MyOPE: Malicious securitY for Oblivious Polynomial Evaluation

Malika Izabachène[1], Anca Nitulescu[2], Paola de Perthuis[1,3,4], and David Pointcheval[3,4]

[1] Cosmian
[2] Protocol Labs
[3] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[4] INRIA, Paris, France

Abstract. Oblivious Polynomial Evaluation (OPE) schemes are interactive protocols between a sender with a private polynomial and a receiver with a private evaluation point where the receiver learns the evaluation of the polynomial in their point and no additional information. They are used in Private Set Intersection (PSI) protocols.
We introduce a scheme for OPE in the presence of malicious senders, enforcing honest sender behavior and consistency by adding verifiability to the calculations. The main tools used are FHE for input privacy and arguments of knowledge for the verifiability property. MyOPE deploys sublinear communication costs in the sender's polynomial degree and one to five rounds of interaction.
In other words, it can be used as a verifiable computation scheme for polynomial evaluation over FHE ciphertexts. While classical techniques in pairing-based settings allow generic succinct proofs for such evaluations, they require large prime order subgroups which highly impact the communication complexity, and prevent the use of FHE with practical parameters. MyOPE builds on generic secure encodings techniques that allow composite integers and enable real-world FHE parameters and even RNS-based optimizations. It is best adapted for the unbalanced setting where the degree of the polynomial and the computing power of the sender are large.
MyOPE can be used as a building block in specialized two-party protocols such as PSI (this use-case is hereafter described), oblivious keyword search, set membership and more using the OPE instantiation.
As another contribution, our techniques are generalized to applications other than OPE, such as Symmetric Private Information Retrieval (SPIR), to make them secure against a malicious sender.

1 Introduction

1.1 Oblivious Polynomial Evaluation

Secure Two-Party Computations (2PC). Secure two-party computations enable two parties to mutually run a protocol computing the execution of a function $f(\cdot,\cdot)$ on their private inputs x, y, and allowing the parties to learn the output $f(x, y)$ but nothing else about the inputs. Seminal results from the 80s, e.g. [Yao86] have shown that with 2PC it is possible to securely evaluate any boolean circuit. Since these first feasibility results, a long line of works focused on improving the efficiency of the computational and communication costs in 2PC protocols. Two approaches were followed in these efforts: (1) improving generic protocols that compute any boolean or arithmetic circuit and (2) designing tailored protocols for practical functions. The latter approach focuses on taking advantage of these functions' particular structure to gain efficiency. Some examples of such 2PC are schemes designed for search problems [HT10,Ver11], RSA key generation [Gil99], set intersection [JL09,HN12], or polynomial evaluation [NP99].

Malicious Security of 2PC. An important aspect to consider when designing 2PC schemes is the adversarial model. In the *semi-honest* (or passive) adversarial model, a.k.a. honest-but-curious, the corrupted parties follow the protocol, but try to learn more about the private inputs of other parties, so there is no impact on the correctness of computation results. On the other hand, in the *malicious* (active) adversarial model, corrupted parties can collaborate in any way and misbehave arbitrarily, without following the protocol description. Therefore, assuring not only the privacy of the inputs, but also the correctness of the outputs (robustness) is essential in such scenarios. Most often, a 2PC protocol execution preserves the privacy of the inputs against malicious adversaries. Regarding the correctness of the output, giving consistency guarantees on the outputs of the honest parties generally comes with big overheads.

Oblivious Polynomial Evaluation. A particular case of 2PC is Oblivious Polynomial Evaluation, where the function to be evaluated is a polynomial $\mathsf{f}(X)$ of fixed degree N secretly chosen by the sender. The receiver chooses the secret evaluation point m. After running OPE, the receiver obtains the value $\mathsf{f}(\mathsf{m})$ without learning anything else about the polynomial f and without giving the sender any information about the point m.

OPE is an important building block for various 2PC schemes that generally require multiple executions of an OPE protocol for the same polynomial and different evaluation points. However, the standard definition of receiver privacy does not preclude the sender from cheating by using a polynomial of higher degree than expected or changing the polynomial between multiple executions. Therefore, extending the security to malicious senders is essential for such applications.

