Boosting Verifiable Computation on Encrypted Data

Dario Fiore[1], Anca Nitulescu[2], and David Pointcheval[3,4]

[1] IMDEA Software Institute, Madrid, Spain
[2] COSMIAN, Paris, France (work done while being at[3])
[3] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[4] INRIA, Paris, France

Abstract. We consider the setting in which an untrusted server stores a collection of data and is asked to compute a function over it. In this scenario, we aim for solutions where the untrusted server does not learn information about the data and is prevented from cheating. This problem is addressed by verifiable and private delegation of computation, proposed by Gennaro, Gentry and Parno (CRYPTO'10), a notion that is close to both the active areas of homomorphic encryption and verifiable computation (VC). However, in spite of the efficiency advances in the respective areas, VC protocols that guarantee privacy of the inputs are still expensive. The only exception is a protocol by Fiore, Gennaro and Pastro (CCS'14) that supports arithmetic circuits of degree at most 2. In this paper we propose new efficient protocols for VC on encrypted data that improve over the state of the art solution of Fiore et al. in multiple aspects. First, we can support computations of degree higher than 2. Second, we achieve public delegatability and public verifiability whereas Fiore et al. need the same secret key to encode inputs and verify outputs. Third, we achieve a new property that guarantees that verifiers can be convinced about the correctness of the outputs without learning information on the inputs. The key tool to obtain our new protocols is a new SNARK that can efficiently handle computations over a quotient polynomial ring, such as the one used by Ring-LWE somewhat homomorphic encryption schemes. This SNARK in turn relies on a new commit-and-prove SNARK for proving evaluations on the same point of several committed polynomials. We propose a construction of this scheme under an extractability assumption over bilinear groups in the random oracle model.

1 Introduction

Due to the ubiquity of the Internet and the advent of cloud computing, it is increasingly common for users to exchange and receive information processed on remote machines. Online storage services are already widely available on the Internet, and allow users to store, access and share their data from anywhere and from multiple devices. This phenomenon includes not only storage: it is more and more common to rely on computation performed on third party machines.

While this shift in the computing trend brings several benefits, new security challenges also emerge. These challenges are related to a main question: *what happens if the remote machine is not trusted?* In this work we are particularly concerned with two security problems in this space. First, we would like to ensure that the untrusted machine can perform the computation without learning the private data of the users. Second, we would like to enable the receivers of computation results to efficiently check that such results are correct. Both problems are in the scope of two important research lines in cryptography.

Privacy-Preserving Computation. The first problem is related to fully homomorphic encryption (FHE) [RAD78, Gen09]. While for a long time it was only known how to construct homomorphic encryption schemes supporting a single operation (e.g., only addition [Pai99] or multiplication [ElG84]), Gentry's breakthrough showed the first FHE scheme that enables computing any function on encrypted data. If Gentry's first FHE was mostly a feasibility result, research in this area has progressed significantly giving rise to many new FHE schemes (e.g., [SV10, BV11, BGV12, GSW13, DM15, CGGI16, CGGI17]) that are efficient and see their first practical applications.

Ensuring Correctness of Computation. The second problem is related to verifiable computation (VC) [GGP10] and related notions such as interactive proofs [GMR85], probabilistically checkable proofs [AS92] and succinct arguments [Kil92]. Briefly speaking, these are protocols that enable a powerful prover to convince a verifier that a statement (e.g., correctness of a computation, $y = f(x)$) is true in such a way that the verifier can run with fewer resources, e.g., faster than re-executing the function. Similarly to FHE, also in this research area, results have been confined to theory for long time. However, several recent works have shown a change in this trend, and today we have several VC protocols that are efficient and have been experimented in practical scenarios, see e.g., [GKR08, CMT12, GGPR13, PHGR13, BCG+13, Gro16, ZGK+17, WJB+17, AHIV17, WTs+18, BCG+18, BBC+18, BCR+19, MBKM19, CFQ19, XZZ+19].

1.1 Ensuring Correctness of Privacy-Preserving Computation

In spite of the research mentioned above, the problem of ensuring both the correctness and the privacy of computation performed on untrusted machines has received much less attention in the literature. There are three main works that considered explicitly this problem.

The first one is the seminal paper of Gennaro et al. [GGP10] who introduced the notion of non-interactive verifiable computation. In [GGP10] they indeed show how to combine garbled circuits and FHE in order to build a VC scheme for arbitrary functions that also preserves the privacy of the computation's inputs and outputs against the computing machine.

The second work is that of Goldwasser et al. [GKP+13] that shows how to use their succinct single-key functional encryption scheme in order to build a VC protocol that preserves the privacy of the inputs (but not of the outputs).

Both these two solutions [GGP10, GKP+13] are however not very satisfactory in terms of efficiency. The main issue in the construction of [GGP10] is that

they need the full power of FHE to perform homomorphic evaluations of garbled circuits. Some of the efficiency issues in [GKP+13] include the use of several instances of an attribute-based encryption that must support an expressive class of predicates (at NC1 circuits), and an inherent design limitation (due to following the approach of [PRV12]) by which their scheme supports functions with a single bit of output (which in practical scenarios like computing on large integers would require multiple instances of their protocol).

A third work that considered the problem of ensuring correctness of privacy-preserving computation is the one of Fiore et al. [FGP14] who proposed a solution that combines an FHE and a VC scheme. The idea of their generic construction is rather simple and consists into using a VC in order to prove that the homomorphic evaluation on ciphertexts has been done correctly. As discussed in [FGP14], even this solution may encounter efficiency limits. This is due to the fact that the VC scheme must be executed on a computation that, due to the FHE ciphertext expansion, is of much larger representation than the computation that would be executed on plain text. Motivated by this issue, [FGP14] also proposed an efficient solution that, for the case of quadratic functions, can avoid this issue. The efficient construction in [FGP14] overcomes the problem of ciphertext expansion in two ways: (1) they consider homomorphic encryption schemes working in the Ring-LWE setting in which ciphertexts are represented by polynomials in a given polynomial ring; (2) they develop, as the VC building block, an homomorphic MAC scheme especially tailored to handle messages that are polynomials in which the prover execution can be independent of the degree of such polynomials. However, for reasons that we will detail later (see Section 3), their technique is inherently bound to computations of multiplicative depth 1. Also, by using an homomorphic MAC as VC, verification requires a secret key, the same secret key used to encode the inputs. This limits the applicability of these solutions to scenarios where users and verifiers are either the same entity or they share a secret key.

1.2 Our Contributions

We propose a new protocol for verifiable computation on encrypted data that improves on the state-of-the-art solution of Fiore et al. [FGP14] in multiple aspects. Notably, we can support HE computations of multiplicative depth larger than 1. Second, we achieve public verifiability whereas [FGP14] is only privately verifiable. Finally, our scheme has an additional property that guarantees that verifiers may be convinced of outputs correctness without learning information on the original inputs. This latter property is particularly relevant in the publicly verifiable setting where the users who encrypt the data and the verifiers are distinct entities. Technically, we achieve this property because our protocol allows for re-randomizing the encrypted results, which was not possible in [FGP14] that only considered deterministic HE evaluations.

Our key tool to obtain this result is a new SNARK that can efficiently handle computations that are arithmetic circuits f over a quotient polynomial ring $\mathbb{R}_q := \mathbb{Z}_q[X]/\langle R(X)\rangle$ (exactly like the popular choice for many Ring-LWE schemes) in which the prover's costs have a minimal dependence on the degree d of

$R(X)$. Specifically, let f be the circuit over $\mathbb{R}_q$ and $\hat{f}$ be the corresponding circuit over $\mathbb{Z}_q$ (i.e., the one that would be computed on plaintexts where additions and multiplications in $\mathbb{R}_q$ are replaced by the corresponding operations in $\mathbb{Z}_q$). Then, whereas a naive application of [FGP14]'s generic solution would incur a cost for proof generation at least $O(d \cdot |\hat{f}|)$ where $|\hat{f}|$ is $\hat{f}$'s circuit size, our scheme lets proof generation be doable in time $O(d \cdot n + |\hat{f}|)$ where n is $\hat{f}$'s input size. We stress that here we are considering the cost of proof generation, after having performed the HE computation; or in other words we consider the cost of generating the proof once the witness is available. To see how this efficiency feature concretely improves, consider for example an $\hat{f}$ that is a multivariate polynomial of degree $c \geqslant 2$ by which $|\hat{f}|$ can be n^c, and consider that for Ring-LWE security the degree d can be a rather large integer (e.g., $d \approx 8000$). Then removing the multiplicative factor $d \cdot |\hat{f}|$ can significantly speed up the prover's costs. Let us also notice that the factor $d \cdot n$ is unavoidable as the prover must read the input.

Our SNARK for arithmetic circuits over polynomial rings is built in a modular way using two building blocks: a commit-and-prove SNARK for arithmetic circuits ($\mathsf{AC\text{-}\Pi}$), and a commit-and-prove SNARK for multiple polynomial evaluations ($\mathsf{MUniEv\text{-}\Pi}$).

To instantiate $\mathsf{AC\text{-}\Pi}$, we can use any commit-and-prove SNARK for arithmetic circuits that supports the same commitment key as our scheme $\mathsf{MUniEv\text{-}\Pi}$. Given the recent result of Campanelli et al. [CFQ19], $\mathsf{MUniEv\text{-}\Pi}$ can be instantiated with a variety of schemes including the efficient commit-and-prove variant of Groth16 [Gro16] proposed in [CFQ19].

For scheme $\mathsf{MUniEv\text{-}\Pi}$, we propose a construction based on the Strong Diffie-Hellman and Power Knowledge of Exponent (PKE) assumptions in bilinear groups, in the random oracle model. We believe this scheme can also have other applications. Slightly more in detail, $\mathsf{MUniEv\text{-}\Pi}$ allows one to prove the following statement: given a commitment C to ℓ degree-d polynomials $\{P_j(X)\}_j$, a commitment C' to a vector of ℓ $\mathbb{Z}_q$-elements $\{p_j\}_j$, and a public point k, show that $p_j = P_j(k)$ for all $j = 1$ to ℓ. In comparison to using an existing general-purpose commit-and-prove SNARK for arithmetic circuits (e.g., LegoGroth16 from [CFQ19]), our scheme $\mathsf{MUniEv\text{-}\Pi}$ has slightly smaller proofs and proving time at least three times faster (cf. Section 7.3 for more details).

Finally, we note that our scheme $\mathsf{MUniEv\text{-}\Pi}$ is in turn constructed from a SNARK $\mathsf{BivPE\text{-}\Pi}$ for the partial evaluation of a committed bivariate polynomial, i.e., given commitments C and C' to $P(X, Y)$ and $Q(Y)$ respectively, prove that $Q(Y) = P(k, Y)$ for some public point k. We construct $\mathsf{BivPE\text{-}\Pi}$ by extending the univariate polynomial commitment techniques of Kate et al. [KZG10]. It is worth mentioning that other works [PST13, ZGK+17] extended [KZG10] to support the evaluation of multivariate polynomials, which include bivariate ones. However, the main difference (crucial for our application) is that we can support *partial* evaluations in one variable while keeping the result polynomial also committed.

1.3 Organization

In Section 2 we introduce notation and basic cryptographic definitions. Section 3 describes our SNARK for arithmetic computations in quotient polynomial rings. In

Section 4 we show how to combine our SNARK from Section 3 together with Ring-LWE-based HE schemes in order to build a verifiable computation scheme with input and output privacy, and also how to achieve the new property of preserving privacy of the inputs from the verifier. In Section 5 we state the computational assumptions needed by our schemes, and we present the $\mathsf{BivPoly}.\mathcal{C}om$ commitment. In Section 6 we build our SNARK $\mathsf{BivPE}\text{-}\Pi$ for bivariate polynomials partial evaluation and then in Section 7 we show how turn it into an efficient $\mathsf{MUniEv}\text{-}\Pi$ for the simultaneous evaluation of many univariate polynomials. In Section 8 we prove the security of our scheme $\mathsf{BivPE}\text{-}\Pi$.

2 Notation and Definitions

Notation. An adversary is denoted by $\mathcal{A}$ and is assumed to be probabilistic Turing machines that run in polynomial time, i.e., PPT. For two PPT machines A, B, with the writing $(\mathsf{A}\|\mathsf{B})(x)$ we denote the execution of A followed by the execution of B on the same input x and with the same random coins.

