cdot \vec{a}| \geqslant T\sigma\,\|\vec{a}\|] < 2\exp(-\pi T^2). \tag{1}$$

At this point, this corollary follows:

**Corollary 16.** *Let* $\vec{s} \leftarrow\!\!\$\, \mathbf{Z}_q^n$ *be a secret key and* $\vec{m} = (m_1, \ldots, m_d) \in \mathbf{Z}_p^d$ *be a vector of messages. Let* $\vec{\mathsf{ct}}$ *be a vector of* $d$ *fresh encodings so that* $\vec{\mathsf{ct}}_i \leftarrow \mathsf{E}(\vec{s}, m_i)$*, and* $\vec{c} \in \mathbf{Z}_p^d$ *be a vector of coefficients. If* $q > 2p^2\sigma\sqrt{\frac{\kappa d}{\pi}}$*, then* $\mathsf{Eval}\left(\vec{c}, \vec{\mathsf{ct}}\right)$ *outputs a valid encoding of* $\vec{m} \cdot \vec{c}$ *under the secret key* $\vec{s}$ *with probability overwhelming in* $\kappa$*.*

*Proof.* The fact that the message part is $\vec{m} \cdot \vec{c}$ is trivially true by simple homomorphic linear operations on the encodings. Then the final encoding is valid if the error does not grow too much during these operations. Let $\vec{e} \in \mathbf{Z}_p^d$ be the vector of all the error terms in the $d$ encodings, and let $T = \sqrt{\kappa/\pi}$. Then by Lemma 15 we have:

$$\Pr\left[\vec{e} \leftarrow \chi_\sigma^d \;:\; |\vec{e} \cdot \vec{c}| \geqslant \sqrt{\frac{\kappa}{\pi}}\sigma\,\|\vec{c}\|\right] < 2\exp(-\kappa).$$

For correctness we need the absolute value of the final noise to be less than $q/2p$ (cf. Lemma 14). Since it holds that $\forall \vec{c} \in \mathbf{Z}_p^d,\ \|\vec{c}\| \leqslant p\sqrt{d}$, we can state that correctness holds if

$$\sqrt{\frac{\kappa}{\pi}}\sigma p\sqrt{d} < \frac{q}{2p}$$

which gives $q > 2p^2\sigma\sqrt{\dfrac{\kappa d}{\pi}}$. □

| $\mathsf{K}(1^\lambda)$ | $\mathsf{E}(\vec{s}, m)$ | $\mathsf{D}(\vec{s}, (\vec{c}_0, c_1))$ |
|---|---|---|
| $\Gamma := (p, q, n, \alpha) := \mathsf{Pg}(1^\lambda)$ | $\Gamma := (p, q, n, \alpha) := \mathsf{Pg}(1^\lambda)$ | $\Gamma := (p, q, n, \alpha) := \mathsf{Pg}(1^\lambda)$ |
| $\vec{s} \leftarrow\!\!\$\ \mathbf{Z}_q^n$ | $\vec{a} \leftarrow\!\!\$\ \mathbf{Z}_q^n$ | **return** $(\vec{c}_0 \cdot \vec{s} + c_1) \pmod p$ |
| **return** $(\Gamma,\ \vec{s})$ | $\sigma := q\alpha;\quad e \leftarrow \chi_\sigma$ | |
| | **return** $(-\vec{a},\ \vec{a} \cdot \vec{s} + pe + m)$ | |

**Fig. 4.** An encoding scheme based on LWE.

**Smudging.** When computing a linear combination of encodings, the distribution of the error term in the final encoding depends on the coefficients of the combination, and it could therefore potentially leak information to whoever holds the secret key. We can solve this problem with the well known technique of noise smudging (or flooding): roughly speaking, adding a term large enough to the noise cancels out any dependency on the coefficients we want to hide.

**Lemma 17 (Noise Smudging, [BGGK17]).** *Let $B_1 = B_1(\kappa)$ and $B_2 = B_2(\kappa)$ be positive integers. Let $x \in [-B_1, B_1]$ be a fixed integer and $y \leftarrow\!\!\$\ [-B_2, B_2]$. Then the distribution of $y$ is statistically indistinguishable from that of $y + x$, as long as $B_1/B_2 = \mathsf{negl}(\kappa)$.*

*Proof.* Let $\Delta$ denote the statistical distance between the two distributions. By its definition:

$$\Delta = \frac{1}{2} \sum_{v=-(B_1+B_2)}^{B_1+B_2} |\Pr[y = v] - \Pr[y = v - x]| = \frac{1}{2} \left( \sum_{v=-(B_1+B_2)}^{-B_2} \frac{1}{B_2} + \sum_{v=B_2}^{B_1+B_2} \frac{1}{B_2} \right) = \frac{B_1}{B_2}.$$

The result follows immediately. □

In order to preserve the correctness of the encoding scheme, we need once again $q$ to be large enough to accommodate for the flooding noise. In particular, $q$ will have to be at least superpolynomial in the statistical security parameter $\kappa$.

**Corollary 18.** *Let $\vec{s} \in \mathbf{Z}_q^n$ be a secret key and $\vec{m} = (m_1, \ldots, m_d) \in \mathbf{Z}_p^d$ be a vector of messages. Let $\vec{\mathsf{ct}}$ be a vector of $d$ encodings so that $\vec{\mathsf{ct}}_i$ is a valid encoding of $m_i$, and $\vec{c} \in \mathbf{Z}_p^d$ be a vector of coefficients. Let $e_{\mathsf{Eval}}$ be the noise in the encoding output by $\mathsf{Eval}(\vec{\mathsf{ct}}, \vec{c})$ and $B_{\mathsf{Eval}}$ a bound on its absolute value. Finally, let $B_{sm} = 2^\kappa B_{\mathsf{Eval}}$, and $e_{sm} \leftarrow\!\!\$\ [-B_{sm}, B_{sm}]$. Then the statistical distance between the distribution of $e_{sm}$ and that of $e_{sm} + e_{\mathsf{Eval}}$ is $2^{-\kappa}$. Moreover, if $q > 2p\, B_{\mathsf{Eval}}(2^\kappa + 1)$ then the result of $\mathsf{Eval}(\vec{\mathsf{ct}}, \vec{c}) + \left(\vec{0}, e_{sm}\right)$ is a valid encoding of $\vec{m} \cdot \vec{c}$ under the secret key $\vec{s}$.*

*Proof.* The claim on the statistical distance follows immediately from Lemma 17 and the fact that the message part is $\vec{m} \cdot \vec{c}$ is true by simple homomorphic linear operations on the encodings. In order to ensure that the final result is a valid encoding, we need to make sure that the error in this output encoding remains smaller than $q/2p$. The final error is upper bounded by $B_{\mathsf{Eval}} + B_{sm}$, so we have

$$B_{\mathsf{Eval}} + B_{sm} < \frac{q}{2p} \implies B_{\mathsf{Eval}} + 2^\kappa B_{\mathsf{Eval}} < \frac{q}{2p} \implies q > 2p\, B_{\mathsf{Eval}}(2^\kappa + 1).$$

□

**Error testing.** By making non-blackbox use of our LWE encoding scheme, it is possible to define an implementation of the function $\mathsf{test\text{-}error}$ (cf. Section 2) that will be used later in our construction in order to guarantee the existence of a security reduction. In fact, for LWE encodings, it is sufficient to use the secret key, recover the error, and enforce an upper bound on its norm (namely, the norm of the error must still allow for some homomorphic operations while holding correctness). A possible implementation of $\mathsf{test\text{-}error}$ is displayed in Figure 5.

