Blind zkSNARKs for Private Proof Delegation and Verifiable Computation over Encrypted Data

Mariana Gama[1], Emad Heydari Beni[1,2], Jiayi Kang[1], Jannik Spiessens[1], and Frederik Vercauteren[1]

[1] COSIC, KU Leuven, Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] Nokia Bell Labs, Antwerp, Belgium
emad.heydari_beni@nokia-bell-labs.com

Abstract. In this paper, we show for the first time it is practical to privately delegate proof generation of zkSNARKs to a single server for computations of up to 2^{20} R1CS constraints. We achieve this by computing zkSNARK proof generation over homomorphic ciphertexts, an approach we call blind zkSNARKs. We formalize the concept of blind proofs, analyze their cryptographic properties and show that the resulting blind zkSNARKs remain sound when compiled using BCS compilation. Our work follows the framework proposed by Garg et al. (Crypto'24) and improves the instantiation presented by Aranha et al. (Asiacrypt'24), which implements only the FRI subprotocol. By delegating proof generation, we are able to reduce client computation time from 10 minutes to mere seconds, while server computation time remains limited to 20 minutes. We also propose a practical construction for vCOED supporting constraint sizes four orders of magnitude larger than the current state-of-the-art verifiable FHE-based approaches. These results are achieved by optimizing Fractal for the GBFV homomorphic encryption scheme, including a novel method for making homomorphic NTT evaluation packing-friendly by computing it in two dimensions. Furthermore, we make the proofs publicly verifiable by appending a zero-knowledge Proof of Decryption (PoD). We propose a new construction for PoDs optimized for low proof generation time, exploiting modulus and ring switching in GBFV and using the Schwartz-Zippel lemma for proof batching; these techniques might be of independent interest. Finally, we implement the latter protocol in C and report on execution time and proof sizes.

Keywords: vCOED · zkDel · Blind zkSNARKs · Proofs of Decryption

1 Introduction

Zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) are undeniably one of the most promising advanced cryptographic protocols developed in the last decade. Simply put, a zkSNARK allows one to generate a proof $\pi \leftarrow \mathsf{Prove}_F(u, y)$ that some output y is the result of $F(u)$. The proof π can be verified in sublinear cost compared to the computation F itself, a property known as *succinctness*. Additionally, the proof π does not leak any private data required for (or generated by) computing F, a property known as *zero-knowledge*. Because of these unique properties, zkSNARK proofs are particularly useful in two main categories of real-world applications.

The first category is Privacy-Enhancing Technologies (PETs), where zkSNARKs and their subcomponents allow one to produce an efficiently verifiable statement of any complexity while retaining privacy. Examples of applications are anonymous credential systems [77] (for proving possession of some private credentials) and private transaction systems [10] (for proving that a private ledger has been correctly updated). Note that in these settings, it is the user who performs the costly proving operation, limiting the complexity of the statement to be proven. Solutions for this problem were proposed in [75,35,48], where multi-party computation was used to build a zero-knowledge proof delegation (zkDel) scheme. Even though this approach is more efficient than single-server delegation, it requires selecting a group of external servers who are trusted not to collude and is, therefore, harder to bootstrap in practice.

Secondly, zkSNARKs are also used to add verifiability to outsourced computations (e.g., in Cloud Computing scenarios [44,30]). The user outsources a computation to the server and then verifies a zkSNARK proof to check whether the returned output is the result of the requested computation. Since the prover cannot compute the proof without knowing the witness, the user has to reveal their inputs and outputs to the server. This can be avoided by using verifiable Computation Over Encrypted Data (vCOED). Possible constructions for vCOED were first described by Gennaro et al. [51] and have received more academic interest in the last years following the emergence of zkSNARKs. These schemes focus on proving the correct execution of homomorphic computations using proof systems and are better known as verifiable Fully Homomorphic Encryption (vFHE). Although recent works [6] have achieved leaps in performance, they struggle with arithmetizing the maintenance operations in homomorphic encryption (HE) schemes.

In [49], Garg et al. proposed a technique for constructing both zkDel and vCOED schemes by applying HE schemes to proof systems (as opposed to applying proof systems to HE schemes, as in vFHE). Generally, an HE scheme $\mathcal{E}$ allows generating a ciphertext $\mathtt{ct}[u] \leftarrow \mathsf{Enc}(u)$ and computing $\mathtt{ct}[y] \leftarrow \mathsf{Hom}_F(\mathtt{ct}[u])$, where $\mathtt{ct}[y]$ is and encryption of $y = F(u)$; in other words, it allows homomorphically computing on ciphertexts. The technique presented in [49] can be described as replacing the prover computation by $\mathtt{ct}[\pi] \leftarrow \mathsf{Hom}_{\mathsf{Prove}_F}(\mathtt{ct}[u], \mathtt{ct}[y])$, i.e., homomorphically computing the proving computation. In the zkDel setting, this allows the prover to delegate the costly proof generation to a single untrusted server. In the vCOED setting, it promises better performance than vFHE since it proves the operations in the plaintext space instead of the ciphertext space, avoiding the need to arithmetize HE schemes. In [5], Aranha et al. consider a similar approach and present a concrete instantiation of this technique for the vCOED setting. Our paper further extends this line of research; we refer to Section 3 for a thorough comparison with previous works. In particular, we answer the following fundamental questions:

1. What security and privacy guarantees do these constructions provide?
2. Can these constructions achieve practical performance?
3. Can these constructions be efficiently publicly verified?

To answer the first question, we construct a theoretical framework for a new primitive we call blind proofs by defining a blind variant of holographic Interactive Oracle Proofs (hIOPs) and zkSNARKs. We also define the completeness, zero-knowledge and soundness properties and prove that they hold for blind hIOPs. We then show that the corresponding properties also hold for blind zkSNARKs, which can be obtained from blind hIOPs by applying the usual BCS compilation [12].

To answer the second question, we provide the first complete method for computing blind zkSNARKs, which remains practical even for large computations. We optimize the Fractal zkSNARK [37] so that it can be efficiently evaluated using the recent GBFV homomorphic encryption scheme [50] and design specialized homomorphic circuits using two-dimensional NTTs.

Finally, we address the last question by showing that public verifiability can be achieved through Proofs of Decryption (PoDs). We propose a state-of-the-art construction incorporating two techniques from homomorphic encryption, modulus switching and ring switching, and propose a novel method for batching PoDs. We implement[3] our PoD in C and show that proving decryption of several thousands of ciphertexts can be done in a matter of seconds and results in a small 12KB proof.

2 Technical overview

As discussed above, instead of computing Prove_F, the prover computes $\mathsf{Hom}_{\mathsf{Prove}_F}$ to obtain a proof for valid computation of F *without seeing the values that* F *was computed on*. The blindly generated proof should then be decrypted before it can be verified. We coin these schemes *blind proofs* due to their similarity to blind signatures [29], which allow one to sign a message without revealing its content.

Consider, for instance, an interactive proof system that proves the claim $\vec{a} \odot \vec{b} = \vec{c}$ for some vectors of field elements that were previously committed to. Define polynomials f_a, f_b, f_c that interpolate these vectors over some evaluation domain H. Now, notice that a polynomial Z_H that vanishes on H, i.e. $Z_H(a) = 0 \Leftrightarrow a \in H$, will divide the polynomial $f_a \cdot f_b - f_c$ if and only if the claim holds. In other words, the original claim has been reduced to the claim that $Q \coloneqq (f_a \cdot f_b - f_c)/Z_H$ is a polynomial of bounded degree. More generally, this technique can be used to reduce claims about generic computations to low-degree tests of rational functions of polynomials, also known as rational constraints [37].

[3] https://github.com/KULeuven-COSIC/blind_zkSNARKs

Moving this proof system to the blind setting requires the prover to perform these proving steps homomorphically, starting from the HE encryptions of $\vec{a}$, $\vec{b}$ and $\vec{c}$. Akin to all HE applications, achieving practical efficiency requires tailoring the proving computation to homomorphic evaluation. In particular, it includes avoiding nonlinear computations since they require costly multiplications between ciphertexts that can be orders of magnitude more expensive than scalar multiplications depending on the specific HE scheme.

Hence, we select the Fractal proof system [37] since it reduces – using a limited number of non-linear computations – the claim of a generic computation (in the form of an R1CS constraint system) to rational constraints. The remaining proving steps, namely the low-degree tests, can be performed using the FRI [9] proof system, which is completely linear. In broad terms, FRI proves that the evaluation $Q\big|_L$ at some domain $|L| > |H|$ is sufficiently close to a Reed-Solomon (RS) code, i.e., it corresponds to a polynomial of bounded degree. Notice that if the commitments to $\vec{a}, \vec{b}, \vec{c}$ are in the form of vector commitments to $f_a\big|_L, f_b\big|_L, f_c\big|_L$, then the RS code of $Q(X)$ does not need to be explicitly computed, as each of its elements can be efficiently reconstructed from openings to those commitments (e.g. computing $Q(\ell_i)$ from $f_a(\ell_i), f_b(\ell_i)$ and $f_c(\ell_i)$). The fact that Fractal and FRI are especially suitable for the blind setting due to their linearity was first noticed by Garg et al [49].

However, homomorphic operations accumulate noise in the ciphertexts, and HE parameters have to be large enough to accommodate this noise growth. Therefore, besides avoiding non-linear operations, one should also limit the total depth of the homomorphic circuit. Take, for example, the computation of the "domain extension" from $\vec{a} = f_a\big|_H$ to $f_a\big|_L$, which is also completely linear. One usually computes the coefficients of f_a using an inverse NTT followed by an NTT over the larger domain to compute $f_a\big|_L$. Although NTTs are usually performed using a base-2 Fast Fourier Transform (FFT) to minimize the total number of operations, Aranha et al. [5] have suggested computing them in a larger base b such that the circuit depth is $\log_b(N)$ for a vector of size N. Together with several other optimizations, they were able to provide a practical construction for computing blind FRI for polynomials up to degree 2^{15}.

As a first step towards making blind zkSNARKs practical, we provide new algorithms that are even better tailored to HE evaluations. Thereby, we are able to compute blind FRI for polynomials up to degree 2^{20} in a practical estimated runtime. Crucially, we construct algorithms that can better exploit the Single Instruction Multiple Data (SIMD) capabilities that many HE schemes natively support. This allows us to encrypt a vector $\vec{a}$ of size N into N/P ciphertexts encrypting P plaintext elements each. HE operations performed on these packed ciphertexts will then correspond to element-wise operations on the packing vectors of size P. Therefore, our algorithms are not only fairly linear and correspond to low-depth circuits, they also accommodate some level of parallelism to exploit the SIMD property.

We now revisit the example of blindly computing an NTT. It is well-known that the base-2 FFT algorithm supports local element-wise operations if and only if it simultaneously performs a bit-reversal permutation. In the blind setting, such permutation would significantly increase the cost of subsequent homomorphic operations. Reversing the permutation before the next FFT invocation would also be costly since it requires swapping elements that are packed in different ciphertexts. In this paper, we describe a different approach that orders the N/P ciphertexts in two dimensions – one orthogonal to the SIMD packing and one parallel to the packed vectors – and then computes a two dimensional NTT (2D-NTT) [40]. This allows us to select an optimal homomorphic circuit that balances operation count, circuit depth and parallelism. Specifically, the NTT performed parallel to the packing is performed as a matrix-vector multiplication that uses the baby-step giant-step algorithm to minimize costly HE operations. The second NTT can be computed as a large base butterfly FFT since the permutation it causes will be orthogonal to the ciphertext packing. In Section 6.1, we show that the permutation caused by the 2D-NTT does not significantly increase the HE operation count.

Another challenge that arises when blindly computing proof systems is that they generally rely on large field sizes ($\log|\mathbb{F}| \approx 128$ to 256) for soundness and one would therefore require an HE scheme that supports large plaintext spaces. HE schemes such as BFV [23,46]/BGV [24] have the plaintext space $\frac{\mathbb{Z}[X]}{(X^n+1,p)} \cong \mathbb{Z}_p^n$ and therefore support a large packing size n, but have the disadvantage that noise growth scales linearly with p. CLPX [31], on the other hand, uses the plaintext space $\frac{\mathbb{Z}[X]}{(X^n+1,X-b)} \cong \mathbb{Z}_p$ of size $p = b^n + 1$ and therefore supports no packing. However, this allows encrypting plaintexts using lower norm polynomial representatives, thereby significantly decreasing the noise growth. We avoid this dichotomy by using a recent HE scheme named GBFV [50] that uses the generalized plaintext space $\frac{\mathbb{Z}[X]}{(\Phi_m(X),t(X))}$, where $\Phi_m(X)$ is the m-th cyclotomic polynomial and $t(X)$ is the plaintext modulus. We select parameters Φ_m and t such that the plaintext space becomes isomorphic to $\mathbb{F}_{p^2}^{\ell}$, where $p = 2^{64} - 2^{32} + 1$ is the Goldilocks prime and ℓ is the SIMD capacity. The field extension $\mathbb{F}_{p^2}$ is a popular choice in zkSNARK implementations because of its efficient arithmetic [17]. We notice that this extension is also especially suitable to the blind zkSNARK setting since it causes small noise growth. In GBFV, the SIMD packing capacity $\ell \in \{96, 192, 384, 768, 1536, 3072\}$ is smaller than the lattice dimension $\deg(\Phi_m)$. However, for our homomorphic circuit, this does not lower performance since we find the lowest operation count for $l = 384$.

Thus far, we have addressed our techniques for the homomorphic evaluation of Fractal's hIOP proof system. These techniques are covered in more detail in Section 6, where we provide an explicit construction for the Blind hIOP (BhIOP) compiled from the Fractal hIOP. Ben-Sasson et al. [12] have shown that using BCS compilation one can compile an IOP into a zkSNARK in the Random Oracle Model (ROM). However, naively applying this compilation in the blind setting would require homomorphically computing commitments and hash functions. Fortunately, we prove that it is possible to simply apply BCS compilation to BhIOPs, as previously claimed by Garg et al. and by Aranha et al. We build upon the existing theoretical framework by Holmgren [60] and Block et al. [16] to prove that any BhIOP compiled from a round-by-round knowledge sound hIOP will be round-by-round *plaintext* knowledge sound (rbr-pks), a property we define for BhIOPs. We also extend the property of *non-adaptivity*, defined by Nassar et al. [73] for IOPs, to the blind setting, and show that BCS compilation can only be applied to rbr-pks non-adaptive BhIOPs, resulting in Designated-Verifier Blind zkSNARKs (DV-BzkSNARK). To summarize, it is sufficient for the blind prover to commit to the HE ciphertext representations of the hIOP prover messages instead if the verifier is the holder of the secret key that can decrypt them.

Note that DV-BzkSNARKs are applicable to the vCOED setting. The client encrypts the inputs to some computation F using an HE scheme and sends them to the server. The server computes F homomorphically and sends the encrypted outputs and accompanying DV-BzkSNARK proof back to the client, who can then decrypt them and verify the proof. In the zkDel setting, the client is a prover that privately outsources the proving computation by sending HE ciphertexts of their private inputs (or the entire computation trace) to the server. However, the returned proof is designated-verifier and can thus only be verified by the client prover. We solve this by extending our compiler such that the secret key holder can construct a Publicly Verifiable BzkSNARK (PV-BzkSNARK) from a DV-BzkSNARK by appending decryptions along with a zero-knowledge Proof of Decryption (PoD), as suggested in [49]. By doing so, any party can verify a normal zkSNARK proof on plaintexts and subsequently check that they correspond to the ciphertexts committed to in BCS compilation – without learning the secret key that could be used to decrypt the private prover inputs.

The viability of PV-BzkSNARKs for zkDel depends on the computation cost that is required from the client prover to generate the PoDs. Therefore, we construct a prover-efficient PoD scheme specifically for this setting. In particular, we want to prove that the inherent noise in some ciphertext (c_0, c_1) is bounded by $\|v_{inh}\|_\infty < \mathcal{B}_q$, where $v_{inh} := c_0 + c_1 \cdot \mathtt{sk} - \lfloor (q/t) \cdot m \rceil$ for some message m, secret key $\mathtt{sk}$ and ciphertext modulus q. We build upon the lattice-based sigma-protocol LNP22 by Lyubashevsky et al. [68]; here, we discuss a few difficulties regarding compatability with our setting. Firstly, because LNP22 uses the Jonhsonn-Lindenstrauss lemma, it can only prove 2-norm bounds with some relaxation factor $\psi^{(L2)}$; because we intend to prove infinity norms, this factor is increased to $\sqrt{n}\psi^{(L2)}$. As a consequence we can only construct a valid proof that $\|v_{inh}\|_\infty < \mathcal{B}_q$ if $\|v_{inh}\|_\infty < \mathcal{B}_q / (\sqrt{n}\psi^{(L2)})$. We observe that in the context of HE ciphertexts this can be interpreted as a $\log(\sqrt{n}\psi^{(L2)})$-bit noise loss. Secondly, to accommodate our parameter selection with a non-power-of-two cyclotomic, we have to prove the above linear relation in vectorized form instead of proving it over the smaller power-of-two rings that are used in LNP22. Furthermore, LNP22 uses a modulus $q'' = 5 \bmod 8$ that is not optimal for efficient HE computations; we address this by modulus switching down to modulus q'' before committing to the ciphertexts in BCS compilation. Reducing the modulus is possible since in both the vCOED and zkDel settings, this modulus should only accommodate limited subsequent noise growth. It also has the added benefit of decreasing the proof size by approximately a factor of $\log(q/q'')$.

However, the LNP22-based PoDs described above would require a proving runtime of multiple hours because they have $\mathcal{O}(\mathrm{r}\, n^2)$ proving complexity, and FRI-based BzkSNARKs for constraint systems of size 2^{20} would require proving $\mathrm{r} \geq 2000$ decryptions. We can decrease proving time by using the following observation: the previous modulus switching operation permits switching to a smaller lattice dimension while retaining the same level of RLWE-based security. Therefore, we provide the prover with the required key material to ringswitch [54] down to a smaller dimension n', which accelerates the client prover computation and simultaneously decreases the proof size by a factor of n/n'. Finally, using a new technique for batching HE PoDs, we can further decrease the client prover runtime from minutes to seconds. We exploit the additive homomorphism of the HE ciphertexts $[\mathtt{ct}_i]_{i\in[\mathrm{r}]}$ to compute a random linear combination $\mathtt{ct}^* := \sum \alpha_i \mathtt{ct}_i$ that corresponds to the message $m^* := \sum \alpha_i m_i$. Using the Schwartz-Zippel lemma, we show it is sufficient to prove that $\mathtt{ct}^*$ decrypts to m^*. Thereby, we decrease the asymptotic prover runtime from $\mathcal{O}(\mathrm{r}\, n^2)$ to $\mathcal{O}(n^2 + \mathrm{r}\, n \log n)$ by computing more of the total proving computation as ring operations. On the other hand, since soundness relies on the correctness of the HE scheme for this linear combination, we require extra noise depth from the HE scheme. We can again increase efficiency by modulus switching to a modulus q'' in the PoD after performing these homomorphic operations, since this enables a subsequent ringswitch down to an even smaller lattice dimension n''.

3 Related Works

The two most relevant works are Garg et al. [49] and Aranha et al. [5]. In [49], the authors first establish FRI on hidden values leveraging linearly-homomorphic encapsulation (LHEncap), a concept more general than HE. They use it to construct a polynomial commitment scheme on hidden values, which is then combined with a polynomial IOP-based SNARK to enable verifiable private delegation of computation (corresponding to vCOED in our paper) and private outsourcing zkSNARKs to a single server (corresponding to zkDel in our paper). They also provide a construction for weighted threshold signatures without trusted setup.

In [5], the authors introduce a general transformation for IOPs to work over HE, denoted as HE-IOP (where the adaptation to holographic IOPs corresponds to the BhIOP in our paper), in the interest of building vCOED. They focus on making FRI over HE (HE-FRI) practical by proposing several optimizations, including a shallow fold algorithm that reduces the Fold operation depth from n to 1 at the cost of increasing complexity from $\mathcal{O}(2^n)$ to $\mathcal{O}(n \cdot 2^n)$ for input size 2^n. This shallow fold algorithm is also utilized in our instantiation of blind zkSNARKs with Fractal. Furthermore, the authors provide an open-source implementation. Based on this implementation, for a batch of 4096 polynomials of degree up to 2^{15}, the prover needs 207.8 seconds to run HE-FRI (including the Reed-Solomon code encoding) using 32 threads.

Below we present detailed comparisons with these two prior works.

Comparison with Garg et al. [49] Our work constructs vCOED and zkDel based on the observation by Garg et al. that Fractal and FRI can efficiently be computed on hidden values due to their linearity. Their work provided the first theoretical framework for vCOED and zkDel and discusses efficiency in asymptotic terms. We extend their work by providing a concrete compiler for constructing vCOED and zkDel, defining the required properties of the hIOP/HE scheme and proving that the resulting constructions achieve the completeness, soundness and zero-knowledge properties. Notably, their framework compiles Polynomial IOPs (PIOPs) to blind zkSNARKs by using polynomial commitment schemes on hidden values, which they construct using FRI on hidden values. Our compiler constructs blind zkSNARKs starting from Reed-Solomon hIOPs (RS-hIOPs) by using FRI directly, avoiding the need of a polynomial commitment scheme on hidden values.

We further extended their work by providing concrete algorithms for blind Fractal and a proof of decryption scheme. We also select specific parameters for Fractal, FRI and the HE scheme which allows us to gauge concrete efficiency and identify practical challenges. For example, as pointed out in [5], the assumption of *decryptable* LHEncap in [49] overlooked the challenge that the evaluation correctness of HE schemes needs to support the homomorphic operations in the blind proofs, posing a significant obstacle for noise management in practice. Our work is the first to demonstrate the feasibility of such assumptions in the context of Fractal, as shown in Table 1. In fact, our algorithm not only ensures the resulting noise remains below the decryption bound but also leaves a sufficient noise gap to enable circuit privacy (via noise flooding) in vCOED and accommodate the relaxation factor in the PoD of the zkDel scheme.

Comparison with Aranha et al. [5] Aranha et al. [5] demonstrated the feasibility of FRI in the blind setting (for polynomials of degree 2^{15}) through implementation. We extend their work by also demonstrating the practicality of blind FRI (for polynomials of degree 2^{20}) through microbenchmarking[4], i.e. deriving runtime estimates from operation counts. Furthermore, we also demonstrate the practicality of the Fractal zkSNARK in the blind setting, of which blind FRI is a subroutine. Fractal contains subprocedures that are more computationally intensive than FRI, such as computing Az, Bz, Cz in the R1CS constraint system and performing the lincheck protocol. Our work also provides improved algorithms for computing FRI in the blind setting. We list technical differences below.

[4] The GBFV scheme, which serves as the HE scheme used in our construction, has not yet been implemented for non-power-of-two cyclotomics and large plaintext spaces.

- The security requirement of FRI necessitates the use of a large field $\mathbb{F}_{p^d}$ with a size of at least 128 bits. In [5], the arithmetic of $\mathbb{F}_{p^d}$ is emulated through $\mathbb{F}_p$, where d ciphertexts are needed to encrypt a single value in $\mathbb{F}_{p^d}$, resulting in memory and efficiency overhead. In contrast, we leverage the recent GBFV [50] homomorphic encryption scheme, which natively supports the arithmetic of $\mathbb{F}_{p^d}$ while offering flexibility in balancing packing capacity P and noise growth. As shown in Sections 6 and 8, we select parameter sets that offer a sufficient number of SIMD slots to achieve a practical performance as well as sufficient noise capacity for computing blind FRI after blindly computing the Fractal hIOP.
- The NTT algorithms in [5] are used for Reed-Solomon encoding in FRI, and their implementation performs the NTT for a batch of 4096 polynomials that are encoded in the 4096 SIMD slots of BGV/BFV ciphertexts. However, if one has to perform the NTT for only $k \ll 4096$ polynomials at once, their approach will only use k slots while leaving the rest unused, resulting in higher memory usage and reduced efficiency. Note that in Fractal, one will have to perform the NTT for at most $k = 4$ polynomials at once. Therefore, we take an alternative approach that allows using all slots in a ciphertext, even when processing a single polynomial ($k = 1$). We refer to it as 2D-NTT [40], which is also known as the Map-Reduce NTT in [78] and NTT with tensorial decomposition in [71]. While the 2D-NTT has been previously explored in the context of horizontal scalability for distributed zero-knowledge proof system [80] and hardware acceleration [1,14], its compatibility with HE packing has not been studied in prior research, to the best of our knowledge.
- To verify the consistency of the messages sent by the prover, the verifier (in vCOED) needs to decrypt to obtain a subset of messages from ciphertexts. To reduce the cost of decrypting a subset, [5] employs a repacking procedure. This involves first converting slot-encoded ciphertexts into the coefficient encoding, then extracting the relevant coefficients as LWE samples, which are subsequently repacked using a key-switching algorithm [38,39]. In our work, we use different FHE techniques to reduce the decryption cost, namely reducing the lattice dimension from ring switching [54] and reducing the ciphertext modulus from modulus switching. These techniques do not require transformation to coefficient encoding, and more importantly, their combination with the slot-wise Schwartz-Zippel lemma significantly reduces cost of PoD in the zkDel setting.

Finally, our formalization of blind proofs presents theoretical improvements over [5]. This includes highlighting the necessity of non-adaptivity in BCS compilation (although to the best of our knowledge, all known hIOPs possess this property) and deriving a tighter RBR soundness error bound, where we conclusively prove that the correctness error of the HE scheme does not help dishonest provers.

Related works regarding proof of decryption. Verifiable decryption was first introduced in [26] together with verifiable encryption, and lattice-based constructions have been proposed for the BGV homomorphic encryption scheme [24] in [66,56,4,28,61] and for the Kyber key encapsulation scheme [20] in [69,68].

The work [66] addresses the special case where the plaintext modulus is 2, and the works [56,4,28,61] discuss *distributed* verifiable decryptions, where the secret key is shared among multiple parties. While [28] also employs LNP22, their construction (as well as the construction in [61]) requires committing to both the NTT and the coefficient representations of the secret key (an element in the ring $\mathcal{R}_{m,q}$ where m is a power-of-two) and uses the NTT representation to facilitate polynomial multiplications in this ring. In contrast, we commit only to the coefficient representation and perform polynomial multiplication via matrix-vector multiplication using the rotation matrix of $\mathcal{R}_{m,q}$. This not only reduces the commitment size but also removes the restriction of m being a power-of-two cyclotomic order, making it applicable to our parameters in Section 8, where $m = 7 \cdot 3 \cdot 2^{11}$.

Other methods for vCOED vFHE [64,47,6,79] is an alternative approach to achieve vCOED by applying proof systems to HE schemes. While vFHE ensures security against key-recovery attacks, the state-of-the-art vFHE constructions can only verifiably outsource computations involving hundreds of constraints in a runtime similar to our vCOED approach with 2^{20} constraints; thus, our work achieves an improvement of four orders of magnitude compared to vFHE-based approaches. The effect of key-recovery attacks can be mitigated by avoiding reusing the same HE secret key. As such, at most one bit of information about the plaintext/secret key is leaked (only when the verifier signals to the prover whether verification passes), as carefully analyzed in [5].

4 Preliminaries

4.1 Commitment scheme

We define a commitment scheme following [45,4].