2.1 Commitment Schemes

Definition 1 (Non-Interactive Commitment). *A non-interactive commitment scheme is a tuple of algorithms* $\mathcal{C}om = (\mathsf{ComGen}, \mathsf{Com}, \mathsf{ComVer}, \mathsf{OpenVer})$*:*

- $\mathsf{ComGen}(1^\lambda) \to \mathsf{ck}$*: Generates a commitment public key* ck*. It specifies a message space* M_{ck}*, a randomness (opening) space* R_{ck}*, and a commitment space* C_{ck}*. This algorithm is run by a trusted or distributed authority;*
- $\mathsf{Com}(\mathsf{ck}, m) \to (c, o)$*: Outputs a commitment* c *and an opening* o*. Given a message* $m \in M_{\mathsf{ck}}$*, it samples* $o \in R_{\mathsf{ck}}$ *and computes the commitment* (c, o)*.*
- $\mathsf{ComVer}(\mathsf{ck}, c) \to 0/1$*: Checks whether* c *is a well-formed commitment. If so, it outputs 1, otherwise it outputs 0;*
- $\mathsf{OpenVer}(\mathsf{ck}, c, m, o) \to 0/1$*: Outputs* 1 *if the value* $m \in M_{\mathsf{ck}}$ *is the committed message in the commitment* c *and* 0 *if* (m, o, c) *does not correspond to a valid pair opening-commitment.*

We say $\mathcal{C}om = (\mathsf{ComGen}, \mathsf{Com}, \mathsf{ComVer}, \mathsf{OpenVer})$ *is a secure commitment scheme if it satisfies the following properties:*

Correctness. *Let* $\mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda)$*. Any commitment of* $m \in M_{\mathsf{ck}}$ *honestly generated* $(c, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m)$ *is successfully verified by* $\mathsf{ComVer}(\mathsf{ck}, c)$ *and by* $\mathsf{OpenVer}(\mathsf{ck}, c, m, o)$*.*

Hiding. *It is statistically hard, for any adversary* $\mathcal{A}$*, to generate two messages* $m_0, m_1 \in M_{\mathsf{ck}}$ *such that* $\mathcal{A}$ *can distinguish between their corresponding commitments* c_0 *and* c_1 *where* $(c_0, o_0) \leftarrow \mathsf{Com}(\mathsf{ck}, m_0)$ *and* $(c_1, o_1) \leftarrow \mathsf{Com}(\mathsf{ck}, m_1)$*.*

Binding. *It is computationally hard, for any adversary* $\mathcal{A}$*, to come up with a collision* (c, m_0, o_0, m_1, o_1)*, such that* o_0 *and* o_1 *are valid opening values for two different pre-images* $m_0 \neq m_1$ *for* c*. For any adversary* $\mathcal{A}$*, the following probability is negligible*

$$\Pr\left[\begin{array}{c} \mathsf{OpenVer}(\mathsf{ck}, c, m_0, o_0) = 1 \\ \wedge\ \mathsf{OpenVer}(\mathsf{ck}, c, m_1, o_1) = 1 \\ \wedge\ m_0 \neq m_1 \end{array} \middle| \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda) \\ (c, (m_0, o_0), (m_1, o_1)) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right].$$

Knowledge Binding [BL07]. *For every adversary $\mathcal{A}$ that produces a valid commitment c associated to a message that verifies, i.e. such that* $\mathsf{ComVer}(\mathsf{ck}, c) = 1$*, there is an extractor* $\mathsf{Ext}_{\mathcal{A}}$ *that is able to output a pre-image m and a valid opening o of c, with overwhelming probability:*

$$\Pr\left[\mathsf{OpenVer}(\mathsf{ck}, c, m, o) = 1 \;\middle|\; \begin{array}{r}\mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda)\\ (c; (m, o)) \leftarrow (\mathcal{A} \| \mathsf{Ext}_{\mathcal{A}})(\mathsf{ck})\\ \mathsf{ComVer}(\mathsf{ck}, c) = 1\end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

For the sake of simplicity, throughout this work, we will omit the commitment key ck from the input of the algorithms, and with a slight abuse of notation, we will adopt the writing $\mathsf{Com}(m) \rightarrow (c, o)$.

2.2 SNARKs – Succinct Non-Interactive Arguments of Knowledge

We recall the definition of (zero-knowledge) succinct non-interactive arguments of knowledge (zk-SNARKs).

Definition 2 (SNARK for NP). *A SNARK is defined by three algorithms,*

$\Pi.\mathsf{Gen}(1^\lambda, \mathcal{R}) \rightarrow \mathsf{crs}$**:** *on input a security parameter $\lambda \in \mathbb{N}$ and a* NP *relation $\mathcal{R}$, the generation algorithm outputs a common reference string* crs*;*

$\Pi.\mathsf{Prove}(\mathsf{crs}, u, w) \rightarrow \pi$**:** *given a prover reference string* crs*, an instance u and a witness w s.t. $(u, w) \in \mathcal{R}$, this algorithm produces a proof π;*

$\Pi.\mathsf{Ver}(\mathsf{crs}, u, \pi) \rightarrow b$**:** *on input a verification state* crs*, an instance u, and a proof π, the verifier algorithm outputs $b = 0$ (reject) or $b = 1$ (accept);*

satisfying completeness*,* succinctness*,* knowledge-soundness *as described below:*

Correctness. *For all valid statement $(u, w) \in \mathcal{R}$,*

$$\Pr\left[\begin{array}{c}\mathsf{Ver}(\mathsf{crs}, u, \pi) = 0\\ \wedge\ \ (u, w) \in \mathcal{R}\end{array}\;\middle|\; \begin{array}{l}\mathsf{crs} \leftarrow \Pi.\mathsf{Gen}(1^\lambda, \mathcal{R})\\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, u, w)\end{array}\right] = \mathsf{negl}(\lambda);$$

Succintness. *The size of the proof is linear in the security parameter λ, i.e. independent of the size of the computation or the witness;*

Knowledge-Soundness [BG93]. *A non-interactive proof system Π is* knowledge-sound *for the class $\mathcal{Z}$ of auxiliary input generators if for any* PPT *adversary $\mathcal{A}^{\mathsf{KS}}$ there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ such that:*

$$\Pr\left[\begin{array}{c}\mathsf{Ver}(\mathsf{crs}, u, \pi) = 1\\ \wedge\ \ \mathcal{R}(u, w) = 0\end{array}\;\middle|\; \begin{array}{l}\mathsf{crs} \leftarrow \Pi.\mathsf{Gen}(1^\lambda, \mathcal{R}), \mathsf{aux} \leftarrow \mathcal{Z}(\mathsf{crs})\\ ((u, \pi); w) \leftarrow (\mathcal{A}^{\mathsf{KS}} \| \mathsf{Ext}_{\mathcal{A}})(\mathsf{crs}, \mathsf{aux})\end{array}\right] = \mathsf{negl}(\lambda).$$

Zero Knowledge. *A Π protocol is a (statistical) zero-knowledge for a relation $\mathcal{R}$ if there exists a stateful interactive polynomial-size simulator* $\mathsf{Sim} = (\mathsf{Sim}^{\mathsf{crs}}, \mathsf{Sim}^{\mathsf{Prove}})$ *such that for all stateful interactive distinguishers $\mathcal{D}$, for every large enough security parameter $\lambda \in \mathbb{N}$, every auxiliary input* aux*, the two probabilities are negligibly close:*

$$\begin{aligned}&\Pr[(u, w) \in \mathcal{R} \wedge \mathcal{D}(\pi) = 1 \,|\, (\mathsf{crs}) \leftarrow \mathsf{Gen}(1^\lambda), (u, w) \leftarrow \mathcal{D}(\mathsf{crs}, \mathsf{aux}),\\ &\qquad\qquad \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, u, w)];\\ &\Pr[(u, w) \in \mathcal{R} \wedge \mathcal{D}(\pi) = 1 \,|\, (\mathsf{crs}, \mathsf{trap}) \leftarrow \mathsf{Sim}^{\mathsf{crs}}(1^\lambda), (u, w) \leftarrow \mathcal{D}(\mathsf{crs}, \mathsf{aux}),\\ &\qquad\qquad \pi \leftarrow \mathsf{Sim}^{\mathsf{Prove}}(\mathsf{crs}, \mathsf{trap}, u, \mathsf{aux})].\end{aligned}$$

Commit and Prove SNARKs. Let $\mathcal{R}(u, w)$ be an NP relation where $w = (\{x_i\}_i, \omega)$. A commit-and-prove SNARK ($\mathsf{CaP\text{-}SNARK}$) for commitment scheme $\mathcal{C}om$ and relation $\mathcal{R}(u, w)$ is a SNARK for the "commit-and-prove relation" $\mathcal{R}^{\mathsf{ck}}(u^*, w^*)$ where $u^* = (u, \{c_i\}_i)$, $w^* = (\{x_i\}_i, \{o_i\}_i, \omega)$ and that holds iff $\mathcal{R}(u, (\{x_i\}_i, \omega))$ holds and $\mathsf{OpenVer}(c_i, m_i, o_i) = 1$ for all i. We adopt the syntax for $\mathsf{CaP\text{-}SNARK}$ used in [CFQ19]:

$\Pi.\mathsf{Gen}(\mathsf{ck}, \mathcal{R}) \to \mathsf{crs}$**:** on input a relation-independent commitment key ck and a NP relation $\mathcal{R}$, it generates the crs;

$\Pi.\mathsf{Prove}(\mathsf{crs}, (u, \{c_i\}_i), (\{x_i\}_i, \{o_i\}_i, \omega)) \to \pi$**:** outputs a proof;

$\Pi.\mathsf{Ver}(\mathsf{crs}, (u, \{c_i\}_i), \pi) \to b$**:** rejects or accepts the proof.

3 Proof Systems for Arithmetic Function Evaluation over Quotient Polynomial Rings

In this section we describe our commit-and-prove SNARK for arithmetic computations in quotient polynomial rings.

Let $\mathbb{R}$ be the quotient ring $\mathbb{Z}/\langle R(X)\rangle$ for some polynomial $R \in \mathbb{Z}[X]$ of degree d. For a prime $q \gg d$ we define $\mathbb{F} = \mathbb{Z}_q$ a finite field and $\mathbb{R}_q = \mathbb{R}/q\mathbb{R}$. We want to construct a succinct non-interactive zero-knowledge argument system for some relation $\mathcal{R}_f$ of correct evaluation of an arithmetic function $f(\cdot) : {\mathbb{R}_q}^n \to \mathbb{R}_q$ taking $n \in \mathbb{N}$ inputs in the quotient ring $\mathbb{R}_q = \mathbb{R}/q\mathbb{R}$. The function f to be evaluated on polynomials $\{P_j\}_{j=1}^n$ in the quotient ring $\mathbb{R}_q$ is considered to be public.

Let $\mathsf{MPoly\text{-}}\mathcal{C}om = (\mathsf{MPoly.ComGen}, \mathsf{MPoly.Com}, \mathsf{MPoly.ComVer}, \mathsf{MPoly.OpenVer})$ be a linearly homomorphic commitment scheme for (many) univariate polynomials, i.e., the message space M consists in vectors of $n \leqslant \ell$ polynomials of degree $d \leqslant \nu$, for some integer bounds ℓ, ν chosen in $\mathsf{MPoly.ComGen}$. In Section 7 we show an efficient instantiation of such a scheme in bilinear groups.

We describe a Commit-and-Prove SNARK, $\mathsf{Rq\text{-}\Pi}$, for commitment scheme $\mathsf{MPoly\text{-}}\mathcal{C}om$ and for the following relation

$$\mathcal{R}_f^{\mathsf{ck}} := \{(u = (C, P); w = (\{P_j\}_{j=1}^n, \rho, T)) : \\ \mathsf{MPoly.OpenVer}(C, \{P_j\}, \rho) = 1 \ \wedge \ P = f(P_j) - TR\}$$

The relation $\mathcal{R}_f^{\mathsf{ck}}$ implicitly contains two bounds ℓ, ν on, respectively, the number of inputs of f and the degree d_f of f as an arithmetic circuit.

In a nutshell, given a compact commitment C and a public polynomial $P \in \mathbb{R}_q$, our $\mathsf{Rq\text{-}\Pi}$ scheme allows to prove that C opens to some polynomials $P_j \in \mathbb{R}\ \forall j = 1 \dots n$ such that P is the result of evaluating the function f on $\{P_j\}_j$, evaluation done in the polynomial ring $\mathbb{R}_q$.

High-Level Description of our $\mathsf{Rq\text{-}\Pi}$ SNARK. We build our $\mathsf{Rq\text{-}\Pi}$ scheme as a combination of the following building blocks:

- $\mathsf{MUniEv\text{-}\Pi} = (\mathsf{MUniEv\text{-}\Pi.Gen}, \mathsf{MUniEv\text{-}\Pi.Prove}, \mathsf{MUniEv\text{-}\Pi.Ver})$: a $\mathsf{CaP\text{-}SNARK}$ for the simultaneous evaluation of n univariate polynomials $\{P_j\}_{j=1}^n$ in a point k, where $\{P_j\}$ are committed with $\mathsf{MPoly\text{-}}\mathcal{C}om$. Proposing efficient constructions of $\mathsf{MPoly\text{-}}\mathcal{C}om$ and $\mathsf{MUniEv\text{-}\Pi}$ are key technical contributions of this paper; these are detailed in Section 7.

– $\mathsf{AC}\text{-}\Pi = (\mathsf{AC}\text{-}\Pi.\mathsf{Gen}, \mathsf{AC}\text{-}\Pi.\mathsf{Prove}, \mathsf{AC}\text{-}\Pi.\mathsf{Ver})$: a $\mathsf{CaP}\text{-}\mathsf{SNARK}$ for arithmetic circuits over $\mathbb{Z}_q$ where inputs and outputs are committed (as a vector of degree-0 polynomials) with the $\mathsf{MPoly}\text{-}\mathcal{C}om$ scheme.
Various instantiations of $\mathsf{AC}\text{-}\Pi$ compatible with our pairing-based $\mathsf{MPoly}\text{-}\mathcal{C}om$ commitment can be obtained by using for example the compiler recently proposed in [CFQ19];[5] a particularly efficient one is a commit-and-prove variant of [Gro16].

The two building blocks above are used as follows.

The prover, knowing a quotient polynomial $T \in \mathbb{Z}_q[X]$ such that $f((P_j)_j) = P + TR$, starts by computing a commitment C_T to $T \in \mathbb{Z}_q[X]$ (which may have degree higher than that of R).

Next, the key idea is that instead of directly proving that $P = f((P_j)_j) - TR$ for the committed polynomials $\{P_j\}$ and T (that would require to work with a large arithmetic circuits f), we use the homomorphic properties of the polynomial ring $\mathbb{Z}_q[X]$ to "compress the computation". Namely, to prove $P = f((P_j)_j) - TR$, we evaluate all the polynomials in a random point k and then prove the relation on the resulting scalars, using the fact that:

$$\hat{f}(P_j(k)) - R(k)T(k) = (f(P_j) - RT)(k) = P(k).$$

where $\hat{f} : \mathbb{Z}_q^n \to \mathbb{Z}_q$ is an arithmetic circuit that is the same as f except that every addition (resp. multiplication) in $\mathbb{R}_q$ is replaced by an addition (resp. multiplication) in $\mathbb{Z}_q$.

This idea is similar to the homomorphic hash function defined by Fiore et al. [FGP14]. In [FGP14], they let this idea work by evaluating the polynomials "in the exponent", i.e., they publish a set of group elements g^{k^i}, and then they compute homomorphically over these encodings to get $g^{P(k)}$.

This technique however hits two problems: first, they cannot deal with reductions modulo $R(X)$, and second, to compute homomorphically a multiplication on these encodings, they have to "consume" a pairing, and thus only degree-2 computations can be supported.

In our case, we solve these issues by exploiting the power of the commit and prove paradigm in order to obtain, for every evaluation, a fresh random k. Then, having $k \in \mathbb{Z}_q$ allows us to support higher-degree computations as well as to deal with modular reductions.

To proceed with the protocol, the prover thus needs to get a random point k, not of its choice and independent of the values committed in C_T and C and of the statement P. This is possible by using a random oracle Hash to obtain a value k on which it evaluates the polynomials $\{P_j(k) = p_j\}_{j=1}^n, R(k) = r, P(k) = p$ and $T(k) = t'$.

Next, the prover compactly commits to the respective evaluations $(C', \rho') \leftarrow \mathsf{MPoly.Com}(t', \{p_j\}_{j=1}^n)$. At this point the prover will use:

[5] In particular, relevant to our work is the compiler that shows that commit-and-prove SNARKs for Pedersen-like commitments can be made compatible with one another.

$\underline{\mathsf{Rq\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_f^{\mathsf{ck}}) \to \mathsf{crs}}$

1: $\mathsf{crs}_C \leftarrow \mathsf{MUniEv\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_{\mathsf{eval}}),\ \mathsf{crs}' \leftarrow \mathsf{AC\text{-}\Pi.Gen}(\mathsf{ck}, \hat{\mathcal{R}}_f)$
2: return $\mathsf{crs} := (\mathsf{crs}_C, \mathsf{crs}')$

$\underline{\mathsf{Rq\text{-}\Pi.Prove}(\mathsf{crs}, x, w)}$

1: $(C, P) := x, (\{P_j\}_{j=1}^n, \rho, T) := w$
2: $(C_T, \tau) \leftarrow \mathsf{MPoly.Com}(T)$
3: $k \leftarrow \mathsf{Hash}(C, P, C_T)$
4: $p = P(k), r = R(k),$
5: $t' = T(k), p_j = P_j(k)$
6: $(C', \rho') \leftarrow \mathsf{MPoly.Com}(t', \{p_j\})$
7: $u_C := (C_T \times C, C', k)$
8: $\pi_C \leftarrow \mathsf{MUniEv\text{-}\Pi.Prove}(\mathsf{crs}_C, u_C, w_C),$
9: $\pi' \leftarrow \mathsf{AC\text{-}\Pi.Prove}(\mathsf{crs}', u' = (C', p, r), w')$
10: returns $\pi = (C_T, C', \pi_C, \pi')$

$\underline{\mathsf{Rq\text{-}\Pi.Ver}(\mathsf{crs}, x = (C, P), \pi)}$

1: $\pi := (C_T, C', \pi_C, \pi')$
2: $k \leftarrow \mathsf{Hash}(C, P, C_T)$
3: $p := P(k), r := R(k)$
4: $u_C := (C_T \times C, C', k)$
5: $u' := (C', p, r)$
6: $b_C \leftarrow \mathsf{MUniEv\text{-}\Pi.Ver}(\mathsf{crs}_C, u_C, \pi_C)$
7: $b' \leftarrow \mathsf{AC\text{-}\Pi.Ver}(\mathsf{crs}', u', \pi')$
8: return $(b_C \wedge b') = 1.$

Fig. 1. Our SNARK $\mathsf{Rq\text{-}\Pi}$ for Evaluations over Polynomial Rings

1. the $\mathsf{MUniEv\text{-}\Pi}$ scheme to prove that C' is a commitment to a vector of $n+1$ scalars $(t, \{p_j\}_{j=1}^n)$ that are the results of evaluating in point k a vector of $n+1$ polynomials $(T, \{P_j\}_{j=1}^n)$ that are committed in $C_T \times C$; [6]
2. the $\mathsf{AC\text{-}\Pi}$ scheme to prove that $p = P(k) = \hat{f}((p_j)_j) - rt'$, and that $t', \{p_j\}_{j=1}^n$ are openings of C'.

More formally, the algorithms of the protocol are described in Figure 1. A detailed intuition of the functionalities of each algorithm follows.

3.1 Formal Description of Our Rq-Π Scheme.

We construct a commit-and-prove SNARK scheme $\mathsf{Rq\text{-}\Pi} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ for any relation $\mathcal{R}_f^{\mathsf{ck}}$ with respect to some bounds ℓ, ν on the cardinality of $\{P_j\}_j$ and on the degree d_f of f.

Relations for MUniEv-Π and AC-Π. We define the intermediate statements $\mathcal{R}_{\mathsf{eval}}, \hat{\mathcal{R}}_f$ to be proven using the two SNARKs, $\mathsf{MUniEv\text{-}\Pi}$ and $\mathsf{AC\text{-}\Pi}$:

$\mathcal{R}_{\mathsf{eval}}$**:** We first define the relation for simultaneous evaluation of multiple polynomials on a point k, to be supported by $\mathsf{MUniEv\text{-}\Pi}$. The prover has to convince the verifier that for a given point k (that in our case is random, but part of the statement) and two commitments $C_T \times C$ and C', it knows the corresponding opening values $(T, \{P_j\}_j, \tau + \rho)$ and $(t', \{p_j\}_j, \rho')$ such that $P_j(k) = p_j$ for all j, and $T(k) = t'$.

[6] We consider linearly homomorphic commitment schemes $\mathsf{MPoly}\text{-}\mathcal{C}om$ and we commit in C_T and C to vectors of $n+1 \leqslant \ell$ polynomials $(C_T, \tau) \leftarrow \mathsf{MPoly.Com}(T, 0, 0 \ldots 0)$ and $(C, \rho) \leftarrow \mathsf{MPoly.Com}(0, \{P_j\}_{j=1}^n)$ with an appropriate number of 0's, i.e., $(T, \{P_j\}) = (T, \{0\}) + (0, \{P_j\})$, such that computing $C_T \times C$ results in a commitment $(C_T \times C, \tau + \rho) \leftarrow \mathsf{MPoly.Com}(T, \{P_j\}_{j=1}^n)$ to the concatenation of T, $\{P_j\}$.

More formally, $\mathsf{MUniEv\text{-}\Pi.Prove}$ takes as input a statement $u_C = (C_T \times C, C', k)$, and a witness $w_C = ((T, \{P_j\}), (t', \{p_j\}), \tau + \rho, \rho')$, and $\mathcal{R}_{\mathsf{eval}}$ holds for (u_C, w_C) iff:

$$\mathcal{R}_{\mathsf{eval}} := \{(u_C, w_C) : \forall j,\ p_j = P_j(k) \wedge t' = T(k) \wedge\ (C', \rho') = \mathsf{MPoly.Com}(t', \{p_j\}) \\ \wedge\ (C_T \times C, \tau + \rho) = \mathsf{MPoly.Com}(T, \{P_j\}_j)\}.$$

$\hat{\mathcal{R}}_f$**:** We define the relation for correct computation of $\hat{f}$, to be supported by $\mathsf{AC\text{-}\Pi}$. The prover has to convince the verifier that an equality holds for some scalar values $t', \{p_j\}, p, r \in \mathbb{Z}_q$. The inputs p, r are known by the verifier (they are public) and $t', \{p_j\}$ are given implicitly in a committed form $(C', \rho') = \mathsf{MPoly.Com}(t', \{p_j\})$. More formally, given a statement $u' = (C', p, r)$ and a witness $w' = (\rho', t', \{p_j\})$ for the computation $p = \hat{f}(p_j) - rt'$ and for the opening of C', the relation is defined as follows:

$$\hat{\mathcal{R}}_f := \{(u', w') : p = \hat{f}(p_j) + rt' \ \wedge\ (C', \rho') = \mathsf{MPoly.Com}(t', \{p_j\})\}.$$

CRS Generation. The setup algorithm $\mathsf{Rq\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_f^{\mathsf{ck}})$, given a commitment key $\mathsf{ck} \leftarrow \mathsf{MPoly.ComGen}(1^\lambda)$ that supports commitments up to ℓ different polynomials $P_j \in \mathbb{R}_q$ (all of degrees $\leqslant d$) and one commitment to a polynomial $T \in \mathbb{Z}_q[X]$ of higher degree (up to ν)[7] and the NP relation $\mathcal{R}_f^{\mathsf{ck}}$ including the bound parameters ℓ, ν, outputs a crs enabling the proof and verification of a function f of degree $d_f < \nu$ over a set of polynomials $\{P_j\}_{j=1}^n$ of cardinality $n \leqslant \ell$.

First it runs the generation algorithm for $\mathsf{MUniEv\text{-}\Pi}$ and computes a part of the setup, $\mathsf{crs}_C \leftarrow \mathsf{MUniEv\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_{\mathsf{eval}})$.

Then it generates a common reference string for $\mathsf{AC\text{-}\Pi}$ that will be used for proving computations of $\hat{f}$: $\mathsf{crs}' \leftarrow \mathsf{AC\text{-}\Pi.Gen}(\mathsf{ck}, \hat{\mathcal{R}}_f)$. As an observation, $\mathsf{AC\text{-}\Pi}$ assumes commitments to vectors of scalars; these can be done with MPoly-$\mathcal{C}om$, by seeing them as vectors of degree-0 polynomials.

Prover. Given a reference string crs, statement $u = (C, P)$ and witness $w = (\{P_j\}_{j=1}^n, \rho, T)$ where P is a public polynomial, C is a compact commitment to polynomials $\{P_j\}_{j=1}^n \in \mathbb{R}_q$ with opening ρ, and $T \in \mathbb{Z}_q[X]$ is a quotient polynomial, the prover algorithm produces a proof π that $f((P_j)_j) = P + TR$ as follows:

- The prover commits to $T = \sum_{i=0}^{\nu} T_i X^i$: $(C_T, \tau) \leftarrow \mathsf{MPoly.Com}(T)$.
- The prover then runs $k \leftarrow \mathsf{Hash}(C, P, C_T)$ to obtain a random value k.
- The prover evaluates the polynomials in k : $\{P_j(k) = p_j\}_{j=1}^n, R(k) = r$, $P(k) = p$ and $T(k) = t$.
- The prover commits the respective evaluations $t', \{p_j\}_{j=1}^n$ as $(C', \rho') \leftarrow \mathsf{MPoly.Com}(\mathsf{ck}, t', \{p_j\}_{j=1}^n)$.

[7] The commitment key ck can have some special property for optimization, for example, it may consist of two keys, one for committing to polynomials $P_j \in \mathbb{R}_q$ of degrees $\leqslant d$ and another longer key to commit to polynomials $T \in \mathbb{Z}_q[X]$ of degree ν.

– The prover runs the algorithm for $\mathsf{MUniEv\text{-}\Pi}$ to prove that the opening values $t', \{p_j\}$ of the commitment C' are evaluation in k of the polynomials $T, \{P_j\}$, committed in $C_T \times C$:

$$\pi_C \leftarrow \mathsf{MUniEv\text{-}\Pi.Prove}(\mathsf{crs}_C, u_C, w_C)$$

$$\text{where } u_C = (C_T \times C, C', k),\ w_C = ((T, \{P_j\}), (t', \{p_j\}), \tau + \rho, \rho').$$

– It then runs the proving algorithm of $\mathsf{AC\text{-}\Pi}$ for proving the evaluation of $\hat{f}$ on scalars with a witness w' for the computation $p = \hat{f}(p_j) - rt$ and for the opening of C':

$$\pi' \leftarrow \mathsf{AC\text{-}\Pi}(\mathsf{crs}', u' = (C', p, r), w').$$

– The prover eventually outputs $\pi = (C_T, C', \pi_C, \pi')$.

Verifier. The algorithm Ver on input a statement $u = (C, P)$ and a proof $\pi := (C_T, C', \pi_C, \pi')$ recomputes the randomness k by running $k \leftarrow \mathsf{Hash}(C, P, C_T)$. Then the Verifier has only to evaluate the known polynomials P, R in k obtaining $p := P(k), r := R(k)$. Once it has all the elements to redefine the two statements $u_C := (C_T \times C, C', k)$ and $u' := (C', p, r)$ for the proofs π_C and π' it runs the corresponding verification algorithms of these two SNARKs, $\mathsf{MUniEv\text{-}\Pi.Ver}$ and $\mathsf{AC\text{-}\Pi.Ver}$ to check the proofs and outputs the conjunction of the two answers.

3.2 Security Analysis

Theorem 3. *Assuming that* $\mathsf{AC\text{-}\Pi}$ *and* $\mathsf{MUniEv\text{-}\Pi}$ *are secure commit-and-prove arguments of knowledge, the new construction* $\mathsf{Rq\text{-}\Pi}$ *described above satisfies* completeness, succinctness, zero-knowledge *and* knowledge-soundness.

For lack of space, the proof appears in the full version. Here we provide a short intuition. Correctness is rather straightforward, and zero-knowledge follows from the zero-knowledge property of the two SNARKs and the perfect hiding of the commitment scheme. For knowledge soundness, the proof consists of two main steps. First, we rely on the knowledge-soundness of the two SNARKs to show that for any adversary creating an accepting proof there is a knowledge extractor that, with all but negligible probability, returns witnesses that correctly satisfy the two relations $\mathcal{R}_{\mathsf{eval}}, \hat{\mathcal{R}}_f$ mentioned previously. Second, the only remaining possibility is that the polynomial $V = P^* - f(P_j) + TR$ is nonzero. However, $V(k) = 0$ and this holds for a random point k sampled by the random oracle independently of V, which can happen only with probability $\deg(V)/q$ which is negligible.

4 Applications to Computing on Encrypted Data

In this section we detail on how we can use our scheme $\mathsf{Rq\text{-}\Pi}$ for computations over polynomial rings to build a VC scheme with input and output privacy.

4.1 Verifiable Computation

Here we recall the notion of *verifiable computation* from [GGP10]. We adapt the definitions to fit the setting (that is in the scope of our construction) where we have public verifiability and public delegatability [PRV12], as well as privacy of the inputs and outputs. A VC scheme $\mathcal{VC} = (\mathsf{KeyGen}, \mathsf{ProbGen}, \mathsf{Compute}, \mathsf{Verify}, \mathsf{Decode})$ consists of the following algorithms:

- $\mathsf{KeyGen}(1^\lambda, f) \to (PK_f, SK_f)$: Given the security parameter, the key generation algorithm outputs a public key and a matching secret key for the function f.
- $\mathsf{ProbGen}_{PK_f}(\boldsymbol{x}) \to (\sigma_x, \tau_x)$: The problem generation algorithm uses the public key PK_f to encode the input x into a public value σ_x, to be given to the computing party, and a public value τ_x to be given to the verifier.
- $\mathsf{Compute}_{PK_f}(\sigma_x) \to \sigma_y$: Given the public key PK_f and the encoded input, the compute algorithm returns an encoded version of the function's output.
- $\mathsf{Verify}_{PK_f}(\tau_x, \sigma_y) \to acc$: Given the public key PK_f for function f, and the public verifier information τ_x, the verification algorithm accepts (output $acc = 1$) or rejects (output $acc = 0$) an output encoding σ_y.
- $\mathsf{Decode}_{SK_f}(\sigma_y) \to y$: Given the secret key SK_f for function f, and an output encoding σ_y, the decoding algorithm outputs a value y.

The correctness of a VC scheme is the obvious property: if one runs $\mathsf{Compute}$ on an honestly generated input encoding of $\boldsymbol{x}$, then the output must verify and its decoding should be $y = f(\boldsymbol{x})$.

For security, intuitively we want to say that an adversary that receives the public parameters for a function f and an encoding of an input $\boldsymbol{x}$ cannot create an encoding that passes verification and decodes to $y' \neq f(\boldsymbol{x})$. More formally, we say that a publicly verifiable computation scheme $\mathcal{VC}$ is *secure* for a function f, if for any PPT adversary $\mathcal{A}$, we have that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{PubVerif}[\mathcal{VC}, f, \lambda] = 1] = \mathsf{negl}(\lambda)$, where the experiment $\mathbf{Exp}^{PubVerif}$ is described below.

The *input privacy* notion intuitively says that no information about the inputs is leaked. This is defined using a typical indistinguishability experiment. Note that input privacy implies also *output privacy*. More formally, we say that a publicly verifiable (and publicly delegatable) VC scheme $\mathcal{VC}$ is *private* for a function f, if for any PPT adversary $\mathcal{A}$, we have that $\Pr[\mathbf{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC}, f, \lambda] = 1] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda)$, where the experiment $\mathbf{Exp}^{Priv}$ is described below.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{PubVerif}[\mathcal{VC}, f, \lambda]$
$(PK, SK) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$;
$\boldsymbol{x} \leftarrow \mathcal{A}(PK_f)$;
$(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK_f}(\boldsymbol{x})$;
$\hat{\sigma}_y \leftarrow \mathcal{A}(PK_f, \sigma_x, \tau_x)$;
$\hat{acc} \leftarrow \mathsf{Verify}_{PK_f}(\tau_x, \hat{\sigma}_y)$
$\hat{y} \leftarrow \mathsf{Decode}_{SK_f}(\hat{\sigma}_y)$
If $\hat{acc} = 1$ and $\hat{y} \neq f(\boldsymbol{x})$,
output '1', else '0';

Experiment $\mathbf{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC}, f, \lambda]$
$b \leftarrow \{0, 1\}$;
$(PK_f, SK_f) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$;
$(\boldsymbol{x}_0, \boldsymbol{x}_1) \leftarrow \mathcal{A}(PK_f)$
$(\sigma_b, \tau_b) \leftarrow \mathsf{ProbGen}_{PK_f}(\boldsymbol{x}_b)$;
$\hat{b} \leftarrow \mathcal{A}(PK_f, \sigma_b)$
If $\hat{b} = b$, output '1', else '0'

4.2 Our VC Scheme

We describe our VC scheme below. The construction is essentially an instantiation of the generic solution of Fiore et al. [FGP14] when using an homomorphic encryption scheme whose homomorphic evaluation algorithm fits our relation $\mathcal{R}_f$. This can be obtained by using HE schemes in the Ring-LWE setting where the ciphertext space works over the same ring $\mathbb{R}_q$ supported by our $\mathsf{Rq\text{-}\Pi}$ construction, and where the evaluation algorithm does not involve modulus switches and rounding operations. An example of such a scheme is the one of Brakerski and Vaikunthanatan [BV11].

Let $\mathsf{MPoly}\text{-}\mathcal{C}om = (\mathsf{MPoly.ComGen}, \mathsf{MPoly.ComVer}, \mathsf{MPoly.OpenVer})$ be a polynomial commitment scheme, $\mathsf{Rq\text{-}\Pi} = (\mathsf{Rq\text{-}\Pi.Gen}, \mathsf{Rq\text{-}\Pi.Prove}, \mathsf{Rq\text{-}\Pi.Ver})$ be a CaP zk-SNARK for polynomial rings computation, and let $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ be a homomorphic encryption scheme in the Ring-LWE setting. Then our VC scheme works as follows:

$\mathsf{KeyGen}(1^\lambda, \hat{f}) \to (PK_f, SK_f)$:
- Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(\lambda)$ to generate a key pair for HE.
- Run $\mathsf{crs} \leftarrow \mathsf{Rq\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_f^{\mathsf{ck}})$ to generate the common reference string of $\mathsf{Rq\text{-}\Pi}$ for the relation $\mathcal{R}_f^{\mathsf{ck}}$.
- Set $PK_f = (\mathsf{pk}, \mathsf{crs}, \hat{f})$ and $SK_f = (\mathsf{sk}, \mathsf{crs})$.

$\mathsf{ProbGen}_{PK_f}(\boldsymbol{x}) \to (\sigma_x, \tau_x)$:
- Parse $\boldsymbol{x} = \{x_i\}_{i=1}^n$ and compute ciphertexts $P_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, x_i)$
- Compute the commitment $(C, \rho) = \mathsf{MPoly.Com}(\{P_i\})$ and define $\sigma_x = (C, \{P_i\}, \rho)$ and $\tau_x = C$.

$\mathsf{Compute}_{PK_f}(\sigma_x) \to \sigma_y$:
- Parse $\sigma_x = (C, \{P_i\}, \rho)$;
- Compute the result ciphertext $P \leftarrow \mathsf{HE.Eval}(\mathsf{pk}, \hat{f}, \{P_i\}) = f(\{P_i\})$.
- Run $\pi \leftarrow \mathsf{Rq\text{-}\Pi.Prove}(\mathsf{crs}, (C, P), (\{P_i\}, \rho))$.
- Define $\sigma_y = (P, \pi)$

$\mathsf{Verify}_{PK_f}(\tau_x, \sigma_y) \to acc$: output $b \leftarrow \mathsf{Rq\text{-}\Pi.Ver}(\mathsf{crs}, (C, P), \pi)$.

$\mathsf{Decode}_{SK_f}(\tau_x, \sigma_y) \to y$: Decrypt $y = \mathsf{HE.Dec}(\mathsf{sk}, P)$.

Following the general result in [FGP14], the scheme satisfies correctness, security and privacy. In particular, privacy relies on the semantic security of HE, and security on the soundness of the SNARK.

4.3 Preserving Privacy of the Inputs Against the Verifier

The VC scheme described in the previous section works when the homomorphic computation $P \leftarrow f(\{P_i\})$ on the ciphertexts is deterministic. This can raise the issue that the result ciphertext P may reveal information on the plaintexts $\{x_i\}$ underlying $\{P_i\}$ (e.g., in lattice-based schemes such information may be inferred by looking at the distribution of the noise recovered as P's decryption time).

It would be therefore interesting to capture the setting where one wants to hide information on the x_i's even from the decryptor. Such a property would

turn useful in scenarios where the data encryptor and decryptor are different entities. As an example, consider the case of users that store medical data x on a cloud server which computes some query f on behalf of an analyst, who however is not entitled to learn more than $f(x)$.

In this section, we provide a formal definition of this property, that we call context-hiding, and then describe how our scheme from the previous section can be extended to achieve this additional property.

Defining Context-Hiding. Informally, this property says that output encodings σ_y, as well as the input verification tokens τ_x, do not reveal any information on the input $\boldsymbol{x}$. Notably this should hold even against the holders of the secret key SK_f. We formalize this definition in a zero-knowledge style, requiring the existence of simulator algorithms that, without knowing the input, should generate (τ_x, σ_y) that look like the real ones. More precisely, a VC scheme is context-hiding for a function f if there exist simulator algorithms S_1, S_2 such that:

- the keys (PK_f, SK_f) and (PK'_f, SK'_f) are statistically indistinguishable, where $(PK_f, SK_f) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$ and $(PK_f, SK_f, \mathsf{td}) \leftarrow S_1(1^\lambda, f)$;
- for any input $\boldsymbol{x}$, the following distributions are negligibly close

$$(PK_f, SK_f, \sigma_x, \tau_x, \sigma_y) \approx (PK_f, SK_f, \sigma_x, \tau'_x, \sigma'_y)$$

  where $(PK_f, SK_f, \mathsf{td}) \leftarrow S_1(1^\lambda, f)$, $(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK_f}(\boldsymbol{x})$, $\sigma_y \leftarrow \mathsf{Compute}_{PK_f}(\sigma_x)$, and $(\sigma'_y, \tau'_x) \leftarrow S_2(\mathsf{td}, SK_f, f(\boldsymbol{x}))$.

Our Context-Hiding Secure VC scheme. Before describing the scheme in detail, let us provide some intuition.

The first observation is that for the HE scheme this problem can be solved by adding to the result P an encryption of 0, P_0^*, whose noise can statically hide that in P (a so called noise flooding technique). However if we do this change in our VC scheme we have two issues: (1) the computation is not deterministic anymore; (2) the prover may create a bogus encryption of 0, not of the correct distribution, in order to make decryption fail. We can solve these issues by using the fact that, as underlying tool for verifiability, we are using a SNARK that can handle deterministic computations. In particular, we can do the following.

For (2) we add to the public key s honestly generated encryptions of 0 $\{P_i^*\}_{i=1}^s$, and then ask the untrusted party to compute the result as $P' = P + P_0^*$ with $P_0^* = \sum_{i=1}^n b_i \cdot P_i^*$, for uniformly random bits b_i. By choosing appropriately the noise parameters in the P_i^*'s and by taking $s \approx \lambda$, based on the leftover hash lemma, P_0^* can statistically hide the noise in P.

Formally, adding such a randomization at the end of computing a function f guarantees leveled circuit privacy. In a nutshell, a somewhat-FHE HE is leveled circuit private if there exists a simulator algorithm $\mathsf{HE.S}$ such that $\mathsf{HE.S}(\mathsf{pk}, d, f(\boldsymbol{x})) \approx \mathsf{HE.Eval}(\mathsf{pk}, f, \mathsf{HE.Enc}(\boldsymbol{x}))$ are statistically close. Here the input d taken by the simulator represents information on the depth of f.

For (1), we simply consider proving a slightly different relation, that is:

$$\mathcal{R}_f^* := \{(u = (C, P', \{P_i^*\}_{i=1}^s); w = (\{P_j\}_{j=1}^n, T, \rho, b_1, \ldots, b_s)) : (C, \rho) = \mathsf{MPoly.Com}(\{P_j\}) \ \wedge \ \forall i \in [s]\ b_i \in \{0,1\} \ \wedge \ P' = f(P_j) + \sum_{i=1}^{s} b_i P_i^* - TR \ \}$$

To use our scheme Rq-Π on the above relation, we can do the following. Given a function $f : {\mathbb{R}_q}^n \to \mathbb{R}_q$, define the function $f' : {\mathbb{R}_q}^{n+s} \times \mathbb{Z}_q^s \to \mathbb{R}_q$ that takes $n + 2s$ inputs such that

$$\hat{f}'(x_1, \ldots, x_n, o_1, \ldots, o_s, b_1, \ldots, b_s) = \hat{f}(x_1, \ldots, x_n) + \sum_{i=1}^{s} b_i \cdot o_i.$$

Then we use our Rq-Π on the following relation

$$\mathcal{R}'_f := \{(u = (C', P'); w = (\{P_j\}_{j=1}^n, \{P_i^*\}_{i=1}^s, \{b_i\}_{i=1}^s, T, \rho')) : (C', \rho') = \mathsf{MPoly.Com}(\{P_j\}, \{P_i^*\}, \{b_i\}) \ \wedge \ \forall i \in [s] b_i \in \{0,1\} \ \wedge \ P' = f'(P_j, \{P_i^*\}, \{b_i\}) - TR \ \}$$

where $C' = C \times C^* \times C_b$ and $\rho' = \rho + \rho^* + \rho_b$. It can be seen that $\mathcal{R}'_f$ matches the format $\mathcal{R}_{f'}$ (for the function f' and a larger set of inputs) of relations supported by our Rq-Π scheme. One change however is that the commitment C' cannot be created directly by ProbGen as it contains elements that depend on a specific computation. We can solve this problem by using the homomorphic property of the commitment scheme: namely we assume that at key generation a commitment $(C^*, \rho^*) = \mathsf{MPoly.Com}(\{P_i^*\})$ is created and made public, and that the prover creates a similar commitment $(C_b, \rho_b) = \mathsf{MPoly.Com}(\{b_i\})$ to the random coefficients. Then C' can be obtained as $C \cdot C^* \cdot C_b$ and its opening is $\rho' = \rho + \rho^* + \rho_b$.

A more precise description of the protocol is given below.

$\mathsf{KeyGen}(1^\lambda, \hat{f}) \to (PK_f, SK_f)$:
- Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(\lambda)$ to generate the key pair for HE.
- Run $\mathsf{crs} \leftarrow \mathsf{Rq\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_{f'})$ to generate the Rq-Π crs for the relation $\mathcal{R}_{f'}$.
- For $i = 1$ to s: $P_i^* \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, 0)$ and compute a commitment $(C^*, \rho^*) = \mathsf{MPoly.Com}(\{P_i^*\})$.
- Set $PK_f = (\mathsf{pk}, \{P_i^*\}_{i=1}^s, C^*, \rho^*, \mathsf{crs}, \hat{f})$ and $SK_P = (\mathsf{sk}, \mathsf{crs})$.

$\mathsf{ProbGen}_{PK_f}(\boldsymbol{x}) \to (\sigma_x, \tau_x)$: this is the same as in the previous section.

$\mathsf{Compute}_{PK_f}(\sigma_x) \to \sigma_y$: parsing $\sigma_x = (C, \{P_i\}, \rho)$, do the following:
- Sample $b_1, \ldots, b_s \leftarrow_\$ \{0,1\}$ uniformly at random, and compute a commitment $(C_b, \rho_b) = \mathsf{MPoly.Com}(\{b_i\})$ (thinking of each b_i as a degree-0 polynomial).
- Compute the result ciphertext $P' \leftarrow f(\{P_i\}) + \sum_{i=1}^s b_i P_i^*$.
- Run $\pi \leftarrow \mathsf{Rq\text{-}\Pi.Prove}(\mathsf{crs}, (C \times C^* \times C_b, P'), (\{P_i\}, \{P_i^*\}, \{b_i\}, \rho, \rho^*, \rho_b))$.
- Define $\sigma_y = (P', C_b, \pi)$

$\mathsf{Verify}_{PK_f}(\tau_x, \sigma_y) \to acc$: output $b \leftarrow \mathsf{Rq\text{-}\Pi.Ver}(\mathsf{crs}, (C \times C^* \times C_b, P), \pi)$.
$\mathsf{Decode}_{SK_f}(\tau_x, \sigma_y) \to y$: Decrypt $y = \mathsf{HE.Dec}(\mathsf{sk}, P')$.

Theorem 4. *If* HE *is semantically secure and circuit private, and* Rq-Π *is knowledge sound and zero-knowledge, then the VC described above is correct, secure, private and context-hiding.*

Proof (Sketch). The proof of the result is rather simple. Below we provide a proof sketch. First, notice that based on the correctness of Rq-Π and that of HE, we obtain correctness of our protocol.

The security follows from the knowledge soundness of the SNARK. The only detail to mention is that we also rely on the correctness of the HE scheme in order to make sure that, for honestly generated ciphertexts $\{P_i\}$ of $\{x_i\}$, and $\{P_i^*\}$ for 0, and for binary coefficients $\{b_i\}$, the ciphertext $P' \leftarrow f(\{P_i\}) + \sum_{i=1}^{s} b_i P_i^*$ decrypts to $\hat{f}(x)$.

Finally, we can prove context-hiding via a simple hybrid argument based on the privacy property of the HE scheme and the zero-knowledge of our SNARK. We define the $\mathcal{VC}$ simulators as follows. S_1 proceeds exactly as KeyGen except that it runs the SNARK simulator $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}^{\mathsf{crs}}(\mathcal{R}_{f'}, \lambda)$ instead of Gen, and set its trapdoor to be td. $S_2(\mathsf{td}, SK_f, y)$ first sets $\tau'_x = C$ where C is created as a commitment to some dummy input. Next, it creates C_b as another commitment to a dummy value, and computes P' as an encryption of y using $\mathsf{HE.S}(\mathsf{pk}, d, y)$ (where d is information on the depth of f), and finally it invokes the SNARK simulator $\pi \leftarrow \mathsf{Sim}^{\mathsf{Prove}}(\mathsf{crs}, (C \times C^* \times C_b, P'))$. Then S_2 outputs τ'_x and $\sigma'_y = (P', C_b, \pi)$.

The indistinguishability of the keys is immediate from the zero-knowledge of the SNARK. For the second property, we can define an hybrid simulator S' that, with knowledge of σ_x, runs as S_2 but creates P' as in Compute. It is easy to see that the output of S' is indistinguishable from that of S_2 by the property of HE.Hide, also by the hiding of the commitment and by the zero-knowledge of the SNARK we obtain that the values (τ'_x, σ'_y) generated by S' are indistinguishable from the ones generated using ProbGen and Compute.

5 Bivariate Polynomial Commitment

Our final goal is to build an efficient instantiation of the MUniEv-Π scheme for the evaluation on the same point of *many* univariate polynomials committed with MPoly-$\mathcal{C}om$. This is the key tool for our Rq-Π scheme for computations over polynomial rings presented in Section 3.

We construct MPoly-$\mathcal{C}om$ and MUniEv-Π starting from a commitment scheme BivPoly.$\mathcal{C}om$ for bivariate polynomials and a commit-and-prove argument BivPE-Π for the partial evaluation, in one variable, of a committed bivariate polynomial.

In this section we recall bilinear pairings and the computational assumptions needed by our schemes, and then we present the BivPoly.$\mathcal{C}om$ commitment scheme. The construction of the BivPE-Π commit-and-prove SNARK is described in Section 6, while their conversion into MPoly-$\mathcal{C}om$ and MUniEv-Π appears in Section 7.

5.1 Computational Assumptions

Security of our constructions rely on various computational assumptions. We state here our assumptions over bilinear groups. Some of them are standard q-type assumptions in the frame of DLog-hard groups and others are extractable (non-falsifiable) assumptions, a class of assumptions inherent to the security of SNARKs as shown in [GW11].

Bilinear Groups. Let the generator $\mathcal{G}$ input a security parameter λ and output a description of a bilinear group $\mathsf{gk} := (q, \mathbb{G}, \mathfrak{G}, \mathbb{G}_T, \mathsf{e}) \leftarrow_\$ \mathcal{G}(1^\lambda)$ such that q is a λ-bit prime; $\mathbb{G}, \mathfrak{G}, \mathbb{G}_T$ are cyclic groups of order q; $\mathsf{e} : \mathbb{G} \times \mathfrak{G} \to \mathbb{G}_T$ is a bilinear asymetric map (pairing), which means that $\forall a, b \in \mathbb{Z}_q : \mathsf{e}(g^a, \mathfrak{g}^b) = \mathsf{e}(g, \mathfrak{g})^{ab}$; if g and $\mathfrak{g}$ generate $\mathbb{G}$ and $\mathfrak{G}$ respectively, then $\mathsf{e}(g, \mathfrak{g})$ generates $\mathbb{G}_T$; membership in $\mathbb{G}, \mathfrak{G}, \mathbb{G}_T$ can be efficiently decided, group operations and the pairing e are efficiently computable, generators are efficiently sampleable, and the descriptions of the groups and group elements each have size $O(\lambda)$ bits.

The d–Strong Diffie-Hellman Assumption ($d - \mathsf{SDH}$). The Strong Diffie-Hellman assumption [BB08] says that given $(g, g^s, \ldots, g^{s^d})$ it is infeasible to compute $y = g^{\frac{1}{s-r}}$ for a chosen $r \in \mathbb{Z}_q$. In our applications, a few more group elements Σ are given as input to the adversary:

Assumption 1 ($d - \mathsf{SDH}$) *The d–Strong Diffie-Hellman assumption holds relative to a bilinear group gk if for all PPT adversaries $\mathcal{A}$ we have, on the probability space $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $\Sigma \leftarrow ((g, g^s, \ldots g^{s^d}); (\mathfrak{g}, \mathfrak{g}^s))$, $g \leftarrow_\$ \mathbb{G}$, $\mathfrak{g} \leftarrow_\$ \mathfrak{G}$, and $s \leftarrow_\$ \mathbb{Z}_q$:*

$$\mathsf{Adv}^{\mathrm{d-sdh}}_{\mathcal{A}}(\lambda) := \Pr\left[(r, y) \leftarrow \mathcal{A}(\mathsf{gk}, \Sigma) \ \wedge \ y = g^{\frac{1}{s-r}} \right] = \mathsf{negl}(\lambda).$$

An adaptation of the proof in Boneh and Boyen [BB08] shows that our variant of the $d - \mathsf{SDH}$ assumption holds in the generic bilinear group model.

Knowledge of Exponent Assumptions. The knowledge of exponent (KEA) assumption introduced by Damgard [Dam92] says that given g, g^α in a group $\mathbb{G}$ it is infeasible to create $c, \hat{c}$ so $\hat{c} = c^\alpha$ without knowing a so $c = g^a$ and $\hat{c} = (g^\alpha)^a$. d-Power Knowledge of Exponent Assumption ($d - \mathsf{PKE}$) is another long-standing extractable assumption. It says that given $\{g, g^s, g^{s^2}, \ldots, g^{s^d}, \hat{g}, \hat{g}^s, \hat{g}^{s^2}, \ldots, \hat{g}^{s^d}\}$ with $\hat{g} = g^\alpha$, it is infeasible to create $c, \hat{c}$ where $\hat{c} = c^\alpha$ without knowing $a_0, a_1, \ldots a_d$ that satisfy $c = \prod_{i=0}^{d} (g^{s^i})^{a_i}$.

The (d, ℓ)–Bivariate PKE Assumption ($(d, \ell) - \mathsf{BPKE}$). We introduce a bivariate power knowledge of exponent assumption that is a simple extension of the popular $d - \mathsf{PKE}$ assumption.

The (d, ℓ)–Bivariate Power Knowledge of Exponent Assumption for a bilinear group gk, noted by $(d, \ell) - \mathsf{BPKE}$ is a hybrid between PKE assumption for d different powers of s and ℓ powers of t and KEA assumption for input $(h, \hat{h} :=$

$\underline{\mathsf{Biv.ComGen}(1^\lambda, d, \ell) \to \mathsf{ck}}$

1: $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda),\ g, h \leftarrow_\$ \mathbb{G}, \mathfrak{g} \leftarrow_\$ \mathfrak{G},\ \alpha, s, t \leftarrow_\$ \mathbb{Z}_q$
2: $\hat{g} := g^\alpha, \hat{h} := h^\alpha, \hat{\mathfrak{g}} := \mathfrak{g}^\alpha$
3: $g_{ij} := g^{s^i t^j},\ \ \hat{g}_{ij} := \hat{g}^{s^i t^j}\ \ \forall\ i < d, j < \ell$
4: $\mathfrak{g}_1 := \mathfrak{g}^s, h_1 := h^s$
5: return $\mathsf{ck} = \{\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\hat{g}_{ij})_{i,j=0}^{d,\ell}; (h, \hat{h}); (\mathfrak{g}, \hat{\mathfrak{g}}); (\mathfrak{g}_1, h_1)\}$

$\underline{\mathsf{Biv.Com}(\mathsf{ck}, P) \to (C, \rho)}$

1: $P := \sum_{i,j=0}^{d,\ell} a_{ij} X^i Y^j$
2: $\rho \leftarrow_\$ \mathbb{Z}_q$
3: $c = h^\rho \prod_{i=0,j=0}^{d,\ell} g_{ij}^{a_{ij}}$
4: $\hat{c} = \hat{h}^\rho \prod_{i=0,j=0}^{d,\ell} \hat{g}_{ij}^{a_{ij}}$
5: $C \leftarrow (c, \hat{c})$
6: return (C, ρ)

$\underline{\mathsf{Biv.ComVer}(\mathsf{ck}, C) \to b}$

1: $C := (c, \hat{c})$
2: return $b := (\mathsf{e}(c, \hat{\mathfrak{g}}) = \mathsf{e}(\hat{c}, \mathfrak{g}))$

$\underline{\mathsf{Biv.OpenVer}(\mathsf{ck}, C, P, \rho) \to P}$

1: $C := (c, \hat{c}),\ P = \sum_{i,j=0}^{d,\ell} a_{ij} X^i Y^j$
2: $b_1 \leftarrow \mathsf{ComVer}(\mathsf{ck}, C)$
3: $b_2 \leftarrow (c = h^\rho \prod_{i,j=0}^{d,\ell} g_{ij}^{a_{ij}})$
4: return $(b_1 \ \wedge\ b_2)$

Fig. 2. Our BivPoly.$\mathcal{C}om$ for Bivariate Polynomial

$h^\alpha) \in \mathbb{G}^2$. It takes the two basis $(g, \hat{g} := g^\alpha), (h, \hat{h} := h^\alpha)$ and all the powers $\{g^{s^i t^j}, \hat{g}^{s^i t^j}\}_{i,j=0}^{d,\ell}$ and claims that it is infeasible to create $c, \hat{c}$ such that $\hat{c} = c^\alpha$ without knowing $\delta, \{a_{ij}\}_{i,j=0}^{d,\ell}$, that satisfy $c = h^\delta \prod_{i,j=0}^{d,\ell} (g^{s^i t^j})^{a_{ij}}$. More formally:

Assumption 2 ($(d, \ell) - \mathsf{BPKE}$) *The* $(d, \ell) - \mathsf{BPKE}$ *assumption holds relative to a bilinear group* gk *for the class* $\mathcal{Z}$ *of auxiliary input generators if, for every* $\mathsf{aux} \in \mathcal{Z}$ *and PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* Ext *such that, on the probability space* $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda),\ \Sigma \leftarrow (g, \{g^{s^i t^j}\}_{i,j=0}^{d,\ell}, \{\hat{g}^{s^i t^j}\}_{i,j=0}^{d,\ell}; (h, \hat{h}, h^s); (\mathfrak{g}, \hat{\mathfrak{g}}, \mathfrak{g}^s)),$ $\mathsf{aux} \leftarrow \mathcal{Z}(\mathsf{gk}, \Sigma),\ g, h \leftarrow_\$ \mathbb{G},\ \mathfrak{g} \leftarrow_\$ \mathfrak{G},\ \alpha, s, t \leftarrow_\$ \mathbb{Z}_q,\ \hat{g} := g^\alpha,\ \hat{h} := h^\alpha,$ *and* $\hat{\mathfrak{g}} := \mathfrak{g}^\alpha$*:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{d}-\mathsf{pke}}(\lambda) := \Pr\left[\begin{array}{c}(c, \hat{c}; \delta, \{a_{ij}\}_{i,j=0}^{d,\ell}) \leftarrow (\mathcal{A} \| \mathsf{Ext})(\mathsf{gk}, \Sigma; \mathsf{aux}) \\ \mathsf{e}(\hat{c}, \mathfrak{g}) = \mathsf{e}(c, \mathfrak{g}^\alpha) \ \ \wedge \ \ c \neq h^\delta \prod_{i,j=0}^{d,\ell} (g^{s^i t^j})^{a_{ij}}\end{array}\right] = \mathsf{negl}(\lambda).$$

5.2 Knowledge Commitment for Bivariate Polynomials

Based on an efficient construction of a polynomial commitment scheme proposed by [KZG10] we further construct a knowledge commitment scheme for bivariate polynomials that is perfectly hiding and computationally binding. This will later allow us to use commitments in a CaP-SNARK BivPE-Π for polynomial partial evaluation.

The commitment scheme BivPoly.$\mathcal{C}om$ = (Biv.ComGen, Biv.Com, Biv.ComVer, Biv.OpenVer) consists of four algorithms as described in Figure 2 and it is specialized for (bivariate) polynomials $P \in \mathbb{Z}_q[X, Y]$: the message space M_{ck} is defined by polynomials in $\mathbb{Z}_q[X, Y]$ of degree in X bounded by a value d and degree in Y bounded by some value ℓ.

Remark 5. The $\mathsf{Biv.ComGen}$ algorithm computes two extra values $\mathfrak{g}_1 := \mathfrak{g}^s, h_1 := h^s$ to be added to ck (step 4). Although these elements are not used by the commitment scheme, they are useful to construct our Commit and Prove SNARK for partial evaluations of polynomials committed with $\mathsf{BivPoly}.\mathcal{C}om$. In other words, used as a stand alone commitment scheme, $\mathsf{BivPoly}.\mathcal{C}om$ may have a slightly shorter commitment key ck (by removing step 4 from $\mathsf{Biv.ComGen}$).

Security of the Commitment $\mathsf{BivPoly}.\mathcal{C}om$**.** We call $\mathsf{BivPoly}.\mathcal{C}om$ a knowledge commitment, since the prover cannot make a valid commitment without "knowing" the committed values. We will rely on the $(d, \ell) - \mathsf{BPKE}$ assumption for extracting the committed polynomials. We can state the following theorem on the security of $\mathsf{BivPoly}.\mathcal{C}om$, whose proof can be found in the full version.

Theorem 6. *The commitment scheme* $\mathsf{BivPoly}.\mathcal{C}om$ *is perfectly hiding and computationally binding assuming the* $d - \mathsf{SDH}$ *assumption holds in* $\mathbb{G}$*. Moreover, assuming* $(d, \ell) - \mathsf{BPKE}$*, the scheme is knowledge binding.*

6 CaP-SNARK for Bivariate Polynomial Evaluation

In this section we show how to construct a commit-and-prove SNARK BivPE-Π for the partial evaluation in a single variable of bivariate polynomials.

6.1 Relations for Bivariate Polynomial Partial Evaluation

The relation $\mathcal{R}$ for partial evaluation of bivariate polynomials is defined over tuples $(k, P(X,Y), Q(Y)) \in \mathbb{Z}_q \times \mathbb{Z}_q[X,Y] \times \mathbb{Z}_q[Y]$ as follows

$$\mathcal{R} := \{(k, P(X,Y), Q(Y)) : Q(Y) = P(k,Y)\}.$$

The scheme we propose in this section is a Commit-and-Prove (CaP) SNARK for the above $\mathcal{R}$ where $P \in \mathbb{Z}_q[X,Y]$ and $Q \in \mathbb{Z}_q[Y]$ are committed in C and C' respectively using $\mathsf{BivPoly}.\mathcal{C}om$. [8]

Namely, following the definition from Section 2.2, BivPE-Π is a zk-SNARK for the following commit-and-prove relation

$$\begin{aligned} \mathcal{R}_{\mathsf{ck}} := \{(u = (C, C', k); w = (P, Q, \rho, \rho')) : \\ (C, \rho) = \mathsf{Biv.Com}(P) \ \wedge \ (C', \rho') = \mathsf{Biv.Com}(Q) \ \wedge \ Q(Y) = P(k, Y)\}. \end{aligned} \tag{1}$$

6.2 Our BivPE-Π Scheme for Bivariate Polynomial Evaluation

We aim to build an efficient commit-and-prove SNARK, BivPE-Π, dedicated to partial evaluation for bivariate polynomials $P \in \mathbb{Z}_q[X,Y]$ in $X = k \in \mathbb{Z}_q$.

Our scheme is based on an algebraic property of polynomials. We remark that $(X - k)$ perfectly divides the polynomial $P(X,Y) - P(k,Y)$ for $k \in \mathbb{Z}_q$.

$\underline{\mathsf{BivPE\text{-}\Pi.Gen}(\mathsf{ck}, \mathcal{R}_{\mathsf{ck}}) \to \mathsf{crs} := \mathsf{ck}}$

$\underline{\mathsf{BivPE\text{-}\Pi.Prove}(\mathsf{crs}, u, w)}$

1: $(C, C', k) := u, (P, Q, \rho, \rho') := w$
2: $W := \big(P - Q\big)/(X - k)$
3: $(D, \omega) \leftarrow \mathsf{Biv.Com}(W)$
4: $\tilde{g} := h_1/h^k, \quad x, y \leftarrow_{\$} \mathbb{Z}_q$
5: $\mathbb{U} := \mathsf{e}(h^x \tilde{g}^y, \mathfrak{g})$
6: $e \leftarrow \mathsf{Hash}(u, D, \mathbb{U})$
7: $\sigma = x - (\rho' - \rho)e \mod q$
8: $\tau = y - \omega e \mod q$
9: return $\pi := (D, e, \sigma, \tau)$

$\underline{\mathsf{BivPE\text{-}\Pi.Ver}(\mathsf{crs}, u, \pi) \to b}$

1: $(C, C', k) := u, (D, e, \sigma, \tau) := \pi$
2: $(c, \hat{c}) := C, (c', \hat{c}') := C', (d, \hat{d}) := D$
3: $b_1 \leftarrow \mathsf{Biv.ComVer}(C)$
4: $b_2 \leftarrow \mathsf{Biv.ComVer}(C')$
5: $b_3 \leftarrow \mathsf{Biv.ComVer}(D)$
6: $\mathbb{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$
7: $\mathbb{U} := \mathsf{e}(h^\sigma \tilde{g}^\tau, \mathfrak{g})\mathbb{A}^e$, s.t. $\tilde{g} := h_1/h^k$
8: $b_4 \leftarrow (e = \mathsf{Hash}(u, D, \mathbb{U}))$
9: return $(b_1 \ \wedge \ b_2 \ \wedge \ b_3 \ \wedge \ b_4)$

Fig. 3. Our CaP-SNARK for Bivariate Polynomial Partial Evaluation

BivPE-Π works for an ($\mathcal{R}$-independent) bivariate polynomial commitment scheme $\mathsf{BivPoly}.\mathcal{C}om = (\mathsf{Biv.ComGen}, \mathsf{Biv.Com}, \mathsf{Biv.ComVer}, \mathsf{Biv.OpenVer})$, as detailed in Figure 3, and has to satisfy *completeness*, *succinctness*, *zero-knowledge* and *knowledge-soundness*.

Description of Our BivPE-Π Protocol. Let $\mathsf{BivPoly}.\mathcal{C}om$ be a bi-variate polynomial knowledge commitment scheme. We construct a zero-knowledge SNARK scheme for any relation $\mathcal{R}_{\mathsf{ck}}$ with respect to some bounds d, ℓ on the degrees in X and in Y of the polynomials $P \in \mathbb{Z}_q[X, Y]$ supported by $\mathsf{BivPoly}.\mathcal{C}om$. Our protocol is formally depicted in Figure 3.

CRS generation. The setup algorithm outputs a crs enabling the proof and verification of statements for the associated relation $\mathcal{R}_{\mathsf{ck}}$ defined in Eq. (1).

We remark that Gen algorithm is just using the same public information (commitment key) ck from the $\mathsf{BivPoly}.\mathcal{C}om$ scheme.

Prover. Given crs, the statement $u = (C, C', k)$ (two commitments C, C' and an evaluation point k) and the witness $w = (P, Q, \rho, \rho')$ (the corresponding polynomials $P \in \mathbb{Z}_q[X, Y], Q \in \mathbb{Z}_q[Y]$ and their randomness ρ, ρ'), the prover proceeds to compute a proof π that $P(k, Y) = Q(Y)$, $(C, \rho) = \mathsf{Biv.Com}(P)$, and $(C', \rho') = \mathsf{Biv.Com}(Q)$ in two steps:

Step 1. (From 1 to 3 in the Prove algorithm from Figure 3.) The prover computes a witness to the correct (partial) evaluation in $k \in \mathbb{Z}_q$ of the polynomial $P \in \mathbb{Z}_q[X, Y]$ as $P(k, Y) = Q \in \mathbb{Z}_q[Y]$. The witness of this evaluation is a polynomial $W \in \mathbb{Z}_q[X, Y]$ defined as the quotient $W := \frac{P(X,Y) - Q(Y)}{X - k}$. This is a well-defined polynomial in $\mathbb{Z}_q[X, Y]$ if and only if $P(k, Y) = Q \in \mathbb{Z}_q[Y]$. The element of the proof π that enables checking this algebraic property over the polynomials P and Q will be a commitment $(D = (d, \hat{d}), \omega)$ to the polynomial W, where $\omega \leftarrow_{\$} \mathbb{Z}_q$ is a fresh randomness.

[8] Note that, although Q is a uni-variate polynomial in Y, it can also be seen as a bivariate polynomial.

Remark 7. To this point, the verifier should be convinced that the polynomial Q is the good evaluation in k of P, only by checking the corresponding polynomial equation evaluated in a random hidden point $(s,t) : W(s,t)(s-k) = P(k,t)-Q(t)$. This can be translated in terms of commitments $(C,\rho)(C',\rho'),(D,\omega)$ to P,Q,W as a pairing check: $\mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1} = \mathsf{e}(h^{(s-t)\omega-(\rho-\rho')}, \mathfrak{g})$ where $C = (c,\hat{c}), C' = (c',\hat{c}'), D = (d,\hat{d})$.

Because of the hiding property, the verifier does not have access to the openings of the commitments, as it does not know the randomness ρ, ρ', ω.

We therefore need the prover to provide something more together with the commitment D. The prover needs to compute an extra proof of knowledge of the randomnesses ω used to create this comitment and of the correct relation to satisfy with respect to the randomness ρ, ρ' of the statement commitments C, C' such that the pairing expression cancels the respective terms $h^{(\rho-\rho')}$ and $h^{(s-t)\omega}$.

This is easily solved by building a Schnorr proof of knowledge of the exponents $\omega, (\rho'-\rho)$ that appear in $\mathbb{A} = \mathsf{e}(h^{(s-k)\omega-(\rho-\rho')}, \mathfrak{g}) = \mathsf{e}(h^{(\rho'-\rho)}h^{(s-k)\omega}, \mathfrak{g})$. If we define $\tilde{g} \coloneqq h_1/h^k = h^{s-k}$, then this proof is a classical Schnorr proof for the public value $\mathbb{A} = \mathsf{e}(h^{\rho'-\rho}\tilde{g}^{\omega}, \mathfrak{g}) = \mathsf{e}(h, \mathfrak{g})^{\rho'-\rho} \cdot \mathsf{e}(\tilde{g}, \mathfrak{g})^{\omega}$ in the target group $\mathfrak{G}$. But we will show we can make it more efficient.

Step 2. (From 4 to 7 in the Prove algorithm from Figure 3.) This step consists in this non-interactive Schnorr proof associated to the value $\mathbb{A} = \mathsf{e}(h^{\rho'-\rho}\tilde{g}^{\omega}, \mathfrak{g})$:

- Choose $x, y \in \mathbb{Z}_q$,
- Define $\mathbb{U} = \mathsf{e}(h^x\tilde{g}^y, \mathfrak{g})$, this corresponds to the first round in the interactive Schnorr proof protocol, where the prover sends its commitment.
- Sample the challenge to the Schnorr proof by running the random oracle (hash function) on input the statement to be proven and the commitment $\mathbb{U}$: $e \leftarrow \mathsf{Hash}(u, D, \mathbb{U})$,
- Compute the answers $\sigma = x - (\rho'-\rho)e \bmod q$ and $\tau = y - \omega e \bmod q$.