Procedure $\mathsf{test\text{-}error}(\vec{s}, (\vec{c}_0, c_1))$

$\Gamma := (p, q, n, \alpha) := \mathsf{Pg}(1^\lambda)$

$e' := (\vec{c}_0 \cdot \vec{s} + c_1) \;//\; p$

**return** (*Equation (2)*)

**Fig. 5.** The error testing procedure.

Now we give a lemma that will be useful later during the security proof. It essentially defines the conditions under which we can take an encoding, add a smudging term to its noise, sum it with the output of an execution of Eval and finally multiply the result by an element in $\mathbf{Z}_p$.

**Lemma 19 (For reduction).** *Let $\vec{s}$, $\vec{\mathsf{ct}}$, $\vec{c}$, $e_{\mathsf{Eval}}$, $B_{\mathsf{Eval}}$ be as in Corollary 18, and let $\mathsf{ct}' = (-\vec{a}', \vec{s} \cdot \vec{a}' + pe' + m')$ be a valid encoding of a message $m' \in \mathbf{Z}_p$ with noise $e'$ bounded by $B_e$. Let $B_{sm} = 2^\kappa B_e$ and $e_{sm} \hookleftarrow [-B_{sm}, B_{sm}]$ be a "smudging noise". Then, if $q > 2p^2\left(\left(2^\kappa + 1\right) B_e + B_{\mathsf{Eval}}\right)$, it is possible to add the smudging term $e_{sm}$ to $\mathsf{ct}'$, sum the result with the output of $\mathsf{Eval}\left(\vec{\mathsf{ct}}, \vec{c}\right)$, multiply the outcome by a coefficient bounded by $p$, and obtain a valid encoding of $k\left(\vec{m} \cdot \vec{c} + m'\right)$.*

*Proof.* The correctness of the message part comes immediately from performing homomorphic linear operations on encodings, and the final output is valid if the noise remains below a certain threshold. After adding the smudging term and performing the sum, the noise term is at most $B_e + B_{sm} + B_{\mathsf{Eval}} = \left(2^\kappa + 1\right) B_e + B_{\mathsf{Eval}}$. After the multiplication by a coefficient bounded by $p$, it is at most $p\left(\left(2^\kappa + 1\right) B_e + B_{\mathsf{Eval}}\right)$. Thus, the encoding is valid if:

$$p\left(\left(2^\kappa + 1\right) B_e + B_{\mathsf{Eval}}\right) < \frac{q}{2p}, \tag{2}$$

which immediately gives the result. □

**Conditions on the modulus $q$.** Corollaries 16 and 18 and Lemma 19 give the conditions that the modulus $q$ has to respect in order to allow for all the necessary computations. In particular, Corollary 16 gives the condition to be able to homomorphically evaluate a linear combination of fresh encodings through the algorithm Eval; Corollary 18 gives the condition to be able to add a smudging noise to the result of such an evaluation; Lemma 19 gives a condition that will have to be satisfied in the security reduction. They are ordered from the least stringent to the most stringent, so the condition that must be satisfied in the end is the one given by Lemma 19. Let $B_e$ be a bound on the absolute value of $e'$, then the following must hold:

$$q > 2p^2\left(\left(2^\kappa + 1\right) B_e + B_{\mathsf{Eval}}\right) \tag{3}$$

**Practical considerations.** A single encoded value has size $(n+1)\log q = \widetilde{O}(\lambda)$. Therefore, as long as the prover sends a constant number of encodings, the proof is guaranteed to be (quasi) succinct. As a matter of fact, we can generate the random vector $\vec{a}$ that composes the first term of the encoding by extending the output of a seeded PRG. This has been proven secure in the random oracle model [Gal13].

Although the scheme requires the noise terms to be sampled from a discrete Gaussian distribution, for practical purposes we can sample them from a bounded uniform distribution (see, e.g., [MP13] for a formal assessment of the hardness of LWE in this case). In particular, given $\chi_\sigma$, one can choose a coefficient $T \in \mathbf{N}$ such that $\Pr\left[x \leftarrow \chi_\sigma : |x| > T\sigma\right]$ is as small as desired. Finally, the error is sampled from $[-T\sigma, T\sigma]$.

Table 1 shows some practical parameters for which we believe our scheme to be post-quantum secure. The concrete estimate of the bits of security has been done using Albreicht's LWE test estimator[6] [APS15]. For a more extensive analysis of the asymptotic complexity, as well as the concrete efficiency estimates, we direct the reader towards Section 6.

[6] https://bitbucket.org/malb/lwe-estimator

Setup $\Pi.\mathsf{G}(1^\lambda, \mathsf{C})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$(v_0, \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathsf{C})$
$\beta, s \leftarrow_\$ \mathbf{F}$
Compute $\mathsf{crs}$ as per Eq. (4)
$\mathsf{vrs} := (\mathsf{sk}, s, \beta)$
**return** $(\mathsf{vrs}, \mathsf{crs})$

Verifier $\Pi.\mathsf{V}(\mathsf{vrs}, u, \pi)$

$(H, \widehat{H}, \widehat{V}, V_w, B_w) := \pi$
Read $(v_0, \ldots, v_m(x), t(x))$ from $\mathsf{vrs}$
$w_s := \mathsf{D}(V_w)$; $b_s := \mathsf{D}(B_w)$
$h_s := \mathsf{D}(H)$; $\widehat{h}_s := \mathsf{D}(\widehat{H})$
$\widehat{v}_s := \mathsf{D}(\widehat{V})$
$v_s := v_0(s) + \sum_i^{\ell_u} a_i v_i(s) + w_s$
Check Eqs. (eq-pke) to (eq-lin)
**return** $\mathsf{test\text{-}error}(s, B_w)$

Prover $\Pi.\mathsf{P}(\mathsf{crs}, u, w)$

$(v_0, \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathsf{C})$
$u := (a_1, \ldots, a_{\ell_u}) \in \{0,1\}^{\ell_u}$;
$w := (a_{\ell_u+1}, \ldots, a_m)$
$\nu(x) := v_0(x) + \sum_i^m a_i v_i(x) + \gamma t(x)$
$v_{\mathsf{mid}}(x) := \sum_{i>\ell_u}^m a_i v_i(x) + \gamma t(x)$
$h(x) = (\nu(x)^2 - 1)/t(x)$
// Compute the proof terms as per Eq. (6)
$H := \mathsf{Eval}((\mathsf{E}(s^i))_i,\ (h_i)_i) = \mathsf{E}(h(s))$
$\widehat{H} := \mathsf{Eval}((\mathsf{E}(\alpha s^i))_i,\ (h_i)_i) = \mathsf{E}(\alpha h(s))$
$\widehat{V} := \mathsf{Eval}((\mathsf{E}(\alpha s^i)_i,\ (\nu_i)_i) = \mathsf{E}(\alpha \nu(s))$
$B_w := \mathsf{Eval}((\mathsf{E}(\beta v_i(s)))_i \| (\mathsf{E}(\beta t(s))),\ (a_i)_i \| (\gamma))$
$V_w := \mathsf{Eval}((\mathsf{E}(s^i))_i,\ (v_{\mathsf{mid}\,i})_i) = \mathsf{E}(v_{\mathsf{mid}}(s))$
**return** $(H, \widehat{H}, \widehat{V}, V_w, B_w)$

**Fig. 6.** Our zk-SNARK protocol $\Pi$.

## 4 Our designated-verifier zk-SNARK

Let $\mathsf{Enc}$ be an encoding scheme (Definition 9). Let $\mathtt{C}$ be some circuit taking as input an $\ell_u$-bit string and outputting 0 or 1. Let $\ell := \ell_u + \ell_w$, where $\ell_u$ is the length of the "public" input, and $\ell_w$ the length of the private input. The value $m$ corresponds to the number of wires in $\mathtt{C}$ and $n$ to the number of fan-in 2 gates. Let $d := m + n$. We will construct a zk-SNARK scheme for any functions $\ell_u, \ell_w$ and families $\mathcal{R}_\lambda$ of relations $\mathcal{R}$ on pairs $(u, w) \in \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w}$ that can be computed by polynomial size circuits $\mathtt{C}$ with $m$ wires and $n$ gates. Our protocol is formally depicted in Figure 6.

**CRS generation.** The setup algorithm $\mathsf{G}$ takes as input some complexity $1^\lambda$ in unary form and the circuit $\mathtt{C} : \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$. It generates a square span program that verifies $\mathtt{C}$ by running:

$$(v_0(x), \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathsf{C})$$

Finally, it samples $\alpha, \beta, s \leftarrow \mathbf{F}$ such that $t(s) \neq 0$, and returns the CRS:

$$\begin{aligned} \mathsf{crs} := \big(\mathsf{ssp},\ \mathsf{pk},\ & \mathsf{E}(s), \ldots, \mathsf{E}(s^d), \\ & \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^d), \\ & \mathsf{E}(\beta t(s)), (\mathsf{E}(\beta v_i))_i \big) \end{aligned} \tag{4}$$

The verification string simply consists of $\mathsf{vrs} := (\mathsf{sk}, s, \beta)$.

**Prover.** The prover algorithm, on input some statement $u := (a_1, \ldots, a_{\ell_u})$, computes a witness $w := (a_{\ell_u+1}, \ldots, a_m)$ such that $(u, w) = (a_1, \ldots, a_m)$ is a satisfying assignment for the circuit $\mathtt{C}$. The $(a_i)_i$ are such that:

$$t(x) \text{ divides } \left(v_0(x) + \sum_i^m a_i v_i(x)\right)^2 - 1,$$

as per Theorem 2. Then, it samples $\gamma \leftarrow_{\$} \mathbf{F}$ and sets $\nu(x) := v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \gamma t(x)$. Let:

$$h(x) := \frac{(v_0(x) + \sum_i^m a_i v_i(x) + \gamma t(x))^2 - 1}{t(x)} = \frac{\nu(x)^2 - 1}{t(x)}, \tag{5}$$

whose coefficients can be computed from the polynomials provided in the $\mathsf{ssp}$. By affine evaluation it is possible to compute:

$$\begin{aligned} H &:= \mathsf{E}(h(s)), \qquad \widehat{H} := \mathsf{E}(\alpha h(s)), \qquad \widehat{V} := \mathsf{E}\left(\alpha \nu(s)\right), \\ V_w &:= \mathsf{E}\left(\sum_{i=\ell_u+1}^{m} a_i v_i(s) + \gamma t(s)\right), \quad B_w := \mathsf{E}\left(\beta\left(\sum_{i=\ell_u+1}^{m} a_i v_i(s) + \gamma t(s)\right)\right). \end{aligned} \tag{6}$$

In fact, $H$ - respectively, $\widehat{H}$ - can be computed from the encodings of $s, \ldots, s^d$ - respectively, $\alpha s, \ldots, \alpha s^d$ - and the coefficients of Equation (5). The element $\widehat{V}$ can be computed from the encodings of $\alpha s, \ldots, \alpha s^d$. Finally, $V_w$ - respectively, $B_w$ - can be computed from the encodings of $s, \ldots, s^d$ - respectively, $\beta t(s), \beta v_{\ell_u+1}(s), \ldots, \beta v_m(s)$. All these affine evaluations involve at most $d$ terms and the coefficients are bounded by $p$. Using the above elements, the prover returns a proof $\pi := (H, \widehat{H}, \widehat{V}, V_w, B_w)$.

**Verifier.** Upon receiving a proof $\pi$ and a statement $u = (a_1, \ldots, a_{\ell_u})$, the verifier proceeds with the following verifications. First, it uses the quadratic root detection algorithm of the encoding scheme $\mathsf{Enc}$ to verify that the proof satisfies:

$$\widehat{h}_s - \alpha h_s = 0 \text{ and } \widehat{v}_s - \alpha v_s = 0, \tag{eq-pke}$$

$$(v_s^2 - 1) - h_s t_s = 0, \tag{eq-div}$$

$$b_s - \beta w_s = 0. \tag{eq-lin}$$

where $(h_s, \widehat{h}_s, \widehat{v}_s, w_s, b_s)$ are the values encoded in $(H, \widehat{H}, \widehat{V}, V_w, B_w) := \pi$ and $v_s$ is an encoding of $v_0 + \sum_i^{\ell_u} a_i v_i(s) + w_s$ as per Fig. 6. Then, the verifier checks whether it is still possible to perform some homomorphic operations, using the $\mathsf{test\text{-}error}$ procedure described in Section 2, and implemented in Figure 5 for the specific case of lattice encodings. More precisely, the verifier tests whether it is still possible to add another encoding and multiply the result by an element bounded by $p$, without compromising the correctness of the encoded element. This will guarantee the existence of a reduction in the knowledge soundness proof of Section 5.2. If all above checks hold, return **true**. Otherwise, return **false**.

*Remark 20.* Instantiating our encoding scheme on top of a "noisy" encryption scheme like Regev's introduces multiple technicalities that affect the protocol, the security proof, and the parameters' choice. For instance, in order to compute a linear combination of $d$ encodings via $\mathsf{Eval}$ we need to scale down the error parameter and consequently increase the parameters $q$ and $n$ in order to maintain correctness and security. Similarly, for the proof to hold, we need the adversary to be able to perform the same amount of homomorphic operations both in the real protocol as well as in the reductions where we synthesize a CRS based on a $q$-PDH challenge. All these issues will be formally addressed in Section 6.