Definition 1 (Commitment scheme). *A commitment scheme* $\mathcal{CT} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Open})$ *includes the following probabilistic polynomial-time algorithms:*

- $\mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$*: for a given security parameter* λ*, it returns public parameters* pp*, which define a message space* $\mathcal{S}_M$*, a randomness space* $\mathcal{S}_R$ *and a commitment space* $\mathcal{S}_C$*.*
- $\mathcal{CT}.\mathsf{Com}_{\mathit{pp}}(m, r)$*: for a given message* $m \in \mathcal{S}_M$ *and some randomness* $r \leftarrow \mathcal{S}_R$*, it returns a commitment* $\mathcal{C}_m$*.*
- $\mathcal{CT}.\mathsf{Open}_{\mathit{pp}}(m, r, \mathcal{C})$*: for a given tuple* $(m, r, \mathcal{C}) \in \mathcal{S}_M \times \mathcal{S}_R \times \mathcal{S}_C$*, it returns either* acc *or* rej*.*

Binding. *A commitment scheme $\mathcal{CT}$ is computationally binding if for any $pp \leftarrow \mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$ and probabilistic polynomial-time adversary $\mathcal{A}$, it holds that*

$$\Pr\left[m_0 \neq m_1 \;\middle|\; \begin{array}{c} (\mathcal{C}, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(pp) \\ \mathcal{CT}.\mathsf{Open}_{pp}(m_0, r_0, \mathcal{C}) = \mathsf{acc} \\ \mathcal{CT}.\mathsf{Open}_{pp}(m_1, r_1, \mathcal{C}) = \mathsf{acc} \end{array} \right] = \mathsf{negl}(\lambda).$$

Hiding. *A commitment scheme $\mathcal{CT}$ is computationally hiding if for any $pp \leftarrow \mathcal{CT}.\mathsf{KeyGen}(1^\lambda)$ and probabilistic polynomial-time adversary $\mathcal{A}$ it holds that*

$$\Big|\Pr\left[\mathcal{A}(\mathcal{C}) = 1 \;\middle|\; \mathcal{C} \leftarrow \mathcal{CT}.\mathsf{Com}_{pp}(m_0) \right] - \Pr\left[\mathcal{A}(\mathcal{C}) = 1 \;\middle|\; \mathcal{C} \leftarrow \mathcal{CT}.\mathsf{Com}_{pp}(m_1) \right]\Big| \leq \mathsf{negl}(\lambda).$$

We also define a verification oracle $\mathcal{O}^{\mathcal{CT}}(\mathcal{C}_m, m)$ which returns acc *when there exists an $r \in \mathcal{S}_R$ such that $\mathcal{CT}.\mathsf{Open}(m, r, \mathcal{C}_m) = \mathsf{acc}$ and* rej *otherwise.*

4.2 Homomorphic Encryption (HE)

We define a secret-key HE scheme with plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$ following [52,25,57,32].

Definition 2 (Homomorphic Encryption). *An HE scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ includes the following polynomial-time algorithms:*

- $\mathcal{E}.\mathsf{KeyGen}(1^\lambda)$*: given the security parameter λ, it returns a secret key sk and a public evaluation key evk.*
- $\mathcal{E}.\mathsf{Enc}_{sk}(\{m_i\}_{i\in[r]})$*: given the secret key sk and plaintexts $\{m_i\}_{i\in[r]} \in \mathcal{P}^r$, it returns ciphertexts $\{ct_i\}_{i\in[r]} \in \mathcal{C}^r$, which can also be denoted as $\{ct[m_i]\}_{i\in[r]}$.*
- $\mathcal{E}.\mathsf{Dec}_{sk}(\{ct_i\}_{i\in[r]})$*: given the secret key sk and ciphertexts $\{ct_i\}_{i\in[r]} \in \mathcal{C}^r$, it returns plaintexts $\{m_i\}_{i\in[r]} \in \mathcal{P}^r$.*
- $\mathcal{E}.\mathsf{Eval}_{evk}(f, \{ct_i\}_{i\in[\ell]})$*: given the public evaluation key evk, a function $f : \mathcal{P}^\ell \to \mathcal{P}^r$ and a set of ciphertexts $\{ct_i\}_{i\in[\ell]} \in \mathcal{C}^\ell$, it returns ciphertexts $\{ct'_i\}_{i\in[r]} \in \mathcal{C}^r$.*

CPA security. *An HE scheme $\mathcal{E}$ is IND-CPA secure if for any $(sk, evk) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, two plaintexts $m_0, m_1 \in \mathcal{P}$ and probabilistic polynomial-time adversary $\mathcal{A}$, it holds that*

$$\Big|\Pr\left[\mathcal{A}(evk, ct) = 1 \;\middle|\; ct \leftarrow \mathcal{E}.\mathsf{Enc}_{sk}(m_1) \right] - \Pr\left[\mathcal{A}(evk, ct) = 1 \;\middle|\; ct \leftarrow \mathcal{E}.\mathsf{Enc}_{sk}(m_0) \right]\Big| \leq \mathsf{negl}(\lambda).$$

Correctness. *An HE scheme $\mathcal{E}$ is correct for functions in $\mathcal{F}$ if for any $(sk, evk) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$, function $f \in \mathcal{F}$, plaintexts $\{m_i\}_{i\in[\ell]}$ and their encryptions $\{ct_i\} \leftarrow \mathcal{E}.\mathsf{Enc}_{sk}(\{m_i\})$, the relation*

$$\mathcal{E}.\mathsf{Dec}_{sk}\left(\mathcal{E}.\mathsf{Eval}_{evk}(f, ct_1, \ldots, ct_\ell)\right) = f(m_1, \ldots, m_\ell),$$

holds with probability no lower than $1 - \mathsf{negl}(\lambda)$.

Circuit Privacy. *An HE scheme* $\mathcal{E}$ *satisfies computational circuit privacy for functions in* $\mathcal{F}$ *if there exists a probabilistic polynomial-time simulator* Sim *such that for any security parameter* λ*, HE keys* $(\boldsymbol{sk}, \boldsymbol{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$*, set of plaintexts* $\{m_i\}$*, function* $f \in \mathcal{F}$ *and probabilistic polynomial-time distinguisher* D*, it holds that*

$$\Big|\Pr\left[\mathsf{D}(\boldsymbol{ct}, \boldsymbol{sk}) = 1 \;\middle|\; \boldsymbol{ct} \leftarrow \mathsf{Sim}(1^\lambda, \boldsymbol{sk}, f(\{m_i\}))\right] \\ - \Pr\left[\mathsf{D}(\boldsymbol{ct}, \boldsymbol{sk}) = 1 \;\middle|\; \begin{array}{c}\{\boldsymbol{ct}_i\} \leftarrow \mathcal{E}.\mathsf{Enc}_{\boldsymbol{sk}}(\{m_i\}) \\ \boldsymbol{ct} \leftarrow \mathcal{E}.\mathsf{Eval}_{\boldsymbol{evk}}(f, \{\boldsymbol{ct}_i\})\end{array}\right]\Big| \leq \mathsf{negl}(\lambda).$$

This property can be achieved using a technique called noise-flooding [52,18,63].

Remark. For use in Subsection 5.2, we define $\mathcal{E}.\mathsf{len}$ to be the minimal length in bits of a decryptable ciphertext. In the context of Subsection 5.3, the ciphertext should additionally still allow for an efficient proof of decryption.

4.3 Relations and languages

A relation $\mathbf{R}$ on some plaintext space $\mathcal{P}$ is a subset of $(\mathbb{x}, \mathbb{w}) \in \mathcal{P}^* \times \mathcal{P}^*$. For some relation $\mathbf{R}$ we define a language $\mathcal{L}_{\mathbf{R}} = \{\mathbb{x} \mid \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathbf{R}\}$. Similarly we define an indexed relation that is a subset of $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{P}^* \times \mathcal{P}^* \times \mathcal{P}^*$ which in turn defines a relation

$$\mathbf{R}_{\mathbb{i}} = \big\{(\mathbb{x};\, \mathbb{w}) : (\mathbb{i};\, \mathbb{x};\, \mathbb{w}) \in \mathbf{R}\big\} = \big\{(x;\, w) : Az \circ Bz = Cz \text{ for } z = (x, w)\big\},$$

and the second equality shows an example where the index $\mathbb{i}$ consists of a Rank-1 Constraint Satisfiability (R1CS) circuit defined by the matrices A, B and C.

4.4 Zero-knowledge Succinct Non-interactive ARgument of Knowledge (zkSNARK)

We will define pre-processing zkSNARKs in the Random Oracle Model (ROM) following Chiesa et al. [37]. Let us denote with $\mathcal{U}(\lambda)$ the uniform distribution over all functions $\mathcal{P}^* \to \{0, 1\}^\lambda$. A function $\rho \leftarrow \mathcal{U}(\lambda)$ is referred to as a Random Oracle (RO). We denote an algorithm A having oracle access to some object x as $A^{[\![x]\!]}$.

Definition 3 (preprocessing zkSNARK in the ROM). *A preprocessing zkSNARK* $\Pi = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ *is a non-interactive proof system for some indexed relation* $\mathbf{R}$ *that includes the following polynomial-time algorithm:*

– $\Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathbb{i})$*: for a given index* $\mathbb{i}$*, using access to the RO* ρ*, it returns the index keys* $(\boldsymbol{ipk}, \boldsymbol{ivk})$*.*

and the following probabilistic polynomial-time algorithms:

– $\Pi.\mathsf{P}^{[\![\rho]\!]}(\boldsymbol{ipk}, \mathbb{x}, \mathbb{w})$*: for a given index prover key* $\boldsymbol{ipk}$*, statement* $\mathbb{x}$ *and witness* $\mathbb{w}$*, using access to the RO* ρ*, it returns a proof* π*.*
– $\Pi.\mathsf{V}^{[\![\rho]\!]}(\boldsymbol{ivk}, \mathbb{x}, \pi)$*: for a given index verifier key* $\boldsymbol{ivk}$*, statement* $\mathbb{x}$ *and proof* π*, using oracle access to the RO* ρ*, it returns either* acc *or* rej*.*

A zkSNARK should satisfy the following properties.

Completeness. *For any* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$ *and* $\rho \leftarrow \mathcal{U}(\lambda)$ *it holds that*

$$\Pr\left[\Pi.\mathsf{V}^{[\![\rho]\!]}(\boldsymbol{ivk}, \mathbb{x}, \pi) \neq \mathsf{acc} \;\middle|\; \pi \leftarrow \Pi.\mathsf{P}^{[\![\rho]\!]}(\boldsymbol{ipk}, \mathbb{x}, \mathbb{w})\right] \leq \delta$$

where δ *is the completeness error and* $(\boldsymbol{ipk}, \boldsymbol{ivk}) = \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathbb{i})$

Zero-knowledge. *For any* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$ *and* $\rho \leftarrow \mathcal{U}(\lambda)$*, if there exists a probabilistic polynomial-time simulator* Sim *such that for any unbounded distinguisher* D *it holds that*

$$\Big|\Pr\left[\mathsf{D}^{[\![\rho[\mu]]\!]}(\pi) = 1 \;\middle|\; (\mu, \pi) \leftarrow \mathsf{Sim}^{[\![\rho]\!]}(\mathbb{i}, \mathbb{x})\right] - \Pr\left[\mathsf{D}^{[\![\rho[\mu]]\!]}(\pi) = 1 \;\middle|\; \pi \leftarrow \Pi.\mathsf{P}^{[\![\rho]\!]}(\boldsymbol{ipk}, \mathbb{x}, \mathbb{w})\right]\Big| \leq z$$

where $(\boldsymbol{ipk}, \boldsymbol{ivk}) = \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathbb{i})$ *and* $\rho[\mu]$ *equals* $\mu(x)$ *if* μ *is defined on* x *and otherwise equals* $\rho(x)$*, then* Π *has* z*-statistical zero-knowledge. If* D *is probabilistic polynomial-time then* Π *has* z*-computational zero-knowledge.*

Soundness. *For any index* $\mathbb{i}$*, statement* $\mathbb{x} \notin \mathcal{L}_{\mathbf{R}_{\mathbb{i}}}$*, RO* $\rho \leftarrow \mathcal{U}(\lambda)$*, index keys* $(\boldsymbol{ipk}, \boldsymbol{ivk}) = \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathbb{i})$ *and prover* P^* *it holds that*

$$\Pr\left[\Pi.\mathsf{V}^{[\![\rho]\!]}(\boldsymbol{ivk}, \mathbb{x}, \pi) = \mathsf{acc} \;\middle|\; \pi \leftarrow \mathsf{P}^{*[\![\rho]\!]}\right] \leq \boldsymbol{\varepsilon}$$

where $\boldsymbol{\varepsilon}$ *is the soundness error.*

Knowledge soundness. *For any index* $\mathbb{i}$*, statement* $\mathbb{x}$*, index keys* $(\boldsymbol{ipk}, \boldsymbol{ivk}) \leftarrow \Pi.\mathsf{Ind}^{[\![\rho]\!]}(\mathbb{i})$ *and prover* P^* *there exists a polynomial-time extractor* Ext *such that*

$$\Pr\left[(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_{\mathbb{i}} \;\middle|\; \mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(\mathbb{i}, \mathbb{x})\right] \geq \Pr\left[\Pi.\mathsf{V}^{[\![\rho]\!]}(\boldsymbol{ivk}, \mathbb{x}, \pi) = \mathsf{acc} \;\middle|\; \pi \leftarrow \mathsf{P}^{*[\![\rho]\!]}\right] - \boldsymbol{\varepsilon}_{\mathsf{k}}$$

where $\boldsymbol{\varepsilon}_{\mathsf{k}}$ *is the knowledge error and* $\mathsf{Ext}^{\mathsf{P}^*}$ *may interact with* P^* *by rewinding it in a black-box manner.*

Remark. Some authors [15,37] define stronger adaptive versions of these properties. For example in knowledge soundness they have the prover P^* choose the index $\mathbb{i}$ and statement $\mathbb{x}$. Although it is possible to define all these and the following properties adaptively, for ease of notation, we will refrain.

Remark. Note that this definition of zkSNARKs has no restriction of proof length or verifier cost and is therefore not necessarily succinct. However, instead of using a different name such as Non-interactive Random Oracle Proof (NIROP) [12], we follow Chiesa et al. [36] and use the popularized term zkSNARK.

4.5 Interactive Oracle Proofs (IOPs)

First introduced by Ben-Sasson et al. [12], an Interactive Oracle Proof (IOP) is a form of interactive proof where the prover sends $\mu + 1$ messages in the form of oracles $[\![m_i]\!]$ to proof strings $m_i \in \mathcal{P}^*$ and the verifier responds with some challenges $c_i \in \mathcal{C}h_i \subseteq \mathcal{P}^*$. They can be seen as a μ-round generalization of Probabilistically Checkable Proofs (PCPs). For $i \in [\mu]$, we define the concatenation $m_1\|c_1\|\dots\|m_i\|c_i$ as an i-round partial transcript and $m_1\|c_1\|\dots\|m_\mu\|c_\mu\|m_{\mu+1}$ as a full transcript. An holographic IOP is an extension where an encoding of some index $\mathbb{i}$ is generated in a preprocessing step for oracle access to the verifier [37].

Definition 4 (holographic IOP (hIOP)). *An hIOP* $\Pi = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ *is a* μ*-round interactive proof system for some indexed relation* $\mathbf{R}$ *that includes the following polynomial-time algorithm:*

- $\Pi.\mathsf{Ind}(\mathbb{i})$*: for a given index* $\mathbb{i}$*, it returns the encoding of the index* $\mathsf{e}[\mathbb{i}]$*.*

and the following probabilistic polynomial-time algorithms:

- $\Pi.\mathsf{P}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w})$*: for a given index encoding* $\mathsf{e}[\mathbb{i}]$*, statement* $\mathbb{x}$ *and witness* $\mathbb{w}$ *for the relation* $\mathbf{R}_{\mathbb{i}}$*, it returns a round message*

$$m_i \leftarrow \Pi.\mathsf{P}_i(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w}, \boldsymbol{tr})$$

in round $i \in [\mu+1]$ *where* $\boldsymbol{tr}$ *is the current* $(i-1)$*-round partial transcript.*
- $\Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!],[\![\boldsymbol{tr}]\!]}(\mathbb{x})$*: for a given statement* $\mathbb{x}$*, using oracle access to the encoded index* $\mathsf{e}[\mathbb{i}]$ *and the current partial transcript* $\boldsymbol{tr} = m_1\|\dots\|c_{i-1}\|m_i$ *to obtain queries* $\{\mathsf{e}[\mathbb{i}]_i\}, \{\boldsymbol{tr}_i\}$*, it returns a round challenge*

$$c_i \leftarrow \Pi.\mathsf{V}_i(\{\mathsf{e}[\mathbb{i}]_i\}, \mathbb{x}, \{\boldsymbol{tr}_i\})$$

when $i \in [\mu]$ *and returns either* acc *or* rej *when* $i = \mu + 1$*.*

We also define a function qr *that maps a query index to its corresponding message index. The hIOP is public-coin if all messages sent by the verifier are random elements of some subset of the plaintext space independent of the current partial transcript. Without loss of generality, we can assume that a public-coin verifier performs all queries after receiving the final prover's message. An hIOP should satisfy the following properties.*

Completeness. *For any* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$ *it holds that*

$$\Pr\left[\Pi.\left\langle \mathsf{P}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w}), \mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]}(\mathbb{x})\right\rangle \neq \mathsf{acc}\right] \leq \delta$$

where δ *is the completeness error,* $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$ *and the bracket notation* $\langle A, B\rangle$ *represents the output of* $B^{[\![\boldsymbol{tr}]\!]}$ *where* $\boldsymbol{tr}$ *is a full transcript resulting from interaction with* A*.*

Honest-verifier zero-knowledge. *For any* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$*, if there exists a probabilistic polynomial-time simulator* Sim *such that for any unbounded distinguisher* D *it holds that*

$$\Big|\Pr\left[\mathsf{D}(\mathbb{i}, \pi) = 1 \;\middle|\; \pi \leftarrow \mathsf{Sim}(\mathbb{i}, \mathbb{x})\right] - \Pr\left[\mathsf{D}(\mathbb{i}, \pi) = 1 \;\middle|\; \pi \leftarrow \mathsf{View}\left\langle \Pi.\mathsf{P}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w}), \Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]}(\mathbb{x})\right\rangle\right]\Big| \leq z$$

where $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$ *and* $\mathsf{View}\langle\cdot\rangle$ *is a random variable that contains all the query responses the verifier receives during the protocol along with the verifier's randomness, then* Π *has* z*-statistical zero-knowledge. If* D *is probabilistic polynomial-time then* Π *has* z*-computational zero-knowledge.*

Soundness. *For any index* $\mathbb{i}$*, statement* $\mathbb{x} \notin \mathcal{L}_{\mathbf{R}_\mathbb{i}}$ *and unbounded prover* P^* *it holds that*

$$\Pr\left[\left\langle \mathsf{P}^*(\mathsf{e}[\mathbb{i}], \mathbb{x}), \Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]}(\mathbb{x})\right\rangle = \mathsf{acc}\right] \leq \boldsymbol{\varepsilon}$$

where $\boldsymbol{\varepsilon}$ *is the soundness error and* $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$*.*

Knowledge soundness. *For any index* $\mathbb{i}$*, statement* $\mathbb{x}$ *and unbounded prover* P^* *there exists a polynomial-time extractor* Ext *such that*

$$\Pr\left[(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_\mathbb{i} \mid \mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(\mathbb{i}, \mathbb{x})\right] \geq \Pr\left[\left\langle \mathsf{P}^*, \Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]}(\mathbb{x})\right\rangle = \mathsf{acc}\right] - \boldsymbol{\varepsilon}_\mathsf{k}$$

where $\boldsymbol{\varepsilon}_\mathsf{k}$ *is the knowledge error and* $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$*. Note that* $\mathsf{Ext}^{\mathsf{P}^*}$ *may interact with* P^* *by rewinding it in a black-box manner.*

The soundness and knowledge soundness properties ensure the security of the hIOP scheme. Respectively, they guarantee (except with some small error) that a prover interacting with a verifier cannot result in acc for a statement that has no valid witness or for which the valid witness is not known. Note that knowledge soundness thus implies normal soundness. However, since hIOPs are compiled into non-interactive proofs [12], their security is best described round-by-round [27,36]. Following Holmgren [60] and Block et al. [16], we will define round-by-round soundness using a doomed set and round-by-round knowledge soundness using a knowledge doomed set.

Definition 5 (Doomed set). *Given a public-coin holographic hIOP* Π *that proves an indexed relation* $\mathbf{R}$*, a doomed set* $\mathcal{D}^\Pi$ *for index* $\mathbb{i}$ *and error* $\boldsymbol{\varepsilon}$ *is a set that satisfies the following properties:*

1. *For any statement* $\mathbb{x}$*, if* $\mathbb{x} \notin \mathcal{L}_{\mathbf{R}_\mathbb{i}}$ *then* $(\mathbb{x}, \emptyset) \in \mathcal{D}^\Pi$*.*
2. *For any* $(\mathbb{x}, \boldsymbol{tr}) \in \mathcal{D}^\Pi$ *where* $\boldsymbol{tr}$ *is a* $(i-1)$*-round partial transcript for* $i \in [\mu + 1]$ *and any next prover message* m_i*, it holds that*
$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}, \boldsymbol{tr} \| m_i \| c_i) \notin \mathcal{D}^\Pi] \leq \boldsymbol{\varepsilon}.$$
3. *For any full transcript* $\boldsymbol{tr}$*, if* $(\mathbb{x}, \boldsymbol{tr}) \in \mathcal{D}^\Pi$ *then* $\Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!],[\![\boldsymbol{tr}]\!]}(\mathbb{x}) = \mathsf{rej}$*.*

Definition 6 (Knowledge doomed set). *Given a public-coin holographic hIOP Π that proves an indexed relation $\mathbf{R}$, a knowledge doomed set $\mathcal{D}_{\mathsf{k}}^{\Pi}$ for index $\mathbb{i}$ and error ε_{k} is a set for which there exists a polynomial-time extractor Ext such that the following properties are satisfied.*

1. *For any statement $\mathbb{x}$, it holds that $(\mathbb{x}, \emptyset) \in \mathcal{D}_{\mathsf{k}}^{\Pi}$.*
2. *For any $(\mathbb{x}, \boldsymbol{tr}) \in \mathcal{D}_{\mathsf{k}}^{\Pi}$ where $\boldsymbol{tr}$ is a $(i-1)$-round (partial) transcript $\boldsymbol{tr}$ for $i \in [\mu+1]$ and next prover message m_i, it holds that if*

$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}, \boldsymbol{tr} \| m_i \| c_i) \notin \mathcal{D}_{\mathsf{k}}^{\Pi}] > \varepsilon_{\mathsf{k}}.$$

   then $\mathbb{w} \leftarrow \mathsf{Ext}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \boldsymbol{tr} \| m_i)$ such that $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_{\mathbb{i}}$.
3. *For any full transcript $\boldsymbol{tr}$, if $(\mathbb{x}, \boldsymbol{tr}) \in \mathcal{D}_{\mathsf{k}}^{\Pi}$ then $\Pi.\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!],[\![\boldsymbol{tr}]\!]}(\mathbb{x}) = \mathsf{rej}$.*

Remark. In property 2 in Definition 5 and 6 we slightly abuse notation by having $c_{\mu+1} \leftarrow \mathcal{C}h_{\mu+1}$ denote the public-coin verifier's additional randomness used in the final verification check.

Definition 7 (Round-by-round (knowledge) soundness). *An hIOP Π for the indexed relation $\mathbf{R}$ is round-by-round sound with error ε or round-by-round knowledge sound with error ε_{k} if for every index $\mathbb{i}$ there exists a doomed set $\mathcal{D}^{\Pi}$ for error ε or knowledge doomed set $\mathcal{D}_{\mathsf{k}}^{\Pi}$ for error ε_{k} respectively.*

Along with the definition of IOPs, Ben-Sasson et al. [12] additionally introduced BCS compilation which compiles an IOP into a zkSNARK in the ROM. Later it was extended to hIOPs [37], round-by-round soundness notions [27], the quantum ROM [36] and recently proven unconditionally UC-secure in the ROM [34]. Many of the zkSNARKs that are deployed in practice are constructed using this compilation. We defer a high-level description of this compilaiton to Section 5.2 and describe its properties in Theorem 1. We define two complexity measures for hIOPs. The proof length $p = \sum_{i=1}^{\mu+1} |m_i|$ is the sum of the lengths of all prover messages and the query complexity q is the number of queries performed by the verifier.

Theorem 1 (BCS compiler [12,36]). *Any hIOP Π for indexed relation $\mathbf{R}$ with completeness error δ, proof length p, query complexity q, round-by-round soundness error ε, round-by-round knowledge soundness error ε_{k} and z-statistical honest-verifier zero-knowledge can be compiled into a zkSNARK $\amalg$ in the ROM with RO query bound Q, security parameter λ and:*

- *Completeness error δ,*
- *Proof length p' upper bound by $\lambda(\mu + 1 + \sum_{j=1}^{q}(3 + \lceil \log_2 |m_{\mathsf{qr}(j)}| \rceil))$,*
- *Soundness error ε' where $\varepsilon' = Q\varepsilon + 3(Q^2+1)2^{-\lambda}$,*
- *Knowledge soundness error $\varepsilon_{\mathsf{k}}'$ where $\varepsilon_{\mathsf{k}}' = Q\varepsilon_{\mathsf{k}} + 3(Q^2+1)2^{-\lambda}$,*
- *z'-Statistical honest-verifier zero-knowledge where $z' = z + p2^{-\lambda/4+2}$,*

where m_i *is* $\Pi.\mathsf{P}$*'s* i*th message and* $|\cdot|$ *denotes length in* λ *bits rounded up. Both soundness and knowledge soundness error are* $\Theta(Q\varepsilon)$ *and* $\Theta(Q\varepsilon_{\mathsf{k}})$ *respectively when considering quantum adversaries that perform no more than* $Q - \mathcal{O}(q \log p)$ *RO queries.*

Remark. Technically, all these error values, proof lengths, etc. can be functions of both the statement and the index but let us disregard that here since it has no influence on what follows.

5 Blind Proofs

In this section, we introduce a new type of proof system called blind proofs, where one proves that some encrypted statement is in the language of a blind relation $\mathcal{E}[\mathbf{R}]$ with respect to some commitment $\mathtt{C}_{\mathtt{sk}}$ to a secret key $\mathtt{sk}$. This blind relation represents ciphertexts of statement-witness pairs such that the underlying plaintexts are in the relation $\mathbf{R}$. In other words, blind proofs allow the prover to generate a proof using $(\mathtt{ct}[x], \mathtt{ct}[w])$ – without knowledge of the plaintext (x, w) – that proves plaintext knowledge of (x, w) such that $x \in \mathcal{L}_{\mathbf{R}}$ for the holder of the secret key $\mathtt{sk}$ committed to in $\mathtt{C}_{\mathtt{sk}}$. We start by defining a blind relation.

Definition 8 (Blind relation). *For a given HE scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *and commitment scheme* $\mathcal{CT}$ *with security parameter* λ*, and indexed relation* $\mathbf{R}$ *we define the indexed blind relation*

$$\mathcal{E}[\mathbf{R}] = \left\{ \begin{array}{c} (\mathbb{i}; \mathbb{x}; \mathbb{w}) = (\mathbb{i};\ \mathcal{C}_{sk}, ct[x];\ ct[w]) : \\ \mathcal{E}.\mathsf{Dec}_{sk}((ct[x], ct[w])) = (x, w) \in \mathbf{R}_{\mathbb{i}} \wedge \\ sk \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^{\lambda}) \ \wedge\ \mathcal{O}^{\mathcal{CT}}(\mathcal{C}_{sk}, sk) = \mathsf{acc} \end{array} \right\}$$

which defines the blind relation $\mathcal{E}[\mathbf{R}_{\mathbb{i}}] = \{(\mathbb{x};\ \mathbb{w}) : (\mathbb{i};\ \mathbb{x};\ \mathbb{w}) \in \mathcal{E}[\mathbf{R}]\}$.

Theorem 2. *Any probabilistic polynomial-time adversary has only negligible advantage in distinguishing the underlying* $(x, w) \in \mathbf{R}_{\mathbb{i}}$ *given the corresponding* $(\mathbb{x}, \mathbb{w}) \in \mathcal{E}[\mathbf{R}_{\mathbb{i}}]$ *if* $\mathcal{E}$ *is an IND-CPA secure HE scheme and* Com *is computationally hiding.*

Proof. This follows from a standard hybrid argument. Let us define an adversary $\mathcal{A}$ in the game of distinguishing elements of a blind relation, which we denote as game G_0. Now let us define game G_1 where the LR oracle responds with a randomly sampled $\mathtt{C}_{\mathtt{sk}}$ instead of a commitment to the used secret key. The advantage of $\mathcal{A}$ in G_1 should be negligible because $\mathcal{E}$ is IND-CPA secure and the difference between G_0 and G_1 should be negligible because Com is hiding. Therefore we can conclude that $\mathcal{A}$ has negligible advantage in game G_0. □

5.1 Blind hIOP (BhIOP)

We define a blind version of the hIOP proof system introduced in Section 4.5.

Definition 9 (Blind hIOP (BhIOP)). *For a given HE scheme $\mathcal{E}$, commitment scheme $\mathcal{CT}$ and hIOP Π for indexed relation $\mathbf{R}$, a blind hIOP $\mathcal{E}[\Pi] = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ includes the following probabilistic polynomial-time algorithms:*

– $\mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathbb{i})$*: for a given index $\mathbb{i}$ and security parameter λ, it returns the encoding $\mathsf{e}[\mathbb{i}] = \Pi.\mathsf{Ind}(\mathbb{i})$, the keys $(\boldsymbol{sk}, \boldsymbol{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and the commitment $\mathcal{C}_{\boldsymbol{sk}} \leftarrow \mathcal{CT}.\mathsf{Com}(\boldsymbol{sk})$.*

– $\mathcal{E}[\Pi].\mathsf{P}_{\boldsymbol{evk}}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w})$*: for a given statement $\mathbb{x} = (\mathcal{C}_{\boldsymbol{sk}}, \boldsymbol{ct}[x])$ and witness $\mathbb{w} = \boldsymbol{ct}[w]$ of the blind relation $\mathcal{E}[\mathbf{R}_{\mathbb{i}}]$, and evaluation key $\boldsymbol{evk}$, it returns a round message*

$$\mathcal{E}.\mathsf{Eval}_{\boldsymbol{evk}}(\Pi.\mathsf{P}_i, (\mathsf{e}[\mathbb{i}], \boldsymbol{ct}[x], \boldsymbol{ct}[w], \boldsymbol{tr}'))$$

in round $i \in [\mu+1]$ where $\boldsymbol{tr}'$ is the current $(i-1)$-round partial transcript.

– $\mathcal{E}[\Pi].\mathsf{V}_{\boldsymbol{sk}}^{[\![\mathsf{e}[\mathbb{i}]]\!][\![\boldsymbol{tr}']\!]\mathcal{O}^{\mathcal{CT}}}(\mathbb{x})$*: for a given statement $\mathbb{x} = (\mathcal{C}_{\boldsymbol{sk}}, \boldsymbol{ct}[x])$ and secret key $\boldsymbol{sk}$, using oracle access to $\mathsf{e}[\mathbb{i}]$ and the current (partial) transcript $\boldsymbol{tr}'$ to obtain queries $\{\mathsf{e}[\mathbb{i}]_i\}, \{\boldsymbol{tr}'_i\}$, it returns*

$$\Pi.\mathsf{V}_i(\{\mathsf{e}[\mathbb{i}]_i\}, \mathcal{E}.\mathsf{Dec}_{\boldsymbol{sk}}(\boldsymbol{ct}[x], \{\boldsymbol{tr}'_i\}))$$

in round $i \in [\mu+1]$ if $\mathcal{O}^{\mathcal{CT}}(\mathcal{C}_{\boldsymbol{sk}}, \boldsymbol{sk})$ returns acc*, otherwise it returns* rej.

Remark. The $\mathcal{O}^{\mathcal{CT}}$ oracle of a blind hIOP verifier is instantiated in Theorem 9, similar to how the oracles of an hIOP verifier are instantiated in the compilation of Theorem 1.

A blind hIOP can be public-coin similar to an hIOP. It should satisfy the completeness and soundness properties as defined in Definition 4. Additionally, it should satisfy the following properties.

Plaintext knowledge soundness. *For any index $\mathbb{i}$, statement $\mathbb{x}$, setup $(\mathsf{e}[\mathbb{i}], \boldsymbol{sk}, \boldsymbol{evk}, \mathcal{C}_{\boldsymbol{sk}}) \leftarrow \mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathbb{i})$ and unbounded prover $\mathsf{P}^*_{\boldsymbol{evk}}$ there exists a polynomial-time extractor $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ with access to a decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ such that*

$$\Pr\left[(x, w) \in \mathbf{R}_{\mathbb{i}} \;\middle|\; (x, w) \leftarrow \mathsf{Ext}^{\mathsf{P}^*, \mathcal{O}_{\mathsf{Dec}}}(\mathbb{i}, \mathbb{x})\right] \geq \Pr\left[\langle \mathsf{P}^*_{\boldsymbol{evk}}, \mathcal{E}[\Pi].\mathsf{V}_{\boldsymbol{sk}}^{[\![\mathsf{e}[\mathbb{i}]]\!]\mathcal{O}^{\mathcal{CT}}}(\mathbb{x})\rangle = \mathsf{acc}\right] - \varepsilon_{\mathsf{k}}$$

where ε_{k} is the knowledge error. Note that $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ may interact with P^* by rewinding it in a black-box manner. Similar to the plaintext scenario it is possible to define a round-by-round variant (see Definition 10).

Honest-verifier zero-knowledge. *For any security parameter λ, $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{E}[\mathbf{R}]$ and $(\mathsf{e}[\mathbb{i}], \boldsymbol{sk}, \boldsymbol{evk}, \mathcal{C}_{\boldsymbol{sk}}) \leftarrow \mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathbb{i})$ such that $\mathbb{x} = (\mathcal{C}_{\boldsymbol{sk}}, \boldsymbol{ct}[x])$, if there exists a probabilistic polynomial-time simulator Sim such that for any unbounded distinguisher D it holds that*

$$\Big|\Pr\left[\mathsf{D}(\mathbb{i}, \pi, \boldsymbol{sk}) = 1 \;\middle|\; \pi \leftarrow \mathsf{Sim}(1^\lambda, \mathbb{i}, \mathbb{x}, \boldsymbol{sk})\right] - \Pr\left[\mathsf{D}(\mathbb{i}, \pi, \boldsymbol{sk}) = 1 \;\middle|\; \pi \leftarrow \mathsf{View}\left\langle \mathcal{E}[\Pi].\mathsf{P}_{\boldsymbol{evk}}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w}), \mathcal{E}[\Pi].\mathsf{V}_{\boldsymbol{sk}}^{[\![\mathsf{e}[\mathbb{i}]]\!]\mathcal{O}^{\mathcal{CT}}}(\mathbb{x})\right\rangle\right]\Big| \leq z$$

then $\mathcal{E}[\Pi]$ has z-statistical zero-knowledge. If D is probabilistic polynomial-time then $\mathcal{E}[\Pi]$ has z-computational zero-knowledge.

Definition 9 describes how blind hIOPs can be constructed from hIOP and HE schemes. From this contruction, one can show that the resulting blind hIOP satisfies the necessary properties.

Theorem 3. *For security parameter λ, an HE scheme $\mathcal{E}$ and a δ-complete hIOP scheme Π for indexed relation $\mathbf{R}$, the blind hIOP $\mathcal{E}[\Pi]$ is complete with completeness error $\delta + \mathsf{negl}(\lambda)$ for indexed blind relation $\mathcal{E}[\mathbf{R}]$ if $\mathcal{E}$ is correct for the homomorphic circuit $\mathcal{E}[\Pi].\mathsf{P}$.*

Proof. For any $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{E}[\mathbf{R}]$ and $(\mathsf{e}[\mathbb{i}], \mathtt{sk}, \mathtt{evk}, \mathsf{C}_{\mathtt{sk}}) \leftarrow \mathcal{E}[\Pi].\mathsf{Setup}(1^\lambda, \mathbb{i})$ such that $\mathbb{x} = (\mathtt{ct}[x], \mathsf{C}_{\mathtt{sk}})$, it holds that $(x, w) \in \mathbf{R}_{\mathbb{i}}$ for $(x, w) = \mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}[x], \mathbb{w})$. Let E be the event that

$$\mathcal{E}[\Pi].\langle \mathsf{P}_{\mathtt{evk}}(\mathsf{e}[\mathbb{i}], \mathbb{w}), \mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!]\mathcal{O}^{\mathcal{CT}}} \rangle(\mathbb{x}) \neq \mathsf{acc}$$

and E' the event that

$$\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathcal{E}.\mathsf{Eval}_{\mathtt{evk}}(\Pi.\mathsf{P}, (\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathbb{w}))) = \Pi.\mathsf{P}(\mathsf{e}[\mathbb{i}], x, w)$$

over the randomness in both prover and verifier. Then we can show that

$$\begin{aligned} \Pr[E] &= \Pr[E \mid E'] \Pr[E'] + \Pr[E \mid \neg E'] \Pr[\neg E'] \\ &\leq \Pr[E \mid E'] + \Pr[\neg E'] \leq \delta + \mathsf{negl}(\lambda) \end{aligned}$$

since $\Pr[E \mid E']$ represents the completeness error in the corresponding Π and $\Pr[\neg E']$ is determined by the correctness of the HE scheme. $\square$

Notice that zero-knowledge has been defined differently from the hIOP case. Informally, an hIOP is zero-knowledge if some simulator Sim can simulate everything the verifier sees without knowledge of the witness. This is formalized by stating no distinguisher algorithm D has an advantage in distinguishing the simulation from a valid prover output. Therefore, since for blind hIOPs the verifier has knowledge of the secret key $\mathtt{sk}$, this will also be given as an input to D. We show that blind hIOPs can retain zero-knowledge by using circuit private HE schemes.

Theorem 4. *For an HE scheme $\mathcal{E}$ with security parameter λ and a z-computational honest-verifier zero-knowledge hIOP scheme Π for indexed relation R, the blind hIOP $\mathcal{E}[\Pi]$ is $z + \mathsf{negl}(\lambda)$-computational honest-verifier zero-knowledge if $\mathcal{E}$ is circuit-private for the homomorphic circuit $\mathcal{E}[\Pi].\mathsf{P}$.*

Proof. The simulator for the $\mathcal{E}[\Pi]$ scheme can be constructed by combining the simulator for the Π scheme and the simulator for circuit privacy in the $\mathcal{E}$ scheme. More concretely, $\mathcal{E}[\Pi].\mathsf{Sim}$ uses $\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}$ to compute the statement for $\mathbf{R}_{\mathbb{i}}$. Then, it uses $\Pi.\mathsf{Sim}$ to sample some queries $\{q_i\}$ and lastly uses $\mathcal{E}.\mathsf{Sim}$ to generate the ciphertexts $\{\mathtt{ct}[q_i]\}$. The theorem follows from a standard hybrid argument relying on the circuit privacy of $\mathcal{E}$ and the honest-verifier zero-knowledge property of the Π scheme. $\square$

We additionally define honest-verifier zero-knowledge *in the decryption oracle setting*. A blind hIOP with this property satisfies honest-verifier zero-knowledge with a distinguisher D that has access to $\mathcal{O}_{\mathsf{Dec}}$ instead of $\mathtt{sk}$. Clearly, this property is a more relaxed form of zero-knowledge since D can not see the noise in ciphertexts. This setting will however be sufficient when the blind hIOP verifier's access to $\mathtt{sk}$ is also replaced by access to $\mathcal{O}_{\mathsf{Dec}}$. Such verifier corresponds to the zkDel setting as described in Section 1 and will be used later in Theorem 9. We show that zero-knowledge in this setting is achieved trivially since the HE ciphertexts are hiding.

Theorem 5. *For an IND-CPA secure HE scheme $\mathcal{E}$ for security parameter λ and a z-computational honest-verifier zero-knowledge hIOP scheme Π for indexed relation R, the blind hIOP $\mathcal{E}[\Pi]$ is $z+\mathsf{negl}(\lambda)$-computational honest-verifier zero-knowledge* in the decryption oracle setting.

Proof sketch. The simulator $\mathcal{E}[\Pi].\mathsf{Sim}$ is constructed similar to the simulator in Theorem 4 except that it uses $\mathcal{E}.\mathsf{Enc}_{\mathtt{sk}}$ to encrypt the queries $\{q_i\}$. □

One can derive the (plaintext knowledge) soundness of a blind hIOP from the (knowledge) soundness of the underlying hIOP and the correctness of the HE scheme. We discuss only the round-by-round variants since these are relevant for the BCS compilation.

Theorem 6. *The blind hIOP $\mathcal{E}[\Pi]$ is round-by-round sound for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ with error ε if the hIOP Π is round-by-round sound for the indexed relation $\mathbf{R}$ with error ε.*

Proof. By the definition of round-by-round soundness, it is sufficient to show the existence of a doomed set $D' = \mathcal{D}^{\mathcal{E}[\Pi]}$ given the existence of the doomed set $D = \mathcal{D}^{\Pi}$. We construct a doomed set D' as follows: it contains all possible HE ciphertexts that decrypt to some element in D under the secret key $\mathtt{sk}$ corresponding to $\mathtt{C}_{\mathtt{sk}}$.

$$D' = \left\{ \begin{array}{c} (\mathbb{x}', \mathtt{tr}') = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x] \| \mathtt{ct}[m_1] \| c_1 \| \dots \| \mathtt{ct}[m_n]) : \\ 0 \leq n \leq \mu + 1 \ \wedge\ \mathcal{O}^{\mathcal{CT}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{sk}) = \mathsf{acc} \\ \wedge\, \exists x, m_1, \dots, m_n, \mathtt{sk} : (x, m_1, c_1, \dots, m_n) \in D \\ \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{ct}[m_1], \dots, \mathtt{ct}[m_n])) = (x, m_1, \dots, m_n) \end{array} \right\}$$

We prove that this set satisfies all properties of a doomed set for any index $\mathbb{i}$.

1. If $\mathbb{x}' = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x]) \notin \mathcal{L}_{\mathcal{E}[\mathbf{R}_{\mathbb{i}}]}$, then $x = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}[x]) \notin \mathcal{L}_{\mathbf{R}_{\mathbb{i}}}$ where $\mathtt{sk}$ is the opening of $\mathtt{C}_{\mathtt{sk}}$. This means that $(x, \emptyset) \in D$ and therefore $(\mathbb{x}', \emptyset) \in D'$.
2. For any $(\mathbb{x}', \mathtt{tr}') = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], \mathtt{tr}') \in D'$ where $\mathtt{tr}'$ is a $(i-1)$-round partial transcript for $i \in [\mu+1]$, the corresponding plaintext transcript $(\mathbb{x}, \mathtt{tr}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{tr}')) \in D$ where $\mathtt{sk}$ corresponds to the commitment $\mathtt{C}_{\mathtt{sk}}$. Thus, for any next blind hIOP prover message $\mathtt{ct}_i$ and its decryption $m_i = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}_i)$, it holds that

$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}, \mathtt{tr} \| m_i \| c_i) \notin D] = \Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i \| c_i) \notin D'] \leq \varepsilon.$$

3. For any full transcript $\mathtt{tr}'$, if $(\mathbb{x}', \mathtt{tr}') = (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x] \| \mathtt{tr}') \in D'$ then, by definition of D', the decryption $(\mathbb{x}, \mathtt{tr}) \leftarrow \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{tr}')) \in D$. Therefore, by definition of $\mathcal{E}[\Pi].\mathsf{V}$ and the assumption that D is a doomed set, it is clear that $\mathcal{E}[\Pi].\mathsf{V}^{[\![\mathsf{e}[\mathbb{i}]]\!], [\![\mathtt{tr}']\!]}(\mathbb{x}) = \mathsf{rej}$ where $\mathsf{e}[\mathbb{i}] = \mathcal{E}[\Pi].\mathsf{Ind}(\mathbb{i})$.

□

By definition 7, a public-coin hIOP $\Pi = (\mathsf{Ind}, \mathsf{P}, \mathsf{V})$ for an indexed relation $\mathbf{R}$ is round-by-round knowledge sound with error ε_{k} if for every index $\mathbb{i}$ there exists a knowledge doomed set $\mathcal{D}^{\Pi}_{\mathsf{k}}$ for error ε_{k} that uses some polynomial-time extractor Ext. In the case of a blind hIOP for a blind relation $\mathcal{E}[\mathbf{R}_{\mathbb{i}}]$ we define round-by-round knowledge soundness slightly different since it should be able to extract the witness of the corresponding relation $\mathbf{R}_{\mathbb{i}}$.

Definition 10 (Round-by-round plaintext knowledge soundness). *A blind hIOP $\mathcal{E}[\Pi]$ for an indexed blind relation $\mathcal{E}[\mathbf{R}]$ is round-by-round plaintext knowledge sound with error $\varepsilon_{\mathsf{pk}}$ if for every index $\mathbb{i}$ there exists a knowledge doomed set $\mathcal{D}^{\mathcal{E}[\Pi]}_{\mathsf{k}}$ for error $\varepsilon_{\mathsf{pk}}$ that uses an extractor $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ with access to a decryption oracle $\mathcal{O}_{\mathsf{Dec}}$.*

Theorem 7. *The blind hIOP $\mathcal{E}[\Pi]$ is round-by-round plaintext knowledge sound for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ with error ε_{k} if the hIOP Π is round-by-round knowledge sound for the indexed relation $\mathbf{R}$ with error ε_{k}.*

Proof. Similar to Theorem 6, we show that there exists a knowledge doomed set $D' = \mathcal{D}^{\mathcal{E}[\Pi]}_{\mathsf{k}}$ given the existence of the knowledge doomed set $D = \mathcal{D}^{\Pi}_{\mathsf{k}}$. Again we define a set D' to contain all HE ciphertext that decrypt to some transcript in D under the secret key $\mathtt{sk}$ corresponding to $\mathtt{C}_{\mathtt{sk}}$. It is clear that D' satisfies the first and third property of a knowledge doomed set. The second property states that for any $(i-1)$-round partial transcript $(\mathbb{x}', \mathtt{tr}') \in D'$ where $i \in [\mu+1]$, and any next prover message $\mathtt{ct}_i$, it should hold that if

$$\Pr_{c_i \leftarrow \mathcal{C}h_i}[(\mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i \| c_i) \notin D'] > \varepsilon_{\mathsf{k}}$$

then $\mathsf{Ext}_{\mathtt{sk}}(\mathsf{e}[\mathbb{i}], \mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i)$ outputs a valid witness for $\mathbb{x}$. Similarly as in Theorem 6, if we define $m_i = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}_i)$ then $(\mathbb{x}', \mathtt{tr}' \| \mathtt{ct}_i \| c_i) \notin D'$ implies $(\mathbb{x}, \mathtt{tr} \| m_i \| c_i) \notin D$ for any c_i and $(\mathbb{x}, \mathtt{tr}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \mathtt{tr}'))$. Therefore, we can construct the extractor $\mathsf{Ext}^{\mathcal{O}_{\mathsf{Dec}}}$ as first requesting $m_i = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}(\mathtt{ct}_i)$ from $\mathcal{O}_{\mathsf{Dec}}$ and subsequently running the extractor $\mathsf{Ext}(\mathsf{e}[\mathbb{i}], \mathbb{x}, \mathtt{tr} \| m_i)$, the output will be a valid witness for $\mathbb{x}$ since $(\mathbb{x}, \mathtt{tr}) \in D$. □

5.2 Designated-Verifier Blind zkSNARK (DV-BzkSNARK)

From Theorem 1, it is clear that any public-coin hIOP Π for some indexed relation $\mathbf{R}$ can be compiled into a zkSNARK for $\mathbf{R}$ in the ROM using BCS compilation. In practice, the RO is instantiated with some suitable hash function. The compiler functions by committing to every oracle message sent by the prover using a Merkle Tree MT and instead sending the commitment root $\mathtt{C}$. Then, when the verifier queries some oracle message, the prover responds to the query by including the authentication path $\mathtt{ap}$ from the corresponding root to the queried value in the message. Lastly, since Π is public-coin, one can make the proof non-interactive using a Fiat-Shamir-like transform FS to simulate the verifier's challenges and generate a final RO output τ.

Let us now describe *public-coin* hIOP verification as follows. The verifier $\Pi.\mathsf{V}$ receives the statement x and in each round i receives a message m_i, contributing to the current partial transcript $\mathtt{tr}_i$, and responds with a challenge $c_i \leftarrow \mathcal{C}h_i$. After receiving the final prover message, the verifier queries the oracle $[\![\mathtt{tr}]\!]$ to construct a list of queries $\{q_i\}$ (same for the oracle $[\![\mathtt{e}[\mathtt{i}]]\!]$, but we dismiss holography for now for ease of notation). Lastly, the verifier returns acc if and only if some equality $f(x, \{q_i\}, \tau) = 0$ holds where τ represents some randomness. In Figure 1, we illustrate the behaviour of BCS compilation from Theorem 1 using this notation. It describes the zkSNARK verifier resulting from this compilation. Instead of performing queries, the verifier receives query responses and checks their authentication paths and whether they where sampled using the RO.

$\Pi.\mathsf{V}(x, \pi = [\{q_i, \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau])$

foreach j :

  $\mathsf{MT.Open}(q_j, \mathtt{ap}_j, \mathtt{C}_{\mathsf{qr}(j)}) \stackrel{?}{=} \mathsf{acc}$

$\mathsf{FS}(x, \{\mathtt{C}_i\}) \stackrel{?}{=} \tau$

$f(x, \{q_i\}, \tau) \stackrel{?}{=} 0$

Fig. 1: Verifier of the compiled zkSNARK resulting from Theorem 1.

It should be clear that this compilation can likewise be applied to the *public-coin* BhIOP from Definition 9. Since the verifier $\mathcal{E}[\Pi].\mathsf{V}$ is public-coin, the only difference to the hIOP verifier will be after receiving the final prover message. By querying the oracle $[\![\mathtt{tr}]\!]$, the verifier $\mathcal{E}[\Pi].\mathsf{V}$ receives ciphertexts $\{\mathtt{ct}[q_i]\}$ that are decrypted to $\{q_i\}$ using $\mathtt{sk}$. Similarly, the verifier decrypts the statement $\mathtt{ct}[x]$ to x and then computes $b \leftarrow \Pi.\mathsf{V}_{\mu+1}(x, \{q_i\}, \tau)$, which we have previously denoted as checking whether some equality $f(x, \{q_i\}, \tau) = 0$ holds, for some randomness τ. Lastly, the verifier returns b if the commitment $\mathtt{C}_{\mathtt{sk}}$ is a valid commitment to $\mathtt{sk}$. We describe this compilation in Theorem 8 and the resulting verifier in Figure 2. Note that we define a subroutine $\mathsf{PartialVer}$ that performs verification without verifying the correspondence between the queries $\{\mathtt{ct}[q_i]\}$ and their plaintexts $\{q_i\}$.

$\mathcal{E}[\mathsf{dv}\Pi].\mathsf{V}_{\mathtt{sk}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x];\ \pi^{\mathsf{dv}} = [\{\mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau])$

$(x, \{q_i\}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \{\mathtt{ct}[q_i]\}))$

$\mathsf{PartialVer}(\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], x, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau)$

- **foreach** j :
  - $\mathsf{MT.Open}(\mathtt{ct}[q_j], \mathtt{ap}_j, \mathtt{C}_{\mathsf{qr}(j)}) \stackrel{?}{=} \mathsf{acc}$
- $\mathsf{FS}(\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], \{\mathtt{C}_i\}) \stackrel{?}{=} \tau$
- $f(x, \{q_i\}, \tau) \stackrel{?}{=} 0$
- $\mathcal{O}^{\mathcal{CT}}(\mathtt{C}_{\mathtt{sk}}, \mathtt{sk}) \stackrel{?}{=} \mathsf{acc}$

Fig. 2: Verifier of the designated-verifier blind zkSNARK resulting from Theorem 8.

The fact that $\mathcal{E}[\Pi].\mathsf{V}$ requires knowledge of the secret key $\mathtt{sk}$ has two major consequences for BCS compilation. Most notably, the resulting zkSNARK verifier $\mathcal{E}[\mathsf{dv}\Pi].\mathsf{V}$ inherits the same requirement; thus, the compiler outputs a *designated-verifier* blind zkSNARK. Secondly, the public-coin requirement for the verifier $\Pi.\mathsf{V}$ is no longer sufficient. Strictly, it ensures that the hIOP verifier is simulatable by the zkSNARK prover in BCS compilation. It can simulate the random challenges using the Fiat-Shamir tranform and simulate the queries by providing Merkle Tree openings. To ensure that the blind hIOP verifier $\mathcal{E}[\Pi].\mathsf{V}$ is simulatable by the blind zkSNARK prover, we must additionally require that it performs no queries where the query location depends on previously queried values (since those are hidden from the prover). This holds for queries to both the $[\![\mathsf{e}[\mathtt{i}]]\!]$ and the $[\![\mathtt{tr}]\!]$ oracles. Nassar et al. [73] have previously coined hIOPs with such verifiers *non-adaptive*. To our knowledge, such hIOP has never been described and so this requirement forms no restriction.

Theorem 8. *Let $\mathcal{E}[\Pi]$ be a non-adaptive public-coin blind hIOP for the indexed blind relation $\mathcal{E}[\mathbf{R}]$ where Π has completeness error δ, proof length p, query complexity q, round-by-round soundness error ε, round-by-round knowledge soundness error ε_{k} and z-statistical honest-verifier zero-knowledge, and the HE scheme $\mathcal{E}$ is correct for the homomorphic circuit in $\mathcal{E}[\Pi].\mathsf{P}$ with security parameter λ. Such blind hIOP scheme can be compiled into a designated-verifier zero-knowledge non-interactive argument of plaintext knowledge for $\mathcal{E}[\mathbf{R}]$, which we will coin a designated-verifier blind zkSNARK $\mathcal{E}[\mathsf{dv}\Pi]$ in the ROM. Then, against Q-query adversaries, $\mathcal{E}[\mathsf{dv}\Pi]$ has:*

- *Completeness error $\delta + \mathsf{negl}(\lambda)$,*
- *Proof length p' upper bounded by*

$$p' = \lambda(\mu + 1 + \sum_{j=1}^{q}(2 + \lceil \log_2 |m_{\mathsf{qr}(j)}|/\mathcal{E}.\mathsf{len}\rceil)) + q\mathcal{E}.\mathsf{len},$$

– *Soundness error* ε' *where* $\varepsilon' = Q\varepsilon + 3(Q^2+1)2^{-\lambda}$,
– *Knowledge soundness error* $\varepsilon'_{\mathsf{k}}$ *where* $\varepsilon'_{\mathsf{k}} = Q\varepsilon_{\mathsf{k}} + 3(Q^2+1)2^{-\lambda}$,
– z'*-Statistical honest-verifier zero-knowledge where* $z' = z + \mathsf{negl}(\lambda) + p2^{-\lambda/4+2}$.

where m_i *is* $\mathcal{E}[\Pi].\mathsf{P}$*'s* i*th prover message and* $|\cdot|$ *denotes length in* λ *bits rounded up. Both soundness and knowledge soundness error are* $\Theta(Q\varepsilon)$ *and* $\Theta(Q\varepsilon_{\mathsf{k}})$ *respectively when considering quantum adversaries that perform no more than* $Q - \mathcal{O}(q \log p)$ *RO queries.*

Proof. The proof follows trivially from Theorem 1 and the discussion above. The non-adaptivity ensures that prover $\mathcal{E}[\mathsf{dv}\Pi].\mathsf{P}$ can partly simulate the verifier $\mathcal{E}[\Pi].\mathsf{V}$ to compute the query locations (it can obviously not simulate the equality checks). The proof length is similar, except for the expansion from HE encryption of the queried values. □

5.3 Publicly-Verifiable Blind zkSNARK (PV-BzkSNARK)

We have shown that using an HE scheme, it is possible to construct a blind zkSNARK in the designated-verifier setting. This primitive already has applications as a vCOED scheme; in this setting, the client would encrypt the statement and the server would compute the encrypted witness, which can then be used to compute the proof. Now, we will show how to compile this designated-verifier blind zkSNARK into a publicly-verifiable blind zkSNARK; this also expands the application of this construction to zkSNARK delegation (the zkDel setting). In this setting, the client computes the witness and then sends the *plaintext* statement and encrypted witness to the server, who then computes the proof. In both scenarios, the client computes a (batched) Proof of Decryption (PoD) to make the proof publicly verifiable. Below we provide a formal definition.