The values sent as Schnorr proof are three scalars e, σ, τ, where e is the output of the hash function $\mathsf{Hash}(u, D, \mathbb{U})$ and does not depend on the size of $\mathbb{U} \in \mathbb{G}_T$. After the two described steps, the prover algorithm outputs $\pi \coloneqq (D, e, \sigma, \tau)$.

Verifier. First, the verifier parses the received statement and proof (steps 1 and 2 in the Ver algorithm from Figure 3), then it makes sure the commitments C, C', D are well-formed (steps 3 to 5 in the Ver algorithm from Figure 3) by running the Biv.ComVer algorithm. If this is not the case, we discard the proof π. To verify the proof π, one needs the polynomial equation $W(X,Y)(X-k) = P(k,Y) - Q(Y)$ to hold for some secret evaluation points (s,t). We can rewrite this equation in terms of pairings applied to the commitments (C, C', D): $\mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$. If the polynomials W, P, Q evaluated in the secret points s, t satisfy the equation $W(s,t)(s-k) = P(k,t) - Q(t)$, then all the exponents in base g cancel out in the pairing expression. It is not the case for the exponents in base h which correspond to the randomness used in the commitments. The important remark is that if D is correct, the remaining value $\mathbb{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$ can be written only in terms of the 3 randomness ρ, ρ', ω used to commit to P, Q, W:

$$\mathbb{A} = \mathsf{e}(h^{(s-k)\omega}h^{(\rho'-\rho)}, \mathfrak{g}) = \mathsf{e}(h^{\rho'-\rho}\tilde{g}^{\omega}, \mathfrak{g}).$$

This can be checked by the usual verification procedure of the Schnorr proof transmitted in π, *i.e.* the values (e, σ, τ): Compute $\mathbb{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$ and $\mathbb{U} = \mathsf{e}(h^\sigma \tilde{g}^\tau, \mathfrak{g}) \cdot \mathbb{A}^e$ then run the Hash function to check whether $e = \mathsf{Hash}(u, D, \mathbb{U})$.

Security of BivPE-Π. The security of our scheme is captured in the following theorem whose proof is elaborated in Section 8:

Theorem 8. *Assuming both the* $d - \mathsf{SDH}$ *and* $(d, \ell) - \mathsf{BPKE}$ *assumptions hold in the bilinear group* gk*, the protocol* CaP-BivPE-Π *is a zero-knowledge Succinct Non-Interactive Argument of Knowledge in the random oracle model.*

Remark 9. We point out that in the case one is *not* interested in hiding the committed bivariate polynomial P and its partial evaluation Q, then it is possible to define a simplified version of our scheme that does not need the Schnorr-style proof and thus is secure *without* random oracles. This protocol is the same as CaP-BivPE-Π except that one would set $\omega = \rho = \rho' = 0$ (so the commitments are no longer hiding); this way the evaluation proof can be just the commitment D and it can be verified with the pairing check $\mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) = \mathsf{e}(c/c', \mathfrak{g})$.

7 CaP-SNARK for Simultaneous Evaluations

In this section we show how we can use our BivPE-Π scheme for the partial evaluation of *one* bivariate polynomial on a point k in order to prove the evaluation of *many* univariate polynomials on the same point k. The resulting scheme MUniEv-Π can be used in the protocol presented in Section 4 for verifiable computation using HE on Ring-LWE.

More precisely, we show how to use our BivPoly.$\mathcal{C}om$ and BivPE-Π to define a commitment scheme and a compact proof system dedicated to multi-polynomials evaluation in the same random point k: given a single compact knowledge commitment C for a set of univariate polynomials $\{P_j(X)\}_j \in \mathbb{Z}_q[X]$ and a public evaluation point $k \in \mathbb{Z}_q$, we want to prove that some values $\{p_j\}_j$ committed in C' are indeed evaluations of the committed polynomials in this point k.

7.1 Commitment for Multiple Univariate Polynomials

We describe below, MPoly-$\mathcal{C}om$, our new knowledge commitment for a set of univariate polynomials. It is obtained in a straightforward way from BivPoly.$\mathcal{C}om$. It is defined as follows, where for simplicity we consider $\ell+1$ committed univariate polynomials $P_j = \sum_{i=0}^{d} p_{ij} X^i$ for all $0 \leqslant j \leqslant \ell, 0 \leqslant i \leqslant d$:

$\mathsf{MPoly.ComGen}(1^\lambda, d, \ell) \to \mathsf{ck}$**:** Given some degree bound d and some maximal bound $\ell + 1$ on the cardinal of the polynomial set to be committed, it runs $\mathsf{ck} \leftarrow \mathsf{Biv.ComGen}(1^\lambda, d, \ell)$, where d, ℓ are the bounds on the degrees on X and Y of the bivariate polynomials in $\mathbb{Z}_q[X, Y]$.

$\mathsf{MPoly.Com}(\mathsf{ck}, \{P_j\}_{0 \leqslant j \leqslant \ell}) \to (C, \rho)$**:** Given a set $\{P_j\}$ of $\ell + 1$ polynomials in $\mathbb{Z}_q[X]$, with coefficients $\{p_{ij}\}_{i,j=0}^{i \leqslant d, j \leqslant \ell}$ we can define the bivariate polynomial $P = \sum_{i,j=0}^{d,\ell} p_{ij} X^i Y^j$ and run $(C, \rho) \leftarrow \mathsf{Biv.Com}(\mathsf{ck}, P)$;

$\mathsf{MPoly.ComVer}(\mathsf{ck}, C = (c, \hat{c})) \to 0/1$**:** Runs $b \leftarrow \mathsf{Biv.ComVer}(\mathsf{ck}, C = (c, \hat{c}))$;

$\mathsf{MPoly.OpenVer}(\mathsf{ck}, C, \{P_j\}_{0 \leqslant j \leqslant \ell}, \rho) \to \{P_j\}_j$**:** Runs $P \leftarrow \mathsf{Biv.OpenVer}(\mathsf{ck}, C, P, \rho)$ where P is parsed as $\sum_{i,j=0}^{d,\ell} p_{ij} X^i Y^j$. then output 1, else output 0 (reject).

We state the following theorem. Its proof (see the full version) simply follows from the way we encode multiple polynomials into a bivariate one.

Theorem 10. *This commitment scheme* MPoly-$\mathcal{C}om$ *is perfectly hiding, computationally binding, and knowledge binding assuming the scheme* BivPoly.$\mathcal{C}om$ *also is so.*

7.2 Succinct Proof of Multiple Evaluations in a Point k

The construction of an efficient MUniEv-Π dedicated to multiple uni-variate polynomial evaluations in some common point k follows as well from the BivPE-Π scheme we built for partial evaluations. More precisely, for some parameters d, ℓ and some given knowledge commitments C, C' for polynomials of maximal degree d, $\{P_j\}_{0 \leqslant j \leqslant \ell} \in \mathbb{Z}_q[X]$ and scalars $\{p_j\}_{0 \leqslant j \leqslant \ell} \in \mathbb{Z}_q$ and a public evaluation point $k \in \mathbb{Z}_q$, we want to prove that p_j is the evaluation $P_j(k)$ for any $0 \leqslant j \leqslant \ell$.

Description of Our CaP MUniEv-Π Protocol. We now describe our protocol for proving multiple uni-variate polynomial evaluations in some common point k, where the j index is always considered as $0 \leqslant j \leqslant \ell$, and thus for $\ell + 1$ polynomials:

$\mathsf{MUniEv\text{-}\Pi.Gen}(1^\lambda, \mathcal{R}_{\mathbf{uni}}) \to \mathsf{crs}$**:** On input a security parameter $\lambda \in \mathbb{N}$ and a NP relation $\mathcal{R}_{\text{uni}} := \{(u = (\{P_j\}_j, k); w = \{p_j\}) : P_j(k) = p_j\}$, define the associated relation $\mathcal{R}_{\text{bi}} := \{(u = (P(X,Y), k); w = Q(Y)) : Q(Y) = P(k,Y)\}$ where $P(X,Y) := \sum_{j=0}^{\ell} P_j Y^j, Q(Y) := \sum_{j=0}^{\ell} p_j Y^j$. Output the common reference string by running $\mathsf{crs} \leftarrow \mathsf{Gen}(\mathsf{ck}, \mathcal{R}_{\text{bi}})$;

$\mathsf{MUniEv\text{-}\Pi.Prove}(\mathsf{crs}, u = (C, C', k), w = (\{P_j\}_j, \{p_j\}_j, \rho, \rho')$**:** Given crs, the instance u and the witness w, the prover defines new bi-variate polynomials $P(X,Y) := \sum_{j=0}^{\ell} P_j Y^j, Q(Y) := \sum_{j=0}^{\ell} p_j Y^j$ and compute the proof π for those: $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, u = (C, C', k), w = (P, Q, \rho, \rho')$. Output $\pi := (D, e, \sigma, \tau)$;

$\mathsf{MUniEv\text{-}\Pi.Ver}(\mathsf{crs}, u, \pi) \to b$**:** Same algorithm as for partial-evaluation BivPE-Π.

Remark 11. The commitment D to the bivariate polynomial $W \in \mathbb{Z}_q[X,Y]$ that appears in the proof can be seen as a commitment to a vector of univariate polynomials $\{W_j\}_j$ using the MPoly-$\mathcal{C}om$ as follows: Write $W_j = \sum_{i=0}^{d} w_{ij} X^i$, then running $\mathsf{MPoly.Com}(\mathsf{ck}, \{W_j\}_j)$ gives the same output (D, ω) as running $\mathsf{Biv.Com}(\mathsf{ck}, W)$.

Theorem 12. *Assuming the* BivPE-Π *is a public coin argument of knowledge of openings of C and C' to some polynomials $P \in \mathbb{Z}_q[X,Y], Q \in \mathbb{Z}_q[Y]$ such that $P(k,Y) = Q(Y)$, then* MUniEv-Π *is a public coin argument of knowledge of openings of C and C' to a set of polynomials $\{P_j\}_j \in \mathbb{Z}_q[X]$ and a set of scalars $\{p_j\}_j \in \mathbb{Z}_q$ such that $P_j(k) = p_j \forall 0 \leqslant j < \ell$.*

For lack of space, the proof appears in the full version. It is almost a straightforward reduction to the properties of BivPE-Π.

7.3 Efficiency and Comparison

We summarize the performance of our scheme $\mathsf{MUniEv\text{-}\Pi}$ in terms of prover and verification time and proof size. The proof consists of 2 elements from the group $\mathbb{G}$ and 3 elements of $\mathbb{Z}_q$. Generating a proof for ℓ polynomials of degree d requires a total of $2\ell d$ exponentiations in $\mathbb{G}$ in order to compute the commitment D, and $O(\ell d \log d)$ operations in $\mathbb{Z}_q$ in order to compute the polynomial $W(X, Y)$ using polynomial division.[9] Verifying a proof requires 5 pairings, and the following number of exponentiations: 6 in $\mathbb{G}$, 1 in $\mathfrak{G}$ and 1 in $\mathbb{G}_T$.[10]

We compare $\mathsf{MUniEv\text{-}\Pi}$ against a solution based on a general-purpose SNARK restricted to proving multiple polynomial evaluations in a commit-and-prove fashion. For the latter, we choose the *LegoGroth16* scheme from [CFQ19], which makes the SNARK of [Gro16] (which is currently among the most efficient SNARKs) to efficiently work with committed inputs, and that achieves the following efficiency. The proof consists of 4 elements of $\mathbb{G}$ and 1 element of $\mathfrak{G}$. Let m and n be the size and degree of the QAP modeling the evaluation of ℓ polynomials of degree d, and note that $m, n \geqslant \ell d$. Proof generation requires $2m + n + \ell d + \ell$ and m exponentiations in $\mathbb{G}$ and $\mathfrak{G}$ respectively, as well as $O(n \log n)$ operations in $\mathbb{Z}_q$ for a polynomial division. Verification requires 7 pairings.

The analysis above shows that our scheme $\mathsf{MUniEv\text{-}\Pi}$ has slightly smaller proofs and, more notably, has faster proof generation. Considering that $m, n \geqslant \ell d$ and that $\mathfrak{G}$ operations are at least twice slower than in $\mathbb{G}$, our prover is at least 3 times faster.

8 Security Analysis of our CaP BivPE-Π

In what follows we prove the main security result of our paper, Theorem 8. We focus on knowledge soundness of our $\mathsf{CaP\ BivPE\text{-}\Pi}$ scheme. We defer the reader to the full version for the proof of correctness and zero-knowledge.

Before going into the technical details of the proof, we provide some intuition about its strategy. The polynomial commitment scheme $\mathsf{BivPoly}.\mathcal{C}om$ requires the prover Prove to exhibit two values $(c, \hat{c})$, that are the same encoding of coefficients of a polynomial $P(X, Y)$ in the exponent, but with respect to different bases. The reason that we require the prover to duplicate its effort w.r.t. α is so that the simulator in the security proof can extract representations of $(c, \hat{c})$ as a polynomial $P(X, Y)$, under the $(d, \ell) - \mathsf{BPKE}$ assumption.

Suppose an adversary $\mathcal{A}$ manages to forge a SNARK of a false statement that nonetheless passes the verification test. The intuition behind the proof is to use the adversary $\mathcal{A}$ and the fact that the commitment scheme $\mathsf{BivPoly}.\mathcal{C}om$ is extractable to be able to solve the $d - \mathsf{SDH}$ assumption for $d = \deg(P)$ in X. There is a similar complementary case that allows this adversary to solve the $d - \mathsf{SDH}$ assumption for $d = \deg(P)$ in Y (actually ℓ in our notations).

[9] Note that W can be computed by aggregating the results of ℓ polynomial divisions of degree d.

[10] The numbers are obtained by observing that the six pairings for $\mathsf{Biv.ComVer}$ can be batched resulting into 2 pairings and 4 exponentiations in $\mathbb{G}$.

To proceed to proving Theorem 8, we first need two preliminary lemmas:

Lemma 13 (Global Extractor). *Assume that* $\mathsf{BivPoly}.\mathcal{C}om$ *is an extractable commitment scheme with perfect hiding and computational binding properties and that* $(d,\ell)-\mathsf{BPKE}$ *assumption holds in the bilinear group* gk*. For any* PPT *adversary* $\mathcal{A}^{\mathsf{KS}}$ *agains the knowledge soundness of* $\mathsf{BivPE}\text{-}\Pi$ *that has non-negligeable probability of success in breaking the scheme, there exists an extractor* Ext *such that:*

$$\Pr\left[\begin{array}{c} C=\mathsf{Biv.Com}(P,\rho) \\ \wedge\ C'=\mathsf{Biv.Com}(Q,\rho') \\ \wedge\ D=\mathsf{Biv.Com}(W,\omega) \end{array}\middle|\begin{array}{r} \mathsf{crs}\leftarrow\mathsf{BivPE}\text{-}\Pi.\mathsf{Gen}(1^\lambda,\mathcal{R}),z\leftarrow\mathcal{Z}(\mathsf{crs}) \\ ((u,\pi);\mathsf{wit})\leftarrow(\mathcal{A}^{\mathsf{KS}}\|\mathsf{Ext}^*)(\mathsf{crs},z) \\ u:=(C,C',k),\ \pi:=(D,\mathbb{U},\sigma,\tau) \\ \mathsf{wit}:=(P,\rho,Q,\rho',W,\omega) \\ \mathsf{BivPE}\text{-}\Pi.\mathsf{Ver}(\mathsf{crs},u,\pi)=1 \end{array}\right]=$$

$$=1-\mathsf{negl}(\lambda).$$

Proof. We show the existence of an extractor Ext^* that will output the polynomials $P(X,Y)$, $Q(Y)$, $W^*(X,Y)$ and the randomness ρ,ρ',ω corresponding to the commitments C,C',D, with overwhelming probability.

Let $\mathcal{A}^{\mathsf{KS}}$ be an adversary that breaks the KS of the protocol $\mathsf{BivPE}\text{-}\Pi$ with overwhelming probability, meaning it outputs a false proof that passes the verifier checks. Consider now the adversary $\mathcal{B}^{\mathsf{BPKE}}$ that takes as input $\sigma\leftarrow(g,\{g^{s^it^j}\}_{i,j=0}^{d,\ell},\{\hat{g}^{s^it^j}\}_{i,j=0}^{d,\ell};(h,\hat{h},h^s);(\mathfrak{g},\mathfrak{g}^\alpha,\mathfrak{g}^s))$ and runs the adversary $\mathcal{A}^{\mathsf{KS}}$ against the scheme. $\mathcal{B}^{\mathsf{BPKE}}$ can provide a valid CRS to $\mathcal{A}^{\mathsf{KS}}$ by using its inputs:

$$\mathsf{crs}=\{\mathsf{gk},(g_{ij})_{i,j=0}^{d,\ell},(\hat{g}_{ij})_{i,j=0}^{d,\ell};(h,\hat{h},h_1);(\mathfrak{g},\mathfrak{g}^\alpha,\mathfrak{g}_1)\}.$$

The statement u, corresponding to $\pi\leftarrow\mathcal{A}^{\mathsf{KS}}(\mathsf{crs})$, contains the values $C:=(c,\hat{c}),C':=(c',\hat{c}')$ that verify $\mathsf{e}(c,\hat{\mathfrak{g}})=\mathsf{e}(\hat{c},\mathfrak{g})$ and $\mathsf{e}(c',\hat{\mathfrak{g}})=\mathsf{e}(\hat{c}',\mathfrak{g})$. The same holds for the value D provided in the proof $\pi=(D,e,\sigma,\tau)$, i.e. $\mathsf{e}(d,\hat{\mathfrak{g}})=\mathsf{e}(\hat{d},\mathfrak{g})$.

Provided that for any adversary $\mathcal{B}^{\mathsf{BPKE}}$ that outputs valid commitment pair $(c,\hat{c})$, there exists an extractor that returns the corresponding witness (the opening). We run the extractor $\mathsf{Ext}_{\mathcal{B}}$ associated to $\mathcal{B}^{\mathsf{BPKE}}$ for each of the inputs $C=(c,\hat{c}),C'=(c',\hat{c}'),D=(d,\hat{d})$. This returns the description of polynomials $P(X,Y),Q(Y),W^*(X,Y)$ and some scalars ρ,ρ',ω. Note that the existence and efficacy of $\mathsf{Ext}_{\mathcal{B}}$ is guaranteed by the $(d,\ell)-\mathsf{BPKE}$ assumption. We will then define a *general* extractor Ext^* associated to the adversary $\mathcal{A}^{\mathsf{KS}}$ by running $\mathsf{Ext}_{\mathcal{B}}$ on the same input. We call this global algorithm composed of the adversary $\mathcal{A}^{\mathsf{KS}}$ and the general extractor Ext^*, machine $M:=\mathcal{A}^{\mathsf{KS}}||\mathsf{Ext}^*$. □

Lemma 14 (Extended Adversary Machine). *Assume that* $(d,\ell)-\mathsf{BPKE}$ *assumption holds in the bilinear group* gk *and that Schnorr proof used in the* $\mathsf{BivPE}\text{-}\Pi$ *protocol is sound. For any* PPT *adversary* $\mathcal{A}^{\mathsf{KS}}$ *against the knowledge soundness of the scheme* $\mathsf{BivPE}\text{-}\Pi$ *that outputs* $u=(C,C',k),\pi=(D,e,\sigma,\tau)$*, where* C,C',D *are well-formed commitments under* $\mathsf{BivPoly}.\mathcal{C}om$ *and the proof* π *verifies,*

i.e. $\mathsf{Ver}(\mathsf{crs}, u, \pi)$*, there exists a machine,* extended adversary $\mathcal{A}^*$ *that outputs the same as* $\mathcal{A}^{\mathsf{KS}}$ *together with an extended witness* $\mathsf{wit} = (P, \rho, Q, \rho', W, \omega, \delta, \gamma)$*, where* $P, W \in \mathbb{Z}_q[X, Y], Q \in \mathbb{Z}_q[Y]$ *are the openings of the commitments* (C, C', D) *under randomness* ρ, ρ', ω *and* δ, γ *are such that* $\mathbb{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$.

Proof. We use the previous defined machine M from Lemma 13 and the rewinding technique [PS00] for proving the soundness of the Schnorr's proof to extract the scalars δ, γ such that $\mathbb{A} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$: Consider the game between the challenger and the machine M against the soundness of the Schnorr's proof. The challenger runs M by fixing the values (C, C', D) and changing the oracle definition to get a fork with $e' \leftarrow \mathsf{Hash}(U, D, \mathbb{U}) \neq e$. The forger M will output two distinct forgeries corresponding to the same random oracle query, but for two distinct answers of the random oracle, e and e'. The Forking Lemma shows that by rewinding the adversary $\mathcal{O}(q_h/\varepsilon)$ times, where q_h is the maximal number of random oracle queries of the machine M and ε its success probability, then one finds two such forgeries $(\sigma, \tau), (\sigma', \tau')$ with constant probability, which enables to compute the values δ, γ such that $\mathbb{A} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$.

Using the existence of Ext^* extractor and of the algorithm that rewinds the machine M in order to obtain the output δ, γ as described before, we can define an aggregate machine $\mathcal{A}^*$ corresponding to the concatenation of both. This machine $\mathcal{A}^*$ takes the same input as $\mathcal{A}^{\mathsf{KS}}$ and outputs the witness corresponding to the commitment openings $(P, \rho), (Q, \rho'), (W, \omega)$ and two scalars δ, γ satisfying $\mathbb{A} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$. □

Knowledge Soundness. We now have all the tools to prove the soundness in two steps.

Step 1. First we show that for every PPT adversary $\mathcal{A}^{\mathsf{KS}}$ against the soundness of the protocol, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ that runs on the same input and random coins as $\mathcal{A}^{\mathsf{KS}}$ and outputs a witness. Defining the extractor $\mathsf{Ext}_{\mathcal{A}}$ is straightforward from the Lemma 13 by running the Ext^* and keeping just the values (P, ρ, Q, ρ') from its output.

Assuming the existence of an adversary $\mathcal{A}^{\mathsf{KS}}$ and extractor $\mathsf{Ext}_{\mathcal{A}}$ that has a non-negligible success probability in winning the soundness game against the protocol $\mathsf{BivPE\text{-}\Pi}$, we now show that we can either solve the discrete logarithm problem, or break the $d - \mathsf{SDH}$ assumption.

Step 2. Suppose the machine $\mathcal{A}^*$ associated to $\mathcal{A}^{\mathsf{KS}}$ defined in the Lemma 14 is able to output a cheating pair statement-proof $u = (C, C', k), \pi = (D, e, \sigma, \tau)$ and a witness $\mathsf{wit} = (\rho, \rho', \omega, P, Q^*, W^*, (\delta, \gamma))$ such that it passes verification checks, but the extracted values $P \in \mathbb{Z}_q[X, Y], Q^* \in \mathbb{Z}_q[Y]$ are not satisfying the expected relation $Q^*(Y) = P(k, Y)$.

For simplicity, we will call $\Delta = \rho' - \rho$. Assuming that the commitment scheme is binding, then one of the following scenarios must hold:

1. The polynomials extracted do not satisfy the correct relation not even when evaluated in s: $W^*(s, t) \neq \frac{P(s,t) - Q^*(t)}{s-k}$. This type of forgery can be reduced to the DLog problem for $(g, h) \in \mathbb{G}$, in the case 1 below (see Lemma 15);

2. The polynomial $W^* \in \mathbb{Z}_q[X,Y]$ committed in D does not satisfy the correct relation with respect to the other extracted values P, Q^*, but still evaluated in s,t we have that $W^*(s,t) = \frac{P(s,t)-Q^*(t)}{s-k}$. We reduce the case to the $d-\mathsf{SDH}$ assumption, in the case 2 below (see Lemma 16).

Lemma 15 (Case 1). *Consider the adversarial machine $\mathcal{A}^*$ associated to $\mathcal{A}^{\mathsf{KS}}$ defined by the Lemma 14 that outputs some values $u = (k, C, C', D, e, \sigma, \tau)$ and $(\rho, P, \rho', Q^*, \omega, W^*, \delta, \gamma)$, such that $P(k,Y) \neq Q^*(Y)$, where $P, W^* \in \mathbb{Z}_q[X,Y]$, $Q^* \in \mathbb{Z}_q[Y]$ and $(P,\rho), (Q^*,\rho'), (W^*,\omega)$ are the openings of the commitments (C, C', D) and (δ, γ) satisfy $\mathbb{A} := \mathsf{e}(h^{\omega} g^{W^*(s,t)}, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(h^{-\Delta} g^{P(s,t)-Q^*(t)}, \mathfrak{g})^{-1} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g})$. Given that the verification check outputs 1 for π, there is a negligible probability that the values k, P, Q^*, W^* are such that $W^*(s,t) \neq \frac{P(s,t)-Q^*(t)}{(s-k)}$ under DLog assumption with respect to the group $\mathbb{G}$.*

Lemma 16 (Case 2). *Consider the adversarial machine $\mathcal{A}^*$ associated to $\mathcal{A}^{\mathsf{KS}}$ defined by the Lemma 14 that outputs some values $u = (k, C, C', D, e, \sigma, \tau)$ and $(\rho, P, \rho', Q^*, \omega, W^*, \delta, \gamma)$, such that $P(k,Y) \neq Q^*(Y)$, where $P, W^* \in \mathbb{Z}_q[X,Y]$, $Q^* \in \mathbb{Z}_q[Y]$ and $(P,\rho), (Q^*,\rho'), (W^*,\omega)$ are the openings of the commitments (C, C', D) and (δ, γ) satisfy $\mathbb{A} := \mathsf{e}(h^{\omega} g^{W^*(s,t)}, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(h^{-\Delta} g^{P(s,t)-Q^*(t)}, \mathfrak{g})^{-1} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g})$. Given that the verification check outputs 1 for π, there is a negligible probability that the values k, P, Q^*, W^* satisfy $W^*(s,t) = \frac{P(s,t)-Q^*(t)}{(s-k)}$ under $d'-\mathsf{SDH}$ assumption with respect to the bilinear group gk, where $d' = \max\{d, \ell\}$.*

For lack of space, the proofs of the lemmas above is in the full version.

Acknowledgments

This work was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud). The first author has been partially supported by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), CRYPTOEPIC (ref. EUR2019-103816), and SECURITAS (ref. RED2018-102321-T), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), and by a research gift from Protocol Labs.

References

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13. IEEE Computer Society Press, October 1992.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699. Springer, Heidelberg, August 2018.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

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

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

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

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BL07] Ahto Buldas and Sven Laur. Knowledge-binding commitments with applications in time-stamping. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 150–165. Springer, Heidelberg, April 2007.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.

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

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 377–408. Springer, Heidelberg, December 2017.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In Shafi Goldwasser, editor, *ITCS 2012*, pages 90–112. ACM, January 2012.

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

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

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 844–855. ACM Press, November 2014.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

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

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

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

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

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

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

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

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

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439. Springer, Heidelberg, March 2012.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Heidelberg, March 2013.

[RAD78] Ron Rivest, Len Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–177, 1978.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443. Springer, Heidelberg, May 2010.

[WJB+17] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, abhi shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2071–2086. ACM Press, October / November 2017.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Heidelberg, August 2019.

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pages 863–880. IEEE Computer Society Press, May 2017.