Rinocchio: SNARKs for Ring Arithmetic

Chaya Ganesh[1], Anca Nitulescu[2], and Eduardo Soria-Vazquez[3]

[1] Indian Institute of Science, India.
[2] Protocol Labs, USA.
[3] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE.**
chaya@iisc.ac.in, anca.nitulescu@protocol.ai, eduardo.soria-vazquez@tii.ae

Abstract. Succinct non-interactive arguments of knowledge (SNARKs) enable non-interactive efficient verification of NP computations and admit short proofs. However, all current SNARK constructions assume that the statements to be proven can be efficiently represented as either Boolean or arithmetic circuits over finite fields. For most constructions, the choice of the prime field $\mathbb{F}_p$ is limited by the existence of groups of matching order for which secure bilinear maps exist.
In this work we overcome such restrictions and enable verifying computations over rings. We construct the first designated-verifier SNARK for statements which are represented as circuits over a broader kind of commutative rings.
Our contribution is threefold:

1. We first introduce Quadratic Ring Programs (QRPs) as a characterization of NP where the arithmetic is over a ring.
2. Second, inspired by the framework in Gennaro, Gentry, Parno and Raykova (EUROCRYPT 2013), we design SNARKs over rings in a modular way. We generalize pre-existent assumptions employed in field-restricted SNARKs to encoding schemes over rings. As our encoding notion is generic in the choice of the ring, it is amenable to different settings.
3. Finally, we propose two applications for our SNARKs.
   – Our first application is verifiable computation over encrypted data, specifically for evaluations of Ring-LWE-based homomorphic encryption schemes.
   – In the second one, we use Rinocchio to naturally prove statements about circuits over e.g. $\mathbb{Z}_{2^{64}}$, which closely matches real-life computer architectures such as standard CPUs.

** Work partially done while at Department of Computer Science, Aarhus University, Aarhus, Denmark.

Table of Contents

1 Introduction

Succinct Non-interactive ARguments of Knowledge (SNARKs) are non-interactive proof systems with short proofs that can be verified very efficiently, that show knowledge of a witness for a given NP statement. Moreover, Zero-knowledge SNARKs (zk-SNARKs) also guarantee that no information is revealed beyond the validity of the statement. Since their introduction, zk-SNARKs proofs have been shown to be very powerful and versatile in the design of secure cryptographic protocols. Many constructions of SNARKs [33, 39, 7, 31, 42, 40, 4, 34], are in pre-processing model, i.e., they require a setup that generates a *structured common reference string* (SRS). The SRS is relation-dependent and can be reused to prove multiple statements.

A recent line of work on zk-SNARK [41, 20] follows a modular approach to construct SNARKs: first, an information-theoretic component is constructed, such as Interactive Oracle Proofs (IOP) or Algebraic Holographic Proofs (AHP); and then this interactive proof system is compiled into an argument using cryptographic tools. Finally, this is made non-interactive in the random oracle model (ROM), to obtain a SNARK. While this approach leads to very efficient SNARKs, the size of the proof is not constant, but logarithmic in the size of the witness. In this work, we focus on SNARKs with constant size proofs that are secure in the standard model avoiding idealised models such as ROM.

1.1 SNARKs for Computation over Rings

Despite the progress we have seen in SNARKs, all existing contructions offer efficiency benefits only for proving statements which can be efficiently represented as very particular forms of computation: The works of [31, 42, 40, 34] consider statements represented as circuit computations, either as Boolean circuits or as arithmetic circuits over a field. The compiler of [3] gives an efficient reduction from the correctness of programs to arithmetic circuit satisfiability for a prime field of suitable size. However, it is clearly interesting to consider computations over more general rings, that better suit applications such as proving evaluations over encrypted data or proving CPU computations. While this can be reduced to computation over a field, emulating ring arithmetic in terms of finite field operations incurs a significant overhead [38]. In addition, fixed and floating-point arithmetic operations that frequently come up in real-world applications (for instance in approximate, rather than exact computations such as in Machine Learning [19]), are more naturally expressed in terms of operations over rings.

Applications. Verifiable computation (VC) allows a computationally weak client to outsource evaluation of a function to a powerful server. The client can then verify that the output returned by the server is indeed correct while performing less work than what is necessary for computing the function itself. SNARKs immediately give a VC scheme, where the server performs the computation and returns a SNARK proof together with the output. Recently, there has been significant progress in constructing protocols and implementing systems for verifiable computation that leverage SNARKs [3, 4, 15, 24]. Nevertheless, the performance of existing constructions deteriorate for functionalities that have "bad" arithmetic circuit representations, as has been noted in prior works [42]. Similarly, the problem of ensuring both correctness and privacy of the computation performed by untrusted machines faces the same bottleneck of circuit representation. A natural construction for such schemes would be to consider a straightforward combination of SNARKs and Fully Homomorphic Encryption (FHE), where FHE allows computation over encrypted data and a SNARK is used to verify the integrity of the results of the computation. However, such a generic construction results in a large overhead even when used with the most performant state-of-the-art SNARKs for arithmetic circuits to prove FHE evaluations. This is due to the limitation of having to use representations over fields for proving computations over ciphertexts which are not naturally expressed as field elements. Therefore, such solutions do not scale well when the evaluation in FHE has to be emulated by arithmetic circuits over fields, and the resulting privacy-preserving VC schemes have very poor efficiency.

1.2 Our Contribution

Our goal is to construct a (zk)-SNARK for ring computations, thus bringing the theory of proof systems closer to practice. We focus on building schemes with security in the standard model as opposed with non-interactive arguments that require the Random Oracle Model (ROM). Along the way, we tackle new technical problems, introduce useful building blocks, such as Quadratic Ring Programs (QRPs) and secure encodings over rings. Finally, we provide two applications for our SNARKs based on the QRP characterization: Privacy-preserving verifiable computation and SNARKs over $\mathbb{Z}_{2^k}$.

Quadratic Programs over Rings. Gennaro et al. [31] introduced the NP representations Quadratic Span Programs (QSP) and Quadratic Arithmetic Programs (QAP) which can be used to compactly encode computations. They show how to convert any Boolean/arithmetic circuit into a QSP/QAP.

We are looking to design a similar characterisation for circuits over rings, and a couple of challenges get in our way: first, not all elements of the ring are invertible, so such a program should be defined over a subset of the ring, second we need a generalised Schwartz-Zippel lemma that gives us the necessary soundness. We treat all the technicalities encountered and we introduce Quadratic Ring Program (QRP) for rings containing big enough *exceptional sets* [6, 1],i.e. sets of elements such that their pairwise differences are invertible. As we discuss in Section 3, there is some 'tightness' to the need of using exceptional sets when capturing ring arithmetic in a black-box way using polynomials.

SNARK for Ring Computations. The QRP characterization allows to test satisfiability of an arithmetic circuit over a ring. To construct a succinct proof, we follow the blueprint of [31, 42], where the QRP test is performed in a probabilistic way. The setup produces a structured reference string CRS that consists of linearly homomorphic encodings, on top of which the prover is expected to compute using the witness. Under knowledge-type assumptions made for the ring encodings, the resulting SNARK can be proved knowledge sound.

We present Rinocchio, a generic framework for building SNARKs for ring arithmetic based on encodings over rings. Depending on how these encodings are instantiated, the resulting SNARK is either public or designated verifiable. One plausible instantiation for publicly-verifiable encodings is based on pairing-friendly composite order groups. However, the structure of such groups is specific and restrictive, in the sense that the ring used to represent the computation would not match any of the important applications considered by this work. Therefore, we focus on more generic secret-key encodings over rings that allow implementations using various rings and can be applied to speed-up proofs for real-world computations. On the other hand, these encodings yield designated-verifier SNARKs.

Our characterization of computation over rings as a QRP and subsequent SNARK construction inherits the need for a trusted CRS generation. However, in the designated-verifier setting, a trusted CRS is acceptable in practice, since if we do not need zero-knowledge property, we can simply have the verifier run the setup and send the CRS to the prover, who can reuse the CRS to prove many statements.

We choose to build Rinocchio in the standard model, on weaker assumptions rather than in idealized models such as Generic Group Model (GGM) or Algebraic Group Model (AGM) used for field-based schemes such as in [34]. Our work sets the stage for future SNARKs over rings with even smaller proof sizes or for other further features, e.g. an updatable structured reference string. We show in Section 6 that we can construct a SNARK along the lines of the construction of Groth16 [34] for general rings based on stronger assumptions for the underlying encodings – we prove its security by assuming that the encoding satisfies "linear only extractability", which roughly means that the only operations that can be performed over the encodings are affine.

Knowledge Assumptions over Rings. We prove Rinocchio secure under variants of the generalized q-PDH and d-PKE assumptions extended to encodings over rings, carefully addressing the technical challenges that arise in the new ring setting. These generalized assumptions were already stated for encodings over fields by prior works as [31, 32] and gained some confidence as a base to build post-quantum SNARKs. Similar to the counterpart of assumptions in the field case, where for instance, the existence of secure bilinear groups limits the choice of the finite fields, our ring assumptions are also cautiously made and assumed to be plausible when care is taken about the particular choice of ring and encoding scheme. In Appendix C we show that if an encryption scheme is assumed to be a linear-only extractable encoding, then that encoding satisfies the generalized q-PDH and q-PKE assumptions over rings. Therefore, if our assumptions turn out to not hold for a non-trivial choice of ring and encoding, that would lead to an efficient encoding scheme over that ring which allows for more than just linear homomorphism, potentially towards a new fully/somewhat homomorphic encryption scheme.

Privacy-Preserving Verifiable Computation. We take a step further to construct better VC schemes with privacy that follows the same blueprint as prior works: combining homomorphic encryption and a zk-SNARK. When instatiating Rinocchio with encoding schemes that take as input ciphertexts of a Ring-LWE-based FHE. The use of our generic SNARK for computation over rings allows for better choices of group order q (not only primes) which improves over the approach from prior works, e.g. [28]. Rinocchio allows for speed up through classical efficiency optimisations in $\mathcal{R}_q$ such as Number-Theoretic Transform (NTT). Also, we provide tools to enable the application of more advanced noise reduction techniques for the Ring-LWE scheme such as modulo switching.

Other Applications. Rinocchio can also help to prove arithmetic computations over rings $\mathbb{Z}_{2^k}$. As opposed to the attempt of simulating arithmetic over $\mathbb{Z}_{2^k}$ in a field $\mathbb{F}_p$, where one has to compute the modular reduction $x \mod 2^k$, a SNARK for ring computation can use a QRPs for the Galois Ring $GR(2^k, \delta)$, which has $\mathbb{Z}_{2^k}$ as a subring. We discuss other nuances of efficient considerations in $\mathbb{Z}_{2^k}$ arithmetic in Section 9.

1.3 Comparison with Related Work

The work of LegoSNARK [16] partially mitigates the efficiency issue of being tied to a unique, particular representation of computation in SNARK constructions. They achieve their results by seeing a computation as naturally consisting of different components and proposing a modular approach that uses the SNARK best suited for each component. Composition of proof gadgets is orthogonal to our work, and by extending our construction to be commit-and prove, the broader class of rings to which we can efficiently apply our SNARK adds yet another tool for works in the spirit of LegoSNARK.

The results of [9] give constructions of a designated verifier Succinct Non-interactive ARGument (SNARG) based on vector encryption over rings under the assumption that the encryption scheme satisfies linear targeted malleability. The subsequent work in [10] constructs a SNARG with quasi-optimal prover complexity. Even though these works use an encoding scheme over a ring to compile the information theoretic object, the statement to be proven is represented as Boolean/arithmetic circuit satisfiability over a field, and the computation is still over $\mathbb{F}_p$. Crucially, in these works the statement to be proved is an arithmetic circuit over a field, whereas our motivation is proving statements that are represented over rings like $\mathbb{Z}_{2^{64}}$ or a polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$ directly. In [37], Kosba et al. generalize the notion of Quadratic Arithmetic Programs over a field $\mathbb{F}$ to that of Quadratic Polynomial Programs (QPPs), which compute circuits whose wires carry values in the ring $\mathbb{F}[X]$. These polynomial circuits, where the addition and multiplication operations are over $\mathbb{F}[X]$, are introduced with the goal of representing (multi-)sets S of elements over $\mathbb{F}$. While the construction in [37] is limited to rings of polynomials over the same fields for which SNARKs à la [42] are secure, our work allows to build SNARKs for any ring R satisfying the property that it has a large subset such that the difference of the elements in the subset are invertible. Furthermore, our definition of QRP also recovers the QPP formulation as an instantiation of the underlying ring R, which we show in Appendix B.1.

Privacy-Preserving Verifiable Computation. To our knowledge, there are few works that consider privacy in the context of VC. The first one is the seminal paper of Gennaro et al. [30] who introduced the notion of non-interactive verifiable computation and builds it from garbled circuits and FHE. Fiore et al. [27] proposed to use homomorphic MACs in order to prove that the evaluation of FHE ciphertexts has been done correctly. Their solution is inherently bound to computations of quadratic functions.

To overcome this, the more recent work in this area by Fiore et al. [28] proposes a new protocol for verifiable computation on encrypted data that supports homomorphic computations of multiplicative depth larger than 1. Towards their VC scheme, [28] build a new SNARK that can efficiently handle computations of arithmetic circuits over a quotient polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$ for a prime number q in which the prover's costs have a minimal dependence on the degree d of $f(Y)$. Although this seems to fit the arithmetic structure for Ring-LWE schemes, it imposes many limitations due to the restriction to rings $\mathcal{R}_q$, where q is a prime which also has to match secure and efficient pairing constructions for some underlying SNARK over $\mathbb{F}_q$. Another significant impact on the performance present in the work of [28] is on the prover's effort to evaluate the circuit C over ciphertexts. In their VC scheme, the prover cannot use directly the transcript obtained by applying the evaluation algorithm over FHE ciphertexts as a witness for generating the proof. Instead, the prover is asked to come up with a different witness by considering the ciphertext of the Ring-LWE scheme as elements of $\mathbb{Z}_q[Y]$ rather than $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$. As a consequence, the degree of the witness polynomial grows linearly with the multiplicative depth of the circuit. This is a significant overhead that reflects, besides on the increased Prover's effort, on the size of the public setup necessary to commit to this polynomial. In another follow-up work, Bois et al. [8] introduced an improved solution. The key idea of their protocol is a new homomorphic hash function, which hashes to Galois rings. This allows for a flexible choice of FHE parameters.

Comparison. While we treat verifiable computation on encrypted data as a use case for Rinocchio scheme, we note that our work is more general and the main focus is on building a general SNARK that is *blackbox* in the choice of the ring. The scheme of [8] is along the lines of the GKR protocol, and therefore admits only circuits that are log-space uniform. Our QRP abstraction yields SNARKs for general circuit computations, albeit making knowledge assumptions similar to analogous SNARKs for fields. Furthermore, our scheme is in the standard model, while [28, 8] require a random oracle for non-interactivity. We give a detailed comparison of our application to privacy-preserving VC with the scheme of [28] in Section 7.4.

2 Preliminaries

Notation. We use κ to denote the security parameter. If $\mathcal{A}$ is a probabilistic polyonomial time (PPT) algorithm, we use $y \leftarrow \mathcal{A}(x)$ to denote that y is the output of $\mathcal{A}$ on x. By writing $\mathcal{A}\|\chi_{\mathcal{A}}(\sigma)$ we denote the execution of $\mathcal{A}$ followed by the execution of $\chi_{\mathcal{A}}$ on the same input σ and with the same random coins. The output of the two are separated by a semicolon.

Whenever we talk about a ring R, unless otherwise specified, we mean a commutative finite ring with identity. We denote the units of such a ring as R^*.

2.1 Succinct Non-interactive ARguments of Knowledge

Let $\mathcal{R}$ be an efficiently computable binary relation which consists of pairs of the form (x, w) where x is a statement and w is a witness. Let $\mathcal{L}$ be the language associated with the relation $\mathcal{R}$, i.e., $\mathcal{L} = \{x \mid \exists w \text{ s.t. } \mathcal{R}(x, w) = 1\}$.

Definition 1 (SNARK). *A triple of polynomial time algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *is a SNARK for an NP relation $\mathcal{R}$, if the following properties are satisfied:*

1. *Completeness. For all $(x, w) \in \mathcal{R}$, the following holds:*

$$\Pr\left(\mathsf{Verify}(\mathsf{vk}, x, \pi) = 1 \;:\; \begin{array}{c}(\sigma, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\kappa)\\ \pi \leftarrow \mathsf{Prove}(\sigma, x, w)\end{array}\right) = 1$$

2. *Knowledge Soundness . For any PPT adversary $\mathcal{A}$, there exists a PPT algorithm $\chi_{\mathcal{A}}$ such that the following probability is negligible in κ:*

$$\Pr\left(\begin{array}{c}\mathsf{Verify}(\mathsf{vk}, \tilde{x}, \tilde{\pi}) = 1\\ \wedge\, \mathcal{R}(\tilde{x}, w') = 0\end{array} \;:\; \begin{array}{c}(\sigma, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\kappa)\\ ((\tilde{x}, \tilde{\pi}); w') \leftarrow \mathcal{A}|\chi_{\mathcal{A}}(\sigma)\end{array}\right)$$

3. *Succinctness. For any x and w, the length of the proof π is given by $|\pi| = \mathsf{poly}(\kappa) \cdot \mathsf{polylog}(|x| + |w|)$.*

Non-black-box extraction. The notion of *knowledge soundness* requires the existence of an extractor that can compute a witness whenever the adversarial prover produces a valid argument. The extractor we defined above is non-black-box and gets full access to the adversary's state, including any random coins.

Definition 2 (zk-SNARK). *A zk-SNARK for a relation $\mathcal{R}$ is a SNARK for $\mathcal{R}$ with the following* zero-knowledge *property: There exists a PPT simulator $(\mathcal{S}_1, \mathcal{S}_2)$ such that $\mathcal{S}_1$ outputs a simulated CRS σ and trapdoor τ; $\mathcal{S}_2$ takes as input σ, a statement x and τ, and outputs a simulated proof π; and, for all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, the following is negligible in κ.*

$$\left| \Pr\left(\begin{matrix} (x,w) \in \mathcal{R} : \wedge \\ \mathcal{A}_2(\pi, \mathsf{st}) = 1 \end{matrix} : \begin{matrix} (\sigma, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\kappa) \\ (x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\kappa, \sigma) \\ \pi \leftarrow \mathsf{Prove}(\sigma, x, w) \end{matrix} \right) - \Pr\left(\begin{matrix} (x,w) \in \mathcal{R} \wedge \\ \mathcal{A}_2(\pi, \mathsf{st}) = 1 \end{matrix} : \begin{matrix} (\sigma, \tau) \leftarrow \mathcal{S}_1(1^\kappa) \\ (x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\kappa, \sigma) \\ \pi \leftarrow \mathcal{S}_2(\sigma, \tau, x) \end{matrix} \right) \right|$$

Public vs Designated verifiability. In a publicly verifiable SNARK, there is no private verification information, i.e. $\mathsf{vk} = \emptyset$. A SNARK is designated verifiable if the proof can be verified only by a party knowing vk. Note that in the designated-verifier case, the verifier's decision bit on a proof potentially leaks some information about vk. Thus, the same common reference string cannot be reused for multiple proofs as in publicly-verifiable case. This was addressed in prior works in verifiable computation [30, 22], by either keeping the decision bit secret from the prover, or running a fresh setup every time a proof fails verification. Note that any sound scheme can tolerate $O(\log \kappa)$ bits of leakage, and assuming that the decision bit leaks only a constant number of bits of information, one would only need to run a new setup after logarithmically-many proof rejections.

Strong Soundness. Multi-statement designated-verifier SNARKs are requiring soundness to hold even against a prover that makes adaptive queries to a proof verification oracle.

2.2 Background in Ring Theory

We now turn to recall useful results from ring theory. While some of the results for fields and euclidean domains (such as $\mathbb{Z}$) carry over to the more general rings we deal with, others do not. For example, one has to be careful about the fact that the rings we consider contain zero divisors, i.e. $d \in R \setminus \{0\}$ for which $\exists\ q \in R \setminus \{0\}$ such that $d \cdot q = 0$.

Lemma 1. *Let R be a finite ring. Then all non-zero elements of R are either a unit or a zero divisor.*

Proof. For every $a \in R \setminus \{0\}$, let $f_a : R \to R$ be the map given by $f_a(x) = a \cdot x$. If f_a is injective, then it has to be surjective, because R is finite. Therefore, in such a case there must exist an $x \in R$ verifying that $f_a(x) = 1$. So we conclude that a is a unit.

Assume that $f_a(x)$ is not injective. Then there exist $b, c \in R, b \neq c$, such that $a \cdot b = a \cdot c$, and thus $a \cdot (b - c) = 0$. In other words, a is a zero divisor. ■

We recall that an ideal of a ring R is an additive subgroup $I \subseteq R$ such that $r \cdot x \in I$ for any $r \in R, x \in I$. Through the paper, (x) will denote the ideal generated by $x \in R$.

Theorem 1. *Let R be a finite commutative ring with identity and let $Z(R)$ denote the set of all its zero divisors. Then the following are equivalent:*

1. *$Z(R)$ is an ideal.*
2. *$Z(R)$ is a maximal ideal.*
3. *R is local.*
4. *Every $x \in Z(R)$ is nilpotent.*

Proof. (1) $\Leftrightarrow$ (2). Assume $Z(R)$ is an ideal and it is not maximal. Then, there must exist some ideal I such that $Z(R) \subsetneq I \subsetneq R$. Which is absurd, as if $Z(R) \subsetneq I$, then I must contain a unit and hence $I = R$.

(2) $\Rightarrow$ (3). Assume R contains another maximal ideal $I \neq Z(R)$. Then either $I \subsetneq Z(R)$, in which case it is not maximal, or otherwise it contains a unit and hence $I = R$.

(3) $\Rightarrow$ (1). Let M be the maximal ideal of R. In order to see that $Z(R)$ is an ideal, let x, y be any two zero-divisors and $(x), (y)$ the ideals they generate. Because R is local, then $(x) \subset M$ and $(y) \subset M$. Since M is a proper ideal of R, then we have that $\forall r \in R$, $r \cdot (x + y) \in M$ and that $r \cdot (x + y)$ cannot be a unit. Hence, by Lemma 1, $r \cdot (x + y) \in Z(R)$.