Definition 11 (Proof of Decryption). *For a given HE scheme* $\mathcal{E}$ *with security parameter* λ *and a commitment scheme* $\mathcal{CT}$*, a Proof of Decryption scheme* $\mathsf{PoD}[\mathcal{E}] = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *includes the following probabilistic polynomial-time algorithms:*

– $\mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$*: for a given security parameter* λ*, it returns some public parameters* $\boldsymbol{pp}$ *which are implicit inputs to the following functions.*
– $\mathsf{PoD}[\mathcal{E}].\mathsf{P}_{\boldsymbol{sk}}(\mathcal{C}_{\boldsymbol{sk}}, \boldsymbol{ct})$*: for a given secret key commitment* $\mathcal{C}_{\boldsymbol{sk}}$*, ciphertext* $\boldsymbol{ct}$ *and secret key* $\boldsymbol{sk}$*, it returns a proof of decryption* π^{PoD} *and plaintext* m.
– $\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathcal{C}_{\boldsymbol{sk}}}(\pi^{\mathsf{PoD}}, \boldsymbol{ct}, m)$*: for a given proof of decryption* π^{PoD}*, ciphertext* $\boldsymbol{ct}$*, plaintext message* m *and secret key commitment* $\mathcal{C}_{\boldsymbol{sk}}$*, it returns either* acc *or* rej.

It should satisfy the following properties.

Completeness. *For any public parameters* $\boldsymbol{pp} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$*, HE keys* $(\boldsymbol{sk}, \boldsymbol{evk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ *and ciphertexts* $\{\boldsymbol{ct}[m_i]\}$ *such that*

$\{m_i\} = \mathcal{E}.\mathsf{Dec}_{sk}(\{\mathtt{ct}[m_i]\})$, it holds that

$$\Pr\left[\begin{array}{c}\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathcal{C}_{sk}}(\pi^{\mathsf{PoD}}, \{\mathtt{ct}[m_i], m_i\}) \\ \neq \\ \mathsf{acc}\end{array}\;\middle|\;\begin{array}{c}\mathcal{C}_{sk} \leftarrow \mathsf{Com}(sk) \\ \pi^{\mathsf{PoD}} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{P}_{sk}(\mathcal{C}_{sk}, \{\mathtt{ct}[m_i]\})\end{array}\right]$$

is less than or equal to some completeness error δ.

Knowledge soundness. *For any public parameters $pp \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$, HE keys $(sk, evk) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and PPT prover P^*, there exists a PPT extractor $\mathsf{Ext}^{\mathsf{P}^*}$ and knowledge error ε_{k} such that*

$$\Pr\left[\begin{array}{c}\mathcal{E}.\mathsf{Dec}_{sk^*}(\{\mathtt{ct}_i\}) \neq \{m_i\} \\ \wedge \\ \mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathcal{C}_{sk}}(\pi^{\mathsf{PoD}}, \{\mathtt{ct}_i, m_i\}) = \mathsf{acc}\end{array}\;\middle|\;\begin{array}{c}(\pi^{\mathsf{PoD}}, \mathcal{C}_{sk}, \{\mathtt{ct}_i, m_i\}) \leftarrow \mathsf{P}^* \\ sk^* \leftarrow \mathsf{Ext}^{\mathsf{P}^*} \\ \mathcal{O}^{\mathcal{CT}}(\mathcal{C}_{sk}, sk^*) = \mathsf{acc}\end{array}\right] \leq \varepsilon_{\mathsf{k}}.$$

Zero-knowledge. *For any public parameters $pp \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Setup}(1^\lambda)$, HE keys $(sk, evk) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$ and ciphertexts $\{\mathtt{ct}[m_i]\}$, if there exists a probabilistic polynomial-time simulator Sim such that for any probabilistic polynomial-time distinguisher D it holds that*

$$\Big|\Pr\left[\mathsf{D}(\pi^{\mathsf{PoD}}, \mathcal{C}_{sk}) = 1 \;\middle|\; (\pi^{\mathsf{PoD}}, \mathcal{C}_{sk}) \leftarrow \mathsf{Sim}(1^\lambda)\right] \\ - \Pr\left[\mathsf{D}(\pi^{\mathsf{PoD}}, \mathcal{C}_{sk}) = 1 \;\middle|\; \begin{array}{c}\mathcal{C}_{sk} \leftarrow \mathsf{Com}(sk) \\ \pi^{\mathsf{PoD}} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{P}_{sk}(\mathcal{C}_{sk}, \{\mathtt{ct}[m_i]\})\end{array}\right]\Big| \leq z$$

then $\mathsf{PoD}[\mathcal{E}]$ has z-computational zero-knowledge.

In the following, we discuss how a proof of decryption for the HE scheme $\mathcal{E}$ allows us to compile a designated-verifier blind zkSNARK $\mathcal{E}[\mathsf{dv}\Pi]$ into a publicly verifiable zkSNARK $\mathcal{E}[\mathsf{pv}\Pi]$. Any party that can verify a proof π^{dv} is able to construct a proof π^{pv} by appending the plaintext queries $\{q_i\}$ along with a PoD proving they are decryptions of the queries $\{\mathtt{ct}[q_i]\}$ that were committed to in $\{\mathtt{C}_i\}$, using the secret key committed to in $\mathtt{C}_{\mathsf{sk}}$. This results in the public verifier described in Figure 3.

$\mathcal{E}[\mathsf{pv}\Pi].\mathsf{V}(x;\, \pi^{\mathsf{pv}} = [\mathtt{C}_{\mathsf{sk}}, \mathtt{ct}[x], \pi^{\mathsf{PoD}}, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau])$	
1 :	$\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathtt{C}_{\mathsf{sk}}}(\pi^{\mathsf{PoD}}, (\mathtt{ct}[x], \{\mathtt{ct}[q_i]\}), (x, \{q_i\})) \stackrel{?}{=} \mathsf{acc}$
2 :	$\mathsf{PartialVer}(\mathtt{C}_{\mathsf{sk}}, \mathtt{ct}[x], x, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau) \stackrel{?}{=} \mathsf{acc}$

Fig. 3: Verifier of the publicly-verifiable zkSNARK resulting from Theorem 9.

In Figure 4, we describe the construction of π^{pv}, and in Theorem 9, we prove that it describes a publicly-verifiable blind zkSNARK. If the PV-BzkSNARK is used in the zkDel setting, one can replace $\mathtt{ct}[x]$ by x in the blind relation and thus $\mathtt{ct}[x]$ requires no PoD and is not included in π^{pv}. Note that it is not necessary for the delegator to send the entire encrypted (extended) witness $\mathtt{ct}[w]$; they can simply send an encryption of the private inputs, from which $\mathtt{ct}[w]$ can be computed homomorphically. This would demand less encryption cost from the verifier and would not increase HE parameters. In the vCOED setting, one depends on Theorem 2 to hide x from the blind prover; therefore, $\mathtt{ct}[x]$ should be included in the proof.

Theorem 9. *For security parameter λ, the protocol in Figure 4 is a publicly-verifiable zkSNARK for the relation $\mathbf{R}$ in the ROM against Q-query adversaries with completeness error $\delta^{\Pi} + \delta^{\mathsf{PoD}} + \mathsf{negl}(\lambda)$ and knowledge soundness error $\varepsilon_{\mathsf{k}}^{\mathsf{PoD}} + Q\varepsilon_{\mathsf{k}}^{\Pi} + 3(Q^2+1)2^{-\lambda}$ that is $z^{\Pi} + \mathsf{negl}(\lambda) + p2^{-\lambda/4+2} + z^{\mathsf{PoD}}$-computational zero-knowledge if $\mathcal{E}$ is an IND-CPA secure and correct HE scheme, PoD is a zero-knowledge proof of decryption with completeness error δ^{PoD} and knowledge soundness error $\varepsilon_{\mathsf{k}}^{\mathsf{PoD}}$ that is z^{PoD}-computational zero-knowlege, and Π is an hIOP scheme with completeness error δ^{Π} and knowledge soundness $\varepsilon_{\mathsf{k}}^{\Pi}$ that is z^{Π}-computational zero-knowledge with proof length p.*

Proof. Completeness follows immediately from the completeness of the PoD scheme along with Theorem 8. Similarly, the zero-knowledge property follows from a hybrid argument using the zero-knowledge property of the PoD and Theorem 8. We discuss knowledge soundness in more detail.

Let us denote P^* as a prover that outputs a proof π^{pv} for the index $\mathbb{i}$ and statement x such that the verifier $\mathcal{E}[\mathsf{pv}\mathbb{\Pi}].\mathsf{V}$ (see Figure 3) accepts with probability p. To prove knowledge soundness, we will show that there exists a polynomial-time extractor Ext that outputs w with probability greater than $p - \varepsilon_{\mathsf{k}}^{\mathsf{PoD}} - \varepsilon_{\mathsf{k}}^{\mathcal{E}[\mathsf{dv}\mathbb{\Pi}]}$, when given access to P^*. Firstly, when $\mathcal{E}[\mathsf{pv}\mathbb{\Pi}].\mathsf{V}$ accepts, the first line in Figure 3 states that $\mathsf{PoD}[\mathcal{E}].\mathsf{V}_{\mathsf{C}_{\mathtt{sk}}}$ also accepts. Therefore, by the knowledge soundness of PoD, the prover P^* can be used to extract a secret key $\mathtt{sk} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{Ext}$ such that

$$(x, \{q_i\}) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}((\mathtt{ct}[x], \{\mathtt{ct}[q_i]\}))$$

and $\mathtt{sk}$ is the secret key committed to in $\mathsf{C}_{\mathtt{sk}}$. Secondly, from Theorem 8 we know that the designated-verifier zkSNARK $\mathcal{E}[\mathsf{dv}\mathbb{\Pi}]$ is plaintext knowledge sound. In other words, any prover that can produce a proof π^{dv} such that the verifier in Figure 2 satisfies, can be used to extract a witness $\mathbb{w} \leftarrow \mathcal{E}[\mathsf{dv}\mathbb{\Pi}].\mathsf{Ext}_{\mathtt{sk}}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathbf{R}$. Note that by assumption, P^* generates proofs that satisfy the $\mathsf{PartialVer}$ subroutine in $\mathcal{E}[\mathsf{dv}\mathbb{\Pi}].\mathsf{V}_{\mathtt{sk}}$. The knowledge soundness of the PoD discussed before ensures that also the first line in Figure 2 is satisfied. Therefore, our prover P^* can be used by the extractor $\mathcal{E}[\mathsf{dv}\mathbb{\Pi}].\mathsf{Ext}_{\mathtt{sk}}$ where $\mathtt{sk}$ is the secret key extracted previously using $\mathsf{PoD}[\mathcal{E}].\mathsf{Ext}$. $\square$

Remark. Note that the execution of $\mathsf{PartialVer}$ by the client is only required in a setting where the blind prover is incentivized to be dishonest. In such setting, it is also required to not reuse the same HE secret key. Note that most HE schemes

zkDel Setting: Delegator		
vCOED Setting: Designated verifier		
$\mathtt{sk}, \mathtt{evk} \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$		
$\mathtt{C}_{\mathtt{sk}} \leftarrow \mathcal{CT}.\mathsf{Com}(\mathtt{sk})$		
$w \leftarrow \mathsf{ExtWit}(\mathbb{i}, x)$		
$(\mathtt{ct}[x], \mathtt{ct}[w]) \leftarrow \mathcal{E}.\mathsf{Enc}_{\mathsf{sk}}((x, w))$	$\xrightarrow{\mathtt{evk}, \mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], \mathtt{ct}[w]}$	**Blind prover**
		$\pi^{\mathsf{dv}} \leftarrow \mathcal{E}[\mathsf{dv}\mathbb{\Pi}].\mathsf{P}_{\mathsf{evk}}(\mathsf{e}[\mathbb{i}], (\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x]), \mathtt{ct}[w])$
$\pi^{\mathsf{dv}} = [\{\mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau]$	$\xleftarrow{\pi^{\mathsf{dv}}}$	
$\{q_i\} \leftarrow \mathcal{E}.\mathsf{Dec}_{\mathsf{sk}}(\{\mathtt{ct}[q_i]\})$		
$\mathsf{PartialVer}(x, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau) \stackrel{?}{=} \mathsf{acc}$		
$\pi^{\mathsf{PoD}} \leftarrow \mathsf{PoD}[\mathcal{E}].\mathsf{P}_{\mathsf{sk}}(\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}[q_i]\}, \{q_i\})$		
$\pi^{\mathsf{pv}} = [\mathtt{C}_{\mathtt{sk}}, \mathtt{ct}[x], \pi^{\mathsf{PoD}}, \{q_i, \mathtt{ct}[q_i], \mathtt{ap}_i\}, \{\mathtt{C}_i\}, \tau])$	$\xrightarrow{\pi^{\mathsf{pv}}}$	**Public verifier**
		$\mathcal{E}[\mathsf{pv}\mathbb{\Pi}].\mathsf{V}(\mathtt{C}_{\mathtt{sk}}, x;\, \pi^{\mathsf{pv}}) \stackrel{?}{=} \mathsf{acc}$

Fig. 4: Compilation of a DV-BzkSNARK into a PV-BzkSNARK.

are vulnerable to key-recovery attacks when the client leaks to the server whether the ciphertexts properly decrypt. In the zkDel setting, the server would learn this when it has access to the publicly-verifiable proof.

Remark. In our PoD construction, the commitment $\mathtt{C}_{\mathtt{sk}}$ is included in the π^{PoD}. Naturally, the proof size of π^{pv} (see Theorem 8) is still larger than a normal proof for the zkSNARK $\mathbb{\Pi}$. This could be mediated by sending π^{pv} to the delegatee to delegate the computation of a recursion step.

6 Instantiation of blind zkSNARKs

In this section, we describe algorithms for computing blind zkSNARKs efficiently. Concretely, for some specific Π, we optimize the computation of $\mathcal{E}[\Pi].\mathsf{P}$ such that the blind zkSNARK resulting from Theorem 8 has efficient proof generation. The input to $\mathcal{E}[\Pi].\mathsf{P}$ is the encrypted trace $\mathtt{ct}[z] = (x, \mathtt{ct}[w])$, where x and w are vectors in some finite field $\mathbb{F}$, and computing $\mathcal{E}[\Pi].\mathsf{P}$ consists of homomorphically evaluating $\Pi.\mathsf{P}$ using $\mathcal{E}.\mathsf{Eval}$.

Most HE schemes naturally support Single Instruction Multiple Data (SIMD) operations since the plaintext space can be interpreted as the vector space $\mathcal{P} = \mathbb{F}^P$ for some finite field $\mathbb{F}$. Operating on plaintexts $\mathtt{pt}$ and/or ciphertexts $\mathtt{ct}$ corresponds to pointwise operations on elements in $\mathcal{P}$, such as pointwise addition and multiplication. Using automorphisms, it is also possible to compute arbitrary linear operations on an encrypted vector, i.e. a matrix-vector multiply.

The inherent noise growth in a ciphertext depends on the type of operation: additions (both `pt-ct` as well as `ct-ct`) cause additive noise growth, whereas multiplication incurs a fixed multiplicative factor depending on the parameters of the scheme. A `pt-ct` multiplication incurs a smaller noise growth than a `ct-ct` multiplication and is also much faster to compute. As an example, for the parameter set used in our implementation, the noise growth would be on average 6.2 bits for a `pt-ct` and 10.3 bits for a `ct-ct`, and a `pt-ct` multiply is $75\times$ faster than a `ct-ct` multiply. An automorphism causes minimal noise growth as it does not change the norm in the canonical embedding; in our implementation, it takes 1/4 of the time of a `ct-ct` multiplication. As will become clear in this section, designing efficient homomorphic circuits is largely a trade-off between the number of operations performed and the amount of noise they require.

Similar to Garg et al. [49], we suggest using the Fractal hIOP Π_{F} [37]; this choice lets us optimize the homomorphic computation of the corresponding blind hIOP in two ways. Firstly, many of the operations in Fractal can be performed element-wise on vectors, which allows us to significantly lower the number of homomorphic operations using SIMD as described above. Secondly, Fractal's linearity implies that many of the homomorphic operations will consist of the cheaper `pt-ct` operations. In Section 8, we select an HE scheme that allows us to exploit both these characteristics. In Section 6.1 below, we introduce some techniques for homomorphically computing Number Theoretic Transforms (NTTs) and arbitrary linear operations. In Section 6.2, we describe the Fractal scheme and discuss an efficient algorithm for computing it blindly, namely the algorithm $\mathcal{E}[\Pi_{\mathsf{F}}].\mathsf{P}$ from Definition 9.

6.1 Building blocks

Packing-friendly 2D-NTT. Consider some finite field $\mathbb{F}$ and an element $\xi_N \in \mathbb{F}$ of order N known as the primitive N-th root of unity. The (inverse) NTT transformation of a vector of polynomial evaluations $\mathbf{a} = [a_0, a_1, \ldots, a_{N-1}] \in \mathbb{F}^N$ is denoted as $\hat{\mathbf{a}} = [\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{N-1}] \in \mathbb{F}^N$ where

$$\hat{a}_k = \frac{1}{N} \sum_{n=0}^{N-1} \xi_N^{-n \cdot k} a_n \quad \forall k \in [0, N-1]. \tag{1}$$

Observe that an inverse NTT differs from the NTT only by using ξ_N^{-1} instead of ξ_N and scaling by $1/N$ at the end. Let us now assume the NTT size N can be written as $N = N_1 \cdot N_2$. As such, we can rewrite Equation (1) by iterating over

$n = N_2 \cdot n_1 + n_2$ and $k = N_1 \cdot k_2 + k_1$ using $k_i, n_i \in [0, N_i - 1] = [N_i]_0$ for $i = 1, 2$

$$\begin{aligned}
\hat{a}_{N_1 \cdot k_2 + k_1} &= \frac{1}{N} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \xi_N^{-(N_2 \cdot n_1 + n_2)(N_1 \cdot k_2 + k_1)} a_{N_2 \cdot n_1 + n_2} \\
&= \frac{1}{N} \sum_{n_2=0}^{N_2-1} \xi_{N_2}^{-n_2 \cdot k_2} \left(\xi_N^{-n_2 \cdot k_1} \sum_{n_1=0}^{N_1-1} \xi_{N_1}^{-n_1 \cdot k_1} a_{N_2 \cdot n_1 + n_2} \right) \quad \forall k_i \in [N_i]_0
\end{aligned}$$

where $\xi_{N_1} = \xi_N^{N_2}$ and $\xi_{N_2} = \xi_N^{N_1}$ are primitive N_1-th and N_2-th roots of unity, respectively. Let us define $f_{n_2,k_1} = \sum_{n_1=0}^{N_1-1} \xi_{N_1}^{-n_1 \cdot k_1} a_{N_1 \cdot n_2 + n_1}$. Now, notice that a size N NTT can be composed into three steps:

1. for $n_2 \in [N_2]_0$ perform a size-N_1 NTT to compute $[f_{n_2,k_1}]_{k_1 \in [N_1]_0}$,
2. multiply by twiddle factors to get $f'_{n_2,k_1} = \xi_N^{-n_2 \cdot k_1} \cdot f_{n_2,k_1}$,
3. for $k_1 \in [N_1]_0$ perform a size-N_2 NTT to compute $[\hat{a}_{N_1 \cdot k_2 + k_1}]_{k_2 \in [N_2]_0}$.

While a variant of the 2D-NTT algorithm has been used to achieve distributed FFT in a distributed zero-knowledge proof system [80], our work demonstrates its effectiveness in a different context, namely for homomorphically performing NTTs on packed HE ciphertexts. Note that [5] also considered exploiting HE packing but only for performing P homomorphic NTTs at once, requiring packing and unpacking operations. They do not perform homomorphic NTTs for a degree N polynomial packed in N/P ciphertexts since it would require expensive HE operations to swap slots between ciphertexts (think of the bit-reversal that the base 2 butterfly algorithm causes). These can only be avoided by using one slot per N ciphertexts, resulting in higher memory usage and reduced efficiency.

Our approach packs the size N input into $N/P = N_1(N_2/P)$ ciphertexts and avoids swapping slots between ciphertexts by exploiting the structure of the 2D-NTT. More precisely, we pack N_2 elements $(a_{N_2 \cdot n_1}, a_{N_2 \cdot n_1 + 1}, \ldots, a_{N_2 \cdot n_1 + N_2 - 1})$ into N_2/P ciphertexts for $n_1 \in [N_1]_0$. Then, the homomorphic NTT evaluation can be performed as follows:

1. homomorphically evaluate N_2/P size-N_1 NTTs using the butterfly algorithm in some base b on packed ciphertexts,
2. perform N/P `pt-ct` multiplications to multiply with twiddle factors,
3. homomorphically evaluate N_1 size-N_2 NTTs as matrix-vector multiplications with vectors of size N_2 packed in N_2/P ciphertexts.

An example is provided in Fig. 5 for $N_2 = P$. Note that the output of the 2D-NTT is packed in column-major order, i.e. N_2 elements $(\hat{a}_{k_1}, \hat{a}_{N_1+k_1}, \ldots, \hat{a}_{N_1 \cdot (N_2-1)+k_1})$ are packed in one ciphertext. Although this ordering may seem problematic, notice that it can be reversed by a subsequent NTT operation that is computed as

$$\begin{aligned}
a_{N_2 \cdot n_1 + n_2} &= \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \xi_N^{(N_2 \cdot n_1 + n_2)(N_1 \cdot k_2 + k_1)} \hat{a}_{N_1 \cdot k_2 + k_1} \\
&= \sum_{k_1=0}^{N_1-1} \xi_{N_1}^{n_1 \cdot k_1} \left(\xi_N^{n_2 \cdot k_1} \sum_{k_2=0}^{N_2-1} \xi_{N_2}^{n_2 \cdot k_2} \hat{a}_{N_1 \cdot k_2 + k_1} \right) \quad \forall n_i \in [N_i]_0
\end{aligned}$$

by doing the following steps:

1. homomorphically evaluate N_1 size-N_2 NTTs as matrix-vector multiplications with vectors of size N_2 packed in N_2/P ciphertexts,
2. perform N/P `pt-ct` multiplications to multiply with twiddle factors,
3. homomorphically evaluate N_2/P size-N_1 NTTs using the butterfly algorithm in some base b on packed ciphertexts.

The algorithms in Section 6.2 require us to compute

$$[a(x)_{x\in D_2}] = \mathrm{NTT}\left(f\left(\mathrm{iNTT}([a(x)]_{x\in D_1})\right)\right) \tag{2}$$

for some evaluation domains D_1, D_2 and some function f that must be computed on ciphertext vectors in some column-major packing. It will always be the case that $|D_2| > |D_1|$ so $[\hat{a}_i]_{i\in[N]_0}$ has to be appended with zeros. Due to the column-major packing, the width $w \coloneqq N_2/p$ has to grow with $|D_2|/|D_1|$ before performing the second 2D-NTT. Importantly, this makes the matrix-vector multiplications of the second 2D-NTT less costly since they can take these zeros into account. Lastly, we remark that these techniques can also be applied when D_1, D_2 are cosets of multiplicative subgroups.

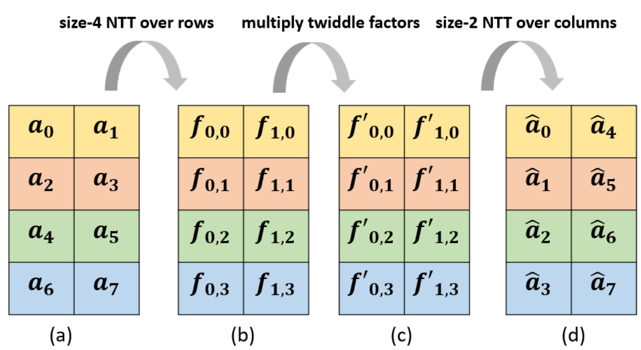

Fig. 5: Illustration of inverse 2D-NTT when $N_1 = 4, N_2 = 2$. Elements within the same row (represented by the same color) are packed in one HE ciphertext.

Matrix-vector multiplication. Let $\mathbf{A} \in \mathbb{F}^{nP\times nP}$ denote a square matrix and $\mathbf{v} \in \mathbb{F}^{nP}$ denote a column vector over a finite field $\mathbb{F}$. For HE schemes that pack P finite field elements per ciphertext, the multiplication of the plaintext matrix $\mathbf{A}$ with encryptions of $\mathbf{v}$ can be visualized as

$$\left(\begin{array}{c|c|c} \mathbf{A_{0,0}} & \cdots & \mathbf{A_{0,n-1}} \\ \hline \vdots & \ddots & \vdots \\ \hline \mathbf{A_{n-1,0}} & \cdots & \mathbf{A_{n-1,n-1}} \end{array}\right) \cdot \left(\begin{array}{c} \mathtt{ct}[\mathbf{v_0}] \\ \hline \vdots \\ \hline \mathtt{ct}[\mathbf{v_{n-1}}] \end{array}\right) \tag{3}$$

where $\mathbf{v_i} = [v_{i\cdot P}, v_{i\cdot P+1}, \dots, v_{i\cdot P+P-1}] \in \mathbb{F}^P$ and $\mathbf{A_{i,j}} \in \mathbb{F}^{P\times P}$ denotes the (i,j)-th block matrix of $\mathbf{A}$. For the base case of $n = 1$, the computation of (3) can be performed using the following two methods

- Full matrix in the diagonal (FD) method [58], which requires $P-1$ automorphisms, P `pt-ct` multiplications, and $P-1$ additions. In terms of noise depth, it incurs the equivalent of 1 `pt-ct` operation, 1 automorphism and $\lceil \log_2 P \rceil$ additions.
- Baby-Step/Giant-Step diagonal (BS/GS) method [59], which requires $2\sqrt{P}-2$ automorphisms, P `pt-ct` multiplications, and $P-1$ additions. In terms of noise depth, it incurs the equivalent of 1 `pt-ct` operation, 2 automorphisms and $2\lceil \log_2 \sqrt{P} \rceil$ additions.

The performance of both methods could be further improved with the *hoisting* technique [59,21], which speeds up FD around 6x and BS/GS around 1.4x for implementations that store ciphertexts in the NTT form (e.g. the HElib library). With hoisting, whether FD or BG/GS gives better performance depends on parameter settings, in particular the dimension P.

For $n > 1$, we notice the automorphisms of $\mathtt{ct}[\mathbf{v_i}]$ in FD can be re-used for multiplications with block matrices in the same block-column, i.e. $\mathbf{A_{0,i}}, \mathbf{A_{1,i}}, \dots, \mathbf{A_{n-1,i}}$. As such, the computation of (3) requires $n(P-1)$ automorphisms, n^2P `pt-ct` multiplications, and $n(nP-1)$ additions. Notice that for BSGS, only the baby step automorphisms can be re-used and so FD can be more efficient for large n. Furthermore, if $\mathbf{A}$ is a sparse matrix with h non-zero elements per row, the expected number of operations can be computed by using the following lemma.

Lemma 1 (Adapted from [76]). *Given n elements grouped into m equal partitions, the expected number of partitions hit when sampling k distinct elements $(k \leq n - n/m)$ is given by*

$$\mathit{PartsHit}(n,m,k) = m \cdot \left[1 - \binom{n-\frac{n}{m}}{k} \Big/ \binom{n}{k}\right].$$

Then, the homomorphic evaluation of (3) requires $n \cdot \mathsf{PartsHit}(nP^2, nP, hP)$ `pt-ct` multiplications, $n \cdot \mathsf{PartsHit}(nP(P-1), P-1, h(P-1))$ automorphisms, and $n \cdot (\mathsf{PartsHit}(nP^2, nP, hP) - 1)$ additions.