Unfortunately, when relying on general 2PC protocols to securely perform the OPE functionality in the malicious setting, some efficiency overheads incur: schemes for Boolean circuits apply the cut-and-choose technique which requires repeating the computation K times in order to prevent cheating except with probability 2^{-K}, while 2PC schemes for arithmetic circuits run an expensive preprocessing model to generate correlated randomness.

1.2 Motivation: Applications of OPE

The first custom OPE protocol was defined and introduced in [NP99] and is secure only against passive adversaries. While few constructions were built since, the OPE functionality can lead to various interesting applications such as data mining [LP00], private set intersection (PSI) (in their original paper [NP99], Naor and Pinkas mention the contact discovery use-case), privacy-preserving keyword search [JL09], set membership (related to PSI), and RSA key generation [Gil99].

Private Set Intersection. Private Set Intersection (PSI) is a well-studied specialized form of 2PC that allows two parties to jointly compute the intersection of their input sets, without revealing any other information about them (other than upper bounds on their sizes). Although protocols for PSI have been built upon generic 2PC, more efficient custom protocols can be achieved by taking advantage of the problem's structure.

A recent line of works reduces interactions in *unbalanced* PSI schemes by using (Leveled) Fully Homomorphic Encryption (FHE) as a tool [CLR17,CHLR18]. In an unbalanced PSI protocol the sender has a set of much larger size than the receiver, and also bigger storage and computational power. While [CLR17] achieves few rounds of interaction, it is secure only against passive adversaries. The follow-up work [CHLR18] extends the security to consider malicious receivers. This is done at the cost of an expensive preprocessing phase with a linear number of interactive rounds in the receiver's set size, and does not address the more relevant case of malicious senders.

The real challenge in PSI is enforcing that the sender performs the correct computation. In the schemes mentioned above, the sender can deviate from the prescribed protocol and make the receiver compute an arbitrary intersection. Since the sender obtains a homomorphically encrypted copy of the receiver's set, they could compute other circuits than the pre-established one. Therefore, if the protocol returns the intersection result to the sender, then not only the result but also the receiver's privacy are compromised. For example, in FHE-based PSI schemes, it is possible for the sender to force the receiver to output their full set as the result of an intersection.

In [CHLR18] the authors heuristically argue that the misbehavior of a *malicious sender* can be mitigated by using a complex hash function H that the sender is unable to evaluate for an encrypted input y. Given the fast-paced advancements made by FHE schemes in recent years, there is no guarantee that a higher degree polynomial could not be evaluated and such a PSI system is insecure when one considers malicious senders. We address these limitations and show an alternative way to build unbalanced PSI with *malicious senders*.

1.3 Related Work

Despite its broad applicability, the study of the OPE functionality includes only few practical and secure protocols, initiated in [NP99] and further continued in works like [CL01,ZB05,HL09].

While [NP99] proposed the first construction for OPE, it relies on a newly introduced intractability assumption: the noisy polynomial interpolation. Naor and Pinkas conjectured that it could be reduced to a more widely studied assumption, the polynomial reconstruction problem. Nevertheless, as shown in [BN00], this conjecture seems not to hold in general.

OPE Schemes with Active Security. Among recent OPE schemes, to the best of our knowledge, the only ones secure against malicious attacks are [HL09,Haz18]. However [HL09] has at least 17 rounds of interaction and the parties send each other $\mathcal{O}(\lambda N)$ Paillier encryptions, where λ is the security parameter and N the degree of the polynomial. Also, their claimed efficiency holds only for sufficiently low degree polynomials. [GNN17] has active security too, but communication is also linear in N.

[Haz18] shows an OPE scheme for polynomial evaluation in the exponent of a DLog group using algebraic Pseudo-Random Functions (PRF). They focus on improving the computational efficiency of [HL09] by reducing the number of modular exponentiations, and removing the trusted setup requirement, while preserving the same number of rounds of interaