zk-SNARKs from Codes with Rank Metrics

Xuan-Thanh Do, Dang-Truong Mac, Quoc-Huy Vu

zk-SNARKs from Codes with Rank Metrics

Xuan-Thanh Do[1], Dang-Truong Mac, and Quoc-Huy Vu[2]

[1] Institute of Cryptography Science and Technology, Vietnam
[2] Léonard de Vinci Pôle Universitaire, Research Center, Paris-La Défense, France

Abstract. Succinct non-interactive zero-knowledge arguments of knowledge (zk-SNARKs) are a type of non-interactive proof system enabling efficient privacy-preserving proofs of membership for NP languages. A great deal of works has studied candidate constructions that are secure against quantum attackers, which are based on either lattice assumptions, or post-quantum collision-resistant hash functions. In this paper, we propose a code-based zk-SNARK scheme, whose security is based on the rank support learning (RSL) problem, a variant of the random linear code decoding problem in the rank metric. Our construction follows the general framework of Gennaro *et al.* (CCS'18), which is based on square span programs (SSPs). Due to the fundamental differences between the hardness assumptions, our proof of security cannot apply the techniques from the lattice-based constructions, and indeed, it distinguishes itself by the use of techniques from coding theory. We also provide the scheme with a set of concrete parameters.

Keywords: Code-based Cryptography, Rank support learning problem, Square span programs, zk-SNARKs

1 Introduction

Zero-knowledge proof systems [GMR85], since its first appearance in 1985, have been become the cornerstone of cryptography. They are an essential component of many privacy-preserving cryptographic systems, including credentials and digital currencies [BG90, CL01, BCC+09, CKL+16, FHS19] as well as group signatures [Cv91, BBS04, DP06, BCC+16] and verifiable computation [GGP10, GGPR13, BCG+18]. In a zero-knowledge proof of knowledge system for an NP relation $\mathcal{R}$, a prover can convince a verifier that a statement is true without revealing anything more about the statement to the verifier. For practical applications, succinct non-interactive zero-knowledge arguments of knowledge (zk-SNARKs) [Kil92, Mic94] are more desirable: we additionally require that (i) the proof should consist of a single message from the prover to the verifier (non-interactivity); (ii) the length of the proof and the verification complexity is sublinear (ideally, polylogarithmic) in the size of the circuit computing $\mathcal{R}$ (succinctness); and (iii) the proof also guarantees that the prover knows the witness (argument of knowledge).

Constructions of succinct non-interactive zero-knowledge can be based on numerous different assumptions, of which one may name collision-resistant hash functions [BBHR19, CMS19], the discrete logarithm assumption [BBB+18], various pairing-based assumptions [Gro16], and lattice-based assumptions [GMNO18, ISW21]. On the other hand, the advancing threat of quantum computers has given tremendous stimulant to the cryptographic community to put more effort into cryptographic constructions that would plausibly withstand the power of quantum attacks. However, present post-quantum zk-SNARKs are only known from hash functions and lattice-based assumptions.

Our result. In this work, following the method of [GMNO18], we introduce the first (designated-verifier) zk-SNARK scheme in the rank metric context. Prior to this work, there has been no construction in the code-based cryptography realm, so the construction herein could be viewed as the first. Furthermore, being based on code-based assumptions with rank metrics, our scheme is plausibly considered to be secure under quantum attacks. We note that the work of Lipmaa [Lip13] makes use of error-correcting codes to improve the performance of span programs and does not concern with code-based assumptions.

Overview of our technique. Our starting point is the framework of Gennaro *et al.* [GMNO18]. Conceptually, based on the techniques of [DFGK14] and [GGPR13], the framework of [GMNO18] uses square span programs to characterize the complexity class NP, leading to a simpler and faster designated-verifier zk-SNARK. The main technical challenge in the framework of [GMNO18] is the growth of noise of

the lattice-based homomorphic operations. As mentioned there, this growth might leak information of the witness to the verifier, thus violate the zero-knowledge property. Fortunately, in the asset of lattice-based techniques, the so-called noise-smudging technique can be used to overcome this leakage problem. The idea is that after doing homomorphic addition, a noise with much larger weight is added to the computed one, thus, the final noise is dominated by that of the adding noise. (One might think of this technique as hiding "leaves" in "forest.") This technical challenge, resolved by smudging, also causes the setting of the common reference string to become involved, that is, the natures of encodings are not the same: some having small noise while others requiring much larger one. The reason underlying this setting is to guarantee success of the reduction.

The naive scheme obtained when one carries out the construction to the rank metric context is even worse since the noise grows linearly with respect to the number of homomorphic operations. And in order to be able to decrypt these ciphertexts, the length of the public code in used must be very large (and thus, the degree of extension field as well) causing parameters of the whole system to be out of concern. (We assume one uses the rank-quasi cyclic (RQC) encryption scheme to design the underlying encoding scheme.) Out of this situation, a natural question arises:

Can we design a SNARK in code-based cryptography (and even in lattice-based cryptography) without using the smudging technique?

We put forth effort to resolve this question in code-based cryptography by making use of the rank support learning problem. We now recall the main technical ingredients of [GMNO18]'s framework, which lie in the way the common reference string (CRS) is constructed. In particular, the CRS in the construction of Gennaro *et al.* consists of encoding elements together with the description of a square span program which computes the statement and public parameters of some additively homomorphic "noisy" encoding scheme. In [GMNO18], their encoding scheme is instantiated from lattice-based assumption. Let E denote this encoding scheme. By examining the form of the CRS, we make the observation that the encodings therein could be divided into three groups. The first group consists of encodings of powers of a hidden element, say, $\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^d)$, where s is kept secret. The second group consists of encodings of elements which are resulted from the first group by a common mask, *i.e.,* $\mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^d)$, where α plays the role of a mask. The third group consists of encodings of elements which are values of polynomials at s masked by a common element, *i.e.,* $\mathsf{E}\big(\beta t(s)\big), \mathsf{E}\big(\beta v_{\ell_u+1}(s)\big), \ldots, \mathsf{E}\big(\beta v_m(s)\big)$. The crucial point is that the error for each of these encodings has to be chosen carefully, so that addition of encodings computationally hides the witness. This is needed when showing the zero-knowledge property of the scheme, whose security proof is based on the smudging technique. Furthermore, when paying closer attention to the way a proof is generated, we observe that homomorphic evaluations (or rather additions) are always performed between elements of each group together with a set of coefficients in the prescribed finite field.

In rank metric code-based cryptography, it seems difficult to do so due to the aforementioned reasons. However, these two observations lead us to the idea of using one and the same vector space of noises for each group. More precisely, let V_1, V_2, V_3 be randomly chosen subspaces of prescribed dimensions, then for $i = 1, 2, 3$, all elements of the ith group are produced by using noises coming from V_i. The effect is that after doing homomorphic additions with coefficients *in the base field,* the noise of the obtained ciphertext has the same magnitude as that of its components. Furthermore, the magnitude of noises in the three groups are slightly different, *i.e.,* the one in the first group is of the smallest value while the other two groups have the same magnitude of noises, and allow "truly" homomorphic addition of order two, that is, any linear combination of two independent encodings/ciphertexts is again a valid encoding/ciphertext. The reason for this requirement will become clear in the proof of security. By further adding another encoding, *i.e.,* a mask, whose noises belong essentially to the same vector space as that of each group, we can argue from this property that the resulted ciphertext does not leak any potential information of the witness.

We also note that though the concrete parameters of our scheme do not compete well with those of [GMNO18], we emphasize that the novelty of our work lies in the way the encoding elements are divided and treated. We believe this method may be of independent interest for other applications.

Organization of the paper. The rest of this work is organized as follows. Section 2 recalls some basic matters needed; Section 3 and 4 describe an encoding scheme and the corresponding zk-SNARK construction. The efficiency and some examples of parameter are the content of Section 6.

2 Preliminaries

2.1 Notations

Vectors are denoted by bold low-case letters, *e.g.,* vector $\mathbf{v}$. Bold capital letters are used to denote matrices, *e.g.,* matrix $\mathbf{A}$. The notation $\mathcal{S}_r^n$ is defined to be the sphere of radius r in $\mathbb{F}_{\mathfrak{q}^{m_0}}^n$ for some positive integer m_0. We use the notation $[n]$ to denote the set $\{0, 1, \ldots, n\}$ for a positive integer n, $\lfloor x \rfloor$ to denote the greatest integer less than or equal to x, and $a \mid b$ to denote a divides b. Negligible functions are denoted by $\mathsf{neg}(\cdot)$.

2.2 Background on Code-Based Cryptography

This section recalls some basic code-based notions as well as ingredients needed, all of which could be found in [BGM22]. Let m_0, n be two positive integers and $\mathfrak{q}$ a power of a prime number. Let $\{\alpha_1, \ldots, \alpha_{m_0}\}$ be a basis of $\mathbb{F}_{\mathfrak{q}^{m_0}}$ over $\mathbb{F}_{\mathfrak{q}}$. This basis can be used to associate any vector $\mathbf{x} \coloneqq (x_1, \ldots, x_n) \in \mathbb{F}_{\mathfrak{q}^{m_0}}^n$ to the corresponding matrix $\mathbf{A}_{\mathbf{x}} \in \mathbb{F}_{\mathfrak{q}}^{n \times m_0}$ as

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1m_0} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nm_0} \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{m_0} \end{pmatrix}.$$

The rank weight of $\mathbf{x}$ is defined to be the rank of matrix $\mathbf{A}_{\mathbf{x}}$, that is, $\|\mathbf{x}\| \coloneqq \mathrm{rank}(\mathbf{A}_{\mathbf{x}})$. In this metric, the distance between two vectors $\mathbf{x}$ and $\mathbf{y}$, denoted by $d(\mathbf{x}, \mathbf{y})$, is defined to be equal the rank weight of $\mathbf{x} - \mathbf{y}$, *i.e.,* $d(\mathbf{x}, \mathbf{y}) \coloneqq \|\mathbf{x} - \mathbf{y}\|$.

Now, let $f(x) \in \mathbb{F}_{\mathfrak{q}^{m_0}}[x]$ be a polynomial of degree n and $\mathcal{R}_f \coloneqq \mathbb{F}_{\mathfrak{q}^{m_0}}[x]/\langle f \rangle$. Consider the following mapping:

$$\begin{aligned} \phi \colon \mathbb{F}_{\mathfrak{q}^{m_0}}^n &\longrightarrow \mathcal{R}_f \\ (a_0, \ldots, a_{n-1}) &\longmapsto a_0 + \cdots + a_{n-1}x^{n-1}. \end{aligned}$$

The inverse mapping, denoted by ϕ^{-1}, simply maps a polynomial to the vector formed by its coefficients. For the sake of simplicity, if $\mathbf{a} \coloneqq (a_0, \ldots, a_{n-1}) \in \mathbb{F}_{\mathfrak{q}^{m_0}}^n$, we let $\phi(\mathbf{a}) = a_0 + \cdots + a_{n-1}x^{n-1} = a(x)$. For $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^n$, their product $\mathbf{a} \cdot \mathbf{b}$ is defined as

$$\mathbf{a} \cdot \mathbf{b} \coloneqq \phi^{-1}\big(a(x) \cdot b(x)\big).$$

Clearly, we have $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$. It is also not hard to see that

$$\mathbf{a} \cdot \mathbf{b} = (a_0, \ldots, a_{n-1}) \cdot \begin{pmatrix} \phi^{-1}\big(b(x)\big) \\ \vdots \\ \phi^{-1}\big(x^{n-1}b(x)\big) \end{pmatrix}. \tag{1}$$

The right-most expression on the right-hand side of Equation 1 is usually referred to as the ideal matrix generated by $b(x)$ with respect to $f(x)$. For ease of notation, vectors are identified with their corresponding polynomials, *i.e.,* $x^k\mathbf{b}$ is understood to be $\phi^{-1}\big(x^k b(x)\big)$. Thus, the ideal matrix of a vector $\mathbf{b}$ with respect to f is written as

$$\vec{b} = \begin{pmatrix} \mathbf{b} \\ x \cdot \mathbf{b} \\ \vdots \\ x^{n-1} \cdot \mathbf{b} \end{pmatrix}.$$

In our construction, we will use 2- and 3-ideal codes. A 2-ideal code of length $2n$ with respect to a polynomial $f(x)$ of degree n over $\mathbb{F}_{\mathfrak{q}^{m_0}}$ is a code whose parity-check matrix is of the form

$$\mathbf{H} = \big[\mathbf{I}_n \mid \vec{h}^T\big], \tag{2}$$

where $\vec{h}$ is the ideal matrix of a vector $\mathbf{h}$ with respect to $f(x)$ in $\mathbb{F}_{\mathfrak{q}^{m_0}}^n$. Similarly, a 3-ideal code of length $3n$ with respect to a polynomial $f(x)$ of degree n over $\mathbb{F}_{\mathfrak{q}^{m_0}}$ is a code whose parity matrix is of the form

$$\mathbf{H} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \vec{h}_1^T \\ \mathbf{0} & \mathbf{I}_n & \vec{h}_2^T \end{pmatrix}. \tag{3}$$

For a given vector $\mathbf{x} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^n$, it is usually associated with the vector space generated by its coordinates.

Definition 1. *Let $\mathbf{x} \coloneqq (x_1, \ldots, x_n) \in \mathbb{F}_{\mathfrak{q}^{m_0}}^n$. The vector space over $\mathbb{F}_{\mathfrak{q}}$ defined by $x_1, \ldots, x_n$ is called the support of $\mathbf{x}$, and denoted by $\mathrm{supp}(\mathbf{x})$. That is,*

$$\mathrm{supp}(\mathbf{x}) \coloneqq \mathrm{Span}_{\mathbb{F}_{\mathfrak{q}}}(x_1, \ldots, x_n).$$

Next, we recall some definitions concerning code-based hardness assumptions.

Definition 2 (Rank Syndrome Decoding Problem). *Let n, k, and w be positive integers, $\mathbf{H}$ a random matrix over $\mathbb{F}_{\mathfrak{q}^{m_0}}^{(n-k)\times n}$, and $\mathbf{y}$ a random vector in $\mathbb{F}_{\mathfrak{q}^{m_0}}^{n-k}$. The rank syndrome decoding problem, $\mathrm{RSD}(n, k, w)$, asks to find a vector $\mathbf{x} \in \mathcal{S}_w^n$ such that $\mathbf{H}\mathbf{x}^T = \mathbf{y}^T$.*

In the following definitions, for $\nu \in \{2, 3\}$, let $S_P(n, \nu)$ be the set of all parity matrices of ν-ideal codes with respect to a polynomial $P(x)$ of degree n over $\mathbb{F}_{q^{m_0}}$, as defined in Equation 2 or 3, respectively.