(1) $\Rightarrow$ (4). Assume $Z(R)$ is an ideal, and assume towards contradiction some $x \in Z(R)$ such that $x^i \neq 0$ for any positive integer i. Then, as R is finite, there must exist some $i > j > 0$ such that $x^i = x^j$, from which we deduce that $x^j \cdot (x^{i-j} - 1) = 0$. As $x^j \neq 0$, then it has to be that $x^{i-j} - 1 \in Z(R)$. Then, as we also now that $x^{i-j} \in Z(R)$ and $Z(R)$ is an ideal, then $(x^{i-j} - 1) - x^{i-j} = -1 \in Z(R)$. Which is absurd, as then we would have that $Z(R) = R$.

(4) $\Rightarrow$ (1). Let $Z(R) = \{x_1, \ldots, x_m\}$. We will prove that $Z(R)$ is an ideal by showing the existence of some $z \in R$ such that $z \cdot x_j = 0$ for all $j \in [m]$, from which follows that $Z(R)$ is an ideal. We construct $z = z_m$ recursively as follows. Because x_1 is nilpotent, there exists an a_1 s.t. $x_1^{a_1+1} = 0$ but $x_1^{a_1} \neq 0$, so we define $z_1 = x_1^{a_1}$. For $i \in [m]$, we define $z_i = z_{i-1} \cdot x_i^{a_i}$, where a_i (which is possibly zero) is chosen such that $z_i \neq 0$ and $z_i \cdot x_i = 0$. Notice that a_i must exist from the fact that x_i is nilpotent. ∎

Theorem 2 (Chinese Remainder Theorem). *Let $I_1, \ldots, I_m$ be m pairwise co-prime*[4] *ideals of R, i.e. $\forall i \neq j, I_i + I_j = R$. Denote $I = I_1 \cdots I_m$. Then the following map is a ring isomorphism:*

$$\begin{aligned} R/I &\to R/I_1 \times \cdots \times R/I_m \\ r \bmod I &\mapsto (r \bmod I_1, \ldots, r \bmod I_m) \end{aligned}$$

Exceptional sets. Elements which satisfy that their pairwise differences are invertible will be fundamental in our constructions. These have received different names in the literature: 'Condition (F)' sets in [6], 'exceptional sequences' in [1] and 'exceptional sets' in [25]. We will stick with the latter denomination.

Definition 3. *Let $A = \{a_1, \ldots, a_n\} \subset R$. We say that A is an exceptional set if $\forall i \neq j, a_i - a_j \in R^*$. We define the Lenstra constant of R to be the size of the biggest exceptional set in R.*

We will need the following generalization of the Schwartz-Zippel lemma.

Lemma 2. *[Generalized Schwartz-Zippel Lemma [6]] Let $f : R^n \to R$ be an n-variate non-zero polynomial. Let $A \subseteq R$ be a finite exceptional set. Let $deg(f)$ denote the total degree of f. Then:*

$$\Pr_{\vec{a} \leftarrow A^n}[f(\vec{a}) = 0] \leq \frac{deg(f)}{|A|}$$

Proof. We prove by induction on the number of variables n. For $n = 1$, let $a_1 \in A$ be a root of $f(x)$. As $(x - a_1)$ is a monic polynomial, we have that $f(x) = (x - a_1)f_1(x)$, where the $deg(f_1) < deg(f)$.

[4] Such ideals are also denoted co-maximal by some authors.

Any other root $a_2 \in A$ has to be a root of $f_1(x)$, as $(a_2 - a_1) \in R^*$ and $f(a_2) = 0$. Hence, we have that $f(x) = (x - a_1)(x - a_2)f_2(x)$, where the $deg(f_2) < deg(f_1)$. By iterating this argument, we conclude that $f(x)$ cannot have more roots in A than $deg(f)$ and hence $\Pr_{a \leftarrow A}[f(a) = 0] \leq (deg(f))/|A|$.

Assume now the result holds for $(n-1)$-variate polynomials. Given any n-variate polynomial $f(\vec{x}) \in R[x_1, \ldots, x_n]$, denote by $k = deg_{x_n}(f)$ the largest power of x_n appearing in any monomial of f. Then we have that:

$$f(\vec{x}) = \sum_{\ell=1}^{k} x_n^{\ell} \cdot g_{\ell}(x_1, \ldots, x_{n-1})$$

Denote by $\mathcal{E}_1$ the event $g_k(\vec{a}) = 0$. By definition of k, we know that $g_k(x_1, \ldots, x_{n-1})$ is a non-zero polynomial, so by induction hypothesis $\Pr_{\vec{a} \leftarrow A^{n-1}}[\mathcal{E}_1] \leq (deg(f) - k)/|A|$. Assuming $\neg\mathcal{E}_1$ and by applying the same reasoning as for $n = 1$, we have that $f(\vec{a}) \in R[x_n]$ has at most k roots in A, so $\Pr_{\vec{a} \leftarrow A}[f(\vec{a}) = 0 | \neg\mathcal{E}_1] \leq k/|A|$. We finalize by noting that (where the probability is taking over the choice of $\vec{a} \leftarrow A^n$):

$$\begin{aligned}\Pr[f(\vec{a}) = 0] &= \Pr[f(\vec{a}) = 0|\neg\mathcal{E}_1] \cdot \Pr[\neg\mathcal{E}_1] + \Pr[f(\vec{a}) = 0|\mathcal{E}_1] \cdot \Pr[\mathcal{E}_1] \\ &\leq \Pr[f(\vec{a}) = 0|\neg\mathcal{E}_1] + \Pr[\mathcal{E}_1] \leq \frac{deg(f) - k}{|A|} + \frac{k}{|A|}\end{aligned}$$

■

Interpolation. Lagrange interpolation for sets of points $(x_i, y_i) \in R^2$ can be computed, as long as all the x_i are part of the same exceptional set $A \subset R$. This follows from either looking at the definition of Lagrange basis polynomials or, more formally, from the Chinese Remainder Theorem (Theorem 2). As an intuition of the latter approach, the ideals $(x - x_i)$ are co-prime, so there is a one-to-one correspondence between any polynomial $p(x) \in R[x]/I$, where $I = \prod_{i=1}^{d+1}(x - x_i)$, and $y_1 = p(x_1), \ldots, y_{d+1} = p(x_{d+1})$. In other words, any $p(x) \in R[x]$ of degree d is uniquely determined by its evaluation at d points of an exceptional set. For more details about the CRT argument, see e.g. [1].

Galois Rings. Galois Rings are the generalization of Galois Fields to the ring case. Informally, a Galois Ring relates to integers modulo p^k in the same way a Galois Field relates to integers modulo a prime p. In the following, we provide a high level overview of their properties and arithmetic. For a more detailed introduction to Galois Rings, see [45].

Definition 4. *A Galois Ring is a ring of the form $R = \mathbb{Z}_{p^k}[X]/(h(X))$, where p is a prime, k a positive integer and $h(X) \in \mathbb{Z}_{p^k}[X]$ a monic polynomial of degree $d \geq 1$ such that its reduction modulo p is an irreducible polynomial in $\mathbb{F}_p[X]$.*

Given a base ring $\mathbb{Z}_{p^k}$, there is a unique degree d Galois extension of $\mathbb{Z}_{p^k}$, which is precisely the Galois Ring provided on the previous definition. Hence, we shall denote such Galois Ring as $GR(p^k, d)$. Note that Galois Rings reconcile the study of finite fields $\mathbb{F}_{p^d} = GR(p, d)$ and finite rings of the form $\mathbb{Z}_{p^k} = GR(p^k, 1)$.

Every Galois Ring $R = GR(p^k, d)$ is a local ring and its unique maximal ideal is (p). Hence, by Theorem 1, all the zero divisors of R are furthermore nilpotent, and they constitute the maximal ideal (p). Furthermore, we have that $R/(p) \cong \mathbb{F}_{p^d}$, and thus a canonical homomorphism $\pi : R \to \mathbb{F}_{p^d}$ which can be computed by 'reducing modulo p'.

Proposition 1 **([1]).** *The Lenstra constant of $R = GR(p^k, d)$ is p^d.*

In this work, we will be particularly interested in Galois Rings of the form $R = GR(2^k, d)$, i.e. of characteristic 2^k, maximal ideal (2) and such that $R/(2) \cong \mathbb{F}_{2^d}$. Whenever we need to explicitly represent elements $a \in R$, we will do so as it follows from Definition 4. In that case, we will say that a is given in its *additive representation*, which consists of the residue classes

$$a \equiv a_0 + a_1 \cdot X + \ldots + a_{d-1} \cdot X^{d-1} \mod h(X), \quad a_i \in \mathbb{Z}_{2^k}. \tag{1}$$

3 Quadratic Programs over Commutative Rings

Notation. We use κ to denote the security parameter. If $\mathcal{A}$ is a probabilistic polyonomial time (PPT) algorithm, we use $y \leftarrow \mathcal{A}(x)$ to denote that y is the output of $\mathcal{A}$ on x. By writing $\mathcal{A} \| \chi_{\mathcal{A}}(\sigma)$ we denote the execution of $\mathcal{A}$ followed by the execution of $\chi_{\mathcal{A}}$ on the same input σ and with the same random coins. Whenever we talk about a ring R, unless otherwise specified, we mean a commutative finite ring with identity. We denote the units of such a ring as R^*.

We now give a characterization for the satisfiability of arithmetic circuits over commutative rings with identity.

Definition 5 (Quadratic Ring Programs (QRP)). *A Quadratic Ring Program (QRP) Q over a finite commutative ring R consists of three sets of polynomials, $\mathcal{V} = \{v_k(x) : k \in [0, m]\}$, $\mathcal{W} = \{w_k(x) : k \in [0, m]\}$, $\mathcal{Y} = \{y_k(x) : k \in [0, m]\}$ and a target polynomial $t(x)$, all in $R[x]$. Let C be an arithmetic circuit over R with n inputs and n' outputs. We say that Q is a QRP that computes C if the following holds:*

$a_1, \ldots, a_n, a_{m-n'+1}, \ldots a_m \in R^{n+n'}$ is a valid assignment to the input/output variables of C if and only if there exist $a_{n+1}, \ldots, a_{m-n'} \in R^{m-n-n'}$ such that:

$$t(x) \text{ divides } p(x),$$

where $p(x) = V(x) \cdot W(x) - Y(x)$, $V(x) = \big(v_0(x) + \sum_{k=1}^{m} a_k \cdot v_k(x)\big)$, $W(x) = \big(w_0(x) + \sum_{k=1}^{m} a_k \cdot w_k(x)\big)$ and $Y(x) = \big(y_0(x) + \sum_{k=1}^{m} a_k \cdot y_k(x)\big)$.

We define the size and degree of Q to be m and $\deg(t(x))$ respectively. Given polynomials $V(x), W(x), Y(x) \in R[x]$ defined as above and corresponding to a valid assignment of the input/output wires, we will call them a QRP solution.

3.1 Construction of a QRP for a Circuit over Rings

Let C be an arithmetic circuit over R. To build a QRP, we will make use of an exceptional set A as follows. We will pick elements $r_g \in A$ for each multiplication gate $g \in C$ and define the target polynomial as $t(x) = \prod_{g \in C}(x - r_g)$. As a consequence of the CRT over rings, the $v_k(x), w_k(x)$ and $y_k(x)$ polynomials can be computed by interpolating over those $r_g \in A$ in the same way one proceeds in the QAP case [31, 42]. In more detail, let $I_1, \ldots I_{\deg(t(x))}$ be the ideals defined by $I_g = (x - r_g)$, which are co-prime since A is an exceptional set. Noting that $p(x) \equiv p(r_g) \mod (x - r_g)$, we have that:

$$\begin{aligned} \phi : R[x]/(t(x)) \; &\simeq R[x]/I_1 \times \ldots \times R[x]/I_{\deg(t(x))} \\ p(x) \; &\mapsto \quad (p(r_1), \; \ldots \; , p(r_{\deg(t(x))})) \end{aligned} \tag{2}$$

In other words, the isomorphism above tells us that $t(x)$ divides $p(x)$ if and only if $p(r_g) = 0$ for every $r_g \in A$, as long as A is an exceptional set. We show that this imposition on A is not only sufficient, but also necessary.

Proposition 2. *Let $t(x) = \prod_{g \in C}(x - r_g)$, $I_g = (x - r_g)$ and $A = \{r_g\}_{g \in C}$. If the map ϕ given by Eq. (2) is an isomorphism, then A is an exceptional set.*

Proof. Assume that A is not exceptional, i.e. that there exist $r_1, r_2 \in A$ such that $r_1 - r_2 \notin R^*$. Since R is a finite ring, then $r_1 - r_2$ is a zero divisor, so $\exists b \in R$ s.t. $b \cdot (r_1 - r_2) = 0$. We show that ϕ is not injective by giving two elements of $R[x]/(t(x))$ that map to the all zeroes vector: 0 and $b \cdot \prod_{r_g \in A \setminus \{r_1\}}(x - r_g)$.

The above proposition highlights the "tightness" of the requirement to use exceptional sets in order to build QRPs. We would like to emphasize that exceptional *sets* have no further algebraic properties (e.g. no closure under addition).

In Appendix B, we show how to build a QRP for a multiplication sub-circuit, and how to compose QRPs to obtain a QRP for any arithmetic circuit.

4 Secure Encoding Schemes over Rings

To construct a SNARK, we follow the framework in [31]. The QRP polynomials are represented by encodings of the polynomials evaluated at a secret point, and the encoding used is additively homomorphic in the ring of computation. We now define these encodings and their properties.

Definition 6 (Encoding scheme). *An encoding scheme* Encode *over a ring R consists of a tuple of algorithms* $(\mathsf{Gen}, \mathsf{E})$.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, *a key generation algorithm that takes as input a security parameter and outputs a secret key* sk, *and public information* pk.
- $s \leftarrow \mathsf{E}(a)$, *a probabilistic encoding algorithm mapping a ring element $a \in R$ to an encoding s in an encoding space S such that the sets $\{\{\mathsf{E}(a)\} : a \in R\}$ partition S, where $\{\mathsf{E}(a)\}$ is the set of encodings of a. Depending on the encoding algorithm,* E *could require the secret state* sk. *To ease notation, we will omit this additional argument.*

  An encoding scheme has to satisfy the following properties:

- ***ℓ-Linearly homomorphic:*** *There is an efficient algorithm* Eval *that on input public information* pk, *encodings $\mathsf{E}(a_1), \dots \mathsf{E}(a_\ell)$ and coefficients $c_1, \dots, c_\ell \in R^\ell$ computes the encoding $\mathsf{E}(\sum_{i=1}^{\ell} c_i \cdot a_i)$.*
- ***Quadratic root detection:*** *There exists an algorithm that given secret key* sk, $(\mathsf{E}(a_1), \dots \mathsf{E}(a_d))$, *and a quadratic polynomial $Q(x_1, \dots, x_t) \in R[x_1, \dots, x_t]$, can distinguish whether $Q(a_1, \dots, a_t) = 0$.*
- ***Image verification:*** *There exists an efficient algorithm that given* sk, *and an element c, can detect if c is a valid encoding of some element in R.*

4.1 Assumptions on Encodings

While the definition of encoding above can be satisfied by, for instance, the identity function, we will only be interested in *secure* encodings, i.e. those which satisfy certain cryptographic assumptions. In the following, A (resp. A^*) denotes an exceptional set of a commutative ring with identity R (resp. R^*, the units of that ring).

Our computational assumptions have been previously used in the discrete-logarithm group setting. Here, we generalize them to encodings over rings. We also show (in Appendix C) how our assumptions are *weaker* than the more intuitive, but stronger, notion of linear-only extractable encodings from [7].

We start by giving a generalized version of the q-PDH problem used in [31]. This assumption has two differences with respect to the original one. First of all, the adversary is able to win the game as long as it outputs a pair (a, y) such that $a \neq 0$ and $y \in \{\mathsf{E}(a \cdot s^{q+1})\}$. In the field case, this is trivially equivalent to the original q-PDH assumption, as $a^{-1} \cdot \mathsf{E}(a \cdot s^{q+1}) = \mathsf{E}(s^{q+1})$. Nevertheless, in the ring case, we need to deal with elements $a \in R$ which might be zero divisors. Second, in order to have the assumption work for any given q, we need to ensure that $s^{2q} \neq 0$. Due to this and additional security reasons, we restrict s to be a unit. Furthermore, we need s to be part of a big enough exceptional set, so that we can prove the soundness of our SNARKs by invoking the Generalized Schwartz-Zippel lemma.

Assumption 1 (Generalized q-PDH) *The generalized q-power Diffie-Hellman assumption holds for an encoding scheme* Encode *if, for every non-uniform* PPT *algorithm $\mathcal{A}$, the following probability is less or equal than* $\frac{2q}{|A^*|} + \mathsf{negl}(\kappa)$*:*

$$\Pr\left(\begin{array}{c} a \neq 0 \, \wedge \\ y \in \{\mathsf{E}(a \cdot s^{q+1})\} \end{array} : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \\ s \leftarrow A^*, \\ \sigma = (\mathsf{pk}, \mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q})), \\ (a, y) \leftarrow \mathcal{A}(\sigma) \end{array} \right).$$

Note that we linked our generalization of q-PDH to the size of the exceptional set A^*. Usually, we will consider A^* to be of exponential size in the security parameter, so that the previous probability is just negligible in the security parameter. Nevertheless, for the purpose of parallel soundness amplification techniques, in some cases it will be useful to consider even exceptional sets of constant size. The reason to bound $\mathcal{A}$'s advantage by $2q/|A^*|$ is the possibility of a generic attack on q-PDH, which was presented in [35]. We generalize such attack to our assumption in Lemma 7, in Appendix C.

We also need a q-power knowledge assumption, which is both augmented to handle the designated verifier setting and generalized to encodings over rings.

Assumption 2 (Generalized Augmented q-PKE) *The generalized augmented q-power knowledge of encoding assumption holds for an encoding scheme* Encode *and for the class $\mathcal{Z}$ of "benign" auxiliary input generators if, for every non-uniform PPT auxiliary input generator $Z \in \mathcal{Z}$ and for all non-uniform* PPT *algorithm $\mathcal{A}$ there exists a non-uniform* PPT *extractor $\chi_{\mathcal{A}}$ such that the following probability is negligible in the security parameter:*

$$\Pr\left(\begin{array}{c} \hat{c} - \alpha c = 0 \\ \wedge \\ c \neq \sum_{i=0}^{q} a_i s^i \end{array} : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \alpha \leftarrow R^*, s \leftarrow A^*, \\ \sigma = (\mathsf{pk}, \mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^q)), \\ z \leftarrow Z(\sigma) \\ (\mathsf{E}(c), \mathsf{E}(\hat{c}); a_0, \ldots, a_q) \leftarrow (\mathcal{A} || \chi_{\mathcal{A}})(\sigma, z) \end{array} \right).$$

In the above, $(x; y) \leftarrow (\mathcal{A}||\chi_{\mathcal{A}})(\sigma, z)$ denotes that on input (σ, z), $\mathcal{A}$ outputs x, and $\chi_{\mathcal{A}}$ given the same input (σ, z), and $\mathcal{A}$'s random tape, outputs y. When we assume that Z is benign, we mean that the auxiliary information z is generated with a dependency on sk, s and α that is limited to the extent that it can be generated efficiently from σ.

5 Rinocchio: A SNARK over Rings

The QRP characterization allows a test for satisfiability of an arithmetic circuit, by checking if the target polynomial divides $p(x) = \left(\sum c_k \cdot v_k(x)\right) \cdot \left(\sum c_k \cdot w_k(x)\right) - \left(\sum c_k \cdot y_k(x)\right)$. If divisibility holds, there is a quotient polynomial that is guaranteed to exist that serves as a witness for this test. To construct a succinct proof, we follow the blueprint of [31, 42]: the QRP test is performed in a probabilistic way at a random point chosen during the setup. Toward this end, the prover is expected to give, in the proof, the polynomials $V(x), W(x), Y(x)$ computed as a linear combination of the QRP polynomials using the intermediate witness values c_k as coefficients. The prover also provides the quotient polynomial $H(x)$, and verification checks whether $V(s) \cdot W(s) - Y(s) = H(s) \cdot t(s)$, where s is the random point that is hidden from the prover. The CRS consists of an encoding of this secret point together with encodings of the QRP polynomials, and the prover homomorphically computes the elements in the proof.

Besides the security assumptions we introduced in the previous section, our designated verifier SNARK construction will rely on the following two technical lemmas. The first one will be useful to define the concrete soundness error of our construction, while the second one is an analogue of [31, Lemma 10]. At a high level, this second lemma will be invoked in the security proof to ensure that, if the adversary outputs a false proof that passes verification that implicitly uses some $V(x)$ that is not in the span of the QRP polynomial set $\{v_k(x)\}$, then the reduction will be able to use that false proof to solve a q-PDH challenge.

Lemma 3. *Given an exceptional set of size n in R, we can construct another exceptional set $A = \{0, a_1, \ldots, a_{n-1} : a_i \in R^*\}$. When an exceptional set has the latter form, we say it is given in its canonical form.*

Proof. Let $B = \{b_1, \ldots, b_n\} \subset R$ be an exceptional set. For all $i \in \{1, \ldots, n-1\}$, define $a_i = b_n - b_i$. By the definition of B, we have that $a_i \in R^*$ and hence so is $(0 - a_i)$. Furthermore, $\forall i \neq j, a_i - a_j = (b_n - b_i) - (b_n - b_j) = b_i - b_j$ which is again a unit by the definition of B. ∎

Lemma 4. *Let $R[x]_{\leq e}$ denote the polynomials in $R[x]$ of degree at most e. Let $R[x]^{\neg(e)}$ denote polynomials over $R[x]$ that have a zero coefficient for x^e. Let $A^* \subset R^*$ be an exceptional set. We define $A^*[x]_{\leq e}$, $A^*[x]^{\neg(e)}$ analogously. Given a set $\mathcal{U} = \{u_i(x)\} \subset R[x]_{\leq e}$ such that $|\mathcal{U}| = m$, let span$(\mathcal{U})$ denote the set of polynomials that can be generated as R-linear combinations of the polynomials in $\mathcal{U}$. Let $a(x) \in A^*[x]_{\leq e+1}$ be generated uniformly at random subject to the constraint that $\{a(x) \cdot u_i(x) : u_i(x) \in \mathcal{U}\} \subset R[x]^{\neg(e+1)}$. Let $s \leftarrow A^*$. Then, if $e > m - 1$, for all algorithms $\mathcal{A}$,*

$$\Pr\left(\begin{matrix} u(x) \in R[x]_{\leq e} \wedge \\ u(x) \notin span(\mathcal{U}) \wedge \\ a(x) \cdot u(x) \in R[x]^{\neg(e+1)} \end{matrix} : u(x) \leftarrow \mathcal{A}(\mathcal{U}, s, a(s))\right) \leq \frac{1}{|A^*|}$$

Proof. Let $u(x) = u_0 + u_1 x + \ldots + u_e x^e \in R[x]$ and $u(x) \notin span(\mathcal{U})$. Define the vector $\boldsymbol{u} = (u_0, \ldots, u_e, 0)$, corresponding to the coefficients of the monomials in u and padded with a zero, and similarly define $\boldsymbol{u}_i = (u_{i,0}, \ldots, u_{i,e}, 0)$ for every $u_i(x) \in \mathcal{U}$. Then, $\boldsymbol{u}$ is not in the span of the vectors $(s^{e+1}, s^e, \ldots, 1) \bigcup_{i \in [m]} \boldsymbol{u}_i$. This follows from the assumption that $u(x) \notin \mathit{span}(\mathcal{U})$ and the fact that the last element of $\boldsymbol{u}$ is a 0 and that of $(s^{e+1}, s^e, \ldots, 1)$ is 1.

This time following the opposite order, define a vector $\boldsymbol{a} = (a_{e+1}, \ldots, a_0)$ from the coefficients of $a(x) = a_0 + \cdots + a_{e+1} \cdot x^{e+1}$. Then, $\mathcal{A}$ has the following information about $a(x)$:

$$\begin{aligned} \langle \boldsymbol{a}, (s^{e+1}, s^e, \ldots, 1) \rangle &= a(s) \\ \langle \boldsymbol{a}, (u_{i,0}, \ldots, u_{i,e}, 0) \rangle &= 0, \quad i \in [m] \end{aligned} \tag{3}$$

Where the second set of equations comes from the fact that $\{a(x) \cdot u_i(x) : u_i(x) \in \mathcal{U}\} \subset R[x]^{\neg(e+1)}$. This provides a system of $m+1$ linear equations on the $e+2$ coefficients $\boldsymbol{a}$, so as $e > m-1$ by hypothesis, $\boldsymbol{a}$ appears uniformly random to $\mathcal{A}$.

Finally, assume that $\mathcal{A}$ manages to satisfy the last missing condition for $u(x)$, that is $a(x) \cdot u(x) \in R[x]^{\neg(e+1)}$, which is equivalent to $\langle \boldsymbol{a}, \boldsymbol{u} \rangle = 0$. Since $\boldsymbol{u}$ is not in the span of $(s^{e+1}, s^e, \ldots, 1) \bigcup_{i \in [m]} \boldsymbol{u}_i$, it is not a linear combination of the equations constituting the system in (3). Hence, since every $a_i \in A^*$ and $\boldsymbol{a}$ appears uniformly random to $\mathcal{A}$ (subject to the constraints provided by the system of linear equations), by looking at $\boldsymbol{u}$ as the coefficients of a polynomial $\tilde{u}(x_1, \ldots, x_{e+2}) = u_0 \cdot x_1 + u_1 \cdot x_2 + \cdots + u_e \cdot x_{e+1} + 0 \cdot x_{e+2}$, we have that $\Pr[\langle \boldsymbol{a}, \boldsymbol{u} \rangle = 0] = \Pr[\tilde{u}(\boldsymbol{a}) = 0] \leq 1/|A^*|$ as a direct consequence of the Generalized Schwartz-Zippel Lemma (Lemma 2). ■

5.1 Construction from QRP

Let C be an arithmetic circuit over R, with m wires and d multiplication gates. Let A be an exceptional set given in canonical form and $A_Q = \{0, a_1, \ldots, a_{d-1}\} \subset A$. Using A_Q, define the QRP $Q = (t(x), \{v_k(x), w_k(x), y_k(x)\}_{k=0}^m)$ which computes C. Let $A^* = A \setminus A_Q$, which satisfies that $A^* \subseteq R^*$, since A is canonical.

We denote by $I_{io} = 1, 2, \ldots \ell$ the indices corresponding to the public input and public output values of the circuit wires and by $I_{mid} = \ell + 1, \ldots m$, the wire indices corresponding to the intermediate values. We construct a SNARK scheme $\mathsf{Rinocchio} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for ring arithmetic as described in Figure 1.

Remark 1. An aspect of our construction that could look surprising to the reader is the definition of A^*: Why do we not include the elements in A_Q used to define the QRP? As previously discussed, we do this in order to precisely define the soundness of our construction. In some cases, as we will discuss in Section 8.3, it could be useful to use parallel repetition strategies for soundness amplification. Previous works in the field setting, using pairings, did not need to make such a concrete analysis, since if circuits are assumed to be of polynomial size in the security parameter, the probability that a randomly sampled $s \leftarrow \mathbb{F}$ would be precisely one of the points used to define the QRP would be negligible, because $\mathbb{F}$ has exponential size in the security parameter. In all rigour, nevertheless, the *concrete* soundness error of those constructions is also bounded by the size of $\mathbb{F}$ minus the size of the QRP[5]. We prefer this concrete analysis even when rings might have exceptional sets of exponential size in the security parameter.

[5] In order to see this, consider a proof that consists purely of encodings of zero. The checks in the verification equations would pass if s happened to coincide with a value in the QRP used to describe a multiplication gate with no connections to input or output wires. This applies to e.g. [42].

Rinocchio

$\underline{\mathsf{Setup}(1^\kappa, \mathcal{R})}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \quad s \leftarrow A^*, \ r_v, r_w \leftarrow R^*, \ r_y = r_v \cdot r_w$

$\alpha, \alpha_v, \alpha_w, \alpha_y \leftarrow R^*, \ \beta \leftarrow R \setminus \{0\}$

$$\mathsf{crs} = \Big(\{\mathsf{E}(s^i)\}_{i=0}^d, \{\mathsf{E}(r_v v_k(s))\}_{k\in I_{mid}}, \{\mathsf{E}(r_w w_k(s))\}_{k\in I_{mid}}, \{\mathsf{E}(r_y y_k(s))\}_{k\in I_{mid}}, \{\mathsf{E}(\alpha s^i)\}_{i=0}^d, \{\mathsf{E}(\beta(r_v v_k(s) + r_w w_k(s) + r_y y_k(s))\}_{k\in I_{mid}}, \mathsf{pk} \Big) \tag{4}$$

$$\mathsf{vk} = (\mathsf{sk}, \mathsf{crs}, s, \alpha, \beta, r_v, r_w, r_y)$$

$\underline{\mathsf{Prove}(\mathsf{crs}, u, w)}$

$u = (a_1, \ldots, a_\ell), \ a_0 = 1,$
$w = (a_{\ell+1}, \ldots, a_m)$
$v(x) = \sum_{k=0}^m a_k v_k(x)$
$v_{mid}(x) = \sum_{k\in I_{mid}} a_k v_k(x)$
$w(x) = \sum_{k=0}^m a_k w_k(x)$
$w_{mid}(x) = \sum_{k\in I_{mid}} a_k w_k(x)$
$y(x) = \sum_{k=0}^m a_k y_k(x)$
$y_{mid}(x) = \sum_{k\in I_{mid}} a_k y_k(x)$
$h(x) = \dfrac{v(x)w(x) - y(x)}{t(x)}$
$L_\beta = \beta\big(r_v v_{mid}(s) + r_w w_{mid}(s) + r_y y_{mid}(s)\big)$
$A = \mathsf{E}(v_{mid}(s)), \ \hat{A} = \mathsf{E}(\alpha v_{mid}(s)),$
$B = \mathsf{E}(w_{mid}(s)), \hat{B} = \mathsf{E}(\alpha w_{mid}(s)),$
$C = \mathsf{E}(y_{mid}(s)), \ \hat{C} = \mathsf{E}(\alpha y_{mid}(s)),$
$D = \mathsf{E}(h(s)), \ \hat{D} = \mathsf{E}(\alpha h(s)), \ F = \mathsf{E}(L_\beta).$
return $\pi = (A, \hat{A}, B, \hat{B}, C, \hat{C}, D, \hat{D}, F)$

$\underline{\mathsf{Verify}(\mathsf{vk}, u, \pi)}$

$\pi = (A, \hat{A}, B, \hat{B}, C, \hat{C}, D, \hat{D}, F),$
$A = \mathsf{E}(V_{mid}), \quad \hat{A} = \mathsf{E}(\hat{V}_{mid}),$
$B = \mathsf{E}(W_{mid}), \hat{B} = \mathsf{E}(\hat{W}_{mid}),$
$C = \mathsf{E}(Y_{mid}), \ \hat{C} = \mathsf{E}(\hat{Y}_{mid}),$
$D = \mathsf{E}(H), \ \hat{D} = \mathsf{E}(\hat{H}), \ F = \mathsf{E}(L)$
$v_{io}(x) = \sum_{k=0}^\ell a_k v_k(x)$
$w_{io}(x) = \sum_{k=0}^\ell a_k w_k(x)$
$y_{io}(x) = \sum_{k=0}^\ell a_k y_k(x)$
$L_{span} = r_v V_{mid} + r_w W_{mid} + r_y Y_{mid}$
$P = (v_{io}(s) + V_{mid}) \cdot (w_{io}(s) + W_{mid}) - (y_{io}(s) + Y_{mid})$

Check: $\hat{V}_{mid} = \alpha V_{mid},$
$\hat{W}_{mid} = \alpha W_{mid},$
$\hat{Y}_{mid} = \alpha Y_{mid},$
$$\hat{H} = \alpha H \tag{5}$$
$$L = \beta L_{span} \tag{6}$$
$$P = H \cdot t(s) \tag{7}$$

Fig. 1. The Rinocchio scheme for (zk-)SNARKs over a ring R.

5.2 Security of our Scheme

We are now ready to state the main theorem and prove that Rinocchio scheme satisfies the properties of a SNARK as stated in Defn. 1.

Theorem 3. *Let R be commutative ring with identity and $A \subseteq R$ an exceptional set. Let d be an upper bound on the degree of the QRP. Assuming that the generalized augmented $(4d+3)$-PKE and the generalized $(4d+4)$-PDH assumptions hold for the encoding scheme* Encode *over R (and A^*), the* Rinocchio *protocol described in Fig. 1 is a SNARK as per Defn. 1, with soundness error $1/|A^*|$.*

Security Proof Sketch. In the following, we provide intuition and an informal sketch of the security reduction for our construction. We refer the reader to Appendix D for the full proof.

The CRS contains encodings of powers of some random secret point s as well as encodings of the QRP polynomials evaluated at s. The construction asks the prover to present encodings computed homomorphically using this CRS. Furthermore, the prover has to duplicate its effort with respect to scalars $\alpha, \alpha_v, \alpha_w, \alpha_y$. This allows the simulator to extract representations of terms as polynomials of a certain degree using the augmented d-PKE extractor. The crs also contains terms multiplied by a value β that enforce the prover to compute its encoding $\mathsf{E}(L)$ as a linear combination of some given encoded polynomials. In the case when a proof $\hat{\pi}$ would be accepted by the verifier but the statement is not true, we can build an adversary $\mathcal{B}$ that is able to solve the q-PDH problem.

The adversary $\mathcal{B}$, given its q-PDH challenge, tailors a CRS by picking values $r'_v, r'_w, r'_y, \alpha, \alpha_v, \alpha_w, \alpha_y$ and β. Since the proof $\hat{\pi}$ verifies but the statement is false, we can show that then one of the following must hold, where $V(x) = \sum_{k \in I_{io}} c_k v_k(x) + V_{mid}(x)$ (similarly $W(x), Y(x)$) and $V_{mid}(x)$ is an extracted polynomial (through the d-PKE assumption).:

Case 1: $V(x) \cdot W(x) - Y(x) \neq H(x) \cdot t(x)$, but Equation (7) holds, therefore, $V(s) \cdot W(s) - Y(s) = H(s) \cdot t(s)$.

Case 2: $U(x) = r'_v x^{d+1} V_{mid}(x) + r'_w x^{2(d+1)} W_{mid}(x) + r'_y x^{3(d+1)} Y_{mid}(x)$ is not in the module S generated by the R-linear combinations of the polynomials $\{u_k(x) = r'_v x^{d+1} v_k(x) + r'_w x^{2(d+1)} w_k(x) + r'_y x^{3(d+1)} y_k(x)\}_{k \in I_{mid}}$.

If the first case holds, then $\gamma(x) = V(x) \cdot W(x) - Y(x) - H(x) \cdot t(x)$ is a nonzero polynomial of degree some $k \leq 2d$ that has s as a root. The simulator can then from $\gamma(x)$ and the PDH challenge subtract off encodings of lower powers of s to get $\mathsf{E}(s^{q+1})$ and solve q-PDH. The second case follows a similar strategy, this time invoking Lemma 4 and reasoning about $U(x)$.

Strong Soundness. We remark that we do not prove *strong* soundness, which demands that soundness holds even when the prover has access to the verification oracle. While some designated-verifier schemes are provably strongly sound, the reduction requires the d-PKEQ assumption (see Assumption 3 in Appendix C) on the encoding scheme to hold. For the sake of keeping Rinocchio as general as possible in the choice of rings and encodings, we do not make that assumption, but our result could be adapted to that case.

5.3 Adding Zero-knowledge: zk-Rinocchio

We can make our construction zero-knowledge by randomizing the elements in the proof π such that the checks verify and the proof is statistically indistinguishable from random encodings. The idea is for the prover to add random multiples of $t(x)$ to the proof terms so that we can define a simulator that "fakes" the proof elements from completely random values. In more detail:

The prover chooses random $\delta_v, \delta_w, \delta_y \leftarrow R^*$, and adds $\delta_v t(s)$ inside the encoding to $v_{mid}(s)$; $\delta_w t(s)$ to $w_{mid}(s)$; and $\delta_y t(s)$ to $y_{mid}(s)$. It is easy to see that the modified value of $p(x)$ remains divisible by $t(x)$. We need to add additional elements to the crs to allow for this computation. The construction zk-Rinocchio is given in Fig. 2.

Theorem 4. *Let R be commutative ring with identity with an exceptional subset A, and d be an upper bound on the degree of the QRP. Assuming that the generalized augmented $(4d+3)$-PKE and the generalized q-PDH assumptions hold for the encoding scheme* Encode *over R (and A^*) for $q = 4d + 4$, the protocol* zk-Rinocchio *described in Fig. 2 is a zk-SNARK as per Defn. 2, with soundness error $1/|A^*|$.*

Proof. We will prove the zero-knowledge property by showing that zk-Rinocchio is statistically ZK. The theorem will follow from the proof of Theorem 3 and the ZK property. Towards this, we first note the following: for a fixed crs and statement u, given the elements $v_{mid}, w_{mid}, y_{mid}$ that are encoded in π, the rest of the elements that are encoded in π are detremined by the constriants given by the verification equations. Fix $V_{mid}, W_{mid}, Y_{mid}$. This fixes $\hat{V}_{mid}, \hat{W}_{mid}, \hat{Y}_{mid}$, and also P since the coefficients $\{a_k\}_{k=0}^{\ell}$ of v_{io}, w_{io}, y_{io} are given by u, and $P = (v_{io}(s) + V_{mid}) \cdot (w_{io}(s) +$

zk-Rinocchio

Setup$(1^\kappa, \mathcal{R})$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \quad s \leftarrow A^*, \; r_v, r_w \leftarrow R^*, \; r_y = r_v \cdot r_w$

$\alpha, \alpha_v, \alpha_w, \alpha_y \leftarrow R^*, \; \beta \leftarrow R \setminus \{0\}$

$$\begin{aligned}\mathsf{crs} = \Big(&\{\mathsf{E}(s^i)\}_{i=0}^d, \{\mathsf{E}(r_v v_k(s))\}_{k\in I_{mid}}, \{\mathsf{E}(r_w w_k(s))\}_{k\in I_{mid}}, \{\mathsf{E}(r_y y_k(s))\}_{k\in I_{mid}},\\ &\{\mathsf{E}(\alpha s^i)\}_{i=0}^d, \{\mathsf{E}(\beta(r_v v_k(s) + r_w w_k(s) + r_y y_k(s))\}_{k\in I_{mid}},\\ &\mathsf{E}(r_v \alpha t(s)), \mathsf{E}(r_w \alpha t(s)), \mathsf{E}(r_y \alpha t(s)), \mathsf{E}(r_v \beta t(s)), \mathsf{E}(r_w \beta t(s)), \mathsf{E}(r_y \beta t(s)), \mathsf{pk}\Big) \qquad (8)\\ \mathsf{vk} = (&\mathsf{sk}, \mathsf{crs}, s, \alpha, \beta, r_v, r_w, r_y)\end{aligned}$$

Prove(crs, u, w)

$u = (a_1, \ldots, a_\ell), \; a_0 = 1,$
$w = (a_{\ell+1}, \ldots, a_m)$
$v(x) = \sum_{k=0}^m a_k v_k(x)$
$v_{mid}(x) = \sum_{k\in I_{mid}} a_k v_k(x)$
$w(x) = \sum_{k=0}^m a_k w_k(x)$
$w_{mid}(x) = \sum_{k\in I_{mid}} a_k w_k(x)$
$y(x) = \sum_{k=0}^m a_k y_k(x)$
$y_{mid}(x) = \sum_{k\in I_{mid}} a_k y_k(x)$
$h(x) = \dfrac{v(x)w(x) - y(x)}{t(x)}$
$\delta_v, \delta_w, \delta_y \leftarrow R^*$
$h'(x) = h(x) + \delta_v w(x) + \delta_w v(x) + \delta_v \delta_w t(x) - \delta_y$
$L_\beta = \beta\big(r_v v_{mid}(s) + r_w w_{mid}(s) + r_y y_{mid}(s)\big)$
$A = \mathsf{E}(v_{mid}(s) + \delta_v t(s))$
$\hat{A} = \mathsf{E}(\alpha(v_{mid}(s) + \delta_v t(s)))$
$B = \mathsf{E}(w_{mid}(s) + \delta_w t(s))$
$\hat{B} = \mathsf{E}(\alpha(w_{mid}(s) + \delta_w t(s)))$
$C = \mathsf{E}(y_{mid}(s) + \delta_y t(s))$
$\hat{C} = \mathsf{E}(\alpha(y_{mid}(s) + \delta_y t(s)))$
$D = \mathsf{E}(h'(s)), \; \hat{D} = \mathsf{E}(\alpha h'(s)), \; F = \mathsf{E}(L_\beta)$
return $\pi = (A, \hat{A}, B, \hat{B}, C, \hat{C}, D, \hat{D}, F)$

Verify(vk, u, π)

$\pi = (A, \hat{A}, B, \hat{B}, C, \hat{C}, D, \hat{D}, F),$
$A = \mathsf{E}(V_{mid}), \quad \hat{A} = \mathsf{E}(\hat{V}_{mid}),$
$B = \mathsf{E}(W_{mid}), \; \hat{B} = \mathsf{E}(\hat{W}_{mid}),$
$C = \mathsf{E}(Y_{mid}), \; \hat{C} = \mathsf{E}(\hat{Y}_{mid}),$
$D = \mathsf{E}(H), \hat{D} = \mathsf{E}(\hat{H}), F = \mathsf{E}(L)$
$v_{io}(x) = \sum_{k=0}^\ell a_k v_k(x)$
$w_{io}(x) = \sum_{k=0}^\ell a_k w_k(x)$
$y_{io}(x) = \sum_{k=0}^\ell a_k y_k(x)$
$L_{span} = r_v V_{mid} + r_w W_{mid} + r_y Y_{mid}$
$P = (v_{io}(s) + V_{mid}) \cdot (w_{io}(s) + W_{mid}) - (y_{io}(s) + Y_{mid})$

Check:
$$\begin{aligned}\hat{V}_{mid} &= \alpha V_{mid},\\ \hat{W}_{mid} &= \alpha W_{mid},\\ \hat{Y}_{mid} &= \alpha Y_{mid},\\ \hat{H} &= \alpha H \qquad (9)\\ L &= \beta L_{span} \qquad (10)\\ P &= H \cdot t(s) \qquad (11)\end{aligned}$$

Fig. 2. The zk-Rinocchio scheme for zk-SNARKs over a ring R.

$W_{mid}) - (y_{io}(s) + Y_{mid})$. Therefore, this fixes $H = P/t(s)$, and also $\hat{H}$. Now, $V_{mid}, W_{mid}, Y_{mid}$ are computed by adding uniformly random values $\delta_v t(s), \delta_w t(s), \delta_y t(s)$ respectively, and are therefore statistically uniform, since $t(s) \in R^*$ with high probability. We now construct a simulator $(\mathcal{S}_1, \mathcal{S}_2)$. $\mathcal{S}_1$ outputs a simulated CRS crs' and sets the trapdoor τ to be $(s, r_v, r_w, \alpha, \alpha_v, \alpha_w, \alpha_y, \beta)$. $\mathcal{S}_2$ takes as input crs', trapdoor τ, statement u and produces a simulated proof π' as follows. $\mathcal{S}_2$ samples random $v(x), w(x), y(x)$ such that $t(x)$ divides $v(x) \cdot w(x) - y(x)$, and sets $h(x)$ to be the quotient polynomial. Using the statement u, $\mathcal{S}_2$ computes $v_{mid}(x) = v(x) - v_{io}(x)$, $w_{mid}(x) = w(x) - w_{io}(x)$ and $y_{mid}(x) = y(x) - y_{io}(x)$. Now, $\mathcal{S}_2$ uses the trapdoor τ to compute elements that are to be encoded as part of the proof; it uses s to compute encodings of $v_{mid}(s), w_{mid}(s), y_{mid}(s), h(s)$, uses knowledge of s, α to compute encodings of $\alpha v_{mid}(s), \alpha w_{mid}(s), \alpha y_{mid}(s), \alpha h(s)$, and knowledge of s, β, r_v, r_w, r_y to compute an encoding of L. The encoded values satisfy the verification equations and are statistically uniform elements just as an honestly generated proof.

6 Groth16-Like Construction based on Linear-Only Encodings

We construct a zk-SNARK scheme for ring computations with efficiency close to its field-restricted counterpart proposed in [34].

Let C be an arithmetic circuit over R, with m wires and d multiplication gates. Let $Q = (t(x), \{v_k(x), w_k(x), y_k(x)\}_{k=0}^{m})$ be a QRP which computes C. We denote by $I_{io} = 1, 2, \ldots \ell$ the indices corresponding to the public input and public output values of the circuit wires and by $I_{mid} = \ell+1, \ldots m$, the wire indices corresponding to non-input, non-output intermediate values. Let $\mathsf{Encode} = (\mathsf{Gen}, \mathsf{E})$ be a secure encoding scheme and $A^* \subset R^*$ an exceptional set.