For our purposes, it will always be more efficient to perform the sparse matrix-vector multiplications with a relatively large n using FD. On the other hand, the matrix-vector products performed in 2D-NTTs always use BSGS (not considering hoisting).

6.2 Computing Fractal blindly

Fractal is a transparent, post-quantum, preprocessing zkSNARK that proves the Rank-1 Constraint Satisfiability (R1CS) $Az \circ Bz = Cz$ of $z = (x, w)$, where x is the statement, w is the (extended) witness, and A, B, C are sparse matrices that represent the computation to be proven. Its construction starts from a type of

IOP named Reed Solomon encoded-holographic IOP (RS-hIOP) that is compiled into an hIOP. In a RS-hIOP, an indexer provides RS codes in an offline phase, the prover's messages are RS codes and the verifier outputs a set of rational constraints on these RS codes. A rational constraint on some RS codes checks that some rational function of the underlying polynomials has a limited degree. For proofs of invalid statements, at least one of these rational constraints will not hold.

Denote by $(f_z, f_{Az}, f_{Bz}, f_{Cz})$ the polynomials that interpolate the vectors (z, Az, Bz, Cz) over some cyclic subgroup H of $\mathbb{F}$; then, the prover's first messages will be the RS codes $(\overrightarrow{f_z}, \overrightarrow{f_{Az}}, \overrightarrow{f_{Bz}}, \overrightarrow{f_{Cz}})$ over some domain $L = \{\ell_i\}_{i\in[|L|]}$. In particular, the RS codes correspond to evaluations of these polynomials on some evaluation domain L and are used to prove three statements that together imply the satisfiability of the R1CS constraint system:

(1) $f_{Az}\big|_H \circ f_{Bz}\big|_H - f_{Cz}\big|_H = 0$
(2) $f_z\big|_I = f_x\big|_I = x$ for some subset I of H
(3) $f_{Mz}\big|_H = M \cdot f_z\big|_H$ for $M \in \{A, B, C\}$.

The previous three statements are proven using some rational constraints on $\overrightarrow{f_z}, \overrightarrow{f_{Az}}, \overrightarrow{f_{Bz}}, \overrightarrow{f_{Cz}}$ and other RS codes derived from z. In our setting, however, the blind prover will have as input the HE encrypted (extended) witness $\texttt{ct}[w]$, leading to the (partially) encrypted trace $\texttt{ct}[z] = (x, \texttt{ct}[w])$. Therefore, all RS codes derived from the trace, as well as all subsequent prover messages required for proving rational constraints, will similarly be encrypted and from now on referred to as Encrypted RS (ERS) codes. Computing them efficiently is a trade-off between minimizing the homomorphic depth and minimizing the execution time (determined by the number of required homomorphic operations and the parameters of the HE system).

6.3 Proving statement (1)

Starting from the encrypted trace $\texttt{ct}[z] = (x, \texttt{ct}[w])$, the prover computes the ERS codes $\texttt{ct}[\overrightarrow{f_{Mz}}]$ for $M \in \{A, B, C\}$. Computing the underlying $\texttt{ct}[Mz] = \texttt{ct}[f_{Mz}\big|_H]$ requires a sparse matrix-vector product of size $|H|$ over ciphertexts. These ciphertext vectors are evaluations on domain H of some polynomial and computing the corresponding ERS codes amounts to evaluating them on some domain L. This is referred to as domain extensions and they are usually implemented using an inverse NTT transform to compute $f(x)$ from $f\big|_H$, followed by an NTT transform to compute $f\big|_L = \vec{f}$. This would result in only $\mathcal{O}(|L| \log |L|)$ operations but require a depth of $2 \log |L|$ `pt-ct` multiplications. Instead one could significantly reduce the `pt-ct` depth by computing the extension from $\{f^{(i)} = f(h^{i-1})\}_{i\in[|H|]}$ where $H = \{h^{i-1}\}_{i\in[n]}$ to domain L by using the barycentric form

$$f(\ell_i) = \sum_{j\in[n]} f^{(i)} \lambda_j^H(\ell_i) = Z_H(\ell_i) \sum_{j\in[n]} \frac{f^{(i)}}{Z'_H(h^{j-1})(\ell_i - h^{j-1})} = \frac{\ell_i^n - 1}{n} \sum_{j\in[n]} f^{(i)} \frac{h^{j-1}}{\ell_i - h^{j-1}}$$

where $i \in [|L|]$. However, even when using the method previously described for homomorphic matrix-vector multiplication, the large number of operations would lead to an unrealistic execution time. A naive hybrid algorithm would perform the first layers of the NTT as matrix-vector products and the remaining layers using the traditional butterfly algorithm (possibly in some other base b). This seems like a trade-off between homomorphic depth and execution time but it has one major problem. The butterfly algorithm would require us to move around elements between slots in different ciphertexts which would be very costly. Therefore we homomorphically evaluate the NTT in 2 dimension as described in Section 6.1. Only the dimension orthogonal to the ciphertext packing uses the butterfly algorithm in some base b and thus only causes permutations over complete ciphertexts which can be reverted by simply permuting rows of the matrix in Fig. 5. Notice that we can still trade off noise depth for execution time by adjusting the base b and the number of ciphertexts per row w.

Now that one has all the relevant ERS codes, one can prove statement (1) by proving the rational constraint $\deg(s) \leq |H| - 2$ for

$$s(X) = \frac{f_{Az}(X) f_{Bz}(X) - f_{Cz}(X)}{Z_H(X)}.$$

Notice that the numerator of s will vanish on H if and only if statement (1) holds. Note that the prover does not have to send $\mathtt{ct}[\vec{s}] = \mathtt{ct}[s\big|_L]$ since any of its elements can be efficiently computed at verification time. This is because the verifier is provided with the required $\mathtt{ct}[\vec{f}_{Mz}]$ and can compute the vector $Z_H\big|_L$ efficiently.

6.4 Proving statements (2) and (3)

Starting from the encrypted trace $\mathtt{ct}[z] = (x, \mathtt{ct}[w])$, the prover computes the ERS code $\mathtt{ct}[\vec{f_w}]$ that corresponds to a polynomial f_w of degree $|H| - |I| - 1$ such that

$$\forall a \in H \setminus I : f_w(a) = \frac{\mathtt{ct}[w]_{\mathsf{Ind}(a)} - f_x(a)}{Z_I(a)}$$

where $\mathsf{Ind} : H \setminus I \to [|H \setminus I|]$ indexes $H \setminus I$, the polynomial f_x interpolates the statement x over I and Z_I is the vanishing polynomial over I. This can be computed using the domain extension described above. From $\mathtt{ct}[\vec{f_w}]$ the prover (and in verification the verifier) can derive the RS code $\mathtt{ct}[\vec{f_z}]$ such that $f_z\big|_H = f_w\big|_H \circ Z_I\big|_H + f_x\big|_H = (x, w)$. Computing the ciphertext vector that f_w interpolates requires one element-wise pt-ct multiplication and addition.

In order to prove statements (2) and (3), i.e. that $f_{Mz}\big|_H = M \cdot f_z\big|_H$ for $f_z(X) = f_w(X) Z_I(X) + f_x(X)$, the Fractal protocol performs a "holographic lincheck" [37]. We only discuss a subprotocol of the holographic lincheck, named the "polynomial sumcheck", since only this particular part would require homomorphic computations in the blind setting. It is a univariate sumcheck protocol that is used to prove $\sum_{b \in H} f_{\mathsf{sc}}(b) = 0$ for

$$f_{\mathsf{sc}}(X) = \alpha(X) f_z(X) + \sum_{M \in \{A,B,C\}} \beta_M(X) f_{Mz}(X) = 0 \tag{4}$$

where α and β_M are polynomials such that f_{sc} is of degree $2|H| - 2$. In the sumcheck protocol the prover sends an RS code of the degree $|H|-2$ polynomial $Xg = f_{\mathsf{sc}} \mod Z_H$, and the verifier checks the rational constraint $\deg(h) \leq |H| - 2$ where $h(X) = \frac{f_{\mathsf{sc}}(X) - Xg(X)}{Z_H(X)}$. Notice that there is no need to send ERS codes for $f_{\mathsf{sc}}(X)$ and $Xg(X)$ since the verifier can efficiently derive any of their elements from the ERS codes $\mathtt{ct}[\overrightarrow{f_z}], \mathtt{ct}[\overrightarrow{f_{Az}}], \mathtt{ct}[\overrightarrow{f_{Bz}}], \mathtt{ct}[\overrightarrow{f_{Cz}}]$ and $\mathtt{ct}[\overrightarrow{g}]$.

Let us now discuss the computation of the ERS code $\mathtt{ct}[\overrightarrow{g}]$ starting from the previously derived ciphertexts. Similar to domain extension one could minimize the homomorphic depth by expressing this computation as one matrix-vector product as follows. First, notice that $g(\ell_i) = \sum_{j \in [|H|]} r_j \ell_i^{j-2}$ where r_j are the coefficients of $r = f_{\mathsf{sc}} \mod Z_H$. Now, w.l.o.g., assume that $\deg(f)+1 = k|H| = kn$ where H is the cyclic subgroup of $\mathbb{F}$ of size n. Then, we have that $Z_H(X) = X^n - 1$ and therefore $r_j = \sum_{s=0}^{k-1} \mathsf{Coeff}(f_{\mathsf{sc}})_{sn+j}$. Lastly, we compute these coefficients $\mathsf{Coeff}(f_{\mathsf{sc}})_i = \sum_{j \in [|L|]} \Lambda_{ij} f(\ell_j)$ where Λ_{ij} are the coefficients of the Lagrange polynomials such that $\lambda_j^L(X) = \sum_{i \in [|L|]} \Lambda_{ij} X^{i-1}$. Therefore, $\overrightarrow{g}$ and similarly $\mathtt{ct}[\overrightarrow{g}]$ could be computed as

$$g(\ell_i) = \sum_{t \in [|L|]} f_{\mathsf{sc}}(l_t) \sum_{j \in [|L|]} \ell_i^{j-2} \sum_{s=0}^{k-1} \Lambda_{sn+j,t} \qquad \text{for } i \in [|L|]$$

which would require $|L|^2$ `pt-ct` multiplications and $|L| \log |L|$ additions. As was the case for domain extension in the barycentric from, we will have to lower the number of required operations in exchange for a larger homomorphic depth. We propose the following algorithm.

Computing the ERS code of g

1 : $\{\alpha\big|_L, \beta_A\big|_L, \beta_B\big|_L, \beta_C\big|_L\} = \mathsf{NTT}(\alpha, \beta_A, \beta_B, \beta_C)$
2 : $\mathtt{ct}[\overrightarrow{f_{\mathsf{sc}}}] = \alpha\big|_L \circ \mathtt{ct}[\overrightarrow{f_z}] + \sum_{M \in \{A,B,C\}} \beta_M\big|_L \circ \mathtt{ct}[\overrightarrow{f_{Mz}}]$
3 : $\mathtt{ct}[\mathsf{Coeff}(f_{\mathsf{sc}})] = \mathsf{iNTT}(\mathtt{ct}[\overrightarrow{f_{\mathsf{sc}}}])$
4 : **foreach** $i \in [n]$:
5 : $\quad \mathtt{ct}[\mathsf{Coeff}(Xg)_i] = \sum_{s=0}^{k-1} \mathtt{ct}[\mathsf{Coeff}(f_{\mathsf{sc}})_{sn+i}]$
6 : $\mathtt{ct}[\overrightarrow{Xg}] = \mathsf{NTT}(\mathtt{ct}[\mathsf{Coeff}(Xg)])$
7 : $\mathtt{ct}[\overrightarrow{g}] = \mathtt{ct}[\overrightarrow{Xg}] \circ \left[l_1^{-1} \; l_2^{-1} \; \ldots \; l_{|L|}^{-1} \right]$

Again we minimize the homomorphic depth of the ciphertext space NTTs on lines 3 and 4 as described before. Notice that we can reuse the domain evaluations of f_z

and f_{Mz} since we will always have $|L| \geq \deg(f_{\mathsf{sc}})$. Remark that the computation on line 5 will have to be performed on ciphertexts in the column-major packing order. We note that this computation still only requires homomorphic additions if the width of the packing used in the iNTT of line 3 is divisible by $\deg(f_{\mathsf{sc}})/|H|$.

6.5 Proving rational constraints

We have so far shown how to blindly compute the Fractal RS-hIOP while Section 5 and specifically Theorem 8 only apply to hIOPs. The computations involved in the compilation from the Fractal RS-hIOP to the hIOP also require homomorphic operations when this hIOP is computed blindly. This compilation utilizes the FRI IOP [9] for checking the rational constraints. We perform this protocol batched, as first described by the authors of Aurora [11]. In batched FRI, one checks whether the rational constraint $f_{\mathsf{FRI}}(X) = \sum_i (\alpha_i + \beta_i X^{d-d_i}) f_i(X)$ has degree $d = \max_i\{d_i\}$ where α_i, β_i are some random challenges provided by the verifier instead of checking whether each rational constraint f_i is of degree d_i. Computing the ERS code for the batched rational constraint $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}}}]$ requires one $\mathtt{pt}$-$\mathtt{ct}$ multiplication.

For the Fractal RS-hIOP, the set of polynomials $\{f_i\}$ will be equal to the following set of rational constraints

$$\left\{ \frac{f_{Az} f_{Bz} - f_{Cz}}{Z_H}, g, \frac{f_{\mathsf{sc}} - Xg}{Z_H}, f_z, f_{Az}, f_{Bz}, f_{Cz}, f_{\mathtt{pt}} \right\}$$

where $f_{\mathtt{pt}}$ is a polynomial that interpolates plaintext values and has therefore not been discussed. A linear combination of these polynomials can be rewritten as a linear combination over the set $\{\mathfrak{f}_i\}$ equal to

$$\left\{ f_{Az} f_{Bz}, g, f_{\mathsf{sc}}, Xg, f_z, f_{Az}, f_{Bz}, f_{Cz}, f_{\mathtt{pt}} \right\}.$$

By this we mean that $\mathtt{ct}[\overrightarrow{s}]$ can be computed using element-wise homomorphic operations on $\mathtt{ct}[\overrightarrow{\mathfrak{f}_i}]$. Every $\mathtt{ct}[\overrightarrow{\mathfrak{f}_i}]$ has been previously computed except $\mathtt{ct}\overrightarrow{f_{Az} f_{Bz}}$ which requires depth of one $\mathtt{ct}$-$\mathtt{ct}$ multiplication. In the FRI IOP, the prover interacts with the verifier in approximately $\log d$ rounds. Let us assume that the evaluation domain L is a multiplicative coset of some cyclic subgroup such that $L = \{g\omega^i\}_{i \in [2^k]}$ for $k = \log_2(|L|)$ and g a field element. In round $j \in [\log_2 d]$, the prover sends the folded evaluation $\{f_{\mathsf{FRI}/2^j}((g\omega^i)^{2^j})\}_{i \in [2^{k-j}]}$ of the degree $d/2^j$ polynomial $f_{\mathsf{FRI}/2^j}$ where

$$f_{\mathsf{FRI}/2^j}((g\omega^i)^{2^j}) = \frac{1 + \alpha_j (g\omega^{-i})^{2^j}}{2} f_{\mathsf{FRI}/2^{j-1}}((g\omega^i)^{2^{j-1}}) + \frac{1 - \alpha_j (g\omega^{-i})^{2^j}}{2} f_{\mathsf{FRI}/2^{j-1}}((g\omega^{2^{k-1}+i})^{2^{j-1}})$$

and α_j is the j-th round verifier challenge. After the last round, the prover sends the remaining $|L|/d$ evaluations to the verifier, who checks that they are colinear. Now in the blind setting, we propose to stop FRI at round $k - p$ (where 2^p elements can be encrypted in a single ciphertext), since this would only amount to sending one ciphertext, the minimal number we can send. It is clear that the computation of $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}/2^j}}]$ from $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}/2^{j-1}}}]$ would require 2 element-wise $\mathtt{pt}$-$\mathtt{ct}$ multiplications on vectors of size 2^{k-j}. As was also noticed by [5], we can trade off the number of operation for homomorphic depth by composing multiple rounds of FRI. In our case we fully compose all FRI rounds.

7 Proof of Decryption

The proof of decryption (PoD) is a key component to build a publicly-verifiable blind zkSNARK, as explained in Section 5.3. In this section, we construct PoDs from our vectorized description of the LNP22 proof system, which is described in detail in Appendix C.

All RLWE-based HE schemes such as BFV [23,46] and the Generalized-BFV (GBFV) scheme [50], but also BGV [24] and CKKS [33], fit in a general framework: the secret key $\mathtt{sk} \in \chi_{key}$ is an element of small norm in $\mathcal{R}_{m,q}$ and a ciphertext $(c_0, c_1) \in \mathcal{C} = \mathcal{R}_{m,q}^2$ encrypts a message $m \in \mathcal{P} = \mathcal{R}_m / \mathcal{I}$ for some ideal $\mathcal{I} \subset \mathcal{R}_m$. For invariant schemes such as GBFV, we have that $\mathcal{I} = (t)$ and the decryption equation is given by

$$c_0 + c_1 \cdot \mathtt{sk} = \lfloor \Delta \cdot m \rceil + v_{inh} \in \mathcal{R}_{q,m} \tag{5}$$

where $\Delta = q/t \in \mathcal{K}_m$ is a scaling factor and v_{inh} is called the inherent noise, i.e. the polynomial with the lowest infinity norm such that the above equation holds. Furthermore, the ciphertext will decrypt correctly as long as the modulus $q \gg B_t := \|t\|_\infty^{\mathrm{can}}$ and $\|v_{inh}\|_\infty < \mathcal{B}_q := \frac{q}{2 \cdot EF_m \cdot h_t \cdot \|t\|_\infty} - \frac{1}{2}$, where h_t is the number of non-zero terms in $t(X)$ and the bound is proven in Appendix B.2. For other schemes such as BGV and CKKS, a slight variation of the above equation describes valid decryption; in particular, in all cases, valid decryption is given by a relation over the ring $\mathcal{R}_{q,m}$, which is linear in the secret key $\mathtt{sk}$ and with the requirement that $\|v_{inh}\|_\infty < \mathcal{B}_q$ for some bound $\mathcal{B}_q$ depending on the parameters of the scheme.

7.1 Relations for the proof of decryption

Let $\mathtt{C_{sk}}$ denote a commitment to a secret key $\mathtt{sk} \in \chi_{key}$. For $1 \le i \le \mathrm{r}$, let $\mathtt{ct}^{(i)} = (c_0^{(i)}, c_1^{(i)}) \in \mathcal{R}_{m,q}^2$ denote a ciphertext that decrypts to $m^{(i)}$ under the secret key $\mathtt{sk}$. Since valid decryption requires the norm of the inherent noise $v_{inh}^{(i)}$ in each ciphertext $\mathtt{ct}^{(i)}$ to be bounded by $\mathcal{B}_q$, we can derive the relation:

$$\mathbf{R_1} = \left\{ \begin{pmatrix} x = (\mathtt{C_{sk}}, \{\mathtt{ct}^{(i)}, m^{(i)}\}_{i\in[\mathrm{r}]}) \\ w = (\mathtt{sk}) \end{pmatrix} \middle| \begin{array}{l} \mathcal{O}^{\mathcal{CT}}(\mathtt{C_{sk}}, \mathtt{sk}) = \mathsf{acc} \\ \wedge \forall i \in [\mathrm{r}] : \left\| v_{inh}^{(i)} \right\|_\infty < \mathcal{B}_q \text{ where} \\ v_{inh}^{(i)} := c_0^{(i)} + c_1^{(i)} \cdot \mathtt{sk} - \lfloor \Delta \cdot m^{(i)} \rceil \end{array} \right\} \tag{6}$$

Any statement-witness pair in $\mathbf{R_1}$ gives r valid plaintext-ciphertext pairs in RLWE-based HE with respect to the secret key committed to in $\mathtt{C_{sk}}$.

In our work, $\mathtt{C_{sk}}$ is instantiated using the ABDLOP commitment scheme. For messages committed under ABDLOP, the LNP22 proof system (described in detail in Appendix C) allows proving various relations over the commitment ring $\mathsf{R}_{q''}$. This includes Approximate Norm bound proofs (ANP) of linear relations in the commitment ring $\mathsf{R}_{q''}$, as detailed in Appendix C.4. While it may seem promising to apply ANP directly to prove the boundness of inherent noises in ciphertexts, the commitment ring $\mathsf{R}_{q''}$ in the LNP22 proof system differs from the HE ciphertext ring $\mathcal{R}_{m,q}$ in two aspects.

- Firstly, the LNP22 commitment ring is defined by a power-of-two cyclotomic polynomial, typically of degree $d = 64, 128$. In the above HE schemes, the ring $\mathcal{R}_m$ is defined modulo the m-th cyclotomic polynomial where m is much larger than 128 and also not necessarily a power of two.
- Secondly, in LNP22, the modulus $q'' = \prod q''_i$ is chosen such that the cyclotomic polynomial $X^d + 1$ has two irreducible factors modulo q''_i. So even in the case where Φ_m would be a power-of-two cyclotomic polynomial, the ciphertext modulus in HE is chosen such that Φ_m fully splits modulo each prime factor of the ciphertext modulus.

To accommodate the first incompatibility, we first represent elements and relations in the ciphertext ring $\mathcal{R}_{m,q}$ as relations on vectors over $\mathbb{Z}_q$ using the coefficient embedding. Thus, the relations for the inherent noises are given by

$$\vec{v}^{(i)}_{inh} = \mathsf{Rot}_{m,q}(c_1^{(i)}) \cdot \vec{\mathtt{sk}} + \vec{c}^{(i)}_0 - \overrightarrow{\left\lfloor \Delta \cdot m^{(i)} \right\rceil} \in \mathbb{Z}_q^n,\ \forall i \in [\mathrm{r}]. \tag{7}$$

In order to prove the boundedness of $\vec{v}^{(i)}_{inh}$, we describe a vectorized version of ANP in Appendix C.5, which is referred to as vec-ANP.

As for the second incompatibility, a natural solution to accommodate different moduli is to include overflows, as in [68, Section 6.3]. Concretely, for a sufficiently large modulus q'', there exist bounded overflows $\{\vec{\ell}^{(i)},\ \forall i \in [\mathrm{r}]\}$ satisfying

$$\vec{v}^{(i)}_{inh} = \mathsf{Rot}_{m,q}(c_1^{(i)}) \cdot \vec{\mathtt{sk}} + \vec{c}^{(i)}_0 - \overrightarrow{\left\lfloor \Delta \cdot m^{(i)} \right\rceil} + q\,\vec{\ell}^{(i)} \in \mathbb{Z}_{q''}^n,\ \forall i \in [\mathrm{r}]. \tag{8}$$

Since inherent noises and overflows are not independent linear combinations of $\vec{\mathtt{sk}}$, proving their bounds would require us to commit to at least one of the two. This not only increases the commitment size, but also requires a higher modulus $q'' > q$ than HE ciphertexts.

To avoid this blow-up, we use a well known technique from HE, namely modulus switching, which allows transforming a valid ciphertext modulo q into a valid ciphertext modulo q'', where q'' is taken to be lower than q in our protocols. Let $\mathtt{ct}[m] = (c_0, c_1) \in \mathcal{R}^2_{m,q}$ denote a ciphertext with ciphertext modulus q and inherent noise v_{inh}. Switching the ciphertext modulus to q'' amounts to computing

$$\mathtt{ct}' = \left(\left\lfloor \frac{q''}{q} c_0 \right\rceil, \left\lfloor \frac{q''}{q} c_1 \right\rceil \right) \in \mathcal{R}^2_{m,q''}\,.$$

In Appendix B.3, we derive the noise bound in $\mathtt{ct}'$ as $\|v'_{inh}\|_\infty \leq \frac{q''}{q}\|v_{inh}\|_\infty + \mathcal{B}_{ms}$, where $\mathcal{B}_{ms}$ is a constant depending on the secret key distribution. As long as we have $\|v'_{inh}\|_\infty \leq \mathcal{B}_{q''}$, the ciphertext $\mathtt{ct}'$ will be valid and thus satisfies the same equation as (7), but with q'' instead of q.

7.2 Relaxed proof of decryption

For modulus switched ciphertexts $\{\mathtt{ct}'^{(i)} \in \mathsf{R}^2_{m,q''}, i \in [\mathrm{r}]\}$, the proof of decryption amounts to proving the relation

$$\mathbf{R_2} = \left\{ \begin{pmatrix} x = (\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}'^{(i)}, m^{(i)}\}_{i\in[\mathrm{r}]}) \\ w = (\mathtt{sk}) \end{pmatrix} \middle| \begin{array}{l} \mathcal{O}^{\mathcal{CT}}(\mathtt{C}_{\mathtt{sk}}, \hat{\mathtt{sk}}) = \mathsf{acc} \\ \wedge \forall i \in [\mathrm{r}] : \left\|\vec{v}^{(i)}_{inh}\right\|_\infty < \mathcal{B}_{q''} \text{ where} \\ \vec{v}^{(i)}_{inh} := \mathsf{Rot}_{m,q''}(c_1'^{(i)}) \cdot \vec{\mathtt{sk}} + \vec{c}_0'^{(i)} - \overrightarrow{\lfloor \Delta \cdot m^{(i)} \rceil} \end{array} \right\},$$

where $\hat{\mathtt{sk}}$ denotes an embedding of $\mathtt{sk}$ into the message space of the commitment scheme $\mathcal{CT}$. In the instantiation of the ABDLOP scheme, for an element $v \in \mathcal{R}_m$, we define its embedding $\hat{\mathbf{v}} \in \mathsf{R}^{\hat{n}}_{q''}$ as $\hat{n} = \lceil \frac{n}{d} \rceil$ elements in the commitment ring $\mathsf{R}_{q''}$, such that the coefficient vector of $\hat{\mathbf{v}}$ equals $\vec{v}$ modulo q''.

In this section, we describe a protocol $\mathsf{PoD}(\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}'^{(i)}, m^{(i)}\}_{i\in[\mathrm{r}]})$ using vec-ANP, which is complete for ciphertexts whose inherent noise satisfy $\left\|\vec{v}^{(i)}_{inh}\right\|_\infty < B_{\mathsf{PoD}}$, where

$$B_{\mathsf{PoD}} := \min\left\{ \frac{\mathcal{B}_{q''}}{\psi^{(L2)}\sqrt{\mathrm{r} \cdot n}}, \frac{q''}{41(\mathrm{r} \cdot n)^{3/2}\psi^{(L2)}} \right\}$$

and the factor $\psi^{(L2)}$ is defined in Appendix C.3. In other words, our protocol is a *relaxed* proof of decryption with a relaxation factor

$$\Phi_r := \mathcal{B}_{q''} / B_{\mathsf{PoD}} \approx \psi^{(L2)}\sqrt{\mathrm{r} \cdot n} \cdot \max\left\{1, \frac{41\,\mathrm{r} \cdot n}{2\delta_m\|t\|_\infty}\right\}.$$

The protocol. To begin with, the prover commits to the secret key $\mathtt{sk}$ using the Ajtai part of the ABDLOP commitment scheme, i.e. $\mathtt{C}_{\mathtt{sk}} = \mathbf{A_1} \cdot \hat{\mathtt{sk}} + \mathbf{A_1} \cdot \mathbf{s_2}$ where $\mathbf{s_2} \in \mathsf{R}^{m_2}_{q''}$ is a small randomness satisfying $\|\mathbf{s_2}\|_\infty \leq \nu$.