## 5 Proofs of security

In this section, we prove our main theorem:

**Theorem 21.** *If the* $q$-PKE, $q$-PKEQ *and* $q$-PDH *assumptions hold for the encoding scheme* $\mathsf{Enc}$, *the protocol* $\Pi$ *on* $\mathsf{Enc}$ *is a zk-SNARK with statistical completeness, statistical zero-knowledge and computational knowledge soundness.*

*Proof (of statistical completeness).* Statistical completeness is straightforward from the encoding scheme and Corollary 16. □

Simulator $\mathsf{Sim}(\mathsf{vrs}, u)$

$(\mathsf{sk}, s, \beta) := \mathsf{vrs}; \quad (a_1, \ldots, a_{\ell_u}) := u$

$\delta_w \leftarrow\!\!\$\ \mathbf{F}$

$h(s) := \left((v_0(x) + \sum_i^{\ell_u} a_i v_i(x) + \delta_w)^2 - 1\right) \ / \ t(x)$

$H \leftarrow \mathsf{E}(h(s)); \quad \widehat{H} \leftarrow \mathsf{E}(\alpha h(s))$

$V_w \leftarrow \mathsf{E}(\delta_w); \quad B_w \leftarrow \mathsf{E}(\alpha v_0(s) + \sum_i^{\ell_u} a_i \alpha v_i(s) + \alpha \delta_w)$

$\widehat{V} \leftarrow \mathsf{E}(\beta \delta_w)$

**return** $(H, \widehat{H}, \widehat{V}, V_w, B_w)$

**Fig. 7.** Simulator for Zero-Knowledge.

### 5.1 Zero-Knowledge

To obtain a zero-knowledge protocol, we do two things: we add a smudging term to the noise of the encoding, in order to make the distribution of the final noise independent of the coefficients $a_i$, and we add randomized factors of the target polynomial $t(x)$ to the answers, in order to achieve zero-knowledge.

*Proof (of zero-knowledge).* The simulator for zero-knowledge is shown in Figure 7. Checking that the proof output by $\mathsf{Sim}$ indeed verifies is trivial. Statistical zero-knowledge follows immediately by observing that both a simulated argument and a real one follow the same distribution. First, we note that in the real world, since $\gamma$ is chosen uniformly at random in $\mathbf{F}$, so is $\gamma t(s)$, because $t(s) \neq 0$. Therefore, $V_w$ is an encoding of some uniformly random value $w_s$. Once $V_w$ is fixed, the verification equation unequivocally defines $B_w$ (which is an encryption of $\beta w_s$ in both worlds), $\widehat{V}$ (which is an encryption of $\alpha v_s$ for $v_s = v_0(s) + \sum_i a_i v_i(s) + w_s$ in both worlds) and $H, \widehat{H}$, which follow the same distribution in both worlds from for the same reasoning as above. □

The zero-knowledge property is certainly interesting, but SNARKs are already appealing on their own, even without this feature. If we are only interested in building SNARKs (and not zk-SNARKs), we can simplify the protocol by removing $\gamma t(x)$ from the computation of $h(x)$. Also, we do not need to "smudge out" the noise anymore, which leads to better bounds on the noise growth. This means that we can scale down our encoding space and make the protocol more efficient. For this reason, in Table 1 we show some choices of parameters, both with and without the zero-knowledge requirement. In the same way, the simulator $\mathsf{Sim}$ must sample the noise from a distribution that is statistically close to the one used in the real world. Concretely, Corollary 18 guarantees that the smudged encoding output by the prover is statistically indistinguishable from the smudged simulated values.

### 5.2 Knowledge Soundness

Before diving into the technical details of the proof of soundness, we provide some intuition in an informal sketch of the security reductions: the CRS for the scheme contains encodings of $\mathsf{E}(s), \ldots, \mathsf{E}(s^d)$, as well as encodings of these terms multiplied by some field elements $\alpha, \beta \in \mathbf{F}$. The scheme requires the prover $\mathsf{P}$ to exhibit encodings computed homomorphically from such CRS.

The reason why we require the prover to duplicate its effort w.r.t. $\alpha$ is so that the simulator in the security proof can extract representations of $\widehat{V}$, $\widehat{H}$ as degree-$d$ polynomials $v(x)$, $h(x)$ such that $v(s) = v_s, h(s) = h_s$, by the $q$-PKE assumption (for $q = d$). The assumption also guarantees that this extraction is efficient. This explains the first quadratic root detection check Equation (eq-pke) in the verification algorithm.

Suppose an adversary manages to forge a SNARK of a false statement and pass the verification test. Then, the soundness of the square span program (Theorem 2) implies that, for the extracted polynomials $v(x), h(x)$ and for the new defined polynomial $v_{\mathsf{mid}}(x) := v(x) - v_0(x) - \sum_i^{\ell_u} a_i v_i(x)$, one of the following must be true:

i. $h(x)t(x) \neq v^2(x) - 1$, but $h(s)t(s) = v^2(s) - 1$, from Equation (eq-div);
ii. $v_{\mathsf{mid}}(x) \notin \mathrm{Span}(v_{\ell_u+1}, \ldots, v_m)$, but $B_w$ is a valid encoding of $\mathsf{E}(\beta v_{\mathsf{mid}}(s))$, from Equation (eq-lin).

If the first case holds, then $p(x) := (v^2(x) - 1) - h(x)t(x)$ is a nonzero polynomial of degree some $k \leqslant 2d$ that has $s$ as a root, since the verification test implies $(v^2(s) - 1) - h(s)t(s) = 0$. The simulator can use $p(x)$ to solve $q$-PDH for $q \geqslant 2d - 1$: using the fact that $\mathsf{E}(0) = \mathsf{E}\left(s^{q+1-k}p(s)\right)$ and subtracting off encodings of lower powers of $s$ to get $\mathsf{E}(s^{q+1})$.

To handle the second case, i.e., to ensure that $v_{\mathsf{mid}}(x)$ is in the linear span of the $v_i(x)$'s with $\ell_u < i \leqslant m$ we use an extra scalar $\beta$, supplement the CRS with the terms $\{\mathsf{E}(\beta v_i(s))\}_{i>\ell_u}, \mathsf{E}(\beta t(s))$, and require the prover to present (encoded) $\beta v_{\mathsf{mid}}(s)$ in its proof. An adversary against $q$-PDH will choose a polynomial $\beta(x)$ convenient to solve the given instance. More specifically, it sets $\beta(x)$ with respect to the set of polynomials $\{v_i(x)\}_{i>\ell_u}$ such that the coefficient for $x^{q+1}$ in $\beta(x)v_{\mathsf{mid}}(x)$ is non-zero. Then, for the values in the $\mathsf{crs}$ it uses $\beta := \beta(s)$. All these allow it to run the SNARK adversary and to obtain from its output $B_w$ an encoding of $s^{q+1}$ and thus solve $q$-PDH.

*Proof (of computational knowledge soundness).* Let $\mathcal{A}^{\Pi}$ be the $\mathsf{PPT}$ adversary in the game for knowledge soundness (Figure 1) able to produce a proof $\pi$ for which $\Pi.\mathsf{V}$ returned **true**. We first claim that it is possible to extract the coefficients of the polynomial $v(x)$ corresponding to the values $v_s$ encoded in $V$. The setup algorithm first generates the parameters $(\mathsf{pk}, \mathsf{sk})$ of an encoding scheme $\mathsf{Enc}$ and picks $\alpha, \beta, s \in \mathbf{F}$, which are used to compute $\mathsf{E}(s), \ldots, \mathsf{E}(s^d), \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^d)$. Fix some circuit $\mathsf{C}$, and let $\mathsf{ssp}$ be an SSP for $\mathsf{C}$. Let $\mathcal{A}^{\mathrm{PKE}}$ be the $d$-$\mathsf{PKE}$ adversary, that takes as input a set of encodings:

$$\sigma := \left(\mathsf{pk}, \mathsf{E}(s), \ldots, \mathsf{E}(s^d), \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^d)\right).$$

The auxiliary input generator $\mathcal{Z}$ is a $\mathsf{PPT}$ machine that upon receiving as input $\sigma$, samples $\beta \leftarrow_{\$} \mathbf{Z}_p$, constructs the remaining terms of the CRS (as per Equation (4)), and outputs them in $z$. Thus, $\mathcal{A}^{\mathrm{PKE}}$ sets $\mathsf{crs} := (\mathsf{ssp} \| \sigma \| z)$ and invokes $\mathcal{A}^{\Pi}(\mathsf{crs})$. As a result, it obtains a proof $\pi = (H, \widehat{H}, \widehat{V}, V_w, B_w)$. On this proof, it computes:

$$V := \mathsf{Eval}\left((V_w), (1), v_0 + \sum_{i=0}^{\ell_u} a_i v_i(s)\right) = \mathsf{E}\left(v_0 + \sum_{i=0}^{\ell_u} a_i v_i(s) + w_s\right), \tag{7}$$

where $(V_w)$ – respectively $(1)$ – is the vector containing only $V_w$ – respectively $1$ –, and $w_s$ is the element encoded in $V_w$. Finally, $\mathcal{A}^{\mathrm{PKE}}$ returns $(\widehat{V}, V)$. If the adversary $\mathcal{A}$ output a valid proof, then by verification equation Eq. (eq-pke) it holds that the two encodings $(V, \widehat{V})$ encode values $v_s, \widehat{v}_s$ such that $\widehat{v}_s - \alpha v_s = 0$. Therefore, by $q$-PKE assumption there exists an extractor $\mathsf{Ext}^{\mathrm{PKE}}$ that, using the same input (and random coins) of $\mathcal{A}^{\mathrm{PKE}}$, outputs a vector $(c_0, \ldots, c_d) \in \mathbf{F}^{d+1}$ such that $V$ is an encoding of $\sum_{i=0}^d c_i s^i$ and $\widehat{V}$ is an encoding of $\sum_{i=0}^d \alpha c_i s^i$. In the same way, it is possible to recover the coefficients of the polynomial $h(x)$ used to construct $(H, \widehat{H})$, the first two elements of the proof of $\mathcal{A}^{\Pi}$ (again, by Eq. (eq-pke)).

Our witness-extractor $\mathsf{Ext}^{\Pi}$, given $\mathsf{crs}$, emulates the extractors $\mathsf{Ext}^{\mathrm{PKE}}$ above on the same input $\sigma$, using as auxiliary information $z$ the rest of the CRS given as input to $\mathsf{Ext}^{\Pi}$. By the reasoning discussed above, $\mathsf{Ext}^{\Pi}$ can recover $(c_0, \ldots, c_d)$ coefficients extracted from the encodings $(V, \widehat{V})$. Consider now the polynomial $v(x) := \sum_{i=0}^d c_i x^i$. If it is possible to write the polynomial as $v(x) = v_0(x) + \sum_i^m a_i v_i(x) + \delta t(x)$ such that $(a_1, \ldots, a_m) \in \{0,1\}^m$ satisfies the assignment for the circuit $\mathsf{C}$ with $u = (a_1, \ldots, a_{\ell_u})$, then the extractor returns the witness $w = (a_{\ell_u+1}, \ldots, a_m)$.

With overwhelming probability, the extracted polynomial $v(x) := \sum_{i=0}^d c_i x^i$ does indeed provide a valid witness $w$. Otherwise, there exists a reduction to $q$-PDH that uses the SNARK adversary

$\mathcal{A}^{\Pi}$. Define the polynomial:

$$v_{\mathsf{mid}}(x) \coloneqq v(x) - v_0(x) - \sum_{i=0}^{\ell_u} a_i v_i(x)$$

We know by definition of SSP (Definition 1) and by Theorem 2 that $\mathsf{C}$ is satisfiable if and only if:

$$t(x) \mid v^2(x) - 1 \;\wedge\; v_{\mathsf{mid}}(x) = \sum_{i}^{d} c_i x^i - v_0(x) - \sum_{i}^{\ell_u} a_i v_i(x) \in \mathrm{Span}(v_{\ell_u+1}, \ldots, v_m, t)$$

Therefore, by contradiction, if the adversary $\mathcal{A}^{\Pi}$ does not know a witness $w \in \{0,1\}^{m-\ell_u}$ for $u$ (such that $(u, w) \in \mathcal{R}_{\mathsf{C}}$), knowing that the two verification checks Eq. (eq-div) and Eq. (eq-lin) passed, we have that either one of the following two cases must hold:

i. $t(x)h(x) \neq v^2(x) - 1$, but $t(s)h(s) = v^2(s) - 1$; or
ii. $v_{\mathsf{mid}}(x) \notin \mathrm{Span}(v_{\ell_u+1}, \ldots, v_m, t)$, but $B_w$ is an encoding of $\beta v_{\mathsf{mid}}(s)$.