Definition 3 (ν-IRSD Distribution). *Let n, w be positive integers, $P(x) \in \mathbb{F}_{\mathfrak{q}}[x]$ an irreducible polynomial of degree n. The ν-IRSD(n,w) distribution chooses uniformly at random a matrix $\mathbf{H} \in S_P(n, \nu)$ together with a vector $\mathbf{x} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^{\nu n}$ such that $\|\mathbf{x}\| = w$ and outputs $\left(\mathbf{H}, \mathbf{H} \cdot \mathbf{x}^T\right)$.*

Definition 4 (Computational ν-IRSD Problem). *Let n, w be positive integers, $P(x) \in \mathbb{F}_{\mathfrak{q}}[x]$ an irreducible polynomial of degree n, $\mathbf{H} \in S_P(n, \nu)$ a random matrix, and $\mathbf{y} \leftarrow \mathbb{F}_{\mathfrak{q}^{m_0}}^n$. The computational ν-IRSD(n,w) problem asks to find a vector $\mathbf{x} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^{\nu n}$ such that $\|\mathbf{x}\| = w$ and $\mathbf{H} \cdot \mathbf{x}^T = \mathbf{y}^T$.*

Definition 5 (Decisional ν-IRSD Problem). *The decisional ν-IRSD(n,w) problem asks to decide with non-negligible advantage whether $(\mathbf{H}, \mathbf{y}^T)$ came from the ν-IRSD(n,w) distribution or the uniform distribution over $S_P(n, \nu) \times \mathbb{F}_{\mathfrak{q}^{m_0}}^n$.*

Next, we recall the rank support learning problem. It made its first appearance in [GHPT17], in the construction of a rank-metric based public-key encryption scheme, and recently, in [BBBG22]. This problem can be viewed as a relaxation of the RSD problem in which, instead of giving one syndrome instance as in the RSD case, it gives a certain number of syndromes, all produced from the very same support of errors. Its definition reads

Definition 6 (Rank Support Learning Problem). *Let n, k, r, N be positive integers. Given a matrix $\mathbf{H} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^{(n-k)\times n}$ and N syndromes $\mathbf{s}_i^T = \mathbf{H}\mathbf{e}_i^T$, where $\mathbf{e}_i \leftarrow V$ for all $i = 1, 2, \ldots, N$, and V is a subspace of $\mathbb{F}_{\mathfrak{q}^{m_0}}^n$ of dimension r, the $\mathrm{RSL}(n, k, r, N)$ problem asks to find V.*

When the number N increases, the problem becomes easier to solve. The attack in [DT18] suggests that parameters should be chosen satisfying $N < kr$. The decisional version of this problem is as follows.

Definition 7 (Decisional RSL Problem). *Given an instance either from $(\mathbf{H}, \mathbf{HE})$ or $(\mathbf{H}, \mathbf{U})$, where $\mathbf{H}$ is a full rank matrix of size $(n-k) \times n$, $\mathbf{U}$ is a random matrix in $\mathbb{F}_{\mathfrak{q}^{m_0}}^{(n-k)\times N}$, and $\mathbf{E}$ is a matrix formed from N randomly chosen vectors $\mathbf{e}_i$'s in a vector space of dimension r, the decisional rank support learning $\mathrm{DRSL}(n, k, r, N)$ asks to decide which is the case.*

For our purpose, we also need another variant of this problem. In addition to a set of N vectors, either produced from preimages of the same support or from the uniform distribution, two additional vectors are also given, which are $\mathbb{F}_{\mathfrak{q}}$-linearly random combinations of these N vectors. The problem now still asks to decide which is the case.

Definition 8 (Variant RSL Problem-vRSL). *Given an instance either from $(\mathbf{H}, \mathbf{HE}, \mathbf{HE}\cdot\mathbf{a}^T, \mathbf{HE}\cdot\mathbf{b}^T)$ or $(\mathbf{H}, \mathbf{U}, \mathbf{U} \cdot \mathbf{a}^T, \mathbf{U} \cdot \mathbf{b}^T)$, where $\mathbf{H} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^{(n-k)\times n}$ is a full rank matrix, $\mathbf{U}$ is a random matrix in $\mathbb{F}_{\mathfrak{q}^{m_0}}^{(n-k)\times N}$, $\mathbf{a}, \mathbf{b}$ are randomly chosen vectors in $\mathbb{F}_{\mathfrak{q}}^N$, and $\mathbf{E}$ is a matrix formed from N randomly chosen vectors $\mathbf{e}_i$'s in a vector space of dimension r, the $\mathrm{vRSL}(n, k, r, N)$ problem asks to decide which is the case.*

The rationale behind this formulation is that what really affects the hardness of the problem is the information of V given in the form of $\mathbf{HE}$ or, equivalently, $\{\mathbf{He}_i^T\}_{i=1}^N$. Adding one or two random $\mathbb{F}_q$-linearly combinations of these vectors does not leak more information about V. In fact, this problem is not easier than the RSL problem. Given an RSL instance, one could create a vRSL instance by randomly picking two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^N$, computing the corresponding linearly combinations; these combinations together with the provided RSL instance then form an instance of the vRSL problem. Therefore, if we can solve the vRSL problem, then we can also solve the RSL problem. The hardness of this problem is used to argue the zero-knowledge property of our scheme. Furthermore, the hardness of this problem also guarantees the zero-knowledge property of the linear coefficients, *i.e.,* the random vectors $\mathbf{a}$ and $\mathbf{b}$. An adversary, if being asked for such a set of coefficients, has either to solve the RSL problem or to make a guess, both of which succeed with negligible probability.

Remark 1. We remark that the problem in the above definition could be generalized to the case in which, in addition to the syndromes (either chosen uniformly or not), a polynomial number of random $\mathbb{F}_q$-linearly combinations of these syndromes are also given. The reduction could be carried in the same manner.

In the above definitions, the matrix $\mathbf{H}$ can be assumed to have ideal structures, and we also make the assumption that the problems corresponding to this situation, namely, the ideal rank support learning (IRSL) problem and its variant (vIRSL), are hard.

2.3 Succinct Non-Interactive Arguments

We recall the definition of (designated-verifier) succinct non-interactive arguments of knowledge (SNARKs) below. We specialize our definitions to the problem of Boolean circuit satisfiability.

Definition 9. *Let $\mathcal{C} \coloneqq \{C_n\}_{n\in\mathbb{N}}$ be a family of Boolean circuits. A designated-verifier non-interactive argument system for an NP relation $\mathcal{R}_\mathcal{C}$ is a triple of algorithms $\Pi = (\mathsf{G}, \mathsf{P}, \mathsf{V})$ such that*

- $\mathsf{G}(1^\lambda, 1^n)$: *On input the security parameter λ and the circuit family parameter n, the setup algorithm G generates a common reference string crs and a verification key vrs.*
- $\mathsf{P}(\mathsf{crs}, u, w)$: *On input the common reference string crs, a statement u, and its witness w, the prover algorithm P generates a proof π.*
- $\mathsf{V}(\mathsf{vrs}, u, \pi)$: *On input the verification key vrs, a statement u and a proof π, the verification algorithm V outputs 1 if the proof π is valid, and 0 otherwise.*

An argument of knowledge system is required to be complete and to have knowledge soundness.

Definition 10 (Completeness). *An argument of knowledge system Π for a relation $\mathcal{R}_\mathcal{C}$ is complete if for all $n \in \mathbb{N}$ and for any pair $(u, w) \in \mathcal{R}_{C_n}$, we have*

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{vrs}) \leftarrow \mathsf{G}(1^\lambda, 1^n) \\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, u, w) \\ s.t.\ \mathsf{V}(\mathsf{vrs}, u, \pi) = 1\end{array}\right] \geq 1 - \mathsf{neg}(\lambda).$$

Definition 11 (Knowledge Soundness). *An argument of knowledge system Π for the relation $\mathcal{R}_\mathcal{C}$ is knowledge-sound if for any PPT adversary $\mathcal{A}$, there exists an extractor $\mathrm{Ext}_\mathcal{A}$, given access to $\mathcal{A}$'s inputs, such that*

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{vrs}) \leftarrow \mathsf{G}(1^\lambda, \mathcal{R}) \\ (u, \pi; w) \leftarrow (\mathcal{A} \parallel \mathrm{Ext}_\mathcal{A})^{\mathsf{V}(\mathsf{vrs}, \cdot)}(\mathsf{crs}) \\ s.t.\ (u, w) \notin \mathcal{R} \wedge \mathsf{V}(\mathsf{vrs}, u, \pi) = 1\end{array}\right] \leq \mathsf{neg}(\lambda),$$

where $(y; z) \leftarrow (\mathcal{A} \parallel \mathrm{Ext}_\mathcal{A})(x)$ signifies that on input x, $\mathcal{A}$ outputs y, and that $\mathrm{Ext}_\mathcal{A}$, given the same input x and $\mathcal{A}$'s random tape, produces z.

Additionally, a system is said to be *succinct* if it satisfies the following property.