Our scheme is based on the assumption of linear-only encodings and consists in 3 algorithms $\mathsf{RingSNARK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ described in Figure 3.

$\underline{\mathsf{Setup}(1^\kappa, \mathcal{R}):}$
$\alpha, \beta, \gamma, \delta \leftarrow R^*, \quad s \leftarrow A^*, \qquad (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$
$\mathsf{crs} = \Big(\mathsf{pk}, \{\mathsf{E}(s^i)\}_{i=0}^{d-1}, \ \{\mathsf{E}(\frac{\beta v_k(s)+\alpha w_k(s)+y_k(s)}{\gamma})\}_{k\in I_{io}},$
$\{\mathsf{E}(\frac{\beta v_k(s)+\alpha w_k(s)+y_k(s)}{\delta})\}_{k\in I_{mid}}, \ \{\mathsf{E}(\frac{s^i t(s)}{\delta})\}_{i=0}^{d-1}\Big)$
$\mathsf{vk} = (\mathsf{sk}, \mathsf{crs}, s, \alpha, \beta, \gamma, \delta)$
return $(\mathsf{crs}, \mathsf{vk})$

$\underline{\mathsf{Prove}(\mathsf{crs}, u, w)}$
$u = (a_1, \ldots, a_\ell), \ a_0 = 1$
$w = (a_{\ell+1}, \ldots, a_m)$
$v(x) = \sum_{k=0}^m a_k v_k(x)$
$v_{mid}(x) = \sum_{k\in I_{mid}} a_k v_k(x)$
$w(x) = \sum_{k=0}^m a_k w_k(x)$
$w_{mid}(x) = \sum_{k\in I_{mid}} a_k w_k(x)$
$y(x) = \sum_{k=0}^m a_k y_k(x)$
$y_{mid}(x) = \sum_{k\in I_{mid}} a_k y_k(x)$
$h(x) = \frac{(v(x)w(x)-y(x))}{t(x)}$
$f_{mid} = \frac{\beta v_{mid}(s)+\alpha w_{mid}(s)+y_{mid}(s)}{\delta}$
$A = \mathsf{E}\big(\alpha + v(s)\big)$
$B = \mathsf{E}\big(\beta + w(s)\big)$
$C = \mathsf{E}(f_{mid} + \frac{t(s)h(s)}{\delta})$
return $\pi = (A, B, C)$

$\underline{\mathsf{Verify}(\mathsf{vk}, u, \pi)}$
$\pi = (A, B, C)$
$A = \mathsf{E}(A_v)$
$B = \mathsf{E}(B_w)$,
$C = \mathsf{E}(C_y)$
$v_{io}(x) = \sum_{i=0}^\ell a_i v_i(x)$
$w_{io}(x) = \sum_{i=0}^\ell a_i w_i(x)$
$y_{io}(x) = \sum_{i=0}^\ell a_i y_i(x)$
$f_{io} = \frac{\beta v_{io}(s)+\alpha w_{io}(s)+y_{io}(s)}{\gamma}$
$F = \mathsf{E}(f_{io})$
Check on encodings
$AB = \mathsf{E}(\alpha)\mathsf{E}(\beta) + \gamma F + \delta C$
i.e.
$A_v B_w = \alpha\beta + \gamma f_{io} + \delta C_y$

Fig. 3. RingSNARK Construction from Linear-only Encodings.

Theorem 5. *Let R be commutative ring with identity with an exceptional subset A, and d be an upper bound on the degree of the QRP. Assuming that the linear-only extractable assumption as per Definition 10 holds for the encoding scheme* Encode *over R (and A^*), the protocol* $\mathsf{RingSNARK}$ *described in Fig. 3 is a SNARK as per Definition 1, with soundness error* $1/|A^*|$.

Proof of Security We first give a variant of the Schwartz-Zippel lemma for Laurent polynomials over rings that we will rely on in the proof.

Lemma 5. *Let A be an exceptional set. Let $h(X) \in R[X_1, X_1^{-1}, \ldots, X_n, X_n^{-1}]$ where no term in any X_i has degree less than $-D$ or larger than D. Let us assume that $h(X)$ is not the zero-polynomial.*

Let $\boldsymbol{a} \in (A)^n$ be chosen uniformly at random. Then

$$\Pr[h(\boldsymbol{a}) = 0] \leq \frac{2nD}{|A|}.$$

Proof. We notice that $f(X) := \prod_{i=1}^{n} X_i^D \cdot h(X)$ is an ordinary polynomial of degree $\leq 2nD$. Since $h(\boldsymbol{a}) = 0$ implies $f(\boldsymbol{a}) = 0$, by the generalized Schwartz-Zippel lemma (Lemma 2), we have that

$$\Pr[h(\boldsymbol{a}) = 0] \leq \Pr[f(\boldsymbol{a}) = 0] \leq \frac{2nD}{|A|},$$

finishing the proof. ■

We are now ready to give the security proof of our scheme RingSNARK:

Theorem 6. *Let R be commutative ring with identity with an exceptional subset A, and d be an upper bound on the degree of the QRP. Assuming that the linear-only extractable assumption as per Definition 10 holds for the encoding scheme* Encode *over R (and A^*), the protocol* RingSNARK *described in Fig. 3 is a SNARK as per Definition 1, with soundness error $1/|A^*|$.*

Proof. **Completeness.** Completeness of the SNARK protocol follows by QRP completeness and by the (statistical) correctness of the Encode scheme.
Knowledge Soundness. We will show the existence of an extrator that on same input and random coins as $\mathcal{A}$ can produce a valid witness whenever the prover $\mathcal{A}$ outputs a valid proof. Let $\mathcal{A}$ be the PPT adversary in the game for knowledge soundness (Definition 1) able to produce a proof π for which the verification algorithm returns true. By linear-only extractable assumption 10 we can run an extractor that gives us a vector of coefficients $A_\alpha, A_\beta, A_\gamma, A_\delta, \{\mathcal{A}_k\}_{k=0}^m$ and polynomials $A(x), A_h(x)$ of degree $d-1, d-2$ such that the value encoded in the proof element A can be written as a linear combination of the initial values encoded in the crs:

$$\begin{aligned} A_v = A_\alpha \alpha + A_\beta \beta + A_\gamma \gamma + A(s) + \sum_{k=0}^{\ell} A_k \frac{\beta v_k(s) + \alpha w_k(s) + y_k(s)}{\gamma} + \\ + \sum_{k=\ell+1}^{m} A_k \frac{\beta v_k(s) + \alpha w_k(s) + y_k(s)}{\delta} + A_h(s) \frac{t(s)}{\delta} \end{aligned} \tag{12}$$

We can write out B_w and C_y in a similar fashion. We can see the verification equation as an equality of multivariate Laurent polynomials. By Lemma 5, $\mathcal{A}$ has negligible success probability unless the verification equation holds when viewing A_v, B_w and C_y as formal polynomials in indeterminates $x_\alpha, x_\beta, x_\gamma, x_\delta, x_s$.

Using the verification test equations and following the same reasoning as the proof in [34] we eliminate coefficient by coefficient until we obtain:

$$A(x) = \sum_{k=0}^{m} a_k v_k(x), \quad B(x) = \sum_{k=0}^{m} a_k w_k(x), \quad C(x) = \sum_{k=0}^{m} a_k y_k(x).$$

This implies that $w = (a_{\ell+1}, \ldots, a_m)$ is a witness for $u = (a_1, \ldots, a_\ell)$. ■

7 SNARKs for Computation over Encrypted Data

In this section we detail how we can apply Rinocchio to the problem of verifiable computation over encrypted data. Our approach is generic, where we just run a proving mechanism – the (zk-)SNARK – on pre-existing Homomorphic Encryption (HE) schemes in a modular way. Taking advantage of our protocol Rinocchio with zero-knowledge (see Section 5.3) this reduces to finding secure encoding schemes over a ring that are compatible with the ciphertext space of the underlying HE scheme.

In Section 7.1 we review some popular homomorphic encryption schemes that are good candidates for realising our privacy-preserving VC scheme. Then, by using a secure encoding scheme E as the ones we provide in Section 7.2, we can invoke Theorem 3 to obtain a DV-SNARK for $\mathcal{R}_q$, as explained in Section 7.3.

7.1 Homomorphic Encryption Schemes and their Parameters

The first fully homomorphic encryption schemes were based on the Learning With Errors (LWE) problem [44], which is the main assumption behind schemes with ciphertexts in the ring $\mathbb{Z}_q$ such as [12]. Nevertheless, the most efficient HE schemes are based on the Ring-LWE problem.

For the Ring-LWE-based schemes we will work with, the ring of plaintexts is $\mathcal{R}_p = \mathbb{Z}_p[Y]/(f(Y))$ and the ring of cyphertexts is $\mathcal{R}_q^2$, where $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$ for some degree-N polynomial $f(Y)$. This is usually picked to be a cyclotomic polynomial, so that it factors into ℓ irreducible factors modulo p. More concretely, $f(Y) \equiv \prod_{i=1}^{\ell} f_i(Y) \mod p$, where each $f_i(Y)$ has degree $\phi(N)/\ell$. By imposing $p \equiv 1 \mod N$, this creates ℓ "plaintext slots", and hence a popular choice is $f(Y) = Y^N + 1$, where N is a power of two. In order to deal with the *noise growth* that affects these schemes, q has to be chosen large (several hundreds of bits) and the rank of the associated lattice, which corresponds to N, has to be high enough to meet security requirements (usually between 2^{10} and 2^{15}).

Frequently, q is chosen so that $q = \prod_{i=1}^{k} p_i$. While this does not affect the asymptotic complexity of operations on ciphertexts, it brings an important gain in practice: The polynomials of $\mathcal{R}_q$ are represented as k polynomials of same degree but with smaller coefficients, thanks to the ring isomorphism given by the CRT. In many cases, these smaller primes fit native (64-bit) integer data types, which speeds up computation and hence this representation is implemented in the SEAL (`https://github.com/Microsoft/SEAL`), Lattigo (`https://github.com/ldsec/lattigo`) and PALISADE (`https://palisade-crypto.org/`) libraries. Being able to efficiently deal with non-prime choices for q is hence a significant advantage of our work, compared to prior results [28]. Depending on the choice of q and $f(Y)$ in the underlying schemes, we have different options for our exceptional sets. Generally speaking, if $q = \prod_{i=1}^{k} p_i$, where $p \leq p_1 < p_2 < \ldots < p_k$ and p comes from the plaintext space $\mathcal{R}_p$, we can always find the exceptional set $A^* = \{1, 2, \ldots, p_1 - 1\} \subset R^*$. Hence, if p_1 is big enough we don't need to worry about anything else. Otherwise, we can move to an extension of the ciphertext ring or apply a parallel soundness amplification strategy as we discuss in Section 8.3.

Concrete Ring-LWE schemes. The HE schemes that we will consider for our privacy-preserving VC are BV [14], BGV [13] and FV [26]. We are interested in "somewhat homomorphic" variants of these schemes, where the parameters are set just large enough so as to enable homomorphic evaluation of some target function which will be represented as a QRP and hence fixed in Rinocchio's crs.

In this setting, schemes like BGV [13] (and a variant of FV [26]) use so-called modulo-switching. They require a chain of moduli $q_0 < \cdots < q_L$ to be able to scale the noise down after each multiplication by switching the ciphertext to a smaller modulus. When evaluating circuits with large multiplicative depth, one needs to choose a large chain of moduli and thus use higher dimensions, resulting in poor performance.

Scale invariant schemes allow to partially overcome this limitation by removing the need of the modulus-switching procedure, which potentially results in the possibility of evaluating circuits with a bigger multiplicative depth. In his seminal work [12], Brakerski introduced a new scale-invariant scheme based on classical LWE where the noise grows only linearly during multiplication. This more effective noise control mechanism makes the scale-invariant schemes particularly interesting. In [26], the scale-invariant scheme of [12] was adapted to the Ring-LWE setting.

Each Ring-LWE scheme is best suited for different types of operations. BGV [13] uses, in general, slow operations, but benefits from optimizations to treat many bits at the same time, while FV [26] allows to perform large vectorial arithmetic operations as long as the multiplicative depth of the evaluated circuit remains small.

7.2 Secure Encodings for (Ring-)LWE ciphertexts

We introduce two different instantiations for the encoding scheme, one suitable for the ciphertext ring $\mathbb{Z}_q$ that appears in LWE-based HE and the other one for a polynomial ring $\mathcal{R}_q$, as in the ciphertext ring of Ring-LWE-based schemes.

Regev-style Encoding. Here, we consider the ring $\mathbb{Z}_q$ as the input space of the encoding (the ring R over which the QRP is defined). This matches the ciphertext ring of LWE-based HE schemes such as [12]. Note that $\mathbb{Z}_q$ need not be a field. In fact, a popular choice for q is a product of co-prime numbers $q = \prod_i q_i$ with some extra conditions on q_i's as discussed in works as [44, 43].

The encoding $\mathsf{E.Regev}$ we consider over the ring $\mathbb{Z}_q$ is the same as the one used to construct lattice-based SNARGs and SNARKs in [32], a slight variation of the classical LWE cryptosystem initially presented by Regev [44]. The encryption scheme is described by parameters $\Gamma \leftarrow (q, Q, n, \alpha)$, with $q, Q, n \in \mathbb{N}$ such that $(q, Q) = 1$, and $0 < \alpha < 1$. We will also consider $\chi_\sigma(S)$, the discrete Gaussian distribution over a discrete set S with mean 0 and parameter σ.

$\mathsf{Gen}(1^\kappa, \Gamma)$: Choose some random string $\boldsymbol{s} \leftarrow \mathbb{Z}_Q^n$. Output $\mathsf{sk} = \boldsymbol{s}$.

$\mathsf{E}_{\mathsf{sk}}(m)$: Given $m \in \mathbb{Z}_q$, sample $\boldsymbol{a} \leftarrow \mathbb{Z}_Q^n$, define $\sigma = Q\alpha$; $\ e \leftarrow \chi_\sigma(\mathbb{Z}_q)$. Output $C = (-\boldsymbol{a},\ \boldsymbol{a} \cdot \boldsymbol{s} + qe + m)$.

$\mathsf{D}_{\mathsf{sk}}(C)$: Parse $\mathsf{sk} = \boldsymbol{s}, C = (\boldsymbol{a}_0, c_1)$ Compute $m = (\boldsymbol{a}_0 \cdot \boldsymbol{s} + c_1) \mod q$.

On the suitability of the encoding. It is easy to see that this is a *statistically-correct* encoding scheme. When encodings are added together and multiplied by scalars, the noise starts to build up. Nevertheless, for any fixed ℓ there is a choice of parameters Γ such that the encoding is *ℓ-linearly-homomorphic*. Consequently, in order to ensure that we obtain a valid encoding of the result, we need to start with sufficiently small noise in each of the initial encodings. A more detailed discussion about the noise growth and the choices for Γ can be found in [9, 10, 32] that specifically address SNARK applications of this encoding. The *quadratic root detection* and *image verification* properties can be implemented using D_{sk}.
Security: Regarding security, this encoding scheme was already used and conjectured as linear-only and secure against generalized q-PDH assumption over fields by prior works. Our generalized q-PDH assumption over rings extends this, taking into account encodings over rings. The constructions

in [9, 10] also employ $\mathsf{E.Regev}$ to instantiate their SNARG(K) and this is assumed to be linear-only extractable which, as we show in Appendix C, is a stronger assumption than our *secure encoding*, i.e. an encoding that satisfies both the Generalized q-PDH and the Generalized Augmented q-PKE assumptions.
Extension to Ring-LWE. Our Regev-style encoding can be naturally extended to an encoding over the ring $\mathcal{R}_q$ used in Ring-LWE based schemes by combining N copies of the encoding above *under the same key*, similar to what we will do in the Torus Encoding or in Section 8.1.

Torus Encoding. We will use a variant of the Torus FHE (TFHE) cryptosystem from [21]. We let $\mathcal{R}_{\mathbb{R}} = \mathbb{R}[Y]/(f(Y))$, $\mathcal{R}_{\mathbb{Z}} = \mathbb{Z}[Y]/(f(Y))$ and $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$ denote the quotient rings with respect to some polynomial $f(Y) = Y^N + 1$, where m is an integer and N is a power of 2. We let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ be the torus, which is a $\mathbb{Z}$-module structure but not a ring.

We consider the $\mathcal{R}_{\mathbb{Z}}$-module $\mathbb{T}_{\mathcal{R}} = \mathcal{R}_{\mathbb{R}}/\mathcal{R}_{\mathbb{Z}}$. The plaintext for the TFHE cryptosystem is the $\mathbb{Z}$-module $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. Our encoding scheme $\mathsf{E.Torus}$ has $\mathbb{Z}_q$ as message space and will be used for encoding of elements in $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$. The key remark is that the ring $\mathcal{R}_q$ can be identified with a subgroup of the torus $\mathbb{T}^N$ via the map $\mathcal{R}_q \simeq \mathbb{Z}_q^N$ that identifies $q^{-1}\mathbb{Z}/\mathbb{Z} \simeq \mathbb{Z}_q$ as an isomorphism of $\mathbb{Z}$-modules. Also, $\mathbb{T}^N \simeq \mathbb{T}_{\mathcal{R}}$ because $\mathbb{T}^N$ can be seen as a vector of coefficients. The module structure of the encoding space $\mathbb{T}^{N+1}$ allows us to conjecture that $\mathsf{E.Torus}$ scheme only supports linear homomorphic operations.

Let $\mathbb{B} = \{0, 1\}$. The encoding scheme $\mathsf{E.Torus}$ is described by parameters $\mathit{\Gamma} \leftarrow (q, N, \alpha)$, with $q, N \in \mathbb{N}$ such that $0 < \alpha < 1$. The noise parameter α is the standard deviation for a concentrated distribution on the torus (more details can be found in [21]). Below, we describe the algorithms of the encoding:

$\mathsf{Gen}(1^\kappa, \mathit{\Gamma})$**:** Choose a random vector $\boldsymbol{s} \in \mathbb{B}^N$. Output $\mathsf{sk} = \boldsymbol{s}$.
$\mathsf{E}_{\mathsf{sk}}(m)$**:** Given $\mathsf{sk} = \boldsymbol{s} \in \mathbb{B}^N$ and $m \in \mathbb{Z}_q$, apply the map $\mathbb{Z}_q \simeq q^{-1}\mathbb{Z}/\mathbb{Z}$ to m and get $m' \in \mathbb{T}$ such that $m' \equiv m/q \mod 1$, sample a vector $\boldsymbol{a} \in \mathbb{T}^N$ and compute $b = \boldsymbol{s} \cdot \boldsymbol{a} + m' + e$ where $e \in \mathbb{T}$ is sampled according to a noise distribution defined by the standard deviation α. Output encoding $C = (\boldsymbol{a}, b)$.
$\mathsf{D}_{\mathsf{sk}}(C)$**:** Parse $\mathsf{sk} = \boldsymbol{s}, C = (\boldsymbol{a}, b)$. Compute $m" = b - \boldsymbol{a} \cdot \boldsymbol{s} = m" + e$. Round $m"$ to the nearest point m' on the torus with respect to a distance function and apply the equivalence $q^{-1}\mathbb{Z}/\mathbb{Z} \simeq \mathbb{Z}_q$ to recover m.

The ring-LWE variant of the torus encoding scheme $\mathsf{E.Torus}_r$ works for message space $\mathcal{R}_q$ as follows:

$\mathsf{Gen}(1^\kappa, \mathit{\Gamma})$**:** Choose a random polynomial $s(Y) \in \mathcal{R}_{\mathbb{Z}}$ with coefficients in $\{0, 1\}$. Output $\mathsf{sk} = s(Y)$.
$\mathsf{E}_{\mathsf{sk}}(m)$**:** Given $\mathsf{sk} = s(Y)$ and $m(Y) \in \mathcal{R}_q \simeq \mathbb{Z}_q^N$, apply the map $\mathbb{Z}_q \simeq q^{-1}\mathbb{Z}/\mathbb{Z}$ to each component of $m(Y)$ and get $m'(Y) \in \mathbb{T}_{\mathcal{R}} \simeq \mathbb{T}^N$, sample a polynomial $a(Y) \in \mathbb{T}^N$ and compute $b(Y) = s(Y) \cdot a(Y) + m' + e(Y)$ where $e(Y) \in \mathbb{T}_{\mathcal{R}}$ is sampled according to a noise distribution defined by the standard deviation α.
$\mathsf{D}_{\mathsf{sk}}(C)$**:** Parse $\mathsf{sk} = s(Y), C = (a(Y), b(Y))$. Compute $m"(Y) = b(Y) - a(Y) \cdot s(Y) = m"(Y) + e(Y)$. Round coefficients of $m"(Y)$ to the nearest ones on the torus to obtain $m'(Y) \in \mathbb{T}_{\mathcal{R}}$ and apply the equivalence $q^{-1}\mathbb{Z}/\mathbb{Z} \simeq \mathbb{Z}_q$ to recover $m(Y) \in \mathcal{R}_q$.

On the suitability of the encoding. It is easy to see that this is a *statistically-correct* encoding scheme and due to the linearly-homomorphic property of the cryptosystem (see Appendix E for specific details), for a fixed ℓ, there is a choice of parameters $\mathit{\Gamma}$ such that we have *ℓ-linearly-homomorphic*. The *quadratic root detection* and *image verification* properties can be implemented using D_{sk}.

Security: $\mathsf{E.Torus}$ is semantically secure under the TLWE assumption, a generalized intractability problem similar to LWE. Also, it is plausible that $\mathsf{E.Torus}$ only permits linear homomorphisms, therefore we conjecture that this is a *secure encoding*, satisfying both q-PDH and q-PKE assumptions. A heuristic argument for believing multiplication of two encoded values is impossible is the torus structure of the encoding space, $\mathbb{T}$ is a $\mathbb{Z}$-module and not a ring (i.e., the product of elements in $\mathbb{T}$ is not well defined), so there is no way for one to compute any missing $\mathsf{E}(s^{q+1})$ to solve q-PDH. Of course, the original TFHE encryption scheme defined in [21] overcomes this limitation: it consists of three major encryption/decryption schemes (each represented by a different plaintext space) and makes use of tools like key-switching, gate bootstrapping and gadget decomposition function to perform computations other than additions. These operations are possible only if some extra keys are available, for example some precomputed ciphertexts of the binary secret key in the case of gate bootstrapping. Since we do not consider all these extensions and we do not provide encodings of the secret key in the crs, our encoding $\mathsf{E.Torus}$ is limited to basic linear operations.