Let $\mathcal{B}^{\mathrm{PDH}}$ be an adversary against the $q$-PDH assumption. Given a $q$-PDH challenge:

$$\left(\mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}\left(s^{q+2}\right), \ldots, \mathsf{E}\left(s^{2q}\right)\right), \qquad \text{for } q \in \{2d-1, d\}$$

adversary $\mathcal{B}^{\mathrm{PDH}}$ samples uniformly at random $\alpha \leftarrow_{\$} \mathbf{F}$, and defines some $\beta \in \mathbf{F}$ (that we will formally construct later) and constructs a CRS as per Equation (4). There are some subtleties in how $\mathcal{B}^{\mathrm{PDH}}$ generates the value $\beta$. In fact, $\beta$ can be generated without knowing its value explicitly, but rather knowing its representation over the power basis $\left\{s^i\right\}_{i=0, i\neq q+1}^{2q}$ – that is, knowing a polynomial $\beta(x)$ and its evaluation in $s$. Some particular choices of $\beta$ will allow us go provide a solution for a $q$-PDH challenge. Using the CRS generated, $\mathcal{B}^{\mathrm{PDH}}$ invokes the adversary $\mathcal{A}^{\Pi}$ as well as the extractor $\mathsf{Ext}^{\Pi}$ on the generated CRS, thus obtaining a proof $\pi$ and the linear combination used by the prover for the polynomials $h(x), v(x)$ and also extracts a witness for the truth of the statement being proved.

In order to simulate the verification oracle and to answer the verification queries of $\mathcal{A}^{\Pi}$, $\mathcal{B}^{\mathrm{PDH}}$ has to compare its encodings (obtained from the extracted coefficients and its input) with $\mathcal{A}$'s proof terms, accepts if the terms match, and rejects otherwise. Because the encoding scheme is not deterministic, adversary $\mathcal{B}^{\mathrm{PDH}}$ invokes the PKEQ extractor and simulates the verification oracle correctly with overwhelming probability. By game of knowledge soundness (Figure 1), if the proof output by the adversary $\mathcal{A}^{\Pi}$ verifies, then we can distinguish two cases:

i. The extracted polynomials $h(x)$ and $v(x)$ satisfy $t(x)h(x) \neq v^2(x) - 1$, but $t(s)h(s) = v^2(s) - 1$. By $q$-PDH assumption this can happen only with negligible probability. We define $p(x) = v^2(x) - 1 - t(x)h(x)$, that in this case is a non-zero polynomial of degree $k \leqslant 2d$ having $s$ as a root. Let $p_k$ be the highest nonzero coefficient of $p(x)$. Write $\tilde{p}(x) = x^k - p_k^{-1} \cdot p(x)$. Since $s$ is a root of $x^k - \tilde{p}(x)$, it is a root of $x^{q+1} - x^{q+1-k}\tilde{p}(x)$. $\mathcal{B}^{\mathrm{PDH}}$ solves $q$-PDH by computing $\mathsf{E}(s^{q+1}) = \mathsf{E}(s^{q+1-k}\tilde{p}(s))$ for $q = 2d - 1$. Since $\deg(\tilde{p}) \leqslant k - 1$, the latter is a known linear combination of encodings $\mathsf{E}(s), \ldots, \mathsf{E}(s^q)$ which are available from the $q$-PDH challenge. More precisely, $\mathcal{B}^{\mathrm{PDH}}$ will compute $\mathsf{Eval}((\mathsf{E}(s^{i+q+1-k}))_i, (\tilde{p}_i)_i^{2d-1})$ on *fresh* encodings $\mathsf{E}(s), \mathsf{E}(s^2), \ldots, \mathsf{E}(s^q)$ solving the $q$-PDH challenge for $q \geqslant 2d - 1$.
ii. In the second case, suppose that the polynomial $v_{\mathsf{mid}}$ extracted as previously described cannot be expressed as a linear combination of $\{v_{\ell_u+1}, \ldots, v_m, t\}$. The proof still passes the verification, so we have a consistent value for $B_w \in \{\mathsf{E}(\beta v_{\mathsf{mid}}(s))\}$.
$\mathcal{B}^{\mathrm{PDH}}$ generates a uniformly random polynomial $a(x)$ of degree $q$ subject to the constraint that all of the polynomials $a(x)t(x)$ and $\{a(x)v_i(x)\}_{i=\ell_u+1}^{m}$ have coefficient 0 for $x^{q+1}$. Remark that for $q = d$, there are $q - (m - \ell_u) > 0$ degrees of freedom in choosing $a(x)$.
$\mathcal{B}^{\mathrm{PDH}}$ defines $\beta$ to be the evaluation of $a(x)$ in $s$, i.e. $\beta \coloneqq a(s)$. Remark that $\mathcal{B}^{\mathrm{PDH}}$ does not know $s$ explicitly, but having access to the encodings of $2q - 1$ powers of $s$, it is able to

generate valid encodings $(\mathsf{E}(\beta v_i(s)))_i$ and $\mathsf{E}(\beta t(s))$ using $\mathsf{Eval}$. Note that, by construction of $\beta$, this evaluation is of $d+1$ elements in $\mathbf{F}$ and that the $(q+1)$-th power of $s$ is never used. Now, since $v_{\mathsf{mid}}(x)$ is not in the proper span, then the coefficient of degree $q+1$ of $xa(x)v_{\mathsf{mid}}(x)$ must be nonzero with overwhelming probability $1-1/|\mathbf{F}|$. The term $B_w$ of the proof must encode a known polynomial in $s$: $\sum_{i=0}^{2q} b_i s^i := \beta v_{\mathsf{mid}}(s) = a(s)v_{\mathsf{mid}}(s)$ where the coefficient $b_{q+1}$ is non-trivial. $\mathcal{B}^{\mathrm{PDH}}$ can subtract off encodings of multiples of other powers of $s$ to recover $\mathsf{E}(s^{q+1})$ and break $q$-PDH. This requires an evaluation on *fresh* encodings:

$$\mathsf{Eval}\left((\mathsf{E}(s^i))_{\substack{i=0\\ i\neq q+1}}^{q+d}, (-b_i)_{\substack{i=0\\ i\neq q+1}}^{q+d}\right). \tag{8}$$

Adding the above to $B_w$ and multiplying by the inverse of the $(q+1)$-th coefficient (using once again $\mathsf{Eval}$) will provide a solution to the $q$-PDH problem for $q=d$.

Since the two above cases are not possible by $q$-PDH assumption, $\mathsf{Ext}^{\Pi}$ extracts a valid witness if the proof of $\mathcal{A}^{\Pi}$ is valid. □

As previously mentioned in Remark 7, the proof of knowledge soundness allows oracle access to the verification procedure. In the context of a weaker notion of soundness, the proof is almost identical, except that there is no need for the $\mathcal{B}^{\mathrm{PDH}}$ adversary to simulate the verification oracle relying on the $q$-PKEQ assumption.

## 6 Efficiency and concrete parameters

The prover's computations are bounded by the security parameter and the size of the circuit, i.e., $\mathsf{P} \in O(\lambda d)$. As in [GGPR13, DFGK14], the verifier's computations depend solely on the security parameter, i.e., $\mathsf{V} \in O(\lambda)$. The proof consists of a constant number (precisely, 5) of LWE encodings, i.e., $|\pi| = 5 \cdot \widetilde{O}(\lambda)$.

Using the propositions from Section 3 and knowing the exact number of homomorphic operations that need to be performed in order to produce a proof, we can now attempt at providing some concrete parameters for our encoding scheme.

We fix the statistical security parameter $\kappa := 32$, as already done in [DM15, CGGI16]. We fix the circuit size $d := 2^{15}$, which is sufficient for some practical applications; see [BCG+14a, PHGR13].

For a first attempt at implementing our solution, we assume a weaker notion of soundness, i.e. that in the KSND game the adversary does not have access to a verification oracle (cf. Figure 1). Concretely, this means that the only bound in the size of $p$ is given by the guessing probability of the witness, and the guessing of a field element. We thus fix $p = 2^{32}$ for the size of the message space.

The CRS is composed of encodings of different nature: some of them are fresh $\big(\mathsf{E}(s), \ldots, \mathsf{E}(s^d)\big)$, some happen to be stale in the construction of $\mathcal{A}^{\mathrm{PKE}}$ and the construction of $\mathcal{B}^{\mathrm{PDH}}$ Section 5.2 (Item i.) $(\mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^d))$, and some are stale from the construction of $\mathcal{B}^{\mathrm{PDH}}$ Section 5.2 (Item ii.) $(\mathsf{E}(\beta t(s)), (\mathsf{E}(\beta v_i(s)))_i)$. They are displayed in Figure 8. Since, as we have seen, $\mathcal{B}^{\mathrm{PDH}}$ manipulates the $q$-PDH challenge via homomorphic operations, we must guarantee that the protocol adversary can *at least* perform the same number of homomorphic operations as in the real-world protocol. Therefore, in the real protocol, we must intentionally increase the magnitude of the noise in the CRS: the terms $\mathsf{E}(\alpha s^i)$ (with $i = 0, \ldots, d$) are generated by multiplying the respective *fresh* encoding $\mathsf{E}(s^i)$ by a term bounded by $p$; the terms $\mathsf{E}(\beta t(s)), \{\mathsf{E}(\beta v_i(s))\}_i$ instead are generated via $\mathsf{Eval}$ of $d+1$ elements with coefficients bounded by $p$. Concretely, when encoding these elements using the encoding scheme of Section 3, the error for $\mathsf{E}(\alpha s^i)$ is sampled from $p \cdot \chi_\sigma$; the error for $\mathsf{E}(\beta t(s)), \mathsf{E}(\beta v_i(s)))$ is sampled from $(p\sqrt{d+1}) \cdot \chi_\sigma$.

The proof $\pi$ consists of five elements $(H, \widehat{H}, \widehat{V}, V_w, B_w)$, as per Equation (6). $H$ and $V_w$ are computed using an affine function on $d$ encodings with coefficients modulo $p$; $\widehat{H}, \widehat{V}$ are computed using a linear function on $d+1$ encodings with coefficients modulo $p$; finally, $B_w$ is computed using a linear combination of $m - \ell_u$ encodings with coefficients in $\{0, 1\}$, except the last one which is

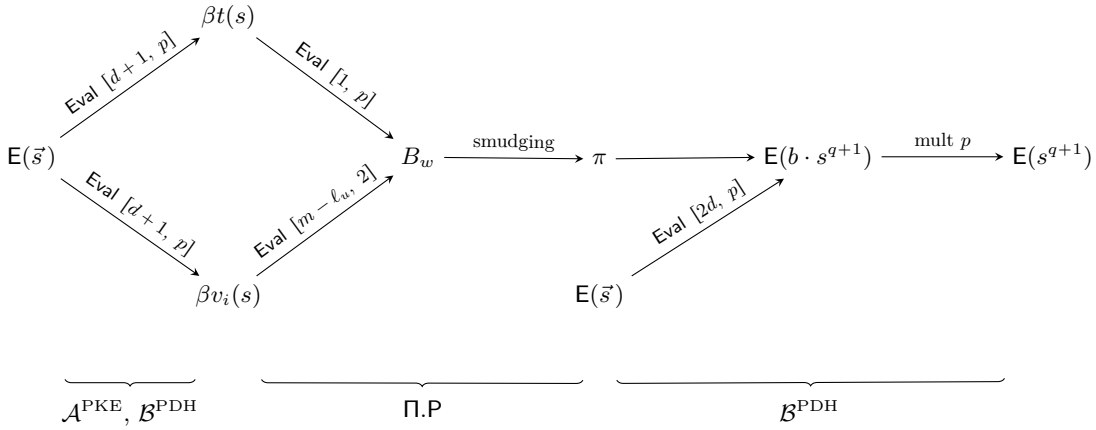


**Fig. 8.** Summary of evaluations in the security proof. The leftmost part of the figure refers to the construction of adversaries for $q$-PKE and $q$-PDH; the central part refers to the protocol itself (i.e., the construction of the proof $\pi$); the rightmost part refers to the construction of the adversary for $q$-PDH (Section 5.2 - Item ii.). The syntax $\mathsf{Eval}\,[d, p]$ denotes an homomorphic evaluation on $d$ encodings with coefficients in $\mathbf{Z}_p$. $\mathsf{E}(\vec{s})$ denotes the PDH challenge.

modulo $p$. Overall, the term that carries the highest load of homomorphic computations is $B_w$. The generation of $B_w$ is outlined in Figure 8, and to it (as well as to the other proof terms) we add a smudging term so for constructing a zero-knowledge proof $\pi$.

In the construction of the adversary $\mathcal{B}^{\mathrm{PDH}}$ (Item ii.) we need to perform some further homomorphic operations on the proof element $B_w$ in order to solve the $q$-PDH challenge, namely one addition (Equation (8)) and one multiplication by a known scalar $b$ bounded by $p$. The result of the first operation is denoted by $\mathsf{E}(b \cdot s^{q+1})$ in Figure 8; the final result is the solution to the $q$-PDH challenge.

We now outline the calculations that we use to choose the relevant parameters for our encoding scheme. In particular, we will focus on the term $B_w$ since, as already stated, it is the one that is involved in the largest number of homomorphic operations. The chain of operations that need to be supported is depicted in Figure 8: we now analyze them one by one. The correctness of the other terms follows directly from Corollary 16.