To generate a proof of decryption for r ciphertext-plaintext pairs whose inherent noises are bounded by B_{PoD}, the prover applies the vec-ANP protocol with inputs

$$\Pi_{\text{vec-ANP}}\left((\mathbf{s_1} = \hat{\mathtt{sk}}, \mathbf{m} = \emptyset, \mathbf{s_2}), (\mathbf{W}, \mathbf{w}, B_w = B_{\mathsf{PoD}})\right),$$

where

$$\mathbf{W} = \begin{bmatrix} \mathsf{Rot}_{m,q''}(c_1^{(1)}) \\ \vdots \\ \mathsf{Rot}_{m,q''}(c_1^{(\mathrm{r})}) \end{bmatrix} \in \mathbb{Z}^{\mathrm{r}\cdot n \times n}_{q''},\ \mathbf{w} = \begin{bmatrix} \vec{c}_0^{(1)} - \overrightarrow{\lfloor \Delta \cdot m^{(1)} \rceil} \\ \vdots \\ \vec{c}_0^{(\mathrm{r})} - \overrightarrow{\lfloor \Delta \cdot m^{(\mathrm{r})} \rceil} \end{bmatrix} \in \mathbb{Z}^{\mathrm{r}\cdot n}_{q''}.$$

Denote the vector $\mathbf{W}\cdot\vec{\mathbf{sk}}+\mathbf{w} = \left[\vec{v}_{inh}^{(1)} \cdots \vec{v}_{inh}^{(\mathrm{r})}\right]^\top$ as $\vec{\mathbf{u}}$; then, the above vec-ANP protocol convinces the verifier that the prover knows $\hat{\mathtt{sk}}$ such that $\mathcal{O}^{\mathcal{CT}}\left(\mathtt{C}_{\mathtt{sk}}, \hat{\mathtt{sk}}\right) = \mathsf{acc}$ and $\|\vec{\mathbf{u}}\|_\infty \le B_{\mathsf{PoD}}\cdot\psi^\infty \le \mathcal{B}_{q''}$. This guarantees the validity of each ciphertext-plaintext pair with respect to the secret key committed in $\mathtt{C}_{\mathtt{sk}}$.

Asymptotic Analysis. With ABDLOP parameters ensuring sufficient hardness of MSIS (for binding) and MLWE (for hiding), the protocol $\mathsf{PoD}(\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}'^{(i)}, m^{(i)}\}_{i\in[\mathrm{r}]})$ achieves a constant amortized proof size (including commitment size, without applying the Huffman coding optimization [68]) with respect to the number of ciphertext-plaintext pairs r.

The computation cost is dominated by a subprotocol $\Pi_{eval}^{(2)}(\cdot)$, where both the prover and the verifer need to compute the function H_j, as detailed in Section C.5. This results in $\mathcal{O}\big(\mathrm{r}\,n^2\big)$ computation costs. In Section 7.3, we describe a protocol that achieves computation cost $\mathcal{O}\big(n^2 + \mathrm{r}\,n\log n\big)$.

7.3 Reducing the computation costs

In Figure 6, we describe another batched proof of decryption protocol that has a reduced computation cost compared to the protocol from Section 7.2. Instead of having the linear relation in the vec-ANP proof grow with the number of ciphertexts r, we prove the decryption of a random linear combination of ciphertexts. The soundness of the protocol is based on the Schwartz-Zippel Lemma. Since the computations on the r ciphertexts are moved to the ring space, the computations are more efficient. In particular, we reduce the cost from $\mathcal{O}\big(\mathrm{r}\,n^2\big)$ to $\mathcal{O}\big(n^2 + \mathrm{r}\,n\log n\big)$. This comes with the change of relaxation factor from $\Phi_r = \mathcal{O}\Big((\mathrm{r}\,n)^{\frac{3}{2}}\Big)$ to $\Phi^{\mathrm{SZ}} = \mathcal{O}\Big(\mathrm{r}\,n^{\frac{5}{2}}\Big)$. Using the Fiat-Shamir transform, this protocol can be compiled into a non-interactive proof in the ROM.

Lemma 2. *Let $\mathcal{P} = \mathbb{F}^P$ denote the plaintext space of an HE scheme with P slots. If $\mathrm{r}\,/\,|\mathbb{F}| = \mathsf{negl}(\lambda)$, then the protocol in Figure 6 is a proof of decryption for* r *ciphertexts $\{\mathtt{ct}'[m^{(i)}], m^{(i)}\}_{i\in[\mathrm{r}]}$ with negligible soundness error and relaxation factor $\Phi^{SZ} := \Phi_1 \cdot N_{\mathtt{ptct}} \cdot 2^{\lceil\log \mathrm{r}\rceil}$ where $N_{\mathtt{ptct}} = \mathcal{O}(n)$ is the noise increase bound for 1 $\mathtt{pt}$-$\mathtt{ct}$ multiplication.*

Proof. We start by discussing soundness. Let us define a function $f : \mathcal{P}^{\mathrm{r}} \to \mathcal{P} : \{m^{(j)}\}_{i\in[\mathrm{r}]} \mapsto \sum_{i\in[\mathrm{r}]} \alpha_i m^{(i)}$ for some set of challenges $\{\alpha_i\}_{i\in[\mathrm{r}]}$ such that each α_i encodes P elements $\{\alpha_{ij}\}_{j\in[P]}$. The proof of decryption that is verified at the end implies that

$$f\left(\{m^{(i)}\}_{i\in[\mathrm{r}]}\right) = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathcal{E}.\mathsf{Eval}\left(f, \{\mathtt{ct}'[m^{(i)}]\}_{i\in[\mathrm{r}]}\right)\right)$$

except with negligible probability. Under the assumption that $\mathcal{E}$ is still correct for f on those ciphertexts, this implies that

$$\begin{aligned} f\left(\{m^{(i)}\}_{i\in[\mathrm{r}]}\right) &= f\left(\mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\{\mathtt{ct}'[m^{(i)}]\}_{i\in[\mathrm{r}]}\right)\right) \\ \Rightarrow \quad \sum_{i\in[\mathrm{r}]}\left(m^{(i)} - \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathtt{ct}'[m^{(i)}]\right)\right)\alpha_i &= 0 \\ \Rightarrow \forall j \in [P] : \sum_{i\in[\mathrm{r}]}\left(m_j^{(i)} - \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathtt{ct}'[m^{(i)}]\right)_j\right)\alpha_{ij} &= 0 \end{aligned}$$

where the subscript j denotes the j-th slot of a plaintext encoding. Now if the values α_{ij} were randomly sampled from $\mathbb{F}$, by the Schwartz-Zippel lemma we can conclude that for each $j \in [P]$ it holds that

$$\forall i \in [\mathrm{r}] : m_j^{(i)} = \mathcal{E}.\mathsf{Dec}_{\mathtt{sk}}\left(\mathtt{ct}'[m^{(i)}]\right)_j$$

except with probability $r/|\mathbb{F}|$. The relaxation factor Φ^{SZ} comes from the relaxation factor required for a PoD on one ciphertext multiplied by the noise factors added by homomorphically computing f. This ensures that the correctness assumption above holds. □

Prover		**Verifier**
	$\xrightarrow{\mathtt{C}_{\mathtt{sk}}, \{\mathtt{ct}'[m^{(i)}], m^{(i)}\}_{i\in[\mathrm{r}]}}$	
$\tilde{\mathtt{ct}} \coloneqq \sum_{i\in[\mathrm{r}]} \mathsf{ptMult}(\mathtt{ct}'[m^{(i)}], \alpha_i)$	$\xleftarrow{\vec{\alpha}}$	$\vec{\alpha} \leftarrow \mathbb{F}^{\mathrm{r}\times P}$
$\tilde{m} \coloneqq \sum_{i\in[\mathrm{r}]} \alpha_i \cdot m^{(i)}$		
$\pi^{\mathsf{PoD}} \leftarrow \mathsf{PoD.P}(\mathtt{C}_{\mathtt{sk}}, \tilde{\mathtt{ct}}, \tilde{m})$	$\xrightarrow{\pi^{\mathsf{PoD}}}$	$\tilde{\mathtt{ct}} \coloneqq \sum_{i\in[\mathrm{r}]} \mathsf{ptMult}(\mathtt{ct}'[m^{(i)}], \alpha_i)$
		$\tilde{m} \coloneqq \sum_{i\in[\mathrm{r}]} \alpha_i \cdot m^{(i)}$
		$\mathsf{PoD.V}(\pi^{\mathsf{PoD}}, \mathtt{C}_{\mathtt{sk}}, \tilde{\mathtt{ct}}, \tilde{m}) \stackrel{?}{=} \mathsf{acc}$

Fig. 6: A PoD protocol for $\{\mathtt{ct}'[m^{(i)}], m^{(i)}\}_{i\in[\mathrm{r}]}$ with reduced computation costs.

7.4 Proving decryptions of a subset

As discussed in Section 6, efficient instantiations of blind zkSNARKs rely on the SIMD capabilities of the used HE scheme. Therefore, when opening the commitment to a ciphertext and proving its decryption in order to reveal a queried value, we are instead revealing an entire batch of elements in the zkSNARK field. To ensure that the zero-knowledge property of the zkSNARK scheme extends to the blind zkSNARK scheme, we must avoid revealing more queried values than intended, by only revealing the queried values. We give two methods to do so.

Masking. The first method consists of simply masking the ciphertext using a plaintext-ciphertext multiplication, where the masking plaintext M encodes a 1 in the slots we want to reveal and a 0 in all other slots. Instead of giving a PoD $(\pi^{\mathsf{PoD}}, \mathtt{ct}, m)$ that some ciphertext $\mathtt{ct} = (c_0, c_1) \in \mathcal{R}^2_{q''}$ decrypts to a plaintext message $m \in \mathcal{P}$, we can simply replace the ciphertext by $\mathtt{ct}^* = \mathsf{ptMult}(\mathtt{ct}, M)$ and give a PoD $(\pi^{\mathsf{PoD}}, \mathtt{ct}^*, m^*)$ where $m^* = M \cdot m$.

Ring switching. The GBFV parameter sets we use in the blind zkSNARK are particular instances of a family of parameter sets. To illustrate this, consider the case where we want to encrypt elements in $\mathbb{F}_{p^2}$ with p the Golidlocks prime; then, the family consists of the following: the plaintext space is given by $\mathbb{Z}[x]/(\Phi_m(x), t(x))$ with $m = 7 \cdot 3 \cdot 2^j$ and $t(x) = x^k - b$, with $k = 7 \cdot 2^{i+j-6}$ and $b = 2^{2^i}$ for some integers $0 \leq i \leq 5$ and $6 \leq j \leq 16$. The blind zkSNARK is executed using the set $j = 11$ and $i = 0$, resulting in a plaintext space corresponding to a vector of 96 elements in $\mathbb{F}_{p^2}$. As explained above, we are only interested in a subset of these elements, which opens up the possibility to construct a valid ciphertext for a smaller parameter set in the same family encrypting only the subset of values we want to reveal. For this we can use a technique called ring switching [54] to map a valid ciphertext for $j = 11$ to the smaller ring defined by $j = 8$, which is the smallest dimension that ensures 100-bit security for the modulus q''' in the relaxed PoD. The resulting protocol can be found in Appendix D. The ring switch decreases the ciphertext size by a factor of 8, which speeds up the PoD by a factor of 64. A similar approach can be taken for the other parameter sets.

8 Implementation

In this section, we demonstrate the practicality of using our protocols in the vCOED setting and the zkDel setting. As discussed before, we select Fractal as the hIOP scheme and GBFV as the HE scheme. In what follows, we will focus on 100-bit security since this is the security level targeted by FRI-based zkSNARK implementations [17].

8.1 GBFV parameter sets

For the Fractal computation, we instantiate GBFV over the ring $\mathcal{R}_m$ for $m = 7 \cdot 3 \cdot 2^{11}$, resulting in the lattice dimension $n := \varphi(m) = 12288$. The ciphertext ring is $\mathcal{R}_{m,q}$ with ciphertext modulus $q \leq 382$ bits, and the plaintext polynomial is $t(X) = X^{7 \cdot 2^7} - 2^4$. The plaintext space is a vector space of dimension 384 over

$\mathbb{F}_{p^2}$, where $p = 2^{64} - 2^{32} + 1$ is the Goldilocks prime, widely used in zkSNARK implementations because of its efficient arithmetic.

Ciphertexts committed to in the BCS compilation are in a smaller ring $\mathcal{R}_{m',q'}$ where $m' = 7 \cdot 3 \cdot 2^9$ (dimension $n' := \varphi(m') = 3072$) and a 96 bit q'. For the PoD, we switch down further to an even smaller ring $\mathcal{R}_{m'',q''}$ where $m'' = 7 \cdot 3 \cdot 2^8$ (with dimension $n'' := \varphi(m'') = 1536$) and a 48 bit q''. All these HE parameters provide at least 100-bit security according to the lattice estimator by Albrecht et al. [3].

8.2 Computing Fractal blindly

We prove that the homomorphic circuit outlined in Section 6 is feasible in practice for computing blind proofs of computations with $\mathbf{C} = 2^{20}$ R1CS constraints. This is achieved by selecting parameters for Fractal and GBFV and then demonstrating the following facts:

- they are secure instantiations of an hIOP and HE scheme respectively,
- the HE scheme will remain correct for that homomorphic circuit,
- the number of required homomorphic computations is feasible.

Fractal is calculated in a field of size $\log_2 \mathbb{F} = 128$, thus the Fractal RS-hIOP will remain sound for circuit sizes up to approximately 2^{28} constraints. The maximal degree on which we will have to perform the FRI IOP will be approximately $\mathbf{C} = 2^{20}$. Therefore, from the recent paper by Block et al. [17], we can derive that FRI will remain secure for this field size when choosing rate $\rho = 1/2$ (so $|L| = \mathbf{C}/\rho = 2^{21}$) and performing $\ell = 101$ repetitions of the query phase. In Appendix D, we argue this implies opening to 3728 $\mathbb{F}_{p^2}$ elements on average.

Let us first discuss the practicality of encrypting and sending a circuit trace of size $\mathbf{C}$, as shown in Figure 4. We propose encrypting the trace into normal BFV ciphertexts and then unpacking them into GBFV ciphertexts on the server side. As described in [50], when instantiating BFV using the same cyclotomic polynomial Φ_m, this can be achieved using one automorphism and one `pt-ct` operation per resulting GBFV ciphertext. For the parameter set with $n = 12288$, the packing size for BFV is 6144 and for GBFV is 384. To facilitate the NTT computation in proving Fractal, we only use vectors of power-of-two sizes; hence we only use 4096 slots in BFV and 256 slots in GBFV. Therefore, the client needs to encrypt $\lceil 2^{20}/4096 \rceil = 256$ ciphertexts, which takes approximately 3.3s. The resulting communication size would be approximately 313MB. However, notice that these are actually upper bounds since one would likely not encrypt and send the entire trace but only the private inputs to the computation. Then, the server could compute the other trace values homomorphically, which might require bootstrapping.

The first operation performed by the server will be unpacking into GBFV ciphertexts. Next, the server computes Fractal as described in Section 6. For the inverse NTT required in domain extension, we choose base $b = 8$ and width

$w = 1$. For all other NTTs we have chosen base $b = 8$ and width $w = 2$. Regarding the FRI computation, we compose all rounds into one to maximally reduce noise depth, as in [5].

During the FRI procedure, the prover outputs GBFV ciphertexts of dimension $n = 12288$, where $P = 256$ slots are used in each ciphertext. Instead of committing to these ciphertexts in the BCS compilation, we perform rings switching to reduce the lattice dimension, resulting in GBFV ciphertexts of dimension $n' = 3072$, where $P' = 64$ slots are used in each ciphertext. As such, opening to 3728 $\mathbb{F}_{p^2}$ elements for the FRI query phase corresponds to opening 2514 dimension-3072 ciphertexts instead of opening 2105 dimension-12288 ciphertexts, as explained in Appendix D.

The number of operations and the noise consumption in each step are presented in Table 1. We also provide the script used to compute this table.[5] Note that we only report on the noise required in the "critical path". For example, in the third row, the reported noise is that of the ciphertexts $\mathtt{ct}[\overrightarrow{f_{Mz}}]$. Also, we reduce the required noise depth by combining subsequent pt-ct operations. For example, all consecutive pt-ct multiplications $\alpha_n(\ldots(\alpha_1 \cdot \mathtt{ct}))$ can be computed using one pt-ct multiplication $(\alpha_1 \cdots \alpha_n) \cdot \mathtt{ct}$.

Computation	Noise (bits)	$C_{\mathtt{add}}$	$C_{\mathtt{ptct}}$	$C_{\mathtt{aut}}$	$C_{\mathtt{ctct}}$
Unpacking	9	0	4096	4096	0
Computing $\mathtt{ct}[Mz]$	14	9421459	9433747	2978354	0
Computing $\mathtt{ct}[\overrightarrow{f_z}]/\mathtt{ct}[\overrightarrow{f_{Mz}}]$	64	4636672	4653056	491520	0
Computing $\mathtt{ct}[\overrightarrow{g}]$	138	6762496	6782976	552960	0
Computing $\mathtt{ct}[\overrightarrow{f_{\mathsf{FRI}}}]$	0	65536	65536	0	8192
Computing FRI	16	98305	106496	0	0
Ringswitching	9	0	196604	589812	0

Table 1: Operation count and noise estimates for computing blind Fractal. See implementation for more details.

Estimated performance for vCOED. In the vCOED setting, we choose $q =$ 375 bit as the starting ciphertext modulus, and apply dynamic scaling (i.e. modulus switching to lower ciphertext moduli) when computing blind Fractal. As described in Section 8.1, the ciphertexts committed to in BCS have dimension $n' = 3072$ and a 96 bit modulus q'. To decrease the proof size and remove

[5] https://github.com/KULeuven-COSIC/blind_zkSNARKs/blindFractal/estimates.m

ringswitching noise, we modswitch to a 60 bit modulus q''. The resulting ciphertexts have 9 bits of noise. Given that the GBFV decryption bound $\mathcal{B}_q \approx 51$ bits, this leaves a $51 - 9 = 42$ bit noise gap, which is sufficient to ensure circuit privacy. Circuit privacy is essential for blind hIOP to maintain zero-knowledge in vCOED. Applying noise flooding [13] (i.e. adding encryptions of zeros with large noise) to ciphertexts with more than 40-bit noise gap gives sufficient security for circuit privacy [42,22].

These noise-flooded ciphertexts are committed to in BCS compilation. The FRI-query phase requires opening on average 2514 GBFV ciphertexts (see Appendix D) of dimension $n' = 3072$ and 60-bit ciphertext modulus. This results in a 116MB proof size. To estimate the execution time for generating a blind Fractal proof, we use the operation counts in Table 1. Since GBFV is currently not implemented for non-power-of-two cyclotomics, we cannot get exact runtimes for the homomorphic operations of $n = 12288$. Therefore, we use the runtimes for the same operations in a power-of-two lattice dimension of larger size, namely 2^{14}, as presented in Table 3. Based on this, we estimate that computing blind Fractal completely sequentially for a circuit size of 2^{20} takes 9.26 hours. With a 32x speedup from parallelization, the expected runtime reduces to 17.4 min.

Estimated performance for zkDel. In the zkDel setting, we choose $q = 382$ bit as the starting ciphertext modulus, and also apply dynamic scaling when computing blind Fractal. As such, the resulting ciphertext has 35.7-bit noise within a 96 bit ciphertext modulus. These ciphertexts are committed to in BCS compilation. Note that in the zkDel setting, circuit privacy is required from the HE scheme. Using the same estimation strategy as in vCOED, the computation of blind Fractal takes 10.9 hours sequentially and 20.5 minutes assuming a 32 parallelization factor.

The FRI-query phase requires opening 2514 GBFV ciphertexts of dimension $n' = 3072$ and 96-bit ciphertext modulus. This results in a 186MB server-to-client communication. Furthermore, the client prover needs to generate a PoD for 2514 masked ciphertexts. Note that since masking is deterministic, we can perform the linear combinations from Figure 6 along with the masking. This results in a ciphertext with 52.9 bits of noise.

To make the PoD protocol less costly, the resulting ciphertext is ringswitched further to dimension $n'' = 1536$ in the PoD. Before this ringswitching, we need to perform an homomorphic trace operation to aggregate messages of the 64 relevant slots in dimension $n' = 3072$ into the 32 remaining slots, and a modulus switching to gaurantee the 100-bit security. Therefore, the protocol from Figure 6 was eventually performed for 32 slots. Before ringswitching to n'', we modswitch to a ciphertext modulus q'' of 48 bits, where q'' is compatible with the requirements of the LNP22 proof system; this results in a noise term of about 5 bits. Finally, we ringswitch down further to $n'' = 1536$, which increases the noise to 14.1 bits. Since the resulting noise is below the threshold $B_{PoD} = 16.9$ bits in Appendix D.1, the final ciphertext-plaintext pair can be proven from our instantiation of the PoD protocol.

Discussion. Due to the lack of GBFV implementation for non power-of-2 cyclotomics, we were unable to provide an implementation for the blind Fractal computation. Instead, we count the number of operations and estimate the runtime using lattice dimension 2^{14} which is $4/3$ times larger than our actual dimension 12288. This conservative approach leaves a margin for the fact that HE operations are slightly slower in non-power-of-two cyclotomic ciphertext space. Furthermore, we also provide runtime estimates with 32x speedup from parallelization. We think it is a reasonable assumption, since all operations in our algorithm are highly parallelizable and the server is expected to have powerful computational resources. Furthermore, the peak memory usage during the blind Fractal computation remains smaller than 100GB.

Comparison with HELIOPOLIS. It is not trivial to compare our estimation to [5] since they only implement blind FRI and choose $\log|\mathbb{F}| \approx 256$ bit field size. Thus, we make a new estimation[6] for only computing FRI using the parameters $m = 2^{14}$ and $t(X) = X^{2^8} - 332$ which results in a HE plaintext space of $\mathbb{F}_p^P$ for $P = 256$ and p approximately 268 bits. Using the same method as before, we estimate a sequential runtime of 71s and a parallelized runtime of 2.2s assuming a 32x speedup. In comparison, the implementation of [5] requires 207s runtime on a 32-thread machine. While achieving a 32x speedup on a a 32-thread machine may not be realistic (despite our method being highly parallelizable), our estimated runtime still demonstrates a significant improvement for blind FRI. Furthermore, we also compare to the operation count required for their NTT algorithm for polynomials of degree 2^{18}, as described in Figure 7 in Appendix E. Our blind 2D-NTT only requires 584704 operations in $n = 2^{13}$ while their blind NTT requires at least 10^7 operations in $n = 2^{12}$. Even though their approach can batch blind evaluation of FRI on many polynomials, we note that in practice FRI is always batched using a random linear combination (as described in Section 6.5) to decrease the proof size.

8.3 Proof of Decryption

We implemented the proof-of-decryption protocol presented in Section 7 using the C programming language[7]. We leveraged basic primitives used in Lazer [72], a library for lattice-based zero-knowledge proofs, and thoroughly extended it to construct our proof of decryption for GBFV ciphertexts.

Below, we present execution times for the $\Pi_{\text{vec-ANP}}$ protocol with increasing number of input ciphertexts. We conclude that, in practice, following the technique from Figure 6 is always beneficial performance-wise, since the client needs to apply $\Pi_{\text{vec-ANP}}$ to a single ciphertext at the comparably negligible expense of executing additional HE operations. Lazer parameters for the complete proof of

[6] https://github.com/KULeuven-COSIC/blind_zkSNARKs/blindFractal/largefield/estimates_FRIonly.m

[7] https://github.com/KULeuven-COSIC/blind_zkSNARKs/proof-of-decryption/README.md

decryption using the protocol from Figure 6 with r = 2514 ciphertexts and noise bound $B_{PoD} = 16.9$ bits are presented in Table 5 (Appendix D.1). The proof size for this parameter set is 12KB and can be computed as described in [68, Section 6.1].

r	$\Pi_{\text{vec-ANP}}$	$\Pi_{eval}^{(2)}$		Total runtime	
	w/out $\Pi_{eval}^{(2)}$	single thread	8 threads	single thread	8 threads
1	0.04	1.15	0.45	1.19	0.49
8	0.14	6.92	1.26	7.06	1.40
64	0.93	53.01	8.09	53.94	9.02
512	7.28	424.17	64.30	431.45	71.58
1024	14.68	846.59	126.89	861.27	141.57
2048	29.40	1688.15	253.55	1717.55	282.95
4096	58.81	3407.10	516.12	3465.91	574.93

Table 2: Runtimes in seconds for the PoD instantiated with the parameters in Table 5 and increasing number of `pt-ct` pairs (r). Using the optimized method given in Figure 6 we can always reduce to the first row in practice.

Experiments. In Table 2, we present the runtimes in seconds for our $\Pi_{\text{vec-ANP}}$ proof of decryption protocol. computed on a machine with an Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz, 8 cores and 377GB RAM. We provide separate numbers for the following two subprotocols:

– The $\Pi_{\text{vec-ANP}}$ protocol up to the execution of $\Pi_{eval}^{(2)}$, including the initial commitment to the FHE secret key and the computation of $\vec{\mathbf{z}} = bR\vec{\mathbf{u}} + \vec{\mathbf{y}}$.
– The $\Pi_{eval}^{(2)}$ protocol for proving that $\vec{\mathbf{z}}$ was computed correctly, including the computation of the quadratic functions H_j.

Constructing the H_j functions involves a relatively large matrix multiplication (for computing $\vec{R}_j \cdot \mathbf{W}$) which represents around 96% of the total runtime. Therefore, we tested two variations of the PoD: (1) a single-threaded version that would be used by a proof delegator with low-end device, and (2) a multi-threaded matrix multiplication using OpenMP leveraging 8 cores for when a more powerful machine is available. We note that the referred matrix multiplication involves only public information, meaning that it would be possible to delegate it to the server computing the zkSNARK. However, the single thread execution is already significantly faster than locally computing the zkSNARK proof itself.

As we discuss in Section 7.3, executing the PoD using the optimized method presented in Figure 6 results in reduced computational costs for the client. In fact, not using this method and instead directly applying the $\Pi_{\text{vec-ANP}}$ to all the ciphertexts resulting from the Fractal query phase would imply executing the protocol with on average r = 2514, as detailed in Appendix D. Conversely, with the protocol in Figure 6, we need to prove correct decryption of a single ciphertext, taking the client less than 2 seconds. Our current implementation consumes 15MB RAM for r = 1, but with better memory management this number could be further optimised.

Acknowledgments. We thank Robin Geelen for helping us use the GBFV implementation and thank Vadim Lyubashevsky and Patrick Steuer for helping us use the Lazer library. This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT - No. 101020788) and by CyberSecurity Research Flanders with reference number VR20192203. Date of this document: February 14, 2025.