7.3 (zk-)SNARKs for Ring-LWE-based homomorphic encryption

We now have everything we need to instantiate the protocol defined in Section 5.1. We pick the ring $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$, to match the ciphertext space of the Ring-LWE schemes from Section 7.1. We can next choose a secure encoding scheme E from the ones in Section 7.2. Assuming that the evaluation algorithm of the underlying homomorphic encryption scheme (e.g. [14]) does not involve modulus switching and rounding operations, we directly obtain a Designated Verifier SNARK for computation on encrypted data by invoking Theorem 3 for $\mathcal{R}_q$, as explained in Section 7.3.

We could choose schemes that employ modulus switching to deal with the quadratic growth of the noise after a multiplication, such as BGV [13] and FV [26]. In these schemes, there is a chain of moduli $q_i = \prod_{j=1}^{i} p_i$ to successively reduce to (from q to q_k, from q_k to q_{k-1} and so on) when noise builds up, typically after every multiplication. An advantage of the fact that we can work over $\mathcal{R}_q$ natively (rather than emulating its arithmetic) is that reducing modulo a q_i in the chain is simply a multiplication by a public constant, which corresponds to $(1, \dots, 1, 0, \dots, 0)$ in CRT representation, where the amount of non-zero elements in the vector is i. As multiplication by constants can be basically considered “for free” in SNARKs, this is a clear advantage of our work compared with field-based counterparts which cannot work natively over $\mathcal{R}_q$. Nevertheless, BGV and FV are less friendly to $\mathsf{Rinocchio}$ than BV [14], since relinearization requires some bit-wise and/or rounding operations which significantly increase the number of constraints in the QRP. Furthermore, since these operations happen *before* modular reduction, in order to work with BGV and (the modulus-switching variant of) FV, $\mathsf{Rinocchio}$ would have to be instantiated over a ring $\mathcal{R}_s$ where $s > q^2$, rather than $\mathcal{R}_q$. Whereas this is an additional overhead, it does not deny the advantages of working with the CRT representation of ciphertexts and the ease to extract a witness from it, since we can pick $s = P \cdot q$, where $P > q$ is a prime that “creates another slot”. We consider a very interesting venue for a less foundational but more experimental work to determine what would be overall more efficient for the prover: Having a more efficient SNARK but using BV or using more state-of-the-art schemes such as BGV and FV at an increased cost in terms of producing a witness and computing the SNARK.

Context hiding. Another challenge for our VC scheme is preserving privacy of the inputs against the verifier. Such a property would turn useful in the following two example scenarios. In the first one, the party holding the secret key for HE and the verifier (who holds the secret key for the encoding) checking the computation over the ciphertexts are different entities. In the second scenario, the prover wants to compute on ciphertext from the verifier using some secret coefficients (e.g. a Machine Learning model, or his own input in a two-party computation scenario) that he wants to remain private.

The *context hiding* property roughly says that output encodings together with input verification tokens do not reveal any information on the input. Note that this is required to hold even against a party that is in possession of the secret key for the encryption scheme. We can make our VC scheme context-hiding using the same techniques as proposed in [28]. In the HE schemes we propose, information about the underlying plaintexts may be inferred from the distribution of the noise recovered during decryption of the result. To address this, the strategy is to statistically hide the noise. In a nutshell, the trick is to add to the public key some honestly generated encryptions of 0 and then ask the prover to add these to the result of the computation.

7.4 Comparison with [28]

We compare our work with its most close counterpart for this specific application, which is [28]. We remind that the result from [8] is not comparable to ours, since it is not succinct for general circuits and turning it to a non-interactive variant requires relying on random oracles and the Fiat-Shamir heuristic.

The advantage of choosing our SNARK for ring computation as a candidate for the VC scheme is that it is compatible with a set of optimisations on the underlying homomorphic encryption schemes, which leads to a total computational overhead smaller than in prior works not only in terms of the Prove algorithm, but also in the work required to obtained a suitable witness for it beyond a non-verifiable evaluation of the desired function on the ciphertexts.

The work by Fiore et al. [28] relies on bilinear-group based primitives such as commitments and SNARKs, and therefore imposes specific parameters to the ciphertext space, the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$, which are not optimal for the relevant homomorphic schemes known today. Rinocchio supports generic rings $\mathcal{R}_q$ with $q = \prod_{i=1}^{L} q_i$ for a chain of moduli $\{q_1, \ldots, q_L\}$ as in the state-of-art leveled HE schemes, whereas [28] requires q to be a prime.

Another drawback of [28] comes from the trick of moving from ciphertexts in $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$ to scalars in $\mathbb{F}_q$. This requires expensive computations on large degree polynomials in $\mathbb{Z}_q[Y]$. The prover needs to carry all the circuit computations on the ciphertext polynomials without reduction modulo $f(Y)$ along the way (where $f(Y)$ is the quotient polynomial that defines $\mathcal{R}_q = \mathbb{Z}_q[Y]/(f(Y))$). Even if this is not counted in the cost of proof generation, it is an overhead for the worker performing the homomorphic evaluation of the HE scheme. Since $f(Y)$ has a large degree d_f in practice (usually between 2^{11} to 2^{15}, see e.g. the analysis in [8] for BV and Appendix E.2 for BGV and FV), this incurs on a very significant overhead just in obtaining the witness. For a depth-D circuit, the $\mathbb{Z}_q[Y]$ polynomials in the output layer can have a degree $m = 2^D \cdot (d_f - 1)$ and since polynomial multiplication has a complexity of at least $O(m \log m)$, this is an overhead that very soon becomes prohibitive as D increases (which moreover requires to quickly increase d_f too for the security of the underlying HE scheme!).

In our work, such an overhead is not necessary, our techniques allow for the worker/prover to use the existing HE schemes with their latest optimisations for computations over ciphertexts. After the HE evaluation, the prover can use the intermediate ciphertexts from the homomorphic evaluation of the circuit as witness to our SNARK. We remark that these are all elements in the ring $\mathcal{R}_q$ as opposed to large degree integer polynomials in $\mathbb{Z}_q[Y]$ computed in Fiore et al. [28].

Even though they are costly, since they incur rounding operations, Rinocchio also enables noise reduction operations such as relinearization in BGV [13] and FV [26]. These are directly impossible in [28], since they homomorphically hash ciphertexts to a single element of $\mathbb{F}_q$ (for a prime q) and rounding/bit-wise operations are not preserved through the homomorphic hash function.

A qualitative difference is that [28] is a commit-and-prove scheme; and has the inherent drawback that it is limited by the choice of schemes which are compatible with both the commitment scheme and the proof system. Our scheme is an instantiation of a SNARK without combining two different proof systems. We believe one could turn our scheme into a commit-and-prove SNARK along the lines of [2] by "extracting" a suitable encoding to act as a commitment to the input wire values from the SNARK. We leave working out the details to obtain a concrete commit-and-prove scheme for ring computation to future work.

8 SNARKs for Computation over $\mathbb{Z}_{2^k}$

We instantiate Rinocchio for the ring R being the Galois Ring $GR(2^k, \delta)$. As R is a free module over $\mathbb{Z}_{2^k}$ of rank δ, we can embed elements from $\mathbb{Z}_{2^k}$ into the first coordinate of R. Hence, a QRP for arithmetic circuits over $\mathbb{Z}_{2^k}$ can be embedded in a QRP for arithmetic circuits over R. We first discuss a suitable encoding scheme for $R = GR(2^k, \delta)$. Then, we provide a simple, direct instantiation of Theorem 3 using said encoding, together with some QRP gadgets to perform useful computations such as bit decomposition.

8.1 A secure encoding for $GR(2^k, \delta)$

We will use the Joye-Libert (JL) cryptosystem [5], which has $\mathbb{Z}_{2^k}$ as message space, as building block for our encoding of Galois Ring elements.

$\mathsf{KeyGen}(1^\kappa, k)$**:** According to the security parameter κ, choose two random primes p, q satisfying the equivalences:

$$p \equiv 1 \pmod{2^k} \quad \text{and} \quad q \equiv 3 \pmod 4.$$

For simplicity, pick $p = 2^k p' + 1$ and $q = 2q' + 1$, where p', q' are primes. Let g be a random generator of both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, $N = p \cdot q$ and $\mu = p'$. We define $\mathsf{pk} = (g, k, N)$ and $\mathsf{sk} = \mu$.

$\mathsf{Enc}_{\mathsf{pk}}(m)$**:** Given $m \in \mathbb{Z}_{2^k}$, sample $x \leftarrow \mathbb{Z}_N^*$ and output $C = g^m \cdot x^{2^k} \pmod N$.

$\mathsf{Dec}_{\mathsf{sk}}(C)$**:** Compute $c = C^\mu \mod p$ and then retrieve m bit by bit as follows. Observe that $c = C^\mu \mod p = (g^\mu)^m \mod p$, where g^μ is an element of order 2^k in $\mathbb{Z}_p^*$. Let $m = \sum_{j=0}^{k-1} 2^j m_j,\ m_j \in \{0,1\}$. We can compute its least significant bit m_0 by computing $c^{2^{k-1}} \mod p$. Set $m_0 = 0$ if $c^{2^{k-1}} \mod p = 1$, and 1 otherwise. After computing $m_{i-1}, \dots, m_0$, compute m_i as follows: Set $m_i = 0$ if and only if

$$\left(\frac{c}{\left(g^{\mu\left(\sum_{j=0}^{i-1} 2^j m_j\right)}\right)}\right)^{2^{k-i-1}} = 1 \mod p$$

Note that whereas the decryption cost is linear in k, there is empirical evidence [18, Section 5] that it can be faster than more common encryption schemes such as Paillier. The JL cryptosystem is secure under the assumption that k-quadratic residuosity is hard [5]. It is linearly homomorphic over $\mathbb{Z}_{2^k}$, and it has already been employed in the context of efficient two-party computation over $\mathbb{Z}_{2^k}$ (see [18] for concrete efficiency estimates).

Let $R = GR(2^k, \delta)$. Given $a \in R$ written in its additive form $a = a_0 + a_1 X + \ldots + a_{\delta-1}X^{\delta-1}$ (see Equation (1)), we define our encoding $\mathsf{E.JL}$ as follows:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$ calls $\mathsf{KeyGen}(1^\kappa, k)$ in the JL cryptosystem and outputs $(\mathsf{pk}, \mathsf{sk})$.
- $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_\delta} \leftarrow \mathsf{E.JL}_{\mathsf{pk}}(a)$ is a probabilistic encoding algorithm mapping a ring element $a \in R$ to an encoding space $Z = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_\delta}$ such that the sets $\{\{\mathsf{E.JL}(a)\} : a \in R\}$ partition Z, where $\{\mathsf{E.JL}(a)\}$ is the set of encodings of a. Concretely:

$$\mathsf{E.JL}(a) = \big(\mathsf{Enc}_{\mathsf{pk}}(a_0), \ldots, \mathsf{Enc}_{\mathsf{pk}}(a_{d-1})\big)$$

On the suitability of the encoding. The scheme $\mathsf{E.JL}$ clearly satisfies all non-security properties required from an encoding. Under the security assumptions of the JL scheme, it is also reasonable to assume that q-PDH (and q-PKE) hold for the encoding scheme too, where $\mathcal{A}$ has a success probability negligible in d. This dependence on d is intrinsic to the arithmetic in R, as $\mathcal{A}$ can simply output $(a, y) = (2^{k-1}, \mathsf{E.JL}(\sum_{\ell=0}^{d-1} s_{\ell,0} \cdot X^\ell))$, where $s_{\ell,0}$ are $\mathcal{A}$'s guesses for the least significant bit of each $s_\ell \in \mathbb{Z}_{2^k}$ that conform the additive representation of s. Note that the fact that Enhanced-CPA cannot be supported by the JL cyptosystem, as brought up by [18], is not an issue here. Such notion requires interaction with an oracle which the Adversary does not have access to in our construction, as we do not aim to provide *strong* soundness.

8.2 A simple construction

We can instantiate the protocol defined in Section 5.1 for $R = GR(2^k, \delta)$. It follows from inspection that, representing elements of R in their additive notation, $A = \{a_i \in R : a_i = \sum_{j=0}^{\delta-1} a_{i,j} \cdot X^j, a_{i,j} \in \{0,1\}\}$ is an exceptional set in canonical form. Let C be a circuit with d multiplication gates and define A^* as described in Section 5.1. Then, using the secure encoding scheme $\mathsf{E.JL}$ from Section 8.1, we can invoke Theorem 3 to obtain a DV-SNARK for $R \supset \mathbb{Z}_{2^k}$ with a soundness error of $|A^*|^{-1} = (2^\delta - d)^{-1}$.

Efficiency considerations. While in this construction δ is logarithmic in the desired soundness error, we emphasize that our QRP does not suffer from the overhead of adding roughly k multiplication gates whenever a modular reduction $x \mod 2^k$ has to be computed, as it would happen if the circuit was to be run by a SNARK over fields (see our discussion in Section 1.1). Hence, avoiding this and using $\mathsf{Rinocchio}$ allows us not blow-up the *degree* of the QRP, which was an efficiency bottleneck in e.g. [42]. We would further like to note that FFT-style techniques can be applied to Galois Rings [17] and that the price of working with circuits over $GR(2^k, \delta)$, rather than $\mathbb{Z}_{2^k}$, has the potential to be amortized, as it has happened in the context of Multi-Party Computation protocols which faced similar limitations (c.f. [1, 25]).

In Appendix B.3 we show how to build QRPs for bit decomposition, which is useful for practical bit-wise operations such as comparisons. Next, we outline a *soundness amplification* technique. We discuss efficiency considerations of this construction in Appendix 9.

8.3 Soundness Amplification

Despite the previous arguments, there is a concern as to what is the practical impact of the extension degree δ in the previous construction. We believe that this is an interesting question to explore in experimental work, for which we provide one more strategy here. Whereas it would seem that we cannot escape from δ being logarithmically proportional to the soundness error, we it is good enough to apply a parallel repetition strategy as we next describe.

Let d be the number of multiplication gates in the QRP Q. If we choose $\delta \in O(\log(d))$ and work over $R = GR(2^k, \delta)$, the soundness error for a single execution of Rinocchio over R is of $|A^*| = |A| - |A_Q| = 2^\delta - d$. Let us analyze the soundness error for r independent instances of Rinocchio over R for the same QRP (that is, r different crs from r independent Setup executions), for each of which the prover computes the Prove step from the (common) QRP witness. Now, the verifier only accepts if all the r proofs pass verification, yielding a soundness error of $|A^*|^{-r} = (2^\delta - d)^{-r}$. The previous analysis considers that the adversary does not break q-PDH. Since $\mathcal{A}$ has a bigger advantage (of $2q/|A^*|$) in breaking that assumption, it would be its best attack strategy. Still, $\mathcal{A}$ would need to break all the q-PDH instances, which only has a success probability of $(2q/|A^*|)^r$. Recall that $q = 4d + 4$. If we pick $\delta = \log(17d)$, then the best cheating strategy has a success probability of roughly 2^{-r}.

Working over $R = GR(2^k, \delta)$ rather than $\tilde{R} = GR(2^k, S)$ improves the computational efficiency without greatly affecting the total proof or crs sizes. This is due to the fact that the size of each value in R encoded using E.JL is reduced by a multiplicative factor of δ/S compared with $\tilde{R}$. Overall, and since we repeat r times the Rinocchio protocol over R, this results on a total proof and crs size which is $r\delta/S$ times the ones that would result from a single execution of Rinocchio over $\tilde{R}$.

9 Efficiency of Privacy-Preserving VC and Similar Instantiations

A natural question is whether the Rinocchio instantiation from Section 8 will beat a QAP over a field, where reduction modulo k is computed after every multiplication gate by using bit decomposition. We would like to emphasize that asymptotic comparison of costs to other approaches is fairly complex. We discuss some of the issues below for the case of computing with the integers modulo 2^k. In this instantiation, the degree of the extension affects the complexity of the Prover, CRS size and the Verifier's complexity. This is in the same way as in pairing-based SNARKs, where the typical matching finite field has to be big enough (and meet other security requirements which rule out e.g. characteristic 2 fields) for soundness. Usually, a 254-bit prime field is chosen.

- Our *soundness amplification* techniques (Sec 8.3) describe how to make the extension degree *logarithmic in the QRP size*, rather than logarithmic in the soundness error, for the particular case of the integers modulo 2^k. The strategy easily generalizes to other rings. This is in contrast to pairing-based SNARKs where no similar strategies are known.
- When using a QAP to emulate ring arithmetic, addition gates are *no longer for free*; in contrast to QRPs with free ring additions. This is due to the fact that, in the QAP-based approach, a modular reduction might be necessary after adding two numbers. Hence, the blowup in the QAP degree of the naïve baseline needs to take addition gates into consideration as well.
- Besides having to take addition gates into account too, the cost of bit decomposition in a QAP over a field $\mathbb{F}$ is not exactly k gates, but one of the following:
  - If there is no enforced upper bound on the value that has to be decomposed (which always happens if inputs are not known and proven to be upper bounded), the cost of bit decomposition is $\log_2(\mathbb{F}) + 1$ multiplication gates. Since the typical bit size of $\mathbb{F}$ is 254 bits for secure and efficient pairings, this means approximately 255 gates per bit decomposition.
  - If the values are provably bounded, the above can be reduced to logarithmic in the maximum value attainable in the wire, e.g. $2k$ for the result of multiplying two k-bit numbers or $k + 1$

for the result of adding them. In order to provably bound values in the circuit, when some inputs are provided in ZK, this would require enforcing the bound by providing them bit-by-bit and ensuring that they are indeed bits (i.e. computing $x(1-x)$ for every alleged bit and checking it equals zero, plus reconstructing the bits into a single value: $k+1$ gates for a k-bit value). Hence, there is an additive overhead depending on the number of ZK-inputs if opting for this approach.

– On the plus side for QAPs, modular reduction is not needed after every operation as long as the value on *every* wire can be upper-bounded. Nevertheless, optimizing the compilation of a QAP so as to reduce the cost of modular reductions is a non-trivial problem for which practitioners have turned to heuristics [38] (see discussion in Sec 1.1). The Rinocchio approach, on the other hand, requires only *black-box* access to the underlying ring operations and removes the need for any compilation step.

These issues are specific to the instantiation of computing with the integers modulo 2^k when $k < \mathbb{F}/2$, i.e. $k < 122$ for the typical field choice. For larger values of k, there are additional problems for QAP-based SNARKs over the typical field choices. For instance, the integers modulo a composite N, there is already an exceptional set of size as big as the smallest prime factor of N, which reduces the impact of this efficiency concern. Furthermore, if the prover and/or the verifier know the factorization of N, they can benefit from using a CRT representation of data during the execution of their respective Prove and Verify algorithms, which is more efficient in practice and incompatible with having to emulate the arithmetic of $\mathbb{Z}_N$ within a field.

For Rinocchio's application to privacy-preserving verifiable computation, we refer the reader to the detailed analysis in Section 7.4.

While there are lookup techniques [11, 29] that can be used to simulate non-arithmetic operations like bit-decomposition, it is not clear how to use them directly in the QAP/QRP setting (without a Random Oracle assumption). Hence, we do not find these approaches to be comparable in terms of assumptions. We view our contributions as a first step towards constructing practical SNARKs for ring arithmetic, setting the stage for further work and improvements. The main advantage of our work is its generality and the ease to instantiate with different commutative rings, due to its black-box nature on the choice of that ring. We leave implementation and experimental studies to compare concrete gains of Rinocchio in our proposed (or different) applications as interesting future work.

References

1. Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via galois rings. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 471–501. Springer, Heidelberg, December 2019.
2. Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673. Springer, Heidelberg, August 2018.
3. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.
4. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 781–796. USENIX Association, August 2014.
5. Fabrice Benhamouda, Javier Herranz, Marc Joye, and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. *Journal of Cryptology*, 30(2):519–549, April 2017.

6. Anurag Bishnoi, Pete L Clark, Aditya Potukuchi, and John R Schmitt. On zeros of a polynomial in a finite grid. *Combinatorics, Probability and Computing*, 27(3):310–333, 2018.
7. 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.
8. Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. LNCS, pages 528–558. Springer, Heidelberg, 2021.
9. 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 III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, April / May 2017.
10. Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255. Springer, Heidelberg, April / May 2018.
11. Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 595–626. Springer, Heidelberg, December 2018.
12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.
13. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.
14. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.
15. Benjamin Braun, Ariel J Feldman, Zuocheng Ren, Srinath Setty, Andrew J Blumberg, and Michael Walfish. Verifying computations with state. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 341–357, 2013.
16. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.
17. David G Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.
18. Dario Catalano, Mario Di Raimondo, Dario Fiore, and Irene Giacomelli. Monza: Fast maliciously secure two party computation on $\mathbb{Z}_{2^k}$. Cryptology ePrint Archive, Report 2019/211, 2019. https://eprint.iacr.org/2019/211.
19. Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. Verifiable computing for approximate computation. Cryptology ePrint Archive, Report 2019/762, 2019. https://eprint.iacr.org/2019/762.
20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.
21. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.
22. Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 483–501. Springer, Heidelberg, August 2010.
23. Anamaria Costache, Kim Laine, and Rachel Player. Evaluating the effectiveness of heuristic worst-case noise analysis in FHE. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 546–565. Springer, Heidelberg, September 2020.
24. Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE, 2015.
25. Anders P. K. Dalskov, Eysa Lee, and Eduardo Soria-Vazquez. Circuit amortization friendly encodingsand their application to statistically secure multiparty computation. In *ASIACRYPT 2020, Part III*, LNCS, pages 213–243. Springer, Heidelberg, December 2020.
26. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

27. Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 844–855. ACM Press, November 2014.
28. Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Heidelberg, May 2020.
29. Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315, 2020. https://ia.cr/2020/315.
30. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.
31. 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.
32. Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 556–573. ACM Press, October 2018.
33. 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.
34. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.
35. Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. pages 212–234. ACM Press, 2021.
36. Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. Cryptology ePrint Archive, Report 2021/977, 2021. https://eprint.iacr.org/2021/977.
37. Ahmed E. Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, Mahmoud F. Sayed, Elaine Shi, and Nikos Triandopoulos. TRUESET: Faster verifiable set computations. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 765–780. USENIX Association, August 2014.
38. Ahmed E. Kosba, Charalampos Papamanthou, and Elaine Shi. xJsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy*, pages 944–961. IEEE Computer Society Press, May 2018.
39. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.
40. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013.
41. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.
42. 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.
43. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.
44. 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.
45. Zhe-Xian Wan. *Lectures on finite fields and Galois rings.* World Scientific Publishing Company, 2003.