First of all, the terms $(\beta v_i(s))_{i \in [m-\ell_u]}$ and $\beta t(s)$ are produced through the algorithm $\mathsf{Eval}$ executed on $d+1$ *fresh* encodings with coefficients modulo $p$. Let $\sigma$ be the discrete Gaussian parameter of the noise terms in fresh encodings; then, by Pythagorean additivity, the Gaussian parameter of encodings output by this homomorphic evaluation is $\sigma_{\mathsf{Eval}} := p\sigma\sqrt{d+1}$. Then the term $\beta t(s)$ is multiplied by a coefficient in $\mathbf{Z}_p$, and the result is added to a subset sum of the terms $(\beta v_i(s))_i$, i.e., a weighted sum with coefficients in $\{0, 1\}$. It is trivial to see that, for the first term, the resulting Gaussian parameter is bounded by $p\sigma_{\mathsf{Eval}}$, whereas for the second term it is bounded by $\sigma_{\mathsf{Eval}}\sqrt{m-\ell_u}$. The parameter of the sum of these two terms is then bounded by $\sigma_{B_w} := \sigma_{\mathsf{Eval}}\sqrt{p^2+m-\ell_u}$. Let us then consider a constant factor $T$ for "cutting the Gaussian tails", i.e., such that the probability of sampling from the distribution and obtaining a value with magnitude larger than $T$ times the standard deviation is as small as desired. We can then write that the absolute value of the error in $B_w$ is bounded by $T\sigma_{B_w}$. At this point we add a *smudging* term, which amounts to multiplying the norm of the noise by $(2^\kappa + 1)$ (cf. Corollary 18). Finally, the so-obtained encoding has to be summed with the output of an $\mathsf{Eval}$ invoked on $2d$ fresh encodings with coefficients modulo $p$ and multiplied by a constant in $\mathbf{Z}_p$. It is easy to calculate that the final noise is then bounded by $Tp\sigma_{B_w}(2^\kappa + 1) + Tp\sigma_{\mathsf{Eval}}$ (cf. Lemma 19). By substituting the values of $\sigma_{\mathsf{Eval}}$, $\sigma_{B_w}$, remembering that $\sigma := \alpha q$ and imposing the condition for having a valid encoding, we

obtain

$$Tp^2\alpha q\sqrt{d+1}\left(\sqrt{p^2+m-\ell_u}\,(2^\kappa+1)+1\right) < \frac{q}{2p}.$$

The above corresponds to Equation (3) with bounds $Be := T\sigma_{B_w}$ and $B_{\mathsf{Eval}} := T\sigma_{\mathsf{Eval}}$. By simplifying $q$ and isolating $\alpha$, we get:

$$\alpha < \frac{1}{2Tp^3\sqrt{d+1}\left(\sqrt{p^2+m-\ell_u}\,(2^\kappa+1)+1\right)}.$$

With our choice of parameters and by taking $T = 8$, we can select for instance $\alpha = 2^{-180}$.

Once $\alpha$ and $p$ are chosen, we select the remaining parameters $q$ and $n$ in order to achieve the desired level of security for the LWE encoding scheme. To do so, we take advantage of Albrecht's estimator [APS15] which, as of now, covers the following attacks: meet-in-the-middle exhaustive search, coded-BKW [GJS15], dual-lattice attack and small/sparse secret variant [Alb17], lattice-reduction with enumeration [LP11], primal attack via uSVP [AFG14, BG14], Arora-Ge algorithm [AG11] using Gröbner bases [ACFP14]. Some possible choices of parameters are reported in Table 1.

Finally, based on these parameters, we can concretely compute the size of the CRS[7] and that of the proof $\pi$. The CRS is composed of $d + (d+1) + (m+1)$ encodings, corresponding to the encodings of the $d$ powers of $s$, the encodings of $\alpha$ multiplied by the $d+1$ powers of $s$, the $m$ encodings of $(\beta v_i)_i$, and the encoding of $\beta t(s)$. This amounts to $(2d+m+2)$ LWE encodings, each of which has size $(n+1)\log q$ bits[8]. For the calculations, we bound $m$ by $d$ and state that the size of the CRS is that of $(3d+2)$ LWE encodings. From an implementation point of view, as already stated in Section 3, we can consider LWE encodings $(\vec{a}, b) \in \mathbf{Z}_q^{n+1}$ where the vector $\vec{a}$ is the output of a seeded PRG. Therefore, the communication complexity is greatly reduced, as sending an LWE encoding just amounts to sending the seed for the PRG and the value $b \in \mathbf{Z}_q$. For security to hold, we can take the size of the seed to be $\lambda$ bits, thus obtaining the final size of the CRS: $(3d+2)\log q + \lambda$ bits. The proof $\pi$ is composed of 5 LWE encodings, therefore it has size $|\pi| = 5(n+1)\log q$ bits. Note that in this case we cannot trivially use the same trick with the PRG, since the encodings are produced through homomorphic evaluations.

**Open problems.** We list here some directions for future research. First of all, the proposed approach requires a very large message space for the SNARK protocol to go through. It might be possible to soften this requirement by employing some decomposition techniques, e.g., by encoding terms bit-by-bit. We leave exploring this kind of optimizations as an interesting open problem. Another natural question that comes up is whether it is possible to have a post-quantum designated-verifier SNARK from QAP in the same spirit of Pinocchio [PHGR13]. A final, more broad, open question is whether it is possible to do publicly-verifiable SNARKs from lattice assumptions. It seems difficult to achieve this without some bilinear pairing map. However, the discovery of such a map would constitute a major breakthrough in cryptography as, among the other things, it would allow for indistinguishability obfuscation [BGI+01] and multilinear maps [BS02].

**Acknowledgments.** We would like to thank Balthazar Bauer and Florian Bourse for insightful discussions. Rosario Gennaro was supported by NSF Award no. 1565403. Michele Minelli was supported by European Union's Horizon 2020 research and innovation programme under grant agreement no. H2020-MSCA-ITN-2014-643161 ECRYPT-NET. Anca Nitulescu was supported by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud). Michele Orrù was supported by ERC grant 639554 (project aSCEND).

[7] We take into account only the encodings that are contained in the CRS. The other terms have considerably smaller impact on its size or can be agreed upon offline (e.g., the $\mathsf{SSP}$).

[8] Note that the magnitude of the noise term, i.e., whether the encoding is fresh or stale, has no impact on the size of an encoding. This size is a function only of $n$ (the number of elements in the vector) and the modulus $q$.

## References


ABL+17. Divesh Aggarwal, Gavin K Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them. *arXiv preprint arXiv:1710.10377*, 2017.

ACFP14. Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. http://eprint.iacr.org/2014/1018.

AFG14. Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

AG11. Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.

Alb17. Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, May 2017.

APS15. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

Ban95. Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices inr n. *Discrete & Computational Geometry*, 13(2):217–231, 1995.

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

BCG+14a. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

BCG+14b. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. Cryptology ePrint Archive, Report 2014/349, 2014. http://eprint.iacr.org/2014/349.

BCI+13. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

BG93. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

BG14. Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

BGGK17. Dan Boneh, Rosario Gennaro, Steven Goldfeder, and Sam Kim. A lattice-based universal thresholdizer for cryptographic systems. Cryptology ePrint Archive, Report 2017/251, 2017. http://eprint.iacr.org/2017/251.

BGI+01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

BISW17. Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 247–277. Springer, Heidelberg, May 2017.

BISW18. Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal snargs via linear multi-prover interactive proofs. Cryptology ePrint Archive, Report 2018/133, 2018. https://eprint.iacr.org/2018/133.

BS02. Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.

BSBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

DM15. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

Gal13. Steven D. Galbraith. Space-efficient variants of cryptosystems based on learning with errors. preprint, 2013. https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

GJS15. Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

KW93. M. Karchmer and A. Wigderson. On span programs. In IEEE Computer Society Press, editor, *In Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111, Gaithersburg, MD, USA, 1993. IEEE Computer Society Press.

LP11. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

MP13. Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.