References

1. Aikata, A., Mert, A.C., Kwon, S., Deryabin, M., Roy, S.S.: REED: Chiplet-based accelerator for fully homomorphic encryption. Cryptology ePrint Archive, Report 2023/1190 (2023), https://eprint.iacr.org/2023/1190
2. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC. pp. 99–108. ACM Press (May 1996). https://doi.org/10.1145/237814.237838
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015), https://eprint.iacr.org/2015/046
4. Aranha, D.F., Baum, C., Gjøsteen, K., Silde, T.: Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 1467–1481. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3616683
5. Aranha, D.F., Costache, A., Guimarães, A., Soria-Vazquez, E.: HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part V. LNCS, vol. 15488, pp. 302–334. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0935-2_10
6. Atapoor, S., Baghery, K., Pereira, H.V.L., Spiessens, J.: Verifiable FHE via lattice-based SNARKs. CiC **1**(1), 24 (2024). https://doi.org/10.62056/a6ksdkp10
7. Baum, C., Bootle, J., Cerulli, A., del Pino, R., Groth, J., Lyubashevsky, V.: Sublinear lattice-based zero-knowledge arguments for arithmetic circuits. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 669–699. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_23
8. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 368–385. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_20
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl (Jul 2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14

10. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press (May 2014). https://doi.org/10.1109/SP.2014.36
11. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
12. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Berlin, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_2
13. Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 201–218. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_13
14. Bertels, J., Pereira, H.V.L., Verbauwhede, I.: FINAL bootstrap acceleration on FPGA using DSP-free constant-multiplier NTTs. Cryptology ePrint Archive, Paper 2025/137 (2025), https://eprint.iacr.org/2025/137
15. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Berlin, Heidelberg (Mar 2013). https://doi.org/10.1007/978-3-642-36594-2_18
16. Block, A.R., Garreta, A., Katz, J., Thaler, J., Tiwari, P.R., Zajac, M.: Fiat-shamir security of FRI and related SNARKs. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part II. LNCS, vol. 14439, pp. 3–40. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8724-5_1
17. Block, A.R., Tiwari, P.R.: On the concrete security of non-interactive FRI. Cryptology ePrint Archive, Paper 2024/1161 (2024), https://eprint.iacr.org/2024/1161
18. Bois, A., Cascudo, I., Fiore, D., Kim, D.: Flexible and efficient verifiable computation on encrypted data. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 528–558. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75248-4_19
19. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Sorniotti, A.: A framework for practical anonymous credentials from lattices. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 384–417. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_13
20. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018. pp. 353–367. IEEE (2018), https://doi.org/10.1109/EuroSP.2018.00032
21. Bossuat, J.P., Mouchet, C., Troncoso-Pastoriza, J.R., Hubaux, J.P.: Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 587–617. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_21
22. Boudgoust, K., Scholl, P.: Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 371–404. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8721-4_12

23. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_50
24. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). https://doi.org/10.1145/2090236.2090262
25. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. Cryptology ePrint Archive, Report 2011/344 (2011), https://eprint.iacr.org/2011/344
26. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_8
27. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1082–1090. ACM Press (Jun 2019). https://doi.org/10.1145/3313276.3316380
28. Chatel, S., Mouchet, C., Sahin, A.U., Pyrgelis, A., Troncoso, C., Hubaux, J.P.: PELTA - shielding multiparty-FHE against malicious adversaries. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 711–725. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623139
29. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology. pp. 199–203. Springer US, Boston, MA (1983)
30. Chen, B.J., Waiwitlikhit, S., Stoica, I., Kang, D.: Zkml: An optimizing system for ml inference in zero-knowledge proofs. In: Proceedings of the Nineteenth European Conference on Computer Systems. p. 560–574. EuroSys '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3627703.3650088
31. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 116–136. Springer, Cham (Apr 2018). https://doi.org/10.1007/978-3-319-76953-0_7
32. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the INDCPA-d security of exact FHE schemes. Cryptology ePrint Archive, Paper 2024/127 (2024), https://eprint.iacr.org/2024/127
33. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8_15
34. Chiesa, A., Fenzi, G.: zkSNARKs in the ROM with unconditional UC-security. Cryptology ePrint Archive, Paper 2024/724 (2024), https://eprint.iacr.org/2024/724
35. Chiesa, A., Lehmkuhl, R., Mishra, P., Zhang, Y.: Eos: Efficient private delegation of zkSNARK provers. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 6453–6469. USENIX Association (Aug 2023)
36. Chiesa, A., Manohar, P., Spooner, N.: Succinct arguments in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 1–29. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_1

37. Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 769–793. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_27
38. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (Jan 2020). https://doi.org/10.1007/s00145-019-09319-x
39. Chillotti, I., Ligier, D., Orfila, J.B., Tap, S.: Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 670–699. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_23
40. Chu, E., George, A.: Inside the FFT black box: serial and parallel fast Fourier transform algorithms. CRC press (1999)
41. Costache, A., Smart, N.P.: Which ring based somewhat homomorphic encryption scheme is best? In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 325–340. Springer, Cham (Feb / Mar 2016). https://doi.org/10.1007/978-3-319-29485-8_19
42. Dahl, M., Demmler, D., Kazdadi, S.E., Meyre, A., Orfila, J., Rotaru, D., Smart, N.P., Tap, S., Walter, M.: Noah's ark: Efficient threshold-fhe using noise flooding. In: Brenner, M., Costache, A., Rohloff, K. (eds.) Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Copenhagen, Denmark, 26 November 2023. pp. 35–46. ACM (2023). https://doi.org/10.1145/3605759.3625259
43. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_38
44. Dokchitser, T., Bulkin, A.: Zero knowledge virtual machine step by step. Cryptology ePrint Archive, Report 2023/1032 (2023), https://eprint.iacr.org/2023/1032
45. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 115–146. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_5
46. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
47. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: SNARKs for ring arithmetic. Journal of Cryptology **36**(4), 41 (Oct 2023). https://doi.org/10.1007/s00145-023-09481-3
48. Garg, S., Goel, A., Jain, A., Policharla, G.V., Sekar, S.: zkSaaS: Zero-knowledge SNARKs as a service. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 4427–4444. USENIX Association (Aug 2023)
49. Garg, S., Goel, A., Wang, M.: How to prove statements obliviously? In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 449–487. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_14
50. Geelen, R., Vercauteren, F.: Fully homomorphic encryption for cyclotomic prime moduli. Cryptology ePrint Archive, Paper 2024/1587 (2024), https://eprint.iacr.org/2024/1587
51. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_25

52. Gentry, C.: A fully homomorphic encryption scheme. Stanford university (2009)
53. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 458–487. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-06944-4_16
54. Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Field switching in bgv-style homomorphic encryption. J. Comput. Secur. **21**(5), 663–684 (2013). https://doi.org/10.3233/JCS-130480
55. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_49
56. Gjøsteen, K., Haines, T., Müller, J., Rønne, P.B., Silde, T.: Verifiable decryption in the head. In: Nguyen, K., Yang, G., Guo, F., Susilo, W. (eds.) ACISP 22. LNCS, vol. 13494, pp. 355–374. Springer, Cham (Nov 2022). https://doi.org/10.1007/978-3-031-22301-3_18
57. Halevi, S.: Homomorphic encryption. In: Lindell, Y. (ed.) Tutorials on the Foundations of Cryptography, pp. 219–276. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-57048-8_5
58. Halevi, S., Shoup, V.: Algorithms in HElib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 554–571. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_31
59. Halevi, S., Shoup, V.: Faster homomorphic linear transformations in HElib. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 93–120. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_4
60. Holmgren, J.: On round-by-round soundness and state restoration attacks. Cryptology ePrint Archive, Report 2019/1261 (2019), https://eprint.iacr.org/2019/1261
61. Hwang, I., Lee, H., Seo, J., Song, Y.: Practical zero-knowledge PIOP for public key and ciphertext generation in (multi-group) homomorphic encryption. Cryptology ePrint Archive, Paper 2024/1879 (2024), https://eprint.iacr.org/2024/1879
62. Hwang, I., Seo, J., Song, Y.: Concretely efficient lattice-based polynomial commitment from standard assumptions. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 414–448. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_13
63. Ishai, Y., Su, H., Wu, D.J.: Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 212–234. ACM Press (Nov 2021). https://doi.org/10.1145/3460120.3484572
64. Knabenhans, C., Viand, A., Merino-Gallardo, A., Hithnawi, A.: vfhe: Verifiable fully homomorphic encryption. In: Bergamaschi, F., Costache, A., Rohloff, K. (eds.) Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Salt Lake City, UT, USA, October 14-18, 2024. pp. 11–22. ACM (2024). https://doi.org/10.1145/3689945.3694806
65. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. DCC **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
66. Luo, F., Wang, K.: Verifiable decryption for fully homomorphic encryption. In: Chen, L., Manulis, M., Schneider, S. (eds.) ISC 2018. LNCS, vol. 11060, pp. 347–365. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-99136-8_19
67. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Berlin, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43

68. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 71–101. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_3

69. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Shorter lattice-based zero-knowledge proofs via one-time commitments. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 215–241. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75245-3_9

70. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_1

71. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Berlin, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_3

72. Lyubashevsky, V., Seiler, G., Steuer, P.: The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy. CCS '24 (2024)

73. Nassar, S., Rothblum, R.D.: Succinct interactive oracle proofs: Applications and limitations. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 504–532. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15802-5_18

74. Nguyen, N.K.: Lattice-Based Zero-Knowledge Proofs Under a Few Dozen Kilobytes. Ph.D. thesis, ETH Zurich, Zürich, Switzerland (2022). https://doi.org/10.3929/ETHZ-B-000574844, https://hdl.handle.net/20.500.11850/574844

75. Ozdemir, A., Boneh, D.: Experimenting with collaborative zk-SNARKs: Zero-knowledge proofs for distributed secrets. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022. pp. 4291–4308. USENIX Association (Aug 2022)

76. Palvia, P., March, S.T.: Approximating block accesses in database organizations. Inf. Process. Lett. **19**(2), 75–79 (1984). https://doi.org/10.1016/0020-0190(84)90101-7

77. Rosenberg, M., White, J.D., Garman, C., Miers, I.: zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. In: 2023 IEEE Symposium on Security and Privacy. pp. 790–808. IEEE Computer Society Press (May 2023). https://doi.org/10.1109/SP46215.2023.10179430

78. Sze, T.: Schönhage-strassen algorithm with mapreduce for multiplying terabit integers. In: Maza, M.M. (ed.) SNC 2011, Proceedings of the 2011 Internation Workshop on Symbolic-Numeric Computation, San Jose, California, USA, June 7-9, 2011. pp. 54–62. ACM (2011). https://doi.org/10.1145/2331684.2331693

79. Thibault, L.T., Walter, M.: Towards verifiable FHE in practice: Proving correct execution of TFHE's bootstrapping using plonky2. Cryptology ePrint Archive, Report 2024/451 (2024), https://eprint.iacr.org/2024/451

80. Wu, H., Zheng, W., Chiesa, A., Popa, R.A., Stoica, I.: DIZK: A distributed zero knowledge proof system. In: Enck, W., Felt, A.P. (eds.) USENIX Security 2018. pp. 675–692. USENIX Association (Aug 2018)

A Supplementary preliminaries

A.1 Number fields, rings and coefficient embedding

For any positive integer m, let $\Phi_m(X)$ denote the m-th cyclotomic polynomial of degree $n = \phi(m)$, where $\phi(\cdot)$ is the Euler totient function. Specifically, when m is a power-of-two, $\Phi_m(X) = X^{m/2} + 1$. The m-th cyclotomic number field is $\mathcal{K}_m = \mathbb{Q}[X]/(\Phi_m(X))$ and the m-th cyclotomic ring is $\mathcal{R}_m = \mathbb{Z}[X]/(\Phi_m(X))$. For $g = \sum_{i=0}^{n-1} g_i X^i \in \mathcal{K}_m$, its coefficient vector $[g_0\ g_1\ \dots\ g_{n-1}]^\intercal \in \mathbb{Q}^n$ is denoted as $\vec{g}$, and its coefficient-wise norms $\|g\|_p = \|\vec{g}\|_p$, e.g.

$$\|g\|_1 = \sum |g_i|, \quad \|g\|_2 = (\sum g_i^2)^{\frac{1}{2}}, \quad \|g\|_\infty = \max\{|g_i|\}.$$

For $c(X), s(X), b(X) \in \mathcal{R}_m$ and $b(X) = c(X) \cdot s(X)$, their coefficient representations satisfy $\vec{b} = \mathsf{Rot}_m(c) \cdot \vec{s}$, where

$$\mathsf{Rot}_m(c) \in \mathbb{Z}^{n\times n} = \begin{bmatrix} | & | & & | \\ \overrightarrow{c_{(0)}} & \overrightarrow{c_{(1)}} & \dots & \overrightarrow{c_{(n-1)}} \\ | & | & & | \end{bmatrix}$$

and $c_{(i)} = X^i \cdot c(X) \bmod \Phi_m(X)$. The expansion factor with respect to the infinity norm is defined as

$$\delta_m = \sup\left\{ \frac{\|g \cdot f \bmod \Phi_m\|_\infty}{\|g\|_\infty \cdot \|f\|_\infty} \mid g, f \in \mathbb{Z}[X] \setminus 0 \text{ and } \deg(g), \deg(f) \le (n-1) \right\}.$$

For elements in $\mathsf{Rot}_m(c)$, let $\left\|\overrightarrow{c_{(i)}}\right\|_\infty \le EF_m \cdot \|c\|_\infty$, which consecutively gives $\delta_m \le n \cdot EF_m$. Specifically, when m is a power-of-two, $EF_m = 1$ and $\delta_m = n$.

For the ring $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$, we use $[-\frac{q}{2}, \frac{q}{2})$ as the representative interval, and for $x \in \mathbb{Z}$, we denote the centered reduction modulo q by $[x]_q \in \mathbb{Z}_q$. Let $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the flooring and ceiling functions respectively, and let $\lfloor\cdot\rceil$ denote the rounding function that rounds half up. All these notations are extended to elements in $\mathcal{K}_m$ and $\mathcal{R}_m$ coefficient-wise.

For a non-zero element $t(X) \in \mathcal{R}_m$, denote the quotient ring of $\mathcal{R}_m$ modulo $t(X)$ as $\mathcal{R}_{m,t(X)} = \mathcal{R}_m / t\mathcal{R}_m$. Specifically, for $q \in \mathbb{Z}$, the quotient ring of $\mathcal{R}_m$ modulo q is denoted as $\mathcal{R}_{m,q}$. Notations for coefficient vectors and norms in $\mathcal{R}_m$ naturally extend to $\mathcal{R}_{m,q}$ using representatives in $[-\frac{q}{2}, \frac{q}{2})$. For $d(X) \in \mathcal{R}_{m,q}$, the rotation matrix $\mathsf{Rot}_{m,q}(d)$ contains columns $\overrightarrow{d_{(i)}}$ where $d_{(i)} = X^i \cdot d(X) \bmod (q, \Phi_m)$, which are bounded as

$$\left\|\overrightarrow{d_{(i)}}\right\|_\infty \le \min\{EF_m \cdot \|d\|_\infty, \frac{q}{2}\},\ 0 \le i \le n-1.$$

Moreover, for an explicit power-of-two cyclotomic order 2^k, let R denote the ring $\mathcal{R}_{2^k} = \mathbb{Z}[X]/\left(X^d + 1\right)$ where $d = 2^{k-1}$, and $\mathsf{R}_q \coloneqq \mathsf{R}/q\,\mathsf{R}$.

A.2 Probability distributions

Given a probability distribution χ, the notation $a \leftarrow \chi$ implies that a is sampled from χ. Let $\mathcal{U}(\mathbb{Z}_q)$ and $\mathcal{U}(\mathcal{R}_{m,q})$ denote the uniform distribution over $\mathbb{Z}_q$ and over $\mathcal{R}_{m,q}$, respectively. For example, $a \leftarrow \mathcal{U}(\mathcal{R}_{m,3})$ is a uniformly random polynomial in $\mathcal{R}_m$ with ternary coefficients.

Let D_σ denote the discrete Gaussian distribution with standard deviation σ over the integers, then the following properties are satisfied [67,7]

$$\Pr\left[|z| > k\sigma \mid z \leftarrow D_\sigma\right] \leq 2e^{-k^2/2} \tag{9}$$

$$\Pr\left[\|\mathbf{z}\|_2 > t\sqrt{r} \cdot \sigma \mid \mathbf{z} \leftarrow D_\sigma^r\right] \leq \left(te^{\frac{1-t^2}{2}}\right)^r . \tag{10}$$

The notation naturally extends to the ring $\mathcal{R}_m$, i.e. $D_{\mathcal{R}_m,\sigma}$ denotes the discrete Gaussian distribution with standard deviation σ over $\mathcal{R}_m$.

Let Bin_κ denote the binomial distribution parameterized by κ, i.e. the distribution $\sum_{i=0}^{\kappa}(a_i - b_i)$ where $a_i, b_i \leftarrow \{0,1\}$. For example, for $c \leftarrow \mathsf{Bin}_1$, $\Pr(c = 0) = \frac{1}{2}$ and $\Pr(c = 1) = \Pr(c = -1) = \frac{1}{4}$.

B Background on the GBFV Scheme [50]

B.1 Canonical embedding

For polynomials in $\mathcal{K}_m$, defining norms on coefficient vectors provides a straightforward measure of sizes. However, analyzing the coefficient norm growth upon multiplication requires the expansion factor δ_m, which depends heavily on the polynomial modulus $\Phi_m(X)$ and often results in loose bounds. This leads to the broad use of canonical norm $\|\cdot\|^{\text{can}}$ [70,71,55,43,41], which is defined from the canonical embedding into $\mathbb{C}^n$. Recall that the canonical embedding is

$$\tau : \mathcal{K}_m \hookrightarrow \mathbb{C}^n : a(X) \mapsto \{a(\xi_m^j)\}_{j \in \mathbb{Z}_m^\times},$$

where $\xi_m = \exp(2\pi i/m)$ is a primitive complex m-th root of unity. The canonical norm is $\|a\|_p^{\text{can}} = \|\tau(a)\|_p$, and common values of p are $1, 2, \infty$.

Lemma 3 (Adapted from [43]). *For all $a, b \in \mathcal{K}_m$, the following properties are satisfied*

- $\|a\|_\infty^{\text{can}} \leq \|a\|_1$
- $\|a\|_\infty \leq c_m \cdot \|a\|_\infty^{\text{can}}$, *where c_m is a constant determined by the cyclotomic order m*
- $\|a \cdot b\|_\infty^{\text{can}} \leq \|a\|_\infty^{\text{can}} + \|b\|_\infty^{\text{can}}$
- $\|a \cdot b\|_p^{\text{can}} \leq \|a\|_\infty^{\text{can}} \cdot \|b\|_p^{\text{can}}$

Specifically, $c_m = 1$ for power-of-two m, and for $m = p_1^{e_1} \cdots p_k^{e_k}$, if $p_1 \cdots p_k \leq 400$ then $c_m \leq 8.6$ [43].

B.2 The inherent noise bound in the GBFV Scheme

Let $\Delta = q/t(X) \in \mathcal{K}_m$ denote the scaling factor in GBFV. The inherent noise in the GBFV Scheme [50] can be defined in the same way as for BFV as follows.

Definition 12. *Let $(c_0, c_1) \in \mathcal{R}_{m,q}^2$ be a ciphertext in the GBFV scheme that decrypts to $m \in \mathcal{R}_{m,t}$, then its inherent noise $v_{inh} \in \mathcal{R}_m$ is the polynomial with the lowest infinity norm such that*

$$c_0 + c_1 \cdot \boldsymbol{sk} = \lfloor \Delta \cdot m \rceil + v_{inh} + aq \in \mathcal{R}_m \tag{11}$$

for some polynomial $a \in \mathcal{R}_m$.

For correct decryption, we present the following inherent noise bound $\mathcal{B}_q$.

Lemma 4. *The ciphertext $(c_0, c_1) \in \mathcal{R}_{m,q}^2$ in the GBFV scheme decrypts to message m correctly if its inherent noise v_{inh} satisfies $\|v_{inh}\|_\infty < \mathcal{B}_q \coloneqq \frac{q}{2 \cdot EF_m \cdot h_t \cdot \|t\|_\infty} - \frac{1}{2}$, where h_t is the number of non-zero terms in $t(X)$.*

Proof. The decryption procedure requires computing

$$\begin{aligned} \left\lfloor \frac{t(X)}{q}(c_0 + c_1 \cdot s) \right\rceil \bmod t(X) &= \left\lfloor \frac{t(X)}{q}(\lfloor \Delta \cdot m \rceil + v_{inh} + aq) \right\rceil \bmod t(X) \\ &= \left\lfloor m + \frac{t(X)}{q}(\epsilon + v_{inh}) \right\rceil, \end{aligned}$$

where $\|\epsilon\|_\infty < \frac{1}{2}$, and the decryption is correct as long as

$$\left\| \frac{t(X)}{q}(\epsilon + v_{inh}) \right\|_\infty < \frac{1}{2}. \tag{12}$$

Let h_t is the number of non-zero terms in $t(X)$, then $\|t(X) \cdot (\epsilon + v_{inh})\|_\infty \leq EF_m \cdot h_t \cdot \|t\|_\infty \cdot (\frac{1}{2} + \|v_{inh}\|_\infty)$, relation (12) is guaranteed by

$$\|v_{inh}\|_\infty < \frac{q}{2 \cdot EF_m \cdot h_t \cdot \|t\|_\infty} - \frac{1}{2}.$$

□

B.3 Modulus switching

Let $\mathtt{ct}[m] = (c_0, c_1) \in \mathcal{R}_{m,q}^2$ denote a ciphertext with ciphertext modulus q and inherent noise v_{inh}, i.e. it satisfies $c_0 + c_1 \cdot \mathtt{sk} = \lfloor \Delta \cdot m \rceil + v_{inh} + aq$ for some $a \in \mathcal{R}_m$. Switching ciphertext modulus to q' amounts to computing

$$\mathtt{ct}' = \left(\left\lfloor \frac{q'}{q} c_0 \right\rceil, \left\lfloor \frac{q'}{q} c_1 \right\rceil \right) \in \mathcal{R}_{m,q'}^2 .$$

The derived ciphertext satisfies

$$\begin{aligned}
\left\lfloor \frac{q'}{q} c_0 \right\rceil + \left\lfloor \frac{q'}{q} c_1 \right\rceil \cdot \mathtt{sk} &= \frac{q'}{q} \left(c_0 + c_1 \cdot \mathtt{sk} \right) + \left(\epsilon_0 + \epsilon_1 \cdot \mathtt{sk} \right) \\
&= \frac{q'}{q} \left(\lfloor \Delta \cdot m \rceil + v_{inh} + aq \right) + \left(\epsilon_0 + \epsilon_1 \cdot \mathtt{sk} \right) \\
&= \frac{q'}{q} \left(\frac{q}{t} \cdot m + \epsilon_3 + v_{inh} + aq \right) + \left(\epsilon_0 + \epsilon_1 \cdot \mathtt{sk} \right) \\
&= \frac{q'}{q} \left(\frac{q}{t} \cdot m + \epsilon_3 + v_{inh} + aq \right) + \left(\epsilon_0 + \epsilon_1 \cdot \mathtt{sk} \right) \\
&= \lfloor \Delta' \cdot m \rceil + \epsilon_4 + \frac{q'}{q} (\epsilon_3 + v_{inh}) + \left(\epsilon_0 + \epsilon_1 \cdot \mathtt{sk} \right) + q' \cdot a
\end{aligned}$$

for $\Delta' = \frac{q'}{t}$ and $\|\epsilon_i\|_\infty \leq \frac{1}{2},\ i \in [4]$. Its inherent noise is

$$v'_{inh} = \frac{q'}{q} \cdot v_{inh} + (\epsilon_4 + \frac{q'}{q} \epsilon_3 + \epsilon_0 + \epsilon_1 \cdot \mathtt{sk}), \tag{13}$$

which can be bounded as $\|v'_{inh}\|_\infty \leq \frac{q'}{q} \|v_{inh}\|_\infty + \mathcal{B}_{ms}$ and $\mathcal{B}_{ms} = 1 + \frac{q'}{2q} + \frac{1}{2} \delta_m \cdot \|\mathtt{sk}\|_\infty$. Moreover, for a ternary secret key with hamming weight h, the bound $\mathcal{B}_{ms}$ can be lower into $(1 + \frac{q'}{2q} + \frac{1}{2} EF_m \cdot h)$ in the worst-case, and $(1 + \frac{q'}{2q} + EF_m \cdot 3 \cdot \sqrt{\frac{h}{12}})$ heuristically.

B.4 Performance of GBFV

Here we present the runtimes used for the microbenchmark in Section 8. They were measured on a MacBook Pro (2021) equipped with an Apple M1 Max processor (10 cores: 8 performance and 2 efficiency), 64 GB of RAM and running macOS Sonoma 14.7.1.

C The LNP22 Proof System

This section provides an overview of the LNP22 proof system, including the ABDLOP commitment, commit-and-prove protocols of qudratic relations and approximate proofs of bounded norms (ANP). The latter is extended into proving relations in the coefficient encoding in C.5, and parameters for our instantiation are provided in D.1. For future works, it would be interesting to prove relations with other encodings, such as the new CLPX-like encoding in [62].

Remark 1. The modulus q in this appendix section corresponds to the LNP-friendly modulus q'' in the main text.

$\log q$ (bits)	$T_{\texttt{add}}$ (ms)	$T_{\texttt{ctct}}$ (ms)	$T_{\texttt{aut}}$ (ms)	$T_{\texttt{ptct}}$ (ms)
120	0.02	5.58	0.84	0.06
180	0.04	9.44	1.66	0.1
240	0.04	9.46	1.66	0.1
300	0.1	18.12	3.98	0.24
360	0.12	23.06	5.52	0.28
420	0.14	28.4	7.24	0.34

Table 3: Timings of operations in GBFV for $n = 2^{14}$ and different sizes for ciphertext modulus q.

C.1 Module-SIS, Module-LWE and the ABDLOP commitment scheme

For some integer k let R denote the ring $\mathcal{R}_{2^k} = \mathbb{Z}[X]/\left(X^d + 1\right)$ where $d = 2^{k-1}$, and $\mathsf{R}_q = \mathsf{R}/q\,\mathsf{R}$. The ABDLOP commitment scheme [68] is defined over the ring R_q and relies on the hardness of the Module-SIS (MSIS) problem and the Module-LWE (MLWE) problem over R_q, as defined below [65].

Definition 13 **($\mathsf{MSIS}_{\kappa,m,q,B}$).** *Given $A \leftarrow \mathsf{R}_q^{\kappa\times m}$, the $\mathsf{MSIS}_{\kappa,m,q,B}$ problem is to find $z \in \mathsf{R}_q^m$ such that $A \cdot z = 0^\kappa \bmod q$ and $\|z\|_2 \leq B$.*

Definition 14 **($\mathsf{MLWE}_{\kappa,m,q,\chi}$).** *Given a distribution χ and parameters κ, the $\mathsf{MLWE}_{\kappa,m,q,\chi}$ problem is to distinguish $(A, A\cdot s + e)$ for $A \leftarrow \mathsf{R}_q^{m\times\kappa}$, secret vector $s \leftarrow \chi^\kappa$ and error vector $e \leftarrow \chi^m$, from $(A, b) \leftarrow \mathsf{R}_q^{m\times\kappa} \times \mathsf{R}_q^m$.*

The hardness of $\mathsf{MSIS}_{\kappa,m,q,B}$ and $\mathsf{MLWE}_{\kappa,m,q,\chi}$ are estimated using $\mathsf{SIS}_{\kappa\cdot d,q,B}$ and $\mathsf{LWE}_{\kappa\cdot d,q,\chi}$ in the lattice estimator by Albrecht et al. [3].

The ABDLOP commitment scheme [68]. The ring modulus in ABDLOP is $q = \prod_i q_i$ where $q_i = 5 \bmod 8$ is a prime and q_1 is the smallest factor. Let σ_i denote an automorphism in R_q where $\sigma_i(X) = X^i$ for odd i. This notation extends to arbitrary vectors $\mathbf{m} \in \mathsf{R}^k$ element-wise, i.e. $\sigma_i(\mathbf{m}) = (\sigma_i(\mathbf{m}[j]))_{1\leq j\leq k}$.

In the ABDLOP commitment scheme, the public parameters pp are generated as

$$\mathsf{pp} = (\mathbf{A_1}, \mathbf{A_2}, \mathbf{B}) \leftarrow \mathsf{R}_q^{\omega\times m_1} \times \mathsf{R}_q^{\omega\times m_2} \times \mathsf{R}_q^{u\times m_2}\,.$$

In order to commit to a small message $\mathbf{s_1} \in \mathsf{R}_q^{m_1}$ where $\|\mathbf{s_1}\| \leq \alpha$ and an arbitrarily large message $\mathbf{m} \in \mathsf{R}_q^u$, one samples a small randomness $\mathbf{s_2} \leftarrow \chi^{m_2}$ where χ is a distribution over R_q with bounded infinity norm ν and computes

$$\mathsf{ABDLOP.Com}(\mathsf{pp}, (\mathbf{s_1}, \mathbf{m}, \mathbf{s_2})) = \begin{bmatrix}\mathbf{t_A}\\ \mathbf{t_B}\end{bmatrix} = \begin{bmatrix}\mathbf{A_1}\\ \mathbf{0}\end{bmatrix}\cdot\mathbf{s_1} + \begin{bmatrix}\mathbf{A_2}\\ \mathbf{B}\end{bmatrix}\cdot\mathbf{s_2} + \begin{bmatrix}\mathbf{0}\\ \mathbf{m}\end{bmatrix} \mod q.$$

As such, the ABDLOP scheme not only allows the commitment of large messages $\mathbf{m}$ as in the BDLOP commitment [8], but also compresses small messages $\mathbf{s_1}$ as in the Ajtai commitment [2]. The commitment $\mathbf{t_B}$ of $\mathbf{m}$ and $\mathbf{t_A}$ of $\mathbf{s_1}$ are referred as the BDLOP part and the Ajtai part of ABDLOP, respectively.