A Verifiable Computation

Verifiable computation [?, 30] addresses the setting where a computationally limited client wishes to outsource the computation of a function to an untrusted, but computationally powerful worker. The goal is to enable to client to outsource the computation and be able to verify the correctness of the result such that this verification is less work than the evaluation of the function itself.

Definition 7 (Verifiable Computation). *A verifiable computation scheme is a tuple of polynomial time algorithms* $(\mathsf{KGen}, \mathsf{ProbGen}, \mathsf{Compute}, \mathsf{Ver})$ *defined as follows.*

- $(\mathsf{SK}, \mathsf{PK}_F) \leftarrow \mathsf{KGen}(1^\kappa, F)$*: A randomized key generation algorithm takes a function* F *as input and outputs a secret key* SK *and a public key* PK_F*.*
- $([x], \mathsf{VK}_x) \leftarrow \mathsf{ProbGen}_{\mathsf{PK}_F}(x)$ *A randomized problem generation algorithm takes the public key* PK_F*, an input* x*, and outputs an encoding of* x*, together with a private verification key* VK_x*.*
- $[y] \leftarrow \mathsf{Compute}_{\mathsf{PK}_F}([x])$ *A deterministic worker computation algorithm takes the public key* PK_F *and an encoded input* $[x]$ *to compute a value* $[y]$*.*
- $y \leftarrow \mathsf{Ver}_{\mathsf{SK}}(\mathsf{VK}_x, [y])$ *A verification algorithm uses the verification key* VK_x*, the worker's output* $[y]$*, and outputs* $y \in \{0,1\}^* \cup \perp$*, where* y *is the output of the computation and* $\perp$ *indicates that the client rejects the worker's output.*

  A verifiable computation scheme satisfies correctness, efficiency and security properties.

- *Correctness. Correctness guarantees that if the worker is honest, the verification test will pass. That is, for all* F*, and for all* x *in the domain of* F*,*

$$\Pr\left(y = F(x) \ : \ \begin{array}{c} (\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{KGen}(1^\kappa, F) \\ ([y], \mathsf{VK}_x) \leftarrow \mathsf{ProbGen}_{\mathsf{PK}}([x]) \\ [y] \leftarrow \mathsf{Compute}_{\mathsf{PK}}([x]) \\ y \leftarrow \mathsf{Ver}_{\mathsf{SK}}(\mathsf{VK}_x, [y]) \end{array} \right) = 1$$

- *Efficiency. The efficiency requirement states that the complexity of the outsourcing algorithm* $\mathsf{ProbGen}$*, and verification algorithm* Ver *together is less than the computation required to evaluate* F*. A VC must satisfy the property that for any* x *and any* $[y]$*, the time required for* $\mathsf{ProbGen}(x)$ *plus the time required for* $\mathsf{Ver}(\mathsf{VK}_x, [y])$ *is* $o(T)$*, where* T *is the time required to compute* $F(x)$*.*
- *Security. A VC scheme is secure if a malicious worker cannot make the verification algorithm accept an incorrect answer. That is, a scheme is secure if the advantage of any PPT adversary* $\mathcal{A}$ *in the game* $\mathsf{Expt}_{\mathcal{A}}^{Ver}$ *defined as* $\Pr\left(\mathsf{Expt}_{\mathcal{A}}^{Ver}[VC, F, \kappa] = 1\right)$ *is negligible.*

A.1 Context-Hiding

An additional property that can be defined for a VC scheme is called context-hiding. This captures the setting where one wants to hide information on the input x even from the verifier. Such a property would turn useful in scenarios where the data encoder and the verifier are different entities. Informally, this property says that output encodings $[y]$, as well as the input verification tokens verification key VK_x do not reveal any information on the input x. Notably this should hold even against the holders of the secret key SK. We formalize this definition in zero-knowledge style, requiring the existence of a simulator algorithm that, without knowing the input, should generate $(\mathsf{VK}_x, [y])$ that look like the real ones. More formally:

```
procedure GAME Expt_A^Ver(VC, F, κ)
    (SK, PK) ← KGen(1^κ, F)
    for i = 1, ..., ℓ = poly(κ) do
        x_i = A(PK, x_1, [x_1], ..., x_{i-1}, [x_{i-1}])
        ([x_i], VK_{x_i}) ← ProbGen_PK(x_i)
    end for
    (i, [y]) = A(PK, x_1, [x_1], ..., x_ℓ, [x_ℓ])
    y ← Ver_SK(VK_{x_i}, [y])
return ((y ≠ ⊥) ∧ (y ≠ F(x_i)))
end procedure
```

Definition 8 (Context-Hiding). *A VC scheme is context-hiding for a function F if there exist simulator algorithms S_1, S_2 such that:*

- *the keys* $(\mathsf{SK}, \mathsf{PK})$ *and* $(\mathsf{SK}', \mathsf{PK}')$ *are statistically indistinguishable, where* $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{KGen}(1^\kappa, F)$ *and* $(\mathsf{SK}', \mathsf{PK}', \mathsf{td}) \leftarrow S_1(1^\kappa, f)$*;*
- *for any input x, the following distributions are negligibly close*

$$(\mathsf{SK}', \mathsf{PK}', \mathsf{VK}_x, [x], [y]) \approx (\mathsf{SK}', \mathsf{PK}', \mathsf{VK}'_x, [x], [y]')$$

where $(\mathsf{SK}', \mathsf{PK}', \mathsf{td}) \leftarrow S_1(1^\kappa, f)$, $([x], \mathsf{VK}_x) \leftarrow \mathsf{ProbGen}_{\mathsf{PK}'}(x)$, $[y] \leftarrow \mathsf{Compute}_{\mathsf{PK}}([x])$*, and* $([y]', \mathsf{VK}'_x) \leftarrow S_2(\mathsf{td}, \mathsf{SK}', F(x))$.

B QRP: Abstraction, Composition and Circuit Representation

We begin by recalling the definition of a QRP, after which we follow with all the results about QRP composition and circuit representation.

Definition 9 (Quadratic Ring Programs (QRP)). *A Quadratic Ring Program (QRP) Q over a finite commutative ring R consists of three sets of polynomials, $\mathcal{V} = \{v_k(x) : k \in [0, m]\}, \mathcal{W} = \{w_k(x) : k \in [0, m]\}, \mathcal{Y} = \{y_k(x) : k \in [0, m]\}$ and a target polynomial $t(x)$, all in $R[x]$. Let C be an arithmetic circuit over R with n inputs and n' outputs. We say that Q is a QRP that computes C if the following holds:*

$a_1, \ldots, a_n, a_{m-n'+1}, \ldots a_m \in R^{n+n'}$ is a valid assignment to the input/output variables of C if and only if there exist $a_{n+1}, \ldots, a_{m-n'} \in R^{m-n-n'}$ such that:

$$t(x) \text{ divides } V(x) \cdot W(x) - Y(x),$$

where $V(x) = \big(v_0(x) + \sum_{k=1}^m a_k \cdot v_k(x)\big)$, $W(x) = \big(w_0(x) + \sum_{k=1}^m a_k \cdot w_k(x)\big)$ and $Y(x) = \big(y_0(x) + \sum_{k=1}^m a_k \cdot y_k(x)\big)$.

We define the size and degree of Q to be m and $\deg(t(x))$ respectively. Given polynomials $V(x), W(x), Y(x) \in R[x]$ defined as above and corresponding to a valid assignment of the input/output wires, we will call them a QRP solution.

Theorem 7. *Let C be a circuit over the ring R containing only one multiplication gate. If C has $m - 1$ inputs and a single output, there is a QRP of size m and degree 1 that computes C.*

Proof. Let $t(x) = x - r$, $r \in A$, where A is the exceptional set. Define $\rho_1(X_1, \ldots, X_{m-1}) = c_0 + \sum_{i=1}^{m-1} c_i \cdot X_i$ (resp. $\rho_2(X_1, \ldots, X_{m-1}) = d_0 + \sum_{i=1}^{m-1} d_i \cdot X_i$) to be the linear polynomial corresponding to the left (resp. right) input wire of the only multiplication gate in C. For $k \in \{0, \ldots, m-1\}$, let $v_k(x) = c_k$, $w_k(x) = d_k$, and $y_k(x) = 0$. Set $v_m(x) = w_m(x) = 0$ and $y_m(x) = 1$. Then we have that:

$$\begin{aligned}\big(v_0(x) + \sum_{k=1}^{m} a_k \cdot v_k(x)\big) \cdot \big(w_0(x) + \sum_{k=1}^{m} a_k \cdot w_k(x)\big) - \big(y_0(x) + \sum_{k=1}^{m} a_k \cdot y_k(x)\big) \\ = \rho_1(a_1, \ldots, a_{m-1}) \cdot \rho_2(a_1, \ldots, a_{m-1}) - a_m = p(x)\end{aligned}$$

We prove that this is a QRP for C. First assume that $a_1, \ldots, a_m \in R^m$ is a valid assignment to the input/output of C. Then $p(x) = 0$, which is trivially divisible by $t(x)$. Conversely, assume that the degree-zero polynomial $p(x)$ is divisible by the degree-one $t(x)$. As r is a root of $t(x)$, then so it has to be of $p(x)$, which implies $p(x) = 0$.

B.1 QRP as an Abstraction

Here, we highlight the generality of our notion of QRP and our construction by outlining how our notion recovers the QPP based construction of [37] for polynomial circuits.

In [37], Kosba et al. generalize the notion of Quadratic Arithmetic Programs over a field $\mathbb{F}$ to that of Quadratic Polynomial Programs (QPPs), which compute circuits whose wires carry values in the ring $\mathbb{F}[Z]$ of polynomials over the base field $\mathbb{F}$. These polynomial circuits, where the addition and multiplication operations are over $\mathbb{F}[Z]$, are introduced with the goal of representing (multi-)sets S of elements over $\mathbb{F}$. Our definition of QRPs and SNARK construction, being more general than those of [37], also covers their work and allows us to see it as an instantiation of Rinocchio for $R = \mathbb{F}[Z]$.

In [37], we have that $A = \mathbb{F} \subset R$, i.e. the degree-zero polynomials, and $A^* = \mathbb{F}^*$. The polynomials $v_k, w_k, z_k \in R[X] = \mathbb{F}[Z][X]$ can be made univariate in X by imposing that the coefficients of public linear combinations in the arithmetic circuit over R are all field elements, rather than elements of $R = \mathbb{F}[Z]$, which is also the approach taken in [37]. The secure encoding $E : R \to S$ consists in, given $c_k(z) \in R$, producing $\tilde{E}(c_k(t)) = g^{c_k(t)}$ for some fixed, secret $t \in \mathbb{F}$ and where $\tilde{E} : \mathbb{F} \to S$ is the same encoding used for QAPs over finite fields, e.g., in Pinocchio.

To cast the construction of [37] in our framework, consider the following encoding $\mathsf{E} : \mathbb{F} \to S$ to encode the QRP polynomials in the CRS: $\mathsf{E}(s) = \{\tilde{\mathsf{E}}(t^i \cdot s)\}_{i=1}^{n}$, where n is determined by the degree of the polynomials on the wires of the computation circuit. When $\tilde{\mathsf{E}}$ is exponentiation in a bilinear group, the encoding E satisfies additive homomorphism and the resulting SNARK achieves public verifiability. The central idea is that even though one has to encode "wire values", which in this case are polynomials and therefore, ring elements, the polynomials can be mapped to an evaluation instead, resulting in a field element which is subsequently encoded during the computation of the proof by the prover. The encoding E is designed to allow the prover to compute this encoding where the evaluation point is the secret t. At a high level, the encoding and the CRS crafted this way means that the secret point of evaluation of the wire polynomials is t, the secret point of evaluation for the QRP polynomials is s, and the prover can compute the correct encodings of the SNARK proof given the encodings in the CRS.

We sketch how the SNARK construction via QPP is a special case of our construction via QRP below.

QPP as an instantiation of QRP. The following definition is recovered by Definition 5, where $R = \mathbb{F}_p[Z], A = \mathbb{F}_p \subset R$, i.e. the degree-zero polynomials, and $A^* = \mathbb{F}_p^*$. The bivariate polynomial $p(x, z)$ accounts for the wire values themselves being polynomials.

Definition B1 (Quadratic Polynomial Program (QPP) [37]) *A QPP Q consists of three sets of polynomials, $\mathcal{V} = \{v_k(x)\}, \mathcal{W} = \{w_k(x)\}, \mathcal{Y} = \{y_k(x)\}$ and a target polynomial $t(x)$. Let C be a polynomial circuit. We say that Q computes C if the following holds:*

$a_1(z), \ldots, a_n(z), a_{m-n'+1}(z), \ldots a_m(z)$ is a valid assignment to the input/output variables of C if and only if there exist polynomials $a_{n+1}(z), \ldots, a_{m-n'}(z)$ such that $t(x)$ divides $p(x, z)$, where

$$p(x,z) = \big(\textstyle\sum_{k=1}^{m} a_k(z) \cdot v_k(x) \big) \cdot \big(\sum_{k=1}^{m} a_k(z) \cdot w_k(x) \big) - \big(\sum_{k=1}^{m} a_k(z) \cdot y_k(x) \big)$$

The degree of Q is said to be $\deg(t(x))$.

B.2 Composing QRPs

Our definition of QRPs and the construction of QRP above, allow for their composition exactly as in the field case [31]. In the following, we use the symbol $\circ$ both for circuit and QRP composition. Note that the composition theorem below holds for the particular QRP construction of Theorem 7, and we make no claims about other constructions that satisfy the QRP definition. In particular, we are careful to pick all the roots of the target polynomials to belong to the same exceptional set A.

For $i \in \{1, 2\}$, let Q_i be a QRP computing an arithmetic circuit f_i. Let $\mathcal{I}_i$ be the set of indices representing all wires in f_i and allow $\mathcal{I}_1 \cap \mathcal{I}_2$ to 'stitch' up to ℓ output wires of $\mathcal{I}_1$ to the inputs of $\mathcal{I}_2$. Denote such stitched circuit as $C = C_2 \circ C_1$. Express Q_i as $\mathcal{V}^{(i)} = \{v_k^{(i)}(x) : k \in \mathcal{I}_i\}, \mathcal{W}^{(i)} = \{w_k^{(i)}(x) : k \in \mathcal{I}_i\}, \mathcal{Y}^{(i)} = \{y_k^{(i)}(x) : k \in \mathcal{I}_i\}$ and target polynomial $t^{(i)}(x)$. Then, let $Q = Q_2 \circ Q_1$ consists of $\mathcal{V} = \{v_k(x) : k \in \mathcal{I}_1 \cup \mathcal{I}_2\}, \mathcal{W} = \{w_k(x) : k \in \mathcal{I}_1 \cup \mathcal{I}_2\}, \mathcal{Y} = \{y_k(x) : k \in \mathcal{I}_1 \cup \mathcal{I}_2\}$ and a target polynomial $t(x)$ which are constructed as follows.

First, define $t(x) = t^{(1)}(x) \cdot t^{(2)}(x)$. Second, for all indices $\tilde{k} \in \mathcal{I}_2 \setminus \mathcal{I}_1$, extend the definition of the wire polynomials in Q_1 as $v_{\tilde{k}}^{(1)}(x) = w_{\tilde{k}}^{(1)}(x) = y_{\tilde{k}}^{(1)}(x) = 0$. Proceed analogously for Q_2 and $\hat{k} \in \mathcal{I}_1 \setminus \mathcal{I}_2$. For all $k \in \mathcal{I}_1 \cup \mathcal{I}_2$ and $i \in \{1, 2\}$, we can now set $v_k(x) \equiv v_k^{(i)}(x) \mod t^{(i)}(x)$, $w_k(x) \equiv w_k^{(i)}(x) \mod t^{(i)}(x)$ and $y_k(x) \equiv y_k^{(i)}(x) \mod t^{(i)}(x)$. Such modular equivalences can be satisfied as long as the target polynomials have no common roots, as we show in the following lemma.

Lemma 6. *Let $t^{(1)}(x)$, $t^{(2)}(x) \in R[x]$ be two polynomials which have roots only on the same exceptional set $A \subset R$ and such that they have no common roots. Let $I_1 = (t^{(1)}(x))$, $I_2 = (t^{(2)}(x))$ and $I = I_1 \cdot I_2$. Then $R[x]/I \xrightarrow{\sim} R[x]/I_1 \times R[x]/I_2$.*

Proof. For $i \in \{1, 2\}$, let $t^{(i)}(x) = \prod_{j_i=1}^{d_i} (x - r_{j_i}^{(i)})$. Define ideals $I_{i,j_i} = (x - r_{j_i}^{(i)})$, where $1 \le j_i \le d_i$. Define $S = \{I_{i,j_i} : 1 \le i \le 2, 1 \le j_i \le d_i\}$. All the ideals in S are pairwise coprime. To see that, take any $K, \tilde{K} \in S$ and re-denote for simplicity $K = (x - k), \tilde{K} = (x - \tilde{k})$. As $k - x \in K$, we have that $k - \tilde{k} = k - x + x - \tilde{k} \in K + \tilde{K}$. Hence, as $k, \tilde{k}$ are two different elements from the same exceptional set $A \subset R$, we have that $k - \tilde{k}$ is a unit and so $K + \tilde{K} = R[x]$.

Given the above, we can apply the CRT (Theorem 2) three times and conclude that

$$R[x]/I_1 \times R[x]/I_2 \xrightarrow{\sim} (\prod_{j_1=1}^{d_1} R[x]/I_{1,j_1}) \times (\prod_{j_2=1}^{d_2} R[x]/I_{2,j_2}) \xrightarrow{\sim} R[x]/I.$$

We prove that the above construction for $Q = Q_2 \circ Q_1$ indeed computes $C = C_2 \circ C_1$.

Theorem 8. *Let C_1 and C_2 be two arithmetic circuits computed by QRPs Q_1 and Q_2. Assume the target polynomials of both QRPs have roots only on the same exceptional set $A \subset R$, but no common roots. Allow also some of the input variables of C_2 to include some ℓ output variables from C_1, but let no other kind of overlapping between the arithmetic circuits be possible. Denote by $C = C_2 \circ C_1$ the circuit obtained by stitching C_1 and C_2 together at those ℓ wires.*

There exists a QRP Q with size $|Q| = |Q_1| + |Q_2| - \ell$ and $\deg(Q) = \deg(Q_1) + \deg(Q_2)$ that computes C. Q's target polynomial is the product of the target polynomials for Q_1 and Q_2.

Proof. Let $\mathcal{I}_{i/o}, \mathcal{I}_{1,i/o}, \mathcal{I}_{2,i/o}$ be the indices of the input/output wires of C, C_1 and C_2, respectively. Suppose $\boldsymbol{a}_{i/o} = \{a_k \in \mathcal{I}_{i/o}\}$ is a valid input/output assignment for C. By definition, such input/output assignment can be extended to a valid assignment to all wires of C and hence in particular we can extend $\boldsymbol{a}_{i/o}$ to a valid assignment $\tilde{\boldsymbol{a}} = \{a_k \in \mathcal{I}_{1,i/o} \cup \mathcal{I}_{2,i/o}\}$. Since Q_1 is a QRP, there exist coefficients $\boldsymbol{b} = \{b_k : k \in \mathcal{I}_1\}$ which are consistent with the valid assignment to $\mathcal{I}_{1,i/o}$ and such that the polynomial

$$\begin{aligned} p^{(1)}(x) = {} & \big(v_0^{(1)}(x) + \sum_{k \in \mathcal{I}_1} b_k \cdot v_k^{(1)}(x)\big) \cdot \big(w_0^{(1)}(x) + \sum_{k \in \mathcal{I}_1} b_k \cdot w_k^{(1)}(x)\big) \\ & - \big(y_0^{(1)}(x) + \sum_{k \in \mathcal{I}_1} b_k \cdot y_k^{(1)}(x)\big) \end{aligned}$$

is a multiple of $t^{(1)}(x)$. The same reasoning can be applied to Q_2, for a polynomial $p^{(2)}(x)$ defined from coefficients $\boldsymbol{c} = \{c_k : k \in \mathcal{I}_2\}$ which must exist by the fact that Q_2 is a QRP. By construction, $\boldsymbol{b}$ and $\boldsymbol{c}$ must be consistent for the indices in $\mathcal{I}_1 \cap \mathcal{I}_2$, as those are contained in both $\mathcal{I}_{1,i/o}$ and $\mathcal{I}_{2,i/o}$, which were fixed by the extended assignment $\tilde{\boldsymbol{a}}$. Therefore, we can define $\boldsymbol{a} = \{a_k \in \mathcal{I}_1 \cup \mathcal{I}_2\}$ as $a_k = b_k$ for all $b_k \in \mathcal{I}_1$ and $a_k = c_k$ for all $c_k \in \mathcal{I}_2$. Let

$$\begin{aligned} p(x) = {} & \big(v_0(x) + \sum_{k \in \mathcal{I}_1 \cup \mathcal{I}_2} a_k \cdot v_k(x)\big) \cdot \big(w_0(x) + \sum_{k \in \mathcal{I}_1 \cup \mathcal{I}_2} a_k \cdot w_k(x)\big) \\ & - \big(y_0(x) + \sum_{k \in \mathcal{I}_1 \cup \mathcal{I}_2} a_k \cdot y_k(x)\big) \end{aligned}$$

where $v_k(x), w_k(x)$ and $y_k(x)$ are defined from $v_k^{(i)}(x), w_k^{(i)}(x)$ and $y_k^{(i)}(x)$, $i \in \{1, 2\}$, as described above (note the hypothesis of Lemma 6 are satisfied). We show that $t(x)$ divides $p(x)$. Since $v_k(x) = v_k^{(1)}(x) \mod t^{(1)}(x)$, $w_k(x) \equiv w_k^{(1)}(x) \mod t^{(1)}(x)$ and $y_k(x) \equiv y_k^{(1)}(x) \mod t^{(1)}(x)$ for all k, and since $v_{\tilde{k}}(x) = w_{\tilde{k}}(x) = y_{\tilde{k}}(x) \equiv 0 \mod t^{(1)}(x)$ for all $\tilde{k} \in \mathcal{I}_2 \setminus \mathcal{I}_1$, we conclude that $t^{(1)}(x)$ divides $p(x)$. Applying analogous reasoning, we can deduce that $t^{(2)}(x)$ divides $p(x)$ and, thus, $t(x) = t^{(1)}(x) \cdot t^{(2)}(x)$ divides $p(x)$

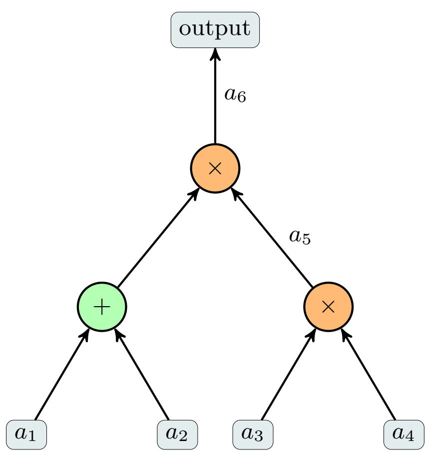
Conversely, let $p(x)$ be defined from the polynomial sets $\mathcal{V}, \mathcal{W}$ and $\mathcal{Y}$ as above and such that $t(x)$ divides $p(x)$. We show that any set of coefficients $\boldsymbol{a}$ enabling such divisibility contains a valid assignment $\boldsymbol{a}_{i/o} = \{a_k \in \mathcal{I}_{i/o}\}$ to the input/output wires of C. As $p(x) \equiv 0 \mod t(x)$, by Lemma 6, $p(x) \equiv 0 \mod t^{(i)}(x)$ for $i \in \{1, 2\}$. Since Q_1 and Q_2 are QRPs, it follows that $\boldsymbol{a}$ must then contain valid assignment to the input/output wires of C_1 and C_2. As $\mathcal{I}_{i/o} \subseteq \mathcal{I}_{1,i/o} \cup \mathcal{I}_{2,i/o}$, we have found a valid assignment $\boldsymbol{a}_{i/o}$ to the input/output wires of C.