Definition 12 (Succinctness). *There exists a fixed polynomial $p(\cdot)$ independent of $\mathcal{C}$ such that for every large enough security parameter $\lambda \in \mathbb{N}$, we have that*

– **Fully Succinct:** G *runs in time* $p(\lambda + \log|C_n|)$, V *runs in time* $p(\lambda + |x| + \log|C_n|)$, *and the length of the proof output by* P *is bounded by* $p(\lambda + \log|C_n|)$.
– **Preprocessing:** G *runs in time* $p(\lambda + |C_n|)$, V *runs in time* $p(\lambda + |x| + \log|C_n|)$, *and the length of the proof output by* P *is bounded by* $p(\lambda + \log|C_n|)$.

If an argument system has the property that the witness(es) is (computationally) hiding, then it is said to be zero-knowledge. This notion is captured by the simulation paradigm: there exists a PPT algorithm $\mathcal{S}$, called simulator, such that given a statement u, it generates a valid proof whose distribution is indistinguishable from that generated in the real protocol.

Definition 13 (Zero-knowledge). *An argument of knowledge system* Π *is zero-knowledge if there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that for any PPT adversary* $\mathcal{A}$ *given access to an oracle* $\mathcal{O}$ *defined as*

$$\begin{array}{l}\underline{\textit{Oracle } \mathcal{O}_b(u,w)\textit{:}}\\ \textit{If } R(u,w) = \mathsf{false}\textit{, return } \bot.\\ \textit{If } b = 1\textit{, then } \pi \leftarrow \mathsf{P}(\mathsf{crs}, u, w)\textit{, else } \pi \leftarrow\\ \mathcal{S}_2(\mathsf{td}, u),\\ \textit{return } \pi,\end{array}$$

we have

$$\Pr\left[\begin{array}{c}(\mathsf{crs},\mathsf{vrs},\mathsf{td}) \leftarrow \mathcal{S}_1(1^\lambda, 1^n)\\ b \leftarrow \{0,1\}\\ b' \leftarrow \mathcal{A}^{\mathcal{O}_b}(\mathsf{vrs})\\ s.t.\ b = b'\end{array}\right] \leq \frac{1}{2} + \mathsf{neg}(\lambda).$$

Definition 14 (zk-SNARK). *A succinct non-interactive zero-knowledge argument of knowledge (zk-SNARK) is a non-interactive argument system that is complete, succinct, knowledge-sound and zero-knowledge.*

2.4 Encoding Schemes

We recall the definition of encoding schemes with noise from [GMNO18], adapted to our secret-key setting.

Definition 15. *An encoding scheme* Enc *over a finite field* $\mathbb{F}_\mathfrak{q}$ *is a tuple of PPT algorithms* $(\mathsf{K}, \mathsf{E}, \mathsf{D})$ *such that:*

– $\mathsf{K}(1^\lambda)$: *The key generating algorithm takes as input the security parameter* λ *and outputs a public information* pk *and a secret state* sk.
– $\mathsf{E}(\mathsf{sk}, m)$: *The non-deterministic encoding algorithm maps an element* $m \in \mathbb{F}_\mathfrak{q}$ *into some encoding space* S *using the secret state* sk, *such that* $\left\{\{\mathsf{E}(a)\} \mid a \in \mathbb{F}_\mathfrak{q}\right\}$ *partitions* S, *where* $\{\mathsf{E}(a)\}$ *denotes the set of the possible evaluations of the algorithm* E *on* a.
– $\mathsf{D}(\mathsf{sk}, \mathbf{c})$: *The decoding algorithm takes as input the secret state* sk, *an encoding* $\mathbf{c}$ *and outputs an element* $m \in \mathbb{F}_\mathfrak{q}$.

An encoding scheme Enc must have the following properties:

– $\mathfrak{d}$**-linearly homomorphic:** there exists a PPT algorithm Eval which takes pk, $\mathfrak{d}$ encodings $\mathsf{E}(m_1), \ldots, \mathsf{E}(m_\mathfrak{d})$, and coefficients $(a_1, \ldots, a_\mathfrak{d}) \in \mathbb{F}_\mathfrak{q}^\mathfrak{d}$ as input and outputs a valid encoding of $\sum_{i=1}^{\mathfrak{d}} a_i m_i$ with overwhelming probability in λ.
– **Quadratic root detection:** there exists a PPT algorithm which takes the public key pk, a set of encodings $\{\mathsf{E}(m_1), \ldots, \mathsf{E}(m_\mathfrak{t})\}$, and a quadratic polynomial $P \in \mathbb{F}_\mathfrak{q}[x_1, \ldots, x_\mathfrak{t}]$ as input and checks for the correctness of the equality $P(m_1, \ldots, m_\mathfrak{t}) = 0$.
– **Image verification:** there exists a PPT algorithm which takes the public key pk and an element $\mathbf{c}$ as input and decides whether $\mathbf{c}$ is a valid encoding of a field element or not.

2.5 Assumptions

The following assumptions are the adaptations of q-PDH and q-PKE assumptions (cf. [GMNO18], [GGPR13]) to the code-based context together with the application of the rank support learning problem. In the following, all the encodings are produced by using *a common vector space* of noise.

Assumption 1 (q-PDH). *Let* $\mathsf{Enc} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be an encoding scheme over a finite field* $\mathbb{F}_{\mathfrak{q}}$. *The q-power Diffie-Hellman assumption, q-PDH, holds for* Enc *if for all PPT adversary* $\mathcal{A}$, *we have*

$$\Pr\left[\begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda), s \leftarrow \mathbb{F}_{\mathfrak{q}} \\ y \leftarrow \mathcal{A}\left(\mathsf{pk}, \mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q})\right) \\ s.t.\ y = \mathsf{E}(s^{q+1})\end{array}\right] \leq \mathsf{neg}(\lambda).$$

Assumption 2 (q-PKE). *Let* $\mathsf{Enc} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *be an encoding scheme over a finite field* $\mathbb{F}_{\mathfrak{q}}$. *The q-power of knowledge of exponent assumption, q-PKE, holds for* Enc *if for all PPT adversary* $\mathcal{A}$, *there exists a non-uniform knowledge extractor* $\mathrm{Ext}_{\mathcal{A}}$, *given access to $\mathcal{A}$'s input, such that*

$$\Pr\left[\begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda), \alpha, s \leftarrow \mathbb{F}_{\mathfrak{q}} \\ \sigma \leftarrow \left(\mathsf{pk}, \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^q)\right) \\ (c, \widehat{c}, (a_0, \ldots, a_q)) \leftarrow (\mathcal{A} \parallel \mathrm{Ext}_{\mathcal{A}})(\sigma, z) \\ s.t.\ \widehat{c} = \alpha c \wedge c \notin \left\{\mathsf{E}\left(\sum_{i=0}^{q} a_i s^i\right)\right\}\end{array}\right] \leq \mathsf{neg}(\lambda),$$

for any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$ *that is generated independently of* α.

2.6 Square Span Programs

We briefly recall here the definition of a square span program [DFGK14].

Definition 16. *A* square span program *over a finite field* $\mathbb{F}_{\mathfrak{q}}$ *consists in a tuple of* $m+1$ *polynomials* $v_0(x), v_1(x), \ldots, v_m(x) \in \mathbb{F}_{\mathfrak{q}}[x]$ *and a target polynomial* $t(x)$ *such that* $\deg v_i \leq \deg t$ *for all* $0 \leq i \leq m$. *We say that the square span program* ssp *has size* m *and degree* $d = \deg t$. *We say that* ssp *accepts an input* $a_1, \ldots, a_\ell \in \{0,1\}$ *if and only if there exist* $a_{\ell+1}, \ldots, a_m \in \{0,1\}$ *satisfying*

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

We say that ssp *verifies a boolean circuit* $\mathtt{C}\colon \{0,1\}^\ell \to \{0,1\}$ *if it accepts exactly those inputs* $(a_1, \ldots, a_\ell) \in \{0,1\}^\ell$ *satisfying*

$$\mathtt{C}(a_1, \ldots, a_\ell) = 1.$$

We follow [GMNO18]'s approach for the SSP generation. That is, on a boolean circuit $\mathtt{C}$ of size d, it generates a finite field $\mathbb{F}_{\mathfrak{q}}$ of $\mathfrak{q}$ elements such that $\mathfrak{q} \geq \max\{d, 8\}$. Next, it randomly picks d elements $r_0, \ldots, r_{d-1}$ and defines $t(x) \coloneqq (x - r_0) \cdots (x - r_{d-1})$. It outputs ssp as

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

where $v_0, \ldots, v_m$ are $m+1$ polynomials of degree at most d as in the above definition.

3 Our Code-based Encoding Scheme

In this section, we describe our instantiation of an encoding scheme from coding theory, which is based on the RQC cryptosystem. The description of the RQC encryption scheme was originally published in [ABD+16], and can be found in [AAB+19] with little changes. In this work, it is turned into a secret-key and used as an encoding scheme in the following manner.

1. $\mathsf{Setup}(1^\lambda)$: Generate parameters $n \coloneqq n(\lambda), k \coloneqq k(\lambda), \delta \coloneqq \delta(\lambda), w \coloneqq w(\lambda), w_e \coloneqq w_e(\lambda), w_r \coloneqq w_r(\lambda)$. The plaintext space is $\mathbb{F}_{\mathfrak{q}^{m_0}}^k$. Output $\mathsf{param} \coloneqq \left(n, k, \delta, w, w_e, w_r, P(x)\right)$, where $P(x) \in \mathbb{F}_{\mathfrak{q}}[x]$ is an irreducible polynomial of degree n which remains irreducible over $\mathbb{F}_{\mathfrak{q}^{m_0}}[x]$.

2. $\mathsf{KeyGen}(\mathsf{param})$: Generate $\mathbf{h} \leftarrow \mathbb{F}_{\mathfrak{q}^{m_0}}^n, \mathbf{x}, \mathbf{y} \leftarrow \mathcal{S}_w^n$, a generator matrix $\mathbf{G} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^{k \times n}$ of a public code $\mathcal{C}$, which is capable of correcting up to δ errors. Output the public parameters $\mathsf{pp} := (\mathsf{param}, \mathbf{h}, \mathbf{G})$ and $\mathsf{sk} := (\mathbf{x}, \mathbf{y})$.
3. $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk}, \mathbf{m})$: To encrypt a message $\mathbf{m} \in \mathbb{F}_{\mathfrak{q}^{m_0}}^k$, randomly choose $\mathbf{r}_1, \mathbf{r}_2 \leftarrow \mathcal{S}_{w_r}^n$ and $\mathbf{e} \leftarrow \mathcal{S}_{w_e}^n$. Compute
$$\begin{cases} \mathbf{c}_1 \leftarrow \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2, \\ \mathbf{c}_2 \leftarrow (\mathbf{x} + \mathbf{h} \cdot \mathbf{y}) \cdot \mathbf{r}_2 + \mathbf{e} + \mathbf{m} \cdot \mathbf{G}. \end{cases}$$
Return $\mathbf{c} := (\mathbf{c}_1, \mathbf{c}_2)$.
We note that the noises vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{e}$ are chosen from a common vector space. Therefore, we also have $w_r = w_e$.
4. $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$: To decrypt, first compute $\mathbf{c}_2 - \mathbf{y} \cdot \mathbf{c}_1$, and then use the decoding algorithm of the code $\mathcal{C}$ to recover $\mathbf{m}$.

From the RQC encryption scheme describe above, our encoding scheme $(\mathsf{K}, \mathsf{E}, \mathsf{D})$ is defined in which K consists of Setup and KeyGen algorithms, the encoding algorithm E is the encryption algorithm Enc, the decoding algorithm D is the decryption algorithm Dec.

We aim at encoding of elements of the finite field $\mathbb{F}_{\mathfrak{q}}$, we do it as follows. For an element $s \in \mathbb{F}_{\mathfrak{q}}$, define $\mathbf{s} = (s, 0, \ldots, 0) \in \mathbb{F}_{\mathfrak{q}}^k$, *i.e.,* the vector $\mathbf{s}$ is formed by placing s in the first entry and 0 elsewhere. An encoding of s is defined to be an encryption of $\mathbf{s}$, *i.e.,* $\mathsf{E}(s) := \mathsf{Enc}(\mathbf{s})$. For two elements $s_1, s_2 \in \mathbb{F}_{\mathfrak{q}}$, if we denote $t = s_1 s_2$, then we have $\mathbf{t} = \mathbf{s}_1 \cdot \mathbf{s}_2 = (s_1 s_2, 0, \ldots, 0)$, and hence, $\mathsf{E}(s_1 s_2) = \mathsf{Enc}(\mathbf{s}_1 \cdot \mathbf{s}_2)$. Therefore, this mapping (from $\mathbb{F}_{\mathfrak{q}}$ to $\mathbb{F}_{\mathfrak{q}^{m_0}}^k$) is well-defined.

To complete the description of our encoding scheme, its properties are defined below.

- $\mathsf{Eval}(\mathsf{pk}, \mathbf{c}_1, \ldots, \mathbf{c}_{\mathfrak{d}}; a_1, \ldots, a_{\mathfrak{d}})$ computes and outputs $\tilde{\mathbf{c}} = (\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2)$, where $\tilde{\mathbf{c}}_b = \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{c}_{b,i}$, for some prescribed positive integer $\mathfrak{d}$ describing the number of desired homomorphic additions and $b \in \{1, 2\}$.
- Quadratic root detection uses the decryption algorithm to invert ciphertexts and evaluates value of the polynomial at the obtained messages.
- Image verification uses the decryption algorithm of RQC to test whether a given vector $\mathbf{c}$ is a valid encoding of some plaintext or not.

By the hardness of IRSL problem, a random vector space of noise can be used a couple of times, which is described in the problem. Our zk-SNARK construction will exploit this variation in subsequent sections. Furthermore, similar to previous work [GMNO18], we will assume that our encoding scheme satisfies the q-PDH and q-PKE assumptions as described in Section 2.5.

3.1 Bound of Noise

The main point of this section is to give a feature of the sum of noises in a particular case, that is, when the sum of noise's weight is much smaller than either the vector length or the degree of the field extension. Simply stated, the weight of the sum is upper-bounded by the sum of every single noise's weight.

Proposition 1. *Let $\ell, m_0, n, w_1, \ldots, w_\ell$ be positive integers such that $m_0, n > d_w$, where $d_w = w_1 + \cdots + w_\ell$. Let t_i be randomly chosen from $\mathcal{S}_{w_i}^n$ for $i = 1, \ldots, \ell$, and $U = \operatorname{supp}\left(\sum_{i=1}^{\ell} \mathsf{t}_i\right)$. Then, we have $\dim U \leq d_w$.*

Proof. The proof is quite straightforwards, since we have

$$U \subseteq \operatorname{supp} \mathsf{t}_1 \oplus \cdots \oplus \operatorname{supp} \mathsf{t}_\ell. \qquad \square$$

As mentioned earlier, for our construction, noises used in the encodings (in each group) share a common vector space and the linear coefficient would be in the based field. Therefore, the noise of the resulted encoding would also belong to the prescribed vector space. The above proposition proves to be helpful in the security proof of our zk-SNARK later, and in fact, the equality holds with overwhelming probability [Mac21].

3.2 Additive Homomorphism

The purpose of this section is to show the additive homomorphism of the RQC scheme. Intuitively, by the result of the previous section, the noise of the homomorphic ciphertext grows linearly with respect to the number of additive components. However, as long as the magnitude of the homomorphic noise is within the decoding capability of the public code $\mathcal{C}$, the decoding algorithm will always succeed.