Moreover, the commitment does not reveal messages if $\left(\begin{bmatrix}\mathbf{A_2}\\ \mathbf{B}\end{bmatrix}, \begin{bmatrix}\mathbf{A_2}\\ \mathbf{B}\end{bmatrix} \cdot \mathbf{s_2}\right)$ is indistinguishable from uniform. In other words, if the $\mathsf{MLWE}_{m_2-(\omega+u),\omega+u,q,\chi}$ problem is hard, then ABDLOP is computationally hiding.

For the proof of opening, with fixed parameters ξ, η and a power-of-two k, the challenge space $\mathcal{C}h$ is defined as

$$\mathcal{C}h = \left\{ c \in \mathsf{R}_q : \|c\|_\infty \leq \xi,\ \sigma_{-1}(c) = c \text{ and } \sqrt[2k]{\|c^{2k}\|_1} \leq \eta \right\},$$

and it should be exponentially large in the security parameter for soundness purposes. Its set of differences is denoted as $\overline{\mathcal{C}h} = \{c - c' : c, c' \in \mathcal{C}h \text{ and } c \neq c'\}$, and elements in $\overline{\mathcal{C}h}$ are invertible if $\xi < \frac{q_1}{2}$. Example parameters for the challenge space taken from [19] are listed in Table 4.

d	ξ	η	k	$\lvert\mathcal{C}\rvert$
64	8	140	32	2^{129}
128	2	59	32	2^{147}

Table 4: Example parameters in [19] to instantiate the challenge space $\mathcal{C}$ assuming $q_1 > 16$

As in other lattice-based commitment schemes [2,8], the opening algorithm in ABDLOP is *relaxed*. For an ABDLOP commitment $[\mathbf{t_A}\ \mathbf{t_B}]^\intercal$, its relaxed opening with respect to the commitment key ck is a tuple $(\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}, \overline{c}) \in \mathsf{R}_q^{m_1} \times \mathsf{R}_q^{u} \times \mathsf{R}_q^{m_2} \times \overline{\mathcal{C}h}$ that satisfies

$$\mathsf{ABDLOP}.\mathsf{Com}(\mathsf{ck}, (\mathbf{s_1}, \mathbf{m}, \mathbf{s_2})) = \begin{bmatrix}\mathbf{t_A}\\ \mathbf{t_B}\end{bmatrix}$$
$$\|\overline{c}\mathbf{s_1}\|_2 \leq B_1 \text{ and } \|\overline{c}\mathbf{s_2}\|_2 \leq B_2,$$

where $B_1 = B_1(\alpha)$ and $B_2 = B_2(\nu)$ are pre-determined constants. Furthermore, as explained in [68, Lemma 3.1] and in [19, Lemma 5.2], if $\mathsf{MSIS}_{\omega,m_1+m_2,4\eta\sqrt{B_1^2+B_2^2}}$ is hard, then ABDLOP is computationally binding with respect to the relaxed openings.

C.2 Commit-and-prove of elementary relations

Let $\mathcal{G} = \{g : \mathsf{R}_q^{2(m_1+u)} \to \mathsf{R}_q\}$ denote the set of quadratic functions over R_q, i.e. any $g \in \mathcal{G}$ can be explicitly written as

$$g(\mathbf{a}) = \mathbf{a}^\intercal \mathbf{G_2} \mathbf{a} + \mathbf{g_1} \mathbf{a} + g_0, \quad \forall \mathbf{a} \in \mathsf{R}_q^{2(m_1+u)}$$

for some $\mathbf{G_2} \in \mathsf{R}_q^{2(m_1+u)\times 2(m_1+u)}$, $\mathbf{g_1} \in \mathsf{R}_q^{2(m_1+u)}$ and $g_0 \in \mathsf{R}_q$.

Given an ABDLOP commitment $(\mathbf{t_A}, \mathbf{t_B})$ to the message $(\mathbf{s_1}, \mathbf{m})$ with randomness $\mathbf{s_2}$, the commit-and-prove protocol in [68, Figure 8] (together with the optimization in [68, Section 4.4]) allows one to prove the knowledge of the message

$$\mathbf{s} = \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \\ \sigma_{-1}(\mathbf{s_1}) \\ \sigma_{-1}(\mathbf{m}) \end{bmatrix} \in \mathsf{R}_q^{2(m_1+u)}$$

such that evaluations of public functions $g_1, \ldots, g_N$ in $\mathcal{G}$ at $\mathbf{s}$ satisfy

$$g_j(\mathbf{s}) = 0 \in \mathsf{R}_q, \forall j \in [N] \tag{14}$$

and evaluations of public functions $G_1, \ldots, G_M$ in $\mathcal{G}$ at $\mathbf{s}$ satisfy

$$\overrightarrow{G_j(\mathbf{s})}[1] = 0 \bmod q, \forall j \in [M], \tag{15}$$

where $\overrightarrow{G_j(\mathbf{s})}[1]$ denotes the constant term of $G_j(\mathbf{s}) \in \mathsf{R}_q$. For convenience, this protocol is denoted as

$$\Pi_{eval}^{(2)}\left((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), \sigma_{-1}, (g_1, \ldots, g_N), (G_1, \ldots, G_M)\right).$$

In other words, condition (14) allows proving quadratic relations of committed messages over R_q, and the vanishing constant condition in (15) allows proving inner products between coefficient vectors of committed messages over $\mathbb{Z}_q$ using the following map T.

Inner product from the T map Given two vectors $\vec{a} = (a_0, \ldots, a_{kd-1})$, $\vec{b} = (b_0, \ldots, b_{kd-1}) \in \mathbb{Z}_q^{kd}$, define the following map

$$\begin{aligned} \mathsf{T} : \mathbb{Z}_q^{kd} \times \mathbb{Z}_q^{kd} &\longrightarrow \mathsf{R}_q \\ (\vec{a}, \vec{b}) &\to \sum_{i=0}^{k-1} \sigma_{-1}\left(\sum_{j=0}^{d-1} a_{id+j} X^j\right) \cdot \left(\sum_{j=0}^{d-1} b_{id+j} X^j\right). \end{aligned}$$

Then the constant coefficient of $\mathsf{T}(\vec{a}, \vec{b})$ is equal to the inner product of $\vec{a}$ and $\vec{b}$ modulo q.

C.3 Approximate range proofs

The approximate range proofs [53,68] allow one to prove smallness of a message $\vec{w} \in \mathbb{Z}^m$, with respect to the proof system modulus q. Firstly, the prover computes a projection $\vec{v} = R\vec{w}$, where $R \leftarrow \mathsf{Bin}_1^{256 \times m}$ is a random challenge from the verifier. Note that [68, Lemma 2.8] provides a probabilistic bound for $\vec{v}$

$$\Pr_{R \leftarrow \mathsf{Bin}_1^{256 \times m}} \left[\|\vec{v}\|_2^2 > 337\beta^2 \right] \leq 2^{-128},$$

where β is an upper bound on $\|\vec{w}\|_2$. Secondly, by using rejection sampling, the prover generates a vector $\vec{z} = \vec{v} + \vec{y}$ whose distribution is independent of $\vec{v}$ and indistinguishable from the masking vector $\vec{y}$. The standard deviation of $\vec{y}$ (hence also $\vec{z}$) is $\mathfrak{s} = \gamma\|\vec{v}\|_2 = \gamma\sqrt{337}\beta$, where γ is a constant defining the rejection sampling repetition rate. The following lemma shows that if $\vec{z}$ is small, then the vector $\vec{w}$ is small with high probability.

Lemma 5 ([68, Lemma 2.9]). *Given q, m, a fixed bound $b \leq q/41m$ and $\vec{w} \in \mathbb{Z}_q^m$ such that $\|\vec{w}\|_2 \geq b$, then for arbitrary $\vec{y} \in \mathbb{Z}_q^{256}$, the following holds*

$$\Pr_{R \leftarrow \mathsf{Bin}_1^{256 \times m}} \left[\|R\vec{w} + \vec{y} \bmod q\|_2 < \frac{1}{2}\sqrt{26}b \right] < 2^{-128}.$$

Following the tail bound in Equation (10) for $t \geq 1.64$, the verifier check $\|\vec{z}\|_2 \leq t\sqrt{256} \cdot \mathfrak{s}$ will hold with overwelming probability for a $\|\vec{w}\|_2 \leq \beta$. By rewriting this check as

$$\begin{aligned} \|\vec{z}\|_2 \leq t\sqrt{256} \cdot \mathfrak{s} &= t\sqrt{256} \cdot \gamma\sqrt{337}\beta \\ &= \frac{1}{2}\sqrt{26} \left(2\sqrt{\frac{256}{26}} t\gamma\sqrt{337} \right) \beta, \end{aligned}$$

it is clear that the vector $\vec{w}$ is proven to be small with negligible soundness error. More precisely, if the prover knows a small $\vec{w}$ where $\|\vec{w}\|_2 \leq \beta$ and computes $\vec{z}$ as described, then the verifier can extract a vector $\overrightarrow{w^*}$ such that $\left\|\overrightarrow{w^*}\right\|_2 \leq \psi^{(L2)} \cdot \beta$, assuming $\psi^{(L2)} \cdot \beta < \frac{q}{41m}$, where the factor $\psi^{(L2)} = 2\sqrt{\frac{256}{26}} t\gamma\sqrt{337}$ is called the slack.

The procedure above can also be applied to generate approximate infinity norm proofs with slack $\psi^{(\infty)} = \psi^{(L2)}\sqrt{m}$. Specifically, consider a prover that knows $\vec{w} \in \mathbb{Z}_q^m$ satisfying $\|\vec{w}\|_\infty \leq \alpha$, then its L2 norm is bounded by $\sqrt{m}\alpha$. The previous procedure allows the verifier to extract a vector $\overrightarrow{w^*}$ where

$$\left\|\overrightarrow{w^*}\right\|_\infty \leq \left\|\overrightarrow{w^*}\right\|_2 \leq \psi^{(L2)} \cdot \|\vec{w}\|_2 \leq \psi^{(L2)}\sqrt{m}\alpha,$$

resulting in a slack $\psi^{(\infty)} = \psi^{(L2)}\sqrt{m}$.

C.4 Approximate proofs of bounded norms

In the Approximate Norm bound Proofs (ANP), the prover knows the secret message $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ which satisfies

$$\left\| \mathbf{E} \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \end{bmatrix} + \mathbf{e} \right\|_\infty \leq B_e \tag{16}$$

for public elements $\mathbf{E} \in \mathsf{R}_q^{\ell_e \times (m_1+u)}$, $\mathbf{e} \in \mathsf{R}_q^{\ell_e}$ and a public bound B_e. After committing $(\mathbf{s_1}, \mathbf{m})$ into $(\mathbf{t_A}, \mathbf{t_B})$, the commit-and-prove protocol convinces the verifier that the prover knows $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ such that $\mathcal{O}^{\mathcal{CT}}((\mathbf{t_A}, \mathbf{t_B}), (\mathbf{s_1}, \mathbf{m})) =$ acc and

$$\left\| \mathbf{E} \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \end{bmatrix} + \mathbf{e} \right\|_\infty \leq \psi^{(\infty)} \cdot B_e, \tag{17}$$

where the infinity norm slack is $\psi^{(\infty)} = \psi^{(L2)}\sqrt{d\ell_e} = 2\sqrt{\frac{256}{26}}t\gamma\sqrt{337}\sqrt{d\ell_e}$. Moreover, approximate proofs are only complete for bounds $B_e \leq \frac{q}{41(d\ell_e)^{3/2}\psi^{(L2)}}$, as explained in Section C.3.

The protocol. Let $B_b \leftarrow \mathsf{R}_q^{1\times m_2}$, $B_y \leftarrow \mathsf{R}_q^{256/d\times m_2}$, $\mathfrak{s} \coloneqq \gamma\sqrt{337}\sqrt{d\ell_e} \cdot B_e$ and rej_0 denote the optimized bimodal rejection sampling [68]. The protocol

$$\Pi_{\text{ANP}}((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), (\mathbf{E}, \mathbf{e}, B_e))$$

gives an approximate bounded norm proof for $\mathbf{u} \in \mathsf{R}_q^{\ell_e} \coloneqq \mathbf{E}\left[\, \mathbf{s_1} \quad \mathbf{m} \,\right]^\top + \mathbf{e}$ and is presented in Figure 7.

Specifically, line 1 samples for a sign b used for bimodal rejection sampling and line 2 samples a vector $\mathbf{y}$ used for masking. In line 3-4, elements b and $\mathbf{y}$ are committed to in the BDLOP part, i.e. the secret messages become

$$\mathbf{s}' \coloneqq (\mathbf{s_1}, (\mathbf{m}, b, \mathbf{y}))\,,$$

and $\dim(\mathbf{s}') = m_1 + u + 1 + 256/d$. The correct computation of $\vec{\mathbf{z}}$ in line 8 and that $b \in \{-1, 1\}$ are proven by calling the $\Pi_{eval}^{(2)}$ in line 12 with appropriate public functions v and V. These public functions are elements of $\mathcal{V} = \{v : \mathsf{R}_q^{2\cdot\dim(\mathbf{s}')} \rightarrow \mathsf{R}_q\}$.

To prove $\vec{\mathbf{z}}$ was computed correctly, public quadratic functions $H_j \in \mathcal{V}$ for $j \in [256]$ are constructed such that their evaluations at $(\mathbf{s}', \sigma_{-1}(\mathbf{s}'))$ satisfy

$$H_j(\mathbf{s}', \sigma_{-1}(\mathbf{s}')) \coloneqq \mathsf{T}\left(b\vec{R}_j, \vec{\mathbf{u}}\right) + \mathsf{T}(\vec{\mathbf{e}_j}, \vec{\mathbf{y}}) - \vec{\mathbf{z}}[j],\ j \in [256], \tag{18}$$

where $\vec{R}_j$ denote the j-th row of R and $\vec{\mathbf{e}_j}$ is the j-th unit vector for $j \in [256]$. The construction of $H_j \in \mathcal{V}$ is detailed later on. Then, $\vec{\mathbf{z}}$ was computed correctly iff the constant term of all equations in (18) are zero modulo q.

Similarly, to prove $b \in \{-1, 1\}$, public quadratic functions $g \in \mathcal{V}$ and $J_k \in \mathcal{V}$ for $k \in [d-1]$ are constructed such that their evaluations at $(\mathbf{s}', \sigma_{-1}(\mathbf{s}'))$ satisfy

$$g\left(\mathbf{s}', \sigma_{-1}(\mathbf{s}')\right) = (b-1)(b+1) \tag{19}$$

$$J_k\left(\mathbf{s}', \sigma_{-1}(\mathbf{s}')\right) = \mathsf{T}\left(\vec{b}, \overrightarrow{X^k}\right), k \in [d-1]. \tag{20}$$

Then $b \in \{-1, 1\}$ iff Equation (19) gives the zero element in R_q and constant terms of all equations in (20) are zero modulo q.

Therefore, for line 12, we define $\mathsf{v} \coloneqq \{g\}$ and $\mathsf{V} \coloneqq \left\{(J_k)_{k\in[d-1]}, (H_j)_{j\in[256]}\right\}$ as inputs for the subprotocol $\Pi^{(2)}_{eval}$.

	Prover		**Verifier**
1 :	$b \leftarrow \{-1, 1\} \subset \mathsf{R}_q$		
2 :	$\mathbf{y} \leftarrow D^{256/d}_{\mathsf{R}_q,\mathfrak{s}}$		
3 :	$t_b = B_b\mathbf{s_2} + b$		
4 :	$\mathbf{t_y} = B_y\mathbf{s_2} + \mathbf{y}$		
5 :		$\xrightarrow{t_b, \mathbf{t_y}}$	
6 :			$R \leftarrow \mathsf{Bin}_1^{256\times\ell_e}$
7 :		$\xleftarrow{R}$	
8 :	$\vec{\mathbf{z}} = bR\vec{\mathbf{u}} + \vec{\mathbf{y}}$		
9 :	If $\mathsf{rej}_0(\vec{\mathbf{z}}, bR\vec{\mathbf{u}}, \mathfrak{s})$		
10 :	Then continue, else abort		
11 :		$\xrightarrow{\vec{\mathbf{z}}}$	
12 :	Run $\Pi = \Pi^{(2)}_{eval}\left((\mathbf{s}', \mathbf{s_2}), \sigma_{-1}, \mathsf{v}, \mathsf{V}\right)$		return acc iff
13 :			• $\lVert\vec{\mathbf{z}}\rVert_2 \leq t\sqrt{256}\mathfrak{s}$
14 :			• Π verifies.

Fig. 7: The protocol $\Pi_{\mathrm{ANP}}\left((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), (\mathbf{E}, \mathbf{e}, B_e)\right)$ that provides an approximate norm proof for $\mathbf{u} = \mathbf{E}\begin{bmatrix}\mathbf{s_1}\\ \mathbf{m}\end{bmatrix} + \mathbf{e}$.

The construction of H_j from (18)**.** We follow the construction in [74, Section 6.4.4] to derive quadratic functions $H_j \in \mathcal{V}$ that satisfy

$$H_j\left(\mathbf{s}', \sigma_{-1}(\mathbf{s}')\right) := \mathsf{T}\left(b\vec{R}_j, \vec{\mathbf{u}}\right) + \mathsf{T}(\vec{\mathbf{e}_j}, \vec{\mathbf{y}}) - \vec{\mathbf{z}}[j],\ j \in [256]. \tag{21}$$

Let $\mathbf{K}_s \in \mathsf{R}_q^{(m_1+u)\times 2\cdot\dim(\mathbf{s}')}$, $\mathbf{K}_b \in \mathsf{R}_q^{1\times 2\cdot\dim(\mathbf{s}')}$ and $\mathbf{K}_y \in \mathsf{R}_q^{256/d\times 2\cdot\dim(\mathbf{s}')}$ denote projection matrices such that

$$\begin{aligned}\begin{bmatrix}\mathbf{s_1}\\ \mathbf{m}\end{bmatrix} &= \mathbf{K}_s\begin{bmatrix}\mathbf{s}'\\ \sigma_{-1}(\mathbf{s}')\end{bmatrix}\\ b &= \mathbf{K}_b\begin{bmatrix}\mathbf{s}'\\ \sigma_{-1}(\mathbf{s}')\end{bmatrix}\\ \mathbf{y} &= \mathbf{K}_y\begin{bmatrix}\mathbf{s}'\\ \sigma_{-1}(\mathbf{s}')\end{bmatrix}.\end{aligned}$$

Let $\mathbf{r_j} \in \mathsf{R}_q^{\ell_e}$ denote a vector of polynomials such that $\vec{\mathbf{r_j}}$ equals $\vec{R}_j$, the j-th row of R. Let $\mathbf{e_j} \in \mathsf{R}_q^{256/d}$ denote a vector of polynomials such that $\vec{\mathbf{e_j}}$ equals the j-th unit vector of dimension 256.

Then the quadratic function $H_j \in \mathcal{V}$ can be explicitly written as

$$H_j(\mathbf{a}) = \mathbf{a}^\intercal\mathbf{G_j}\mathbf{a} + \mathbf{g_j}\mathbf{a} + g_j, \quad \forall \mathbf{a} \in \mathsf{R}_q^{2(\dim(\mathbf{s}'))}$$

where

$$\begin{aligned}\mathbf{G_j} &= \mathbf{K}_b^\top \cdot \sigma_{-1}(\mathbf{r_j})^\top \cdot \mathbf{E} \cdot \mathbf{K}_s\\ \mathbf{g_j} &= \mathbf{e}^\top \cdot \sigma_{-1}(\mathbf{r_j}) \cdot \mathbf{K}_b + \sigma_{-1}(\mathbf{e_j})^\top\mathbf{K}_y\\ g_j &= -\vec{\mathbf{z}}[j].\end{aligned}$$

C.5 Our vectorized description of the approximate norm bound proof in LNP22

In our vectorized version of the approximate norm bound proof (vec-ANP), the prover knows the secret message $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ which satisfies

$$\left\|\mathbf{W}\begin{bmatrix}\vec{\mathbf{s_1}}\\ \vec{\mathbf{m}}\end{bmatrix} + \mathbf{w}\right\|_\infty \le B_w, \tag{22}$$

for public elements $\mathbf{W} \in \mathbb{Z}_q^{\ell_w \times (m_1+u)d}$, $\mathbf{w} \in \mathbb{Z}_q^{\ell_w}$ and a public bound $B_w \le \frac{q}{41\ell_w^{3/2}\psi^{(L2)}}$. After committing $(\mathbf{s_1}, \mathbf{m})$ into $(\mathbf{t_A}, \mathbf{t_B})$, the commit-and-prove protocol convinces the verifier that the prover knows $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ such that

$\mathcal{O}^{\mathcal{CT}}\left((\mathbf{t_A}, \mathbf{t_B}), (\mathbf{s_1}, \mathbf{m})\right) = \mathsf{acc}$ and

$$\left\| \mathbf{W} \begin{bmatrix} \vec{\mathbf{s_1}} \\ \vec{\mathbf{m}} \end{bmatrix} + \mathbf{w} \right\|_\infty \leq \psi^{(\infty)} \cdot B_w, \tag{23}$$

where the slack is $\psi^{(\infty)} = \psi^{(L2)}\sqrt{\ell_w} = 2\sqrt{\frac{256}{26}}t\gamma\sqrt{337}\sqrt{\ell_w}$.

In contrast, ANP in Appendix C.4, proves the knowledge of secret messages $(\mathbf{s_1}, \mathbf{m}) \in \mathsf{R}_q^{m_1+u}$ such that

$$\left\| \mathbf{E} \begin{bmatrix} \mathbf{s_1} \\ \mathbf{m} \end{bmatrix} + \mathbf{e} \right\| \leq B_e, \tag{24}$$

for public bound B_e and public elements $\mathbf{E} \in \mathsf{R}_q^{\ell_e \times (m_1+u)}$, $\mathbf{e} \in \mathsf{R}_q^{\ell_e}$ with slack $\psi^{(\infty)}$. Note that relation (24) is a special case of the relation (22) by taking $\ell_w = \ell_e \cdot d$ and taking $\mathbf{W}$ and $\mathbf{w}$ as concatenations of rotation matrices for elements in $\mathbf{E}$ and $\mathbf{e}$, respectively.

In the vec-ANP, we define $\vec{\mathbf{u}} := \mathbf{W}\left[\, \vec{\mathbf{s_1}} \quad \vec{\mathbf{m}} \,\right]^\top + \mathbf{w} \in \mathbb{Z}_q^{\ell_w}$. The approximate norm proof for $\|\vec{\mathbf{u}}\|_\infty \leq B_w$ is denoted as

$$\mathit{\Pi}_{\text{vec-ANP}}\left((\mathbf{s_1}, \mathbf{m}, \mathbf{s_2}), (\mathbf{W}, \mathbf{w}, B_w)\right),$$

which contains the same steps as ANP in Figure 7, except that the standard deviation in line 2 is $\mathfrak{s} := \gamma\sqrt{337}\sqrt{\ell_w} \cdot B_w$, the projection matrix R in line 6 is $R \leftarrow \mathsf{Bin}_1^{256\times\ell_w}$, and the quadratic functions H_j as inputs for the subprotocol are derived differently.

The construction of H_j in vec-ANP To derive quadratic functions $H_j \in \mathcal{V}$ that satisfy

$$H_j\left(\mathbf{s}', \sigma_{-1}(\mathbf{s}')\right) := \mathsf{T}\left(b\vec{R}_j, \vec{\mathbf{u}}\right) + \mathsf{T}(\vec{\mathbf{e}_j}, \vec{\mathbf{y}}) - \vec{\mathbf{z}}[j],\ j \in [256], \tag{25}$$

we define $\mathbf{K}_s$, $\mathbf{K}_b$ and $\mathbf{K}_y$ as in Appendix C.4.

Let $\mathbf{r_j}^{(W)} \in \mathsf{R}_q^{(m_1+u)}$ denote a vector of polynomials such that $\overrightarrow{\mathbf{r_j}^{(W)}}$ equals $\vec{R}_j \cdot \mathbf{W} \in \mathbb{Z}_q^{d(m_1+u)}$, and $r_j^{(w)}$ denote $\vec{R}_j \cdot \mathbf{w} \in \mathbb{Z}_q$.

Then the quadratic function $H_j \in \mathcal{P}$ can be explicitly written as

$$H_j(\mathbf{a}) = \mathbf{a}^\intercal \mathbf{G_j} \mathbf{a} + \mathbf{g_j} \mathbf{a} + g_j, \quad \forall \mathbf{a} \in \mathsf{R}_q^{2(\dim(\mathbf{s}'))}$$

where

$$\begin{aligned} \mathbf{G_j} &= \mathbf{K}_b^\intercal \cdot \sigma_{-1}(\mathbf{r_j}^{(W)})^\intercal \cdot \mathbf{K}_s \\ \mathbf{g_j} &= r_j^{(w)} \cdot \mathbf{K}_b + \sigma_{-1}(\mathbf{e_j})^\intercal \cdot \mathbf{K}_y \\ g_j &= -\vec{\mathbf{z}}[j]. \end{aligned}$$

D Our instantiation of the PoD protocol

Let us first discuss how we estimate the amount of ciphertexts to decrypt. This is based on the following lemma.

Lemma 6. *For a set B constructed by taking m random values (with repetition) from a set A, it holds that $f_m(n) \coloneqq \mathbb{E}[|B|] = n(1-(1-1/n)^m)$ with $n = |A|$.*

Now notice that for a FRI query phase that is repeated ℓ times over a domain $|L|$ that is packed into vectors of size P, we can compute the expected number of values to open as

$$1 + 2 \cdot 5 \cdot f_\ell\left(\frac{|L|}{2P}\right) + 2 \cdot \sum_{i=2}^{\log_2\left(\frac{|L|}{P}\right)} f_\ell\left(\frac{|L|}{2^i P}\right)$$

since for each of the ℓ queries, we are taking a random evaluation point in half of each evaluation domain and, in Fractal, opening to the first evaluation domain requires opening to 5 polynomials at the same evaluation point. For the parameters discussed in Section 8,

- $P = 64$ results in on average 2514 ciphertexts to open
- $P = 256$ results in on average 2105 ciphertexts to open

For reference, in the non-blind setting, i.e. $P = 1$, one would open to on average 3728 values.

D.1 Parameters for our instantiation of the PoD

parameters	description	value
$\log q''$	# bits of ciphertext and proof system modulus	48
n''	GBFV ring dimension for PoD	1536
r	average number of ciphertext-plaintext pairs	2514
B^{SZ}_{PoD}	noise bound	$2^{16.9}$
d	proof ring dimension	64
ω	height of $\mathbf{A_1}$, $\mathbf{A_2}$ in ABDLOP	11
m_1	length of the Ajtai message $\mathbf{s_1}$	24
u	length of the BDLOP message $\mathbf{m}$	0
λ	$2\cdot$(# of $g_j \in \mathcal{R}_{d,q'}$ for boosting soundness)	4
m_2	length of the randomness $\mathbf{s_2}$ in ABDLOP	43
γ	rejection sampling constant for Π_{ANP}	5
$\mathfrak{s}_{\mathrm{ANP}}$	standard deviation for Π_{ANP}	614147325
$\mathfrak{s}_1$	standard deviation for $\Pi^{(2)}_{eval}$	1587.2
$\mathfrak{s}_2$	standard deviation for $\Pi^{(2)}_{eval}$	50790.4
ξ	max. coeff. of a challenge in $\mathcal{C}h$	8
D	number of low-order bits cut from $\mathbf{t_A}$	8

Table 5: Parameters for our instantiation of the proof of decryption protocol from Figure 6 with 100-bit security.