Finally, we conclude by showing how to build a QRP for any arithmetic circuit by using the previous results from this section.

Theorem 9. *Let C be an arithmetic circuit with n inputs in (a subring of) R and $s < |A|$ multiplication gates, each with fan-in 2. If each output wire of C is the output of a multiplication gate, there is a QRP with size $n + s$ and degree s that computes C.*

Proof. We obtain this result by combining Theorem 7 and Theorem 8, one multiplication gate at a time. As long as $s < |A|$, we can ensure that the target polynomials of the QRPs for each multiplication gate do not have common roots, so that Theorem 8 can be invoked.

There is only one small task remaining. Let C be a circuit with $\tilde{n} \geq 1$ output wires which are not the output of multiplication gates. Our last result does not teach us how to deal with C, but we can build a modified circuit $\tilde{C}$ for which the hypothesis of Theorem 9 are satisfied. As in [31], $\tilde{C}$ has one additional 'dummy' input wire, which is required to be always assigned to the multiplicative identity 1. Furthermore, $\tilde{C}$ has a $\tilde{n}$ additional multiplication gates: For each of them, the left gate-input wire is the 'dummy' circuit-input wire and the right gate-input wire is one of the circuit-output wires which did not satisfy the hypothesis of Theorem 9. It follows that the QRP of size $n + s + \tilde{n} + 1$ and degree $s + \tilde{n}$ that computes $\tilde{C}$ also computes the original C.



Roots	Polynomials in QRP $(\mathcal{V}, \mathcal{W}, \mathcal{Y}, t(x))$		
Gates	Left inputs	Right inputs	Outputs
r_5	$v_3(r_5) = 1$ $v_k(r_5) = 0,$ $k \neq 3$	$w_4(r_5) = 1$ $w_k(r_5) = 0,$ $k \neq 4$	$y_5(r_5) = 1$ $y_k(r_5) = 0,$ $k \neq 5$
r_6	$v_1(r_6) = v_2(r_6) = 1$ $v_k(r_6) = 0,$ $k \neq 1, 2$	$w_5(r_6) = 1$ $w_k(r_6) = 0,$ $k \neq 5$	$y_6(r_6) = 1$ $y_k(r_6) = 0,$ $k \neq 6$

Fig. 4. Arithmetic circuit and equivalent QRP. The polynomials $\mathcal{V} = \{v_k(x) : k \in [6]\}, \mathcal{W} = \{w_k(x) : k \in [6]\}$, $\mathcal{Y} = \{y_k(x) : k \in [6]\}$ and the target polynomial $t(x) = (x - r_5)(x - r_6)$ are defined in terms of their evaluations at two random points belonging to the same exceptional set ($r_5, r_6 \in A$), one for each multiplicative gate.

Given a circuit C, we can construct a QRP for C using the composition theorem above. We can also construct a QRP directly for the given circuit without relying on composition. Let C be a circuit whose gates have fan-in two and fan-out one. To build a QRP, we will make use of a exceptional set A as follows. In order to define the target polynomial, we will pick elements $r_g \in A$ for each multiplication gate $g \in C$ and define $t(x) = \prod_{g \in C}(x - r_g)$. We define the polynomials $v_k(x), w_k(x)$ and $y_k(x)$ by interpolating over those same points in the same way one proceeds in the QAP case [42]. As an example for this procedure, see Figure 4.

B.3 Some useful QRPs

While the QRP construction described in Section 3 would allow us to easily describe arithmetic circuits over e.g. $\mathbb{Z}_{2^k}$ or the rings $\mathcal{R}_q$ used for homomorphic encryption, in practical scenarios one is also interested in performing bit-wise operations such as comparisons, for which we provide a bit decomposition gate.

Bit Decomposition Gate We show how to build a QRP which, given an input $a \in R$, gives as an output wires holding values $a_i \in \{0,1\}$ which correspond to the 'binary representation' of a. Our following description is specialized for $R = GR(2^k, d)$, but it can be easily adapted to other rings such as those employed in Section 7.1.

We provide two different versions of this gate. For the first one, nothing is known about a, whereas in the second case, better efficiency is achieved by assuming that $a \in \mathbb{Z}_{2^k}$. When interested in computation over $\mathbb{Z}_{2^k}$ only, the former version of the gate where potentially $a \notin \mathbb{Z}_{2^k}$ is necessary only if the prover is providing some inputs to the QRP in a zero-knowledge way. Nevertheless, once the inputs from the prover have been asserted to be elements of $\mathbb{Z}_{2^k}$, one can use the more efficient $\mathbb{Z}_{2^k}$-splitter gate during the rest of the circuit. The provers inputs can be tested to be from $\mathbb{Z}_{2^k}$ either by inspection when those are provided in the clear, or when they are provided in ZK, by e.g. applying the general R-splitter gate to them and outputting to the verifier all the wires that should be always equal to zero in a 'binary representation' of an element in $\mathbb{Z}_{2^k} \subset R$. Let $A \subset R$ be the exceptional set.

1. $\mathbb{Z}_{2^k}$-splitter gate: This mini-QRP has one input wire, holding $a \in \mathbb{Z}_{2^k}$, and k output wires holding $a_1, \ldots, a_k \in \{0,1\}$ such that $a = \sum_{i=1}^{k} 2^{i-1} a_i$. Label the input wires as $1, \ldots, k$ and the output wire as $k+1$. Let $t(x) = (x-r)\prod_{i=1}^{k}(x - r_i)$, where $r, r_1, \ldots, r_k \in A$ are pairwise different. In an approach similiar to Pinocchio [42], we set:

$$\begin{aligned} v_0(r) = 0, v_i(r) = 2^{i-1}, \text{ for } 1 \le i \le k, v_{k+1}(r) = 0, \\ w_0(r) = 1, w_i(r) = 0, \text{ for } 1 \le i \le k, w_{k+1}(r) = 0, \\ y_0(r) = 0, y_i(r) = 0, \text{ for } 1 \le i \le k, y_{k+1}(r) = 1 \end{aligned}$$

   For $1 \le j \le k$:

$$\begin{aligned} v_j(r_j) = 1, v_i(r_j) = 0 \text{ for all } i \neq j, \\ w_0(r_j) = 1, w_j(r_j) = -1, w_i(r_j) = 0 \text{ for all } i \neq 0, j, \\ y_i(r_j) = 0 \text{ for all } i \end{aligned}$$

   If $(v_0(x) + \sum a_k v_k(x)) \cdot (w_0(x) + \sum a_k w_k(x)) - (y_0(x) + \sum a_k y_k(x))$ is divisible by $t(x)$, then it must be 0 at r, and therefore, by the first set of equations, this gives, $a = \sum_{i=1}^{k} 2^{i-1} a_i$. The second set of equations guarantee that each r_j is a root, which implies, $a_j(1 - a_j) = 0$. Since all the zero divisors of R belong to the maximal ideal (2), it follows that if a_j is a zero divisor then $a_j \pm 1$ is not, and thence the only solutions for the previous equation are $a_j \in \{0,1\}$. Together, these give the guarantee that all a_i are bits, and are the binary decomposition of a.
2. R-splitter gate: This works essentially as the previous version of the splitter gate repeated δ times in parallel, once for every component of R seen as a free-module of rank δ over $\mathbb{Z}_{2^k}$.

C More on the Security of the Encoding Schemes over Rings

We start by presenting how the attack on q-PDH from [35] extends to the Generalized q-PDH assumption (Assumption 1). This is the reason behind allowing the adversary to have a $2q/|A^*|$ advantage by definition.

Lemma 7. *Let* Encode *be an ℓ-linearly homomorphic encoding scheme over a finite commutative ring R. Let $\ell \geq 2q-1$. There exists an adversary running in time* $\mathsf{poly}(q, \log|R|)$ *which wins the Generalized q-PDH assumption with advantage $2q/|A^*|$.*

Proof. Let $\mathcal{A}$ choose $2q$ random points $z_1, \ldots, z_{2q} \in A^*$. Interpolate the polynomial that has those random points as roots, which we denote by $f(X) = \sum_{i=0}^{2q} \alpha_i x^i$. It holds that $\forall i \in [2q]$, $\alpha_{q+1} z_i^{q+1} = -(\sum_{j=0}^{q} \alpha_j z_i^j) - (\sum_{k=q+1}^{2q} \alpha_k z_i^k)$. Given a generalized q-PDH challenge, $\mathcal{A}$ and since the encoding scheme is ℓ-linearly homomorphic, $\mathcal{A}$ can compute an encoding of the right hand side of the previous equation. Hence, if the secret s sampled for the q-PDH challenges happens to be one among $z_1, \ldots, z_{2q}$, and since the value α_{q+1} is known by $\mathcal{A}$, they win the game. ∎

Consider an encryption scheme which satisfies the properties required for an encoding scheme from Definition 6. If the encryption scheme can be assumed to be linear-only extractable, which is the assumption in [7, 9, 10], then it automatically is a *secure* encoding, i.e. it satisfies both the Generalized q-PDH and the Generalized Augmented q-PKE assumptions. We recall the linear-only extractable definition from [7], which we adapt to the broader context of (non-field) commutative rings with identity.

Definition 10 (Linear-only extractable). *An encoding scheme* $\mathsf{Encode} = (\mathsf{Gen}, \mathsf{E})$ *over R is linear-only extractable if for all probabilistic polynomial time algorithms $\mathcal{A}$, there exists a probabilistic polynomial time extractor $\chi_{\mathcal{A}}$ such that the following probability is negligible in the security parameter.*

$$\Pr\left(c \neq a_0 + \sum_{i=1}^{n} a_i x_i \ : \ \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa), \\ x_1, \ldots, x_n \xleftarrow{R} R, \\ \sigma = (\mathsf{pk}, \mathsf{E}(x_1), \ldots, \mathsf{E}(x_n)), \\ (\mathsf{E}(c); a_0, \ldots, a_n) \leftarrow (\mathcal{A}||\chi_{\mathcal{A}})(\sigma) \end{array} \right).$$

Lemma 8. *If an encoding scheme* $\mathsf{Encode} = (\mathsf{Gen}, \mathsf{E})$ *is IND-CPA secure and linear-only extractable, then it is an encoding scheme that satisfies Generalized Augmented q-PKE (Assumption 2).*

Proof. Let $\sigma = (\mathsf{pk}, \mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^q))$. We will show that Encode satisfies q-PKE, meaning we will show that for any adversary $\mathcal{A}$ able to produce $c, \hat{c}$ such that $\alpha c - \hat{c} = 0$, there exists an extractor $\chi_{\mathcal{A}}$ which outputs coefficients a_i satisfying $c = \sum_{i=0}^{q} a_i s^i$ with non negligible probability.

We define two adversaries $\mathcal{B}_c$ and $\mathcal{B}_{\hat{c}}$ that, upon receiving as input σ, run exactly the same code as $\mathcal{A}$ and output, respectively, c and $\hat{c}$. By our linear-only extractable assumption on E, there exist an extractor χ_c (resp. $\chi_{\hat{c}}$) for $\mathcal{B}_c$ (resp. $\mathcal{B}_{\hat{c}}$) which outputs $a_0, \ldots, a_q, b_0, \ldots, b_q$ (resp. $a'_0, \ldots, a'_q, b'_0, \ldots, b'_q$) such that

$$c = \sum_{i=0}^{q} a_i s^i + \sum_{i=0}^{q} b_i \alpha s^i, \quad \hat{c} = \sum_{i=0}^{q} a'_i s^i + \sum_{i=0}^{q} b'_i \alpha s^i$$

with non negligible probability.

Knowing that $\alpha c - \hat{c} = 0$ implies either that the polynomial

$$P(X,Y) = X^2 \sum_{i=0}^{q} b_i Y^i + X \sum_{i=0}^{q} (a_i - b'_i) Y^i - \sum_{i=0}^{q} a'_i Y^i$$

is the zero polynomial, or that (α, s) are roots of $P(X,Y)$. We rule out the second case by the IND-CPA security of the encoding scheme and the generalized Schwartz-Zippel lemma. Hence, $P(X,Y) = 0$, which gives us that for every $i \in [q]$, $b_i = a'_i = 0$ and $a_i = b'_i$. Therefore, we have defined an extractor $\chi_{\mathcal{A}}$ for the Generalized Augmented q-PKE assumption, which outputs the coefficients a_i obtained from χ_c.

Lemma 9. *If an encoding scheme* $\mathsf{Encode} = (\mathsf{Gen}, \mathsf{E})$ *is IND-CPA secure and linear-only extractable, then it is an encoding scheme that satisfies the Generalized q-PDH assumption (Assumption 1).*

Proof. Consider an adversary $\mathcal{A}$ that breaks q-PDH of the scheme Encode. We construct an adversary $\mathcal{B}$ that breaks IND-CPA. Consider the adversary $\mathcal{B}$ playing left-or-right oracle game where the adversary gets access to an encryption oracle that receives a pair of chosen messages always returns a ciphertext encrypting either the left or the right message. The adversary wins if it guesses the left-or-right bit.

$\mathcal{B}$ samples s_0, s_1 uniformly from an exceptional set $A^* \subset R^*$. $\mathcal{B}$ gets access to the left-or-right encryption oracle, makes queries on pairs (s_0^k, s_1^k) for $k \in \{0, \ldots, q, q+2, \ldots, 2q\}$, and receives $\{\mathsf{E}(s_b^i)\}_{i=0}^{2q, i \neq q+1}$ for challenge bit b. $\mathcal{B}$ now runs the q-PDH adversary $\mathcal{A}$ on $\{\mathsf{E}(s_b^i)\}$. $\mathcal{A}$ returns $y \in \{\mathsf{E}(s_b^{q+1})\}$. $\mathcal{B}$ now invokes the extractor that exists since Encode satisfies linear-only extractability (c.f. Definition 10). $\chi_{\mathcal{A}}$, given the same input as $\mathcal{A}$ and its internal randomness, returns $a_0, \cdots, a_q, a_{q+2}, a_{2q}$ such that $a_0 + \sum_{i=1}^{2q, i \neq q+1} a_i s_b^i = s_b^{q+1}$. Since $\mathcal{B}$ knows s_0, s_1, it checks whether $a_0 + \sum_{i=1}^{2q, i \neq q+1} a_i s_0^i = s_0^{q+1}$ or $a_0 + \sum_{i=1}^{2q, i \neq q+1} a_i s_1^i = s_1^{q+1}$, and outputs the bit b^* for which this holds. Notice that the previous strategy will output a *single* possible value for b^* with high probability, which further matches the challenge bit b. This is because, for the random s_{1-b}, we have that $a_0 + \sum_{i=1}^{2q, i \neq q+1} a_i s_{1-b}^i = s_{1-b}^{q+1}$ will hold only with probability $q/|A^*|$, by the generalized Schwartz-Zippel lemma.

Informally, the linear-only extractability assumption captures the fact that an adversary can perform only affine operations over the encodings provided as input. It can be argued that the PDH asssumption is in some sense weaker than linear-only extractability since the former is implied by the latter. However, if for an encoding scheme like JL, the linear-only extractability property is broken, computing non-linear homomorphisms would be possible which would mean efficient fully homomorphic encryption which is not known using current techniques. In [18], the authors consider a seemingly related notion called enhanced CPA and show that an additively homomorphic encryption scheme over $\mathbb{Z}_{2^k}$ cannot satisfy enhanced CPA. We note that their attack relies on the fact that the adversary has access to an oracle that checks the validity of a ciphertext. In our use of a encoding scheme in constructing a SNARK, we are concerned only with one-time soundness and our setting does not provide access to such oracles to the adversary (see also the remark at the end of Section 2.1). In proving multi-theorem soundness of designated-verifier SNARK constructions, one needs to make a stronger assumption called the q power-knowledge of equality (q-PKEQ) assumption. The following q-PKEQ assumption is needed in the designated verifier setting where the adversary has access to a verification oracle (in the public verification setting, this is for free and the adversary has no additional advantage). This assumption is invoked to prove multi-statement soundness in the proof to test if two (potentially adversarially generated) encodings have the same value underneath without having the secret key.

Assumption 3 (Generalized q-PKEQ) *The generalized q power-knowledge of equality assumption holds for an encoding scheme* Encode *if for all non-uniform probabilistic polynomial time algorithm $\mathcal{A}$, there exists a non-uniform probabilistic polynomial time extractor $\chi_{\mathcal{A}}$ such that the following probability is negligible in the security parameter.*

$$\Pr\left(\begin{array}{c}(b=0 \wedge \hat{c} \in \{\mathsf{E}(c)\})\\ \vee \\ (b=1 \wedge \hat{c} \notin \{\mathsf{E}(c)\})\end{array} : \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa),\\ s \xleftarrow{R} A^*,\\ \sigma = (\mathsf{pk}, \mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q})),\\ (\mathsf{E}(c), \hat{c}; b) \leftarrow (\mathcal{A}||\chi_{\mathcal{A}})(\sigma)\end{array}\right).$$

D Proof of Theorem 3

In this section, we prove that the construction satisfies the properties of a SNARK as stated in Definition 1. We recall the result:

Theorem 10 (Theorem 3, restated). *Let R be commutative ring with identity with an exceptional subset A, and d be an upper bound on the degree of the QRP. Assuming that the generalized augmented d-PKE and the generalized q-PDH assumptions hold for the encoding scheme* Encode *over R (and A^*) for $q = 4d+4$, the protocol* Rinocchio *described in Section 5.1 is a SNARK as per Definition 1, with soundness error $1/|A^*|$.*

Completeness. Assuming the encoding scheme Encode satisfies (statistical) correctness, then it follows by inspection that the verification equations are satisfied by a correctly generated proof π. Therefore (statistical) completeness of the Rinocchio protocol follows by QRP completeness.

Soundness. Assume there exists an adversary $\mathcal{A}$ who returns the proof of a false statement. We use this adversary $\mathcal{A}$ in order to construct an adversary $\mathcal{B}$ who breaks the q-PDH assumption.

Setting up the CRS. Adversary $\mathcal{B}$ is given the description of the encoding scheme, and the challenge $\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q})$. $\mathcal{B}$ provides the crs to $\mathcal{A}$ by constructing it as follows. It samples $r'_v, r'_w, \alpha, \alpha_v, \alpha_w, \alpha_y$ at random from R^* and sets $r'_y = r'_v r'_w$. Let $r_v = r'_v s^{d+1}$, $r_w = r'_w s^{2(d+1)}$, and $r_y = r'_y s^{3(d+1)}$. The value β is chosen as follows. Sample a polynomial $\beta_{poly}(x) \in A^*[X]$ of degree at most $3d+3$ uniformly at random, subject to the constraint that $\beta_{poly}(x) \cdot (r'_v v_k(x) + r'_w x^{(d+1)} w_k(x) + r'_y x^{2(d+1)} y_k(x))$ has a zero coefficient for x^{3d+3} for all k. $\mathcal{B}$ sets $\beta = s^{q-(4d+3)}\beta_{poly}(s)$. Looking ahead in our proof, the polynomial $x^{q-(4d+3)} \cdot \beta_{poly}(x)$ will play the role of $a(x)$ in Lemma 4. $\mathcal{B}$ sets the CRS as follows:

$$\begin{aligned}\mathsf{crs} = \Big(&\{\mathsf{E}(s^i)\}_{i=0}^d, \{\mathsf{E}(r_v v_k(s))\}_{k \in I_{mid}}, \{\mathsf{E}(r_w w_k(s))\}_{k \in I_{mid}}, \{\mathsf{E}(r_y y_k(s))\}_{k \in I_{mid}},\\ &\{\mathsf{E}(\alpha_v r_v v_k(s))\}_{k \in I_{mid}}, \{\mathsf{E}(\alpha_w r_w w_k(s))\}_{k \in I_{mid}}, \{\mathsf{E}(\alpha_y r_y y_k(s))\}_{k \in I_{mid}},\\ &\{\mathsf{E}(\alpha s^i)\}_{i=0}^d, \{\mathsf{E}(\beta(r_v v_k(s) + r_w w_k(s) + r_y y_k(s))\}_{k \in I_{mid}}, \mathsf{pk}\Big)\end{aligned}$$

We now argue that $\mathcal{B}$ can construct the above crs using the terms provided in its challenge. Consider the term in the proof that involves β, which is the final proof term that the prover will have to compute using the CRS.

$$\begin{aligned}&\mathsf{E}(\beta(r_v v_{mid}(s) + r_w w_{mid}(s) + r_y y_{mid}(s))) \\ &= \mathsf{E}(\beta(r'_v s^{d+1} v_{mid}(s) + r'_w s^{2(d+1)} w_{mid}(s) + r'_y s^{3(d+1)} y_{mid}(s))).\end{aligned} \tag{13}$$

In this term, β is multiplied by a polynomial evaluated at s. Note that $\mathcal{B}$ generated β also as a polynomial evaluated at s. If we further rewrite (13) by expressing β in terms of s, we have

$$\begin{aligned}&\mathsf{E}(s^{q-3d-2} r'_v \beta_{poly}(s) v_{mid}(s) + s^{q-2d-1} r'_w \beta_{poly}(s) w_{mid}(s) + s^{q-d} r'_y \beta_{poly}(s) y_{mid}(s)) \\ &= \mathsf{E}(s^{q-3d-2} \beta_{poly}(s)(r'_v v_{mid}(s) + s^{d+1} r'_w w_{mid}(s) + s^{2d+2} r'_y y_{mid}(s))).\end{aligned} \tag{14}$$

Since $\beta_{poly}(x) \cdot (r'_v v_k(x) + r'_w x^{d+1} w_k(x) + r'_y x^{2d+2} y_k(x))$ has a zero coefficient in front of x^{3d+3}, the value underneath the encoding in (14) has a zero in front of s^{q+1}. The powers of s in the encoding go up to $(q-3d-2)+(3d+3)+(2d+2)+d = q+3d+3 \leq 2q$. The polynomials $v_k(x), w_k(x), y_k(x)$ are of degree d, and none of the other elements in the CRS contain s^{q+1} inside the encoding. Since we have $q \geq 4d+4$, all the elements in the CRS can be generated using terms in the challenge.