Proposition 2. *Let $\mathfrak{d}$ be the number of additive operations, w, w_e, w_r be magnitudes of the secret key and noises from an* RQC *scheme whose public code can decode errors of rank weight up to δ. If $\mathfrak{d}(2ww_r+w_e) \leq \delta$, then $\tilde{\mathbf{c}}$, which is the output of* Eval *is a correct encoding.*

Proof. Observe that $\tilde{\mathbf{c}} = (\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2)$, where

$$\begin{cases} \tilde{\mathbf{c}}_1 = \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(1)} + \mathbf{h} \cdot \left(\sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(2)} \right), \\ \tilde{\mathbf{c}}_2 = \mathbf{s} \cdot \left(\sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(2)} \right) + \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{e}_i + \left(\sum_{i=1}^{\mathfrak{d}} a_i \mathbf{m}_i \right) \cdot \mathbf{G}. \end{cases}$$

Since a_i's are elements of $\mathbb{F}_{\mathfrak{q}^{m_0}}$, so $\|a_i \mathbf{r}_i^{(j)}\| = \|\mathbf{r}_i^{(j)}\|$ and $\|a_i \mathbf{e}_i\| = \|\mathbf{e}_i\|$ for all $1 \leq i \leq \mathfrak{d}$ and $j = 1, 2$. By Proposition 1, we get

$$\begin{cases} w_r \mathfrak{d} \leq \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(1)}, \\ w_r \mathfrak{d} \leq \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(2)}, \\ w_e \mathfrak{d} \leq \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{e}_i. \end{cases}$$

Thus,

$$\begin{aligned} & \left\| \mathbf{x} \cdot \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(2)} - \mathbf{y} \cdot \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{r}_i^{(1)} + \sum_{i=1}^{\mathfrak{d}} a_i \mathbf{e}_i \right\| \\ \leq\ & w \cdot w_r \mathfrak{d} + w \cdot w_r \mathfrak{d} + w_e \mathfrak{d} \\ \leq\ & \delta, \end{aligned}$$

which allows successful decryption. □

4 Our Code-based zk-SNARK Scheme

In the following, let $(\mathsf{K}, \mathsf{E}, \mathsf{D})$ be the encoding scheme described in Section 3. We assume Our zk-SNARK scheme Π is detailed as:

– **Setup.** The setup algorithm takes as input the security parameter 1^λ in the unary form and the circuit $\mathtt{C}$. It generates a square span program of degree d over the field $\mathbb{F}_\mathfrak{q}$ of size $\mathfrak{q} \geq d$ that verifies $\mathtt{C}$ by running:

$$\mathsf{ssp} = \big(v_0(x), \ldots, v_m(x), t(x)\big) \leftarrow \mathsf{SSP}(\mathtt{C}).$$

Then, it runs $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$ using our encoding scheme. It samples $\alpha, \beta, s \leftarrow \mathbb{F}_\mathfrak{q}$ such that $t(s) \neq 0$, and returns the crs:

$$\mathsf{crs} \coloneqq \big(\mathsf{ssp}, \mathsf{pp}, \mathfrak{G}_1, \mathfrak{G}_2, \mathfrak{G}_3\big),$$

where

$$\begin{aligned} \mathfrak{G}_1 &\coloneqq \big\{\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^d)\big\}, \\ \mathfrak{G}_2 &\coloneqq \big\{\mathsf{E}(\alpha), \mathsf{E}(\alpha s), \ldots, \mathsf{E}(\alpha s^d)\big\}, \\ \mathfrak{G}_3 &\coloneqq \big\{\mathsf{E}(\beta t(s)), \{\mathsf{E}(\beta v_i(s))\}_{i=\ell_u+1}^m\big\}, \end{aligned}$$

and ℓ_u denotes the size of input u of circuit $\mathtt{C}$. Elements in each group are formed from the encoding scheme which uses three vector spaces of noises V_1, V_2, V_3, respectively. Furthermore, $\dim V_1 = r$, $\dim V_2 = \dim V_3 = \mathfrak{r}$, where $0 < \mathfrak{r} - r$.
Finally, it sets $\mathsf{vrs} = \mathsf{sk}$ and $\mathsf{td} = (\alpha, \beta, s)$ as the verification key and the trapdoor, respectively.

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

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

The prover samples $\gamma \leftarrow \mathbb{F}_{\mathfrak{q}}$, sets $v(x) = v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \gamma t(x)$ and

$$h(x) = \frac{v(x)^2 - 1}{t(x)} \in \mathbb{F}_{\mathfrak{q}}[x].$$

It computes

$$H = \mathsf{E}\big(h(s)\big), \quad \widehat{H} = \mathsf{E}\big(\alpha \cdot h(s)\big), \quad \widehat{V} = \mathsf{E}\big(\alpha \cdot v(s)\big),$$
$$V_w = \mathsf{E}\left(\sum_{i=\ell_u+1}^{m} a_i v_i(s) + \gamma t(s)\right),$$
$$B_w = \mathsf{E}\left(\beta \cdot \left(\sum_{i=\ell_u+1}^{m} a_i v_i(s) + \gamma t(s)\right)\right).$$

The prover returns $\pi \coloneqq \big(H, \widehat{H}, \widehat{V}, V_w, B_w\big)$.
We note that the prover computes H and V_w from the first encoding group $\mathfrak{G}_1$, $\widehat{H}$ and $\widehat{V}$ from the second group $\mathfrak{G}_2$, and B_w from the third group $\mathfrak{G}_3$ in the following manners. Assume that $h(x) = h_0 + h_1 x + \cdots + h_d x^d$, then

1. $H = \mathsf{Eval}\big(\{\mathsf{E}(s^i)\}_{i=0}^{d}; \{h_i\}_{i=0}^{d}\big)$; the same kind of computation is used for V_w;
2. $\widehat{H} = \mathsf{Eval}\big(\{\mathsf{E}(\alpha s^i)\}_{i=0}^{d}; \{h_i\}_{i=0}^{d}\big)$; this is also applied for $\widehat{V}$;
3. $B_w = \mathsf{Eval}\big(\mathsf{E}(\beta t(s)), \{\mathsf{E}(v_i(s))\}_{i=\ell_u+1}^{m}; \gamma, 1, \ldots, 1\big)$ with the observation that

$$\gamma \mathsf{E}\big(\beta t(s)\big) = \mathsf{E}\big(\gamma \beta t(s)\big).$$

– **Verifier.** Upon receiving a proof π and a statement $u = (a_1, \ldots, a_{\ell_u})$, the verifier, in possession of the verification key vrs first checks that

$$\widehat{h}_s - \alpha \cdot h_s = 0, \quad \widehat{v}_s - \alpha \cdot v_s = 0,$$
$$v_s^2 - 1 - h_s \cdot t_s = 0,$$
$$b_s - \beta \cdot \mathfrak{v}_s = 0,$$

where $(h_s, \widehat{h}_s, \widehat{v}_s, \mathfrak{v}_s, b_s)$ are the values encoded in $\pi = \big(H, \widehat{H}, \widehat{V}, V_w, B_w\big)$, and t_s, v_s are computed as $t_s \coloneqq t(s)$ and $v_s \coloneqq v_0 + \sum_{i=1}^{\ell_u} a_i v_i(s) + \mathfrak{v}_s$. (Recall that $t(s)$ and $v_i(s)$ are obtained from the CRS.) The verifier checks that whether it is possible to perform one more homomorphic operation. (Thus, in our scheme, essentially $\mathfrak{d}$ takes the value 2.) If these checks pass, it outputs 1; otherwise, it outputs 0.

Theorem 1. *If the* q-PKE *and* q-PDH *assumptions hold for the encoding scheme* $(\mathsf{K}, \mathsf{E}, \mathsf{D})$ *then the protocol above is a* zk-SNARK *with perfect completeness, computational soundness and computational zero-knowledge.*

The perfectness of the scheme is guaranteed by the fact that the decryption step succeeds with probability 1. Therefore, we only need to concern ourselves with the zero-knowledge and soundness property.

5 Security Analysis of Our zk-SNARK Scheme

5.1 Zero-Knowledge

The idea behind this property is that the distributions of the elements in a proof does not differ from that of its components. The description of the simulator $\mathcal{S}$ is as follows. On input $\mathsf{td} = (\alpha, \beta, s)$ and $u = (a_1, \ldots, a_{\ell_u})$,

1. $\mathcal{S}$ randomly picks an element $\gamma \in \mathbb{F}_{\mathfrak{q}}$ and computes

$$h = \frac{\left(v_0(s) + \sum_{i=1}^{\ell_u} a_i v_i(s) + \gamma\right)^2 - 1}{t(s)}.$$

2. It computes

$$\begin{cases} H = \mathsf{E}(h), \quad \widehat{H} = \mathsf{E}(\alpha h), \\ \widehat{V} = \mathsf{E}\big(\alpha(v_0(s) + \sum_{i=1}^{\ell_u} a_i v_i(s) + \gamma)\big), \quad V_\gamma = \mathsf{E}(\gamma), \\ B_\gamma = \mathsf{E}(\beta\gamma). \end{cases}$$

3. It outputs $(H, \widehat{H}, \widehat{V}, V_\gamma, B_\gamma)$.

Proof. Since the encodings in each group $\mathfrak{G}_i$ share a same vector space V_i for $i = 1, 2, 3$, and all the polynomials in consideration are in $\mathbb{F}_{\mathfrak{q}}[x]$, therefore, after homomorphically adding, the noises in the new encodings belong to the same vector spaces as that of its component encodings.

Observe that in the real protocol, H and V_w are resulted from adding encodings of $\mathfrak{G}_1$, $\widehat{H}$ and $\widehat{V}$ from $\mathfrak{G}_2$, and B_w from $\mathfrak{G}_3$, respectively. By the vIRSL$(2n, n, \mathfrak{r}, N)$ problem (for each group), these outputs are computationally indistinguishable from truly random ones.

On the other hand, by the decisional IRSL$(2n, n, \mathfrak{r}, N)$ problem, the distribution of the outputs of the simulated protocol are computationally indistinguishable from the uniformly random. Therefore, by hybrid argument, we conclude that the outputs distribution of the real execution and that of the simulation are computationally indistinguishable.

□

We note also that the role of $\gamma t(s)$ in the scheme is to hide the witness and is indispensable. Indeed, since the homomorphic linear coefficients are elements of $\mathbb{F}_{\mathfrak{q}}$, so they form a vector which can be viewed as a rank-1 vector over $\mathbb{F}_{\mathfrak{q}^{m_0}}^N$. In this particular situation, finding these low rank vectors can be performed as follows. From the encodings of $\mathfrak{G}_3$, except the first, form an $n \times (m - \ell_u)$ matrix whose columns are the first parts of these encodings. Note that, without adding $\gamma t(s)$, the last term of the proof becomes

$$B_w = \mathsf{E}\left(\beta \cdot \left(\sum_{i=\ell_u+1}^{m} a_i v_i(s)\right)\right) = \mathsf{Eval}\left(\{\mathsf{E}(\beta v_i(s))\}_{i=\ell_u+1}^{m}; \{a_i\}_{i=\ell_u+1}^{m}\right).$$

Regarding as a rank decoding problem and by applying algorithm in [CS96], $(a_{\ell_u+1}, \ldots, a_m)$ could be recovered in polynomial time. Thus, witness must be concealed by the necessary use of the term $\gamma t(s)$.

One may hide the witness by further repeating one more time exactly the encryption step of the RQC scheme, *i.e.,* re-randomizing the resulted encoding before outputting it (thus modifying the evaluation algorithm). By the hardness of the IRSL problem, we can argue the (computational) indistinguishability of the output. This is somewhat similar to the technique of [ISW21].

5.2 Soundness

The idea of the proof follows the frame of [PGHR13] and [GMNO18] with some adaptations to rank-code hardness assumptions.

Proof. Assume that there is an adversary $\mathcal{A}$ who can break the scheme Π with non-negligible probability, we construct an algorithm $\mathcal{B}$ to solve q-PDH or q-PKE problems. First, we show how $\mathcal{B}$ can use $\mathcal{A}$ for this purpose.

Let π be a proof produced by $\mathcal{A}$ which is accepted. Using an extractor of the d-PKE assumption, $\mathcal{B}$ can recover the coefficients of the polynomials $v(x), h(x)$. Define

$$v_{\text{mid}}(x) = v(x) - v_0(x) - \sum_{i=1}^{\ell_u} a_i v_i(x).$$

Since the proof is accepted but the statement is false, so by the same arguments as in the proof in [PGHR13], there are only two possibilities

(*i*) $t(x)h(x) \neq v^2(x) - 1$ but $t(s)h(s) = v^2(s) - 1$, or
(*ii*) B_w is an encoding of $\beta v_{\text{mid}}(s)$ but v_{mid} is not in the linear span of $\{v_{\ell_u+1}, \ldots, v_m, t\}$

Claim. If (*i*) holds, then $\mathcal{B}$ can break the q-PDH assumption with $q = 2d - 1$.

Proof. Let $p(x) = v^2(x) - 1 - t(x)h(x)$. In this case, $p(x)$ is a polynomial of degree at most $2d$ having s as a root. Assume that p_k is the leading coefficient of $p(x)$, define

$$\widehat{p}(x) = x^k - p_k^{-1} p(x).$$

We see that s is a root of $x^k - \widehat{p}(x)$, therefore, it also is a root of $x^{q+1} - x^{q+1-k}\widehat{p}(x)$. Observe that for $q = 2d-1$, $x^{q+1-k}\widehat{p}(x)$ is a polynomial of degree at most $2d-1$. Therefore, $\mathsf{E}\big(s^{q+1-k}\widehat{p}(s)\big)$ can be computed from $\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^{2d-1})$, which form a challenge of the q-PDH assumption for $q = 2d-1$. This means that $\mathcal{B}$ can compute $\mathsf{E}(s^{q+1}) = \mathsf{E}\big(s^{q+1-k}\widehat{p}(s)\big)$ and break the q-PDH assumption for $q = 2d - 1$. □

Claim. If (*ii*) holds, then $\mathcal{B}$ can break the q-PDH assumption with $q = d$.

Proof. First, $\mathcal{B}$ generates a uniformly random polynomial $a(x)$ of degree $q = d$ subject to the constraint that all the polynomials $a(x)t(x)$ and $\{a(x)v_i(x)\}_{i=\ell_u+1}^{m}$ do not contain the term x^q. Since $\deg a(x) = d$, so $\mathcal{B}$ can compute the value $a(s)$ from the challenge of the d-PDH assumption, namely, $\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^d), \mathsf{E}(s^{d+2}), \ldots, \mathsf{E}(s^{2d})$. Thus, when preparing inputs for adversary $\mathcal{A}$, $\mathcal{B}$ sets $\beta = a(s)$. The proof is accepted, so the term B_w must be an encoding of a known polynomial in s, *i.e.,* the polynomial

$$a(s)v_{\text{mid}}(s) = b_0 + b_1 s + \cdots + b_{2q} s^{2q}.$$

Since v_{mid} is not in the linear span of $\{v_{\ell_u+1}, \ldots, v_m, t\}$, so the above polynomial has the term s^{q+1} with overwhelming probability (cf. [GGPR13]). $\mathcal{B}$ performs an evaluation as

$$h = \mathsf{Eval}\Big(\{\mathsf{E}(s^i)\}_{i\in[q+d]\setminus\{q+1\}}, \{-b_i\}_{i\in[q+d]\setminus\{q+1\}}\Big).$$

Then $b_{q+1}^{-1}(h + B_w)$ is an encoding of s^{q+1}, which is a solution to the q-PDH assumption for $q = d$. □

From these above analyses, $\mathcal{B}$ proceeds as follows.

- Target at the q-PDH problem with $q = 2d - 1$. ($\mathcal{B}$ can equally target the q-PDH assumption with $q = d$, and follow the case (*ii*).)
- $\mathcal{B}$, from its challenge $\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^q), \mathsf{E}(s^{q+2}), \ldots, \mathsf{E}(s^{2q})$, prepares inputs for adversary $\mathcal{A}$, *i.e.,* the crs. That is , $\mathcal{B}$ randomly picks $\alpha, \beta \in \mathbb{F}_\mathfrak{q}$ and computes the corresponding terms in the crs from $\mathsf{E}(1), \mathsf{E}(s), \ldots, \mathsf{E}(s^d)$ (depending on which problem $\mathcal{B}$ would target) which form a subset of the set of elements of the challenge. The elements of the first group come directly from the challenge while the elements of the second and third groups are produced by a further step of re-randomization, *i.e.,* by adding some noises from a common vector space to each encoding. (This operation could be viewed as re-randomization.) The preparation for a value β is shown as in (*ii*). All the encodings of the so-generated crs are ciphertexts sharing a common vector space of noise, however, by the IRSL$(2n, n, \mathfrak{r}, N)$ problem, the view of $\mathcal{A}$ on this input is computationally indistinguishable from input of the real protocol.
- By the contradictory assumption, $\mathcal{A}$ outputs a proof which is accepted, however, the statement is false.
- By using the extractor of the d-PKE assumption, $\mathcal{B}$ obtains the coefficients of polynomial $v(x)$ and $h(x)$.
- If (*i*) holds, $\mathcal{B}$ would find a solution for the q-PDH assumption as described above and break the q-PDH assumption for $q = 2d - 1$.
- If (*ii*) holds, $\mathcal{B}$ aborts.

We note that the distribution of the input for $\mathcal{A}$ prepared by $\mathcal{B}$ is computationally indistinguishable from that of the real scheme. Therefore, (*i*) and (*ii*) happen with equal chance. Thus, $\mathcal{B}$ can break the targeted assumption with non-negligible probability. □

6 Efficiency and Parameters

6.1 Efficiency

– A proof consists of 5 encodings, each of which is a ciphertext of the underlying RQC scheme. Therefore, the size of proof is

$$|\pi| = 10m_0 n \log \mathfrak{q}.$$

– A crs contains $m+1$ polynomials v_i's, a polynomial $t(x)$, the public parameters pp, and $(m-\ell_u+2d+3)$ encodings. Each polynomial is of degree at most d, hence needs $(d+1)m_0 \log \mathfrak{q}$ bits for its description. The size of pp is dominated by $(k+1)m_0 n \log \mathfrak{q}$. Thus, the size of crs is

$$(2d+4+k+m-\ell_u)m_0 n \log \mathfrak{q} + (m+2)(d+1)m_0 \log \mathfrak{q} = O(mm_0 d \log \mathfrak{q}).$$

6.2 Parameters

This section provides an example of parameters for the scheme. These parameters are selected to target the security level of 128 bits and soundness error of the same level.

Taking attacks in [DT18, BBBG22] into consideration, for the IRSL$(2n, n, \mathfrak{r}, N)$ problem to be at the 128 bit-level of security, m_0, n, and $\mathfrak{r}$ are chosen such that

$$m_0 > \left\lfloor \frac{m_0 n - N}{2n - \lfloor \frac{N}{\mathfrak{r}} \rfloor} \right\rfloor.$$

And to guarantee the success of decoding, $n, r, \mathfrak{r}, w$, and k are chosen such that $w(r+\mathfrak{r}) \leq \frac{n-k}{2}$. (We use the RQC version in which 1 belongs to the vector space of the secret keys.) Also, the relation between m_0 and n is always $m_0 \geq n$, since a Gabidulin code is employed. To sum up, parameters are chosen to satisfy that

$$\begin{cases} m_0 > \left\lfloor \frac{m_0 n - N}{2n - \lfloor \frac{N}{\mathfrak{r}} \rfloor} \right\rfloor, \\ w(r+\mathfrak{r}) \leq \frac{n-k}{2}, \\ n \leq m_0. \end{cases}$$

The parameter d is fixed to be $d = 2^{13}$, and $N = 4d$ is the number of given "syndromes." Recall that the size of a proof is

$$|\pi| = 10m_0 n \log \mathfrak{q},$$

so we get the result.

N	$\mathfrak{q} = \lvert\mathbb{F}\rvert$	m_0	n	k	r	$\mathfrak{r}$	w	$\lvert\pi\rvert$ (kB)
2^{15}	$\approx 2^{143}$	503	491	3	59	61	2	44147

Acknowledgement

The authors would like to thank Vietnam Institute for Advanced Study in Mathematics (VIASM) for providing a fruitful research environment and working condition. XTD was supported by the KHMM-2022-C07 project. QHV was supported in part by the French ANR projects CryptiQ (ANR-18-CE39-0015), SecNISQ (ANR-21-CE47-0014), TCS-NISQ (ANR-22-CE47-0004), and by the PEPR integrated project EPiQ ANR-22-PETQ-0007 part of Plan France 2030.

References

AAB+19. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, Gilles Zémor, Alain Couvreur, and Adrien Hauteville. RQC. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions`.

ABD+16. Carlos Aguilar, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. Cryptology ePrint Archive, Report 2016/1194, 2016.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BBBG22. Loïc Bidoux, Pierre Briaud, Maxime Bros, and Philippe Gaborit. Rqc revisited and more cryptanalysis for rank-based cryptography. *arXiv preprint arXiv:2207.01410*, 2022.

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO 2019, Part III*, LNCS 11694, pages 701–732. Springer, Heidelberg, August 2019.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, LNCS 3152, pages 41–55. Springer, Heidelberg, August 2004.

BCC+09. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO 2009*, LNCS 5677, pages 108–125. Springer, Heidelberg, August 2009.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *ACNS 16*, LNCS 9696, pages 117–136. Springer, Heidelberg, June 2016.

BCG+18. Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In *ASIACRYPT 2018, Part I*, LNCS 11272, pages 595–626. Springer, Heidelberg, December 2018.

BG90. Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *CRYPTO'89*, LNCS 435, pages 194–211. Springer, Heidelberg, August 1990.

BGM22. Olivier Blazy, Philippe Gaborit, and Dang Truong Mac. A rank metric code-based group signature scheme. In *Code-Based Cryptography*, pages 1–21, Cham, 2022. Springer International Publishing.

CKL+16. Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In *SAC 2015*, LNCS 9566, pages 3–24. Springer, Heidelberg, August 2016.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, LNCS 2045, pages 93–118. Springer, Heidelberg, May 2001.

CMS19. Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In *TCC 2019, Part II*, LNCS 11892, pages 1–29. Springer, Heidelberg, December 2019.

CS96. Florent Chabaud and Jacques Stern. The cryptographic security of the syndrome decoding problem for rank distance codes. In *ASIACRYPT'96*, LNCS 1163, pages 368–381. Springer, Heidelberg, November 1996.

Cv91. David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT'91*, LNCS 547, pages 257–265. Springer, Heidelberg, April 1991.

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

DP06. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *Progress in Cryptology - VIETCRYPT 06*, LNCS 4341, pages 193–210. Springer, Heidelberg, September 2006.

DT18. Thomas Debris-Alazard and Jean-Pierre Tillich. Two attacks on rank metric code-based schemes: RankSign and an IBE scheme. In *ASIACRYPT 2018, Part I*, LNCS 11272, pages 62–92. Springer, Heidelberg, December 2018.

FHS19. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

GGP10. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO 2010*, LNCS 6223, pages 465–482. Springer, Heidelberg, August 2010.

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

GHPT17. Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from codes with rank metric. In *CRYPTO 2017, Part III*, LNCS 10403, pages 194–224. Springer, Heidelberg, August 2017.

GMNO18. Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In *ACM CCS 2018*, pages 556–573. ACM Press, October 2018.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016, Part II*, LNCS 9666, pages 305–326. Springer, Heidelberg, May 2016.

ISW21. Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In *ACM CCS 2021*, pages 212–234. ACM Press, November 2021.

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

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. Cryptology ePrint Archive, Report 2013/121, 2013.

Mac21. Dang Truong Mac. *On Certain Types of Code-Based Signatures*. PhD thesis, University of Limoges, 2021.

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

PGHR13. Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. Cryptology ePrint Archive, Report 2013/279, 2013.