We need to make sure that a crs generated as above has a distribution which is indistinguishable to the one given in our protocol. Note that, as $\beta_{poly}(x)$ is a polynomial of degree at most $3d+3$ and $\beta = s^{q-(4d+3)}\beta_{poly}(s)$, we have that $\Pr[\beta = 0] \leq (3d+3)/|A^*|$ (Lemma 2). This is a bigger chance for $\beta = 0$ than in our protocol, but notice that $\mathcal{A}$ never sees β in the clear, but rather encodings of it. There are hence two cases: Either $\mathsf{E}(0)$ is computationally indistinguishable from any $\mathsf{E}(a)$ where $a \neq 0$, or it is not (as it happens in the exponentiation-based encodings of e.g. [31, 42]). In the former case, $\mathcal{A}$ will accept the crs. In the latter case, $\mathcal{B}$ checks whether $\beta = 0$ by distinguishing whether the last term of crs is $\mathsf{E}(0)$ and, if so, samples a new $\beta_{poly}(x)$ and repeats the process above until the last term is not $\mathsf{E}(0)$.

Extraction. With the CRS set this way, $\mathcal{B}$ can now obtain a purported proof from $\mathcal{A}$. Due to the indistinguishability of simulated CRS and real CRS, $\mathcal{A}$ aborting on input the tailored CRS is negligible. Let $\hat{\pi}$ be a purported proof returned by $\mathcal{A}$, which is parsed as follows:

$$\begin{aligned}\hat{\pi} = \big(\ &\mathsf{E}(r_v V_{mid}), \mathsf{E}(r_w W_{mid}), \mathsf{E}(r_y Y_{mid}), \mathsf{E}(H), \\ &\mathsf{E}(r_v \hat{V}_{mid}), \mathsf{E}(r_w \hat{W}_{mid}), \mathsf{E}(r_y \hat{Y}_{mid}), \mathsf{E}(\hat{H}), \mathsf{E}(L)\ \big)\end{aligned}$$

Since $\mathcal{B}$ knows that $r_v = r'_v s^{d+1}$, $r_w = r'_w s^{2(d+1)}$, and $r_y = r'_y s^{3(d+1)}$, it can reinterpret $\hat{\pi}$ as follows:

$$\begin{aligned}\big(\ &\mathsf{E}_{r'_v}(s^{d+1} V_{mid}), \mathsf{E}_{r'_w}(s^{2d+2} W_{mid}), \mathsf{E}_{r'_y}(s^{3d+3} Y_{mid}), \mathsf{E}(H), \\ &\mathsf{E}_{r'_v}(s^{d+1} \hat{V}_{mid}), \mathsf{E}_{r'_w}(s^{2d+2} \hat{W}_{mid}), \mathsf{E}_{r'_y}(s^{3d+3} \hat{Y}_{mid}), \mathsf{E}(\hat{H}), \mathsf{E}(L)\ \big)\end{aligned}$$

Notice that the proof elements are now being treated as if they belonged to four different encodings: $\mathsf{E}, \mathsf{E}_{r'_v}, \mathsf{E}_{r'_w}, \mathsf{E}_{r'_y}$, where the four latter are defined as $\mathsf{E}_a(b) = \mathsf{E}(a \cdot b)$. It is easy to see that, by the fact that $r'_v, r'_w, r'_y \in R^*$ and the assumption that E is a secure encoding, so are $\mathsf{E}_{r'_v}, \mathsf{E}_{r'_w}, \mathsf{E}_{r'_y}$. Since $\hat{\pi}$

passes verification (in particular Equation (5)), we can apply the following reasoning for E and any of the other three encodings. As $(\mathsf{E}(H), \mathsf{E}(\hat{H}))$ is of the form $(\mathsf{E}(H), \mathsf{E}(\alpha H))$, $\mathcal{B}$ can use the d-PKE extractor χ_A to extract a polynomial $H(x) = \sum_{i=0}^{d} h_i x^i$ of degree at most d such that $H = H(s)$. This is because the CRS given to $\mathcal{A}$ is of the form (σ, z), where:

$$\sigma = (\mathsf{pk}, \{\mathsf{E}(s^i)\}_{i=0}^{d}, \{\mathsf{E}(\alpha s^i)\}_{i=0}^{d}), \quad z = \mathsf{crs} \setminus \sigma$$

Note that the auxiliary information z is independent of α, as the relation between e.g. $\mathsf{E}_{r'_v}(s^{d+1}V_{mid})$ and $\mathsf{E}_{r'_v}(s^{d+1}\hat{V}_{mid})$ is an i.i.d. α_v. If we look at any of the three remaining encodings $\mathsf{E}_{r'_v}(\cdot)$, $\mathsf{E}_{r'_v}(\cdot)$ or $\mathsf{E}_{r'_v}(\cdot)$, we will next show that $\mathcal{B}$ can extract $V_{mid}(x)$ of degree at most d and such that $V_{mid} = V_{mid}(s)$ due to the $(2d+1)$-PKE assumption (resp. $W_{mid}(x)$ due to $(3d+2)$-PKE and $Y_{mid}(x)$ due to $(4d+3)$-PKE). Focusing on $V_{mid}(x)$, notice that $\mathcal{A}$ does not have a $(2d+1)$-PKE challenge, but the following (where the problem is with $\tilde{\sigma}_v$, not with z)

$$\tilde{\sigma}_v = (\mathsf{pk}, \{\mathsf{E}_{r'_v}(s^{d+1}v_k(s))\}_{k \in I_{mid}}, \{\mathsf{E}_{r'_v}(\alpha_v s^{d+1} v_k(s))\}_{k \in I_{mid}}), \quad z = \mathsf{crs} \setminus \tilde{\sigma}_v$$

which differs from the expected $\sigma_v = (\mathsf{pk}, \{\mathsf{E}_{r'_v}(s^i)\}_{i=0}^{2d+1}, \{\mathsf{E}_{r'_v}(\alpha_v s^i\}_{i=0}^{2d+1})$ in two ways: It is completely missing the powers $\{s^i\}_{i=0}^{d}$ and, for those between $d+1$ and $2d+1$, it instead has the evaluation at s of the polynomials $\{x^{d+1}v_k(x)\}_{k \in I_{mid}}$. Informally, since $\mathcal{B}$ can compute $\tilde{\sigma}_v$ from σ_v, we can extract. In more syntactic rigour, $\mathcal{B}$ can send σ_v to a $(2d+1)$-PKE adversary $\mathcal{A}_v$ who runs internally the SNARK prover $\mathcal{A}$ on $\tilde{\sigma}_v$, so as Equation (5) verifies, then, by the PKE assumption there exists an extractor χ_{A_v} which gets a polynomial $x^{d+1}V_{mid}(x) = \sum_{i=0}^{d} v_i x^{d+1+i}$ of degree at most $2d+1$ such that $V_{mid} = V_{mid}(s)$. Applying the same reasoning, we can conclude on the extraction of polynomials $W_{mid}(x), Y_{mid}(x)$ of degree at most d such that $W_{mid} = W_{mid}(s)$ and $Y_{mid} = Y_{mid}(s)$.

Reducing to Generalized q-PDH. Since the proof $\hat{\pi}$ verifies but the statement is false, we show that then one of the following must hold, where $V(x) = \sum_{k \in I_{io}} c_k v_k(x) + V_{mid}(x)$ and similarly $W(x)$ and $Y(x)$:

Case 1: $V(x) \cdot W(x) - Y(x) \neq H(x) \cdot t(x)$, but Equation (7) holds, therefore, $V(s) \cdot W(s) - Y(s) = H(s) \cdot t(s)$.

Case 2: The polynomial

$$U(x) = r'_v x^{d+1} V_{mid}(x) + r'_w x^{2(d+1)} W_{mid}(x) + r'_y x^{3(d+1)} Y_{mid}(x)$$

is not in the module S generated by the R-linear combinations of the polynomials $\{u_k(x) = r'_v x^{d+1} v_k(x) + r'_w x^{2(d+1)} w_k(x) + r'_y x^{3(d+1)} y_k(x)\}_{k \in I_{mid}}$.

We demonstrate that those are the only cases for a false $\hat{\pi}$ by proving that, if none of them holds, then $V(x), W(x)$ and $Y(x)$ are a QRP solution, which would then mean that $\hat{\pi}$ is a valid proof. So, towards contradiction, assume both that $U(x) \in S$ and $V(x) \cdot W(x) - Y(x) = H(x) \cdot t(x)$. Since $U(x) \in S$, it can be expressed as $U(x) = \sum_{k \in I_{mid}} c_k u_k(x)$, where $c_k \in R$. Thus,

$$U(x) = r'_v x^{d+1} v'(x) + r'_w x^{2(d+1)} w'(x) + r'_y x^{3(d+1)} y'(x),$$

where we define $v'(x) = \sum_{k \in I_{mid}} c_k v_k(x)$, $w'(x) = \sum_{k \in I_{mid}} c_k w_k(x)$ and $y'(x) = \sum_{k \in I_{mid}} c_k y_k(x)$. Note that $v'(x), w'(x), y'(x)$ have degree at most d, since they are in the spans of $\{v_k(x)\}_{k \in I_{mid}}, \{w_k(x)\}_{k \in I_{mid}}$

and $\{y_k(x)\}_{k\in I_{mid}}$ respectively. Since $V_{mid}(x), W_{mid}(x), Y_{mid}(x)$ are also polynomials of degree at most d, and since the R-submodules $\{x^{d+1+i} : i \in [0,d]\}$, $\{x^{2(d+1)+i} : i \in [0,d]\}$, and $\{x^{3(d+1)+i} : i \in [0,d]\}$ of $R[x]$ are disjoint (except at zero) we have that

$$\begin{aligned} U(x) &= r'_v x^{d+1} V_{mid}(x) + r'_w x^{2(d+1)} W_{mid}(x) + r'_y x^{3(d+1)} Y_{mid}(x) \\ &= r'_v x^{d+1} v'(x) + r'_w x^{2(d+1)} w'(x) + r'_y x^{3(d+1)} y'(x), \end{aligned}$$

we conclude that $V_{mid}(x) = v'(x)$, $W_{mid}(x) = w'(x)$ and $Y_{mid}(x) = y'(x)$. Therefore, $V(x) = \sum_{k\in I_{io}} c_k v_k(x) + V_{mid}(x) = \sum_{k\in I_{io}} c_k v_k(x) + \sum_{k\in I_{mid}} c_k v_k(x)$, $W(x) = \sum_{k\in I_{io}} c_k w_k(x) + W_{mid}(x) = \sum_{k\in I_{io}} c_k w_k(x) + \sum_{k\in I_{mid}} c_k w_k(x)$, and $Y(x) = \sum_{k\in I_{io}} c_k y_k(x) + Y_{mid}(x) = \sum_{k\in I_{io}} c_k y_k(x) + \sum_{k\in I_{mid}} c_k y_k(x)$. Finally, as we assumed that $V(x) \cdot W(x) - Y(x) = H(x) \cdot t(x)$, we have that $V(x), W(x), Y(x)$ can be written as the same linear combination $\{c_k\}_{k\in I_{io}\cup I_{mid}}$ of their respective sets, and that $t(x)$ divides $V(x) \cdot W(x) - Y(x)$. Therefore, $V(x), W(x), Y(x)$ are a QRP solution.

We now address the two cases corresponding to a false proof $\hat{\pi}$ and show that, in both Case 1 and Case 2, $\mathcal{B}$ can break the Generalized q-PDH (Assumption 1).

Case 1: $V(x) \cdot W(x) - Y(x) \neq H(x) \cdot t(x)$. The non-zero polynomial $\gamma(x) = V(x) \cdot W(x) - Y(x) - H(x) \cdot t(x)$ has degree $k \leq 2d$ and s as a root. Express $\gamma(x) = \gamma_k \cdot x^k + \hat{\gamma}(x)$, where $k \leq 2d$, $\gamma_k \neq 0$ and $\deg(\hat{\gamma}(x)) < k$. Since s is a root of $\gamma(x)$, it is also a root of $x^{q+1-k}\gamma(x)$. Hence, $\gamma_k \cdot s^{q+1} = -s^{q+1-k}\hat{\gamma}(s)$. $\mathcal{B}$ can compute $\mathsf{E}(\gamma_k \cdot s^{q+1})$ by computing $\mathsf{E}(-s^{q+1-k}\hat{\gamma}(s))$, which is a known linear combination of the $\{\mathsf{E}(s^i)\}_{i=0}^{q}$ values belonging to the q-PDH instance. This solves the q-PDH challenge.

Case 2: The polynomials $V_{mid}(x), W_{mid}(x), Y_{mid}(x)$ are not in the required spans. There does not exist $\{c_k\}_{k\in I_{mid}}$ such that $V_{mid}(x) = \sum_{k\in I_{mid}} c_k v_k(x)$, $W_{mid}(x) = \sum_{k\in I_{mid}} c_k w_k(x)$ and $Y_{mid}(x) = \sum_{k\in I_{mid}} c_k y_k(x)$. Then, the polynomial $U(x) = r'_v x^{d+1} V_{mid}(x) + r'_w x^{2(d+1)} W_{mid}(x) + r'_y x^{3(d+1)} Y_{mid}(x)$ is not in the module S generated by the R-linear combinations of the polynomials $\{u_k(x) = r'_v x^{d+1} v_k(x) + r'_w x^{2(d+1)} w_k(x) + r'_y x^{3(d+1)} y_k(x)\}$. Recall that $\mathcal{B}$ chose a polynomial $\beta_{poly}(x) \in A^*[X]$ of degree at most $3d+3$ subject to the constraint that all polynomials in $\{\beta_{poly}(x) \cdot (r'_v v_k(x) + r'_w x^{(d+1)} w_k(x) + r'_y x^{2(d+1)} y_k(x))\}$ have a zero coefficient for x^{3d+3}. Thus, by Lemma 4, the coefficient of x^{q+1} in the polynomial $\omega(x) = x^{q-(4d+3)} \cdot \beta_{poly}(x) \cdot U(x)$ is $a \in R \backslash \{0\}$ with probability $1 - 1/|A^*|$. Furthermore, $\mathcal{B}$ can compute all the coefficients of $\omega(x)$ on its own, so it can subtract from $\mathsf{E}(L) = \mathsf{E}(s^{q-(4d+3)} \beta_{poly}(s) \cdot (s^{d+1} V_{mid}(s) + s^{2(d+1)} W_{mid}(s) + s^{3(d+1)} Y_{mid}(s)))$ all the monomials corresponding to $\mathsf{E}(s^j)$ for $j \neq q+1$ and obtain $\mathsf{E}(a \cdot s^{q+1})$. Note that this is possible even when $\beta_{poly}(s) = 0$. By outputting $(a, \mathsf{E}(a \cdot s^{q+1}))$, $\mathcal{B}$ breaks the generalized q-PDH assumption.

E SNARKs for Computation over Encrypted Data (Cont'd)

E.1 Further details on Torus encoding

Multiplying encoded elements with elements from R: We next show explicitly how our TFHE-based encoding is R-linear homomorphic. $R = \mathbb{Z}_m[Y]/(f(Y))$ is a free module over $\mathbb{Z}_m$ of rank d, i.e. we can find a basis for R. Let ξ be a root of $f(Y)$, we have that $\{1, \xi, \ldots, \xi^{d-1}\}$ is one of such basis. The map $\phi : R \to (\mathbb{Z}_m)^d$, which sends $b = b_0 + \cdots + b_{d-1}\xi^{d-1}$ to $\phi(b) = (b_0, \ldots, b_{d-1})$ is an isomorphism of $\mathbb{Z}_m$-modules. We will make extensive use of this isomorphism going forward.

The encoding we use is the following:

$$\begin{aligned}\mathsf{E}_{\mathsf{pk}} &: R \to (\mathbb{T})^d \\ &a \mapsto \mathsf{E}_{\mathsf{pk}}(a) = (\mathsf{TFHE}(a_0), ..., \mathsf{TFHE}(a_{d-1}))\end{aligned}$$

For our QRPs, we wish to compute values of the form $E(a \cdot b)$, where $a, b \in R$, given $E(a)$ and b. The problem is that $E(a) \in (\mathbb{T})^d$, and the torus does not allow us to simply and directly compute $b \cdot E(a)$ as in previous occasions. Rather, we have to look at the R-module endomorphism $\cdot_b$ which is induced by multiplication of any element of R with b, and use this to manipulate the d individual values $\mathsf{TFHE}(a_0), ..., \mathsf{TFHE}(a_{d-1}) \in \mathbb{T}$.

In a more explicit and step-by-step fashion, $\cdot_b$ is an R-module endomorphism and hence a $\mathbb{Z}_m$-module homomorphism $\cdot_b : (\mathbb{Z}_m)^d \to (\mathbb{Z}_m)^d$. We can therefore represent this operation as follows:

$$\begin{aligned}\cdot_b &: (\mathbb{Z}_m)^d \to (\mathbb{Z}_m)^d \\ &a \mapsto M_b \cdot a\end{aligned}$$

where $M_b \in \mathcal{M}_{d\times d}(\mathbb{Z}_m)$. As a side note, in fact, M_b can be easily defined from the polynomial $f(Y)$ used to construct $R \simeq (\mathbb{Z}_m)^d$. Our goal can now be re-stated as computing $E(\cdot_b(a))$, given $E(a)$ and $b \in R$. We are almost done, as $\mathsf{TFHE}(x) + \mathsf{TFHE}(y) = \mathsf{TFHE}(x+y)$ and $\mathbb{T}$ allows for external multiplication with elements in $\mathbb{Z}$. In full formalism, let $N_b \in \mathcal{M}_{d\times d}(\mathbb{Z})$ such that $N_b \equiv M_b \mod n$. We only need to compute:

$$N_b \cdot E(a) = E(N_b \cdot a) = E(M_b \cdot a) = E(\cdot_b(a)) = E(a \cdot b)$$

E.2 Parameters for BGV and FV

Here, we provide some outputs of the Maple script (`https://github.com/rachelplayer/CLP19-code/blob/master/Comparison/comparison.mpl`) behind the work of Costache, Laine and Player [23]. These provide a more detailed view of the parameters for the BGV and FV schemes than the one provided in [23], which is necessary to understand both the soundness of our scheme and the efficiency impact compared with [28] (see Section 7.4).

Scheme	L	n	$\lvert p_1 \rvert$	$\lvert q \rvert$
BGV	2	2^{12}	23	109
FV	2	2^{11}	22	54
BGV	4	2^{13}	24	218
FV	4	2^{12}	23	109
BGV	6	2^{13}	25	218
FV	6	2^{13}	26	218
BGV	8	2^{14}	28	438
FV	8	2^{13}	24	218
BGV	10	2^{14}	25	438
FV	10	2^{14}	25	438
BGV	12	2^{14}	27	438
FV	12	2^{14}	29	438
BGV	14	2^{14}	28	438
FV	14	2^{14}	26	438
BGV	16	2^{15}	34	881
FV	16	2^{15}	34	881

Table 1. Parameters for BGV and FV with a plaintext space $\mathcal{R}_p$ where $p = 2^8$. L is the amount of levels, n the degree of the quotient polynomial, q the integer modulo of the ciphertext ring $\mathcal{R}_q$ and p_1 the smallest prime factor of q. With $|x|$, we denote the bit-length of x.

Scheme	L	n	$\lvert p_1 \rvert$	$\lvert q \rvert$
BGV	2	2^{12}	47	109
FV	2	2^{13}	48	218
BGV	4	2^{13}	48	218
FV	4	2^{14}	51	438
BGV	6	2^{14}	51	438
FV	6	2^{14}	51	438
BGV	8	2^{15}	51	881
FV	8	2^{14}	50	438
BGV	10	2^{15}	56	881
FV	10	2^{15}	56	881
BGV	12	2^{15}	59	881
FV	12	2^{15}	57	881
BGV	14	2^{15}	51	881
FV	14	2^{15}	50	881
FV	16	2^{15}	51	881

Table 2. Parameters for BGV and FV with a plaintext space $\mathcal{R}_p$ where $p = 2^{32}$. L is the amount of levels, n the degree of the quotient polynomial, q the integer modulo of the ciphertext ring $\mathcal{R}_q$ and p_1 the smallest prime factor of q. With $|x|$, we denote the bit-length of x.

Scheme	L	n	$\|p_1\|$	$\|q\|$
BGV	2	2^{13}	80	218
FV	2	2^{14}	81	438
BGV	4	2^{14}	81	438
FV	4	2^{14}	82	438
BGV	6	2^{15}	84	881
FV	6	2^{15}	84	881
BGV	8	2^{15}	86	881
FV	8	2^{15}	83	881
BGV	10	2^{15}	83	881

Table 3. Parameters for BGV and FV with a plaintext space $\mathcal{R}_p$ where $p = 2^{64}$. L is the amount of levels, n the degree of the quotient polynomial, q the integer modulo of the ciphertext ring $\mathcal{R}_q$ and p_1 the smallest prime factor of q. With $|x|$, we denote the bit-length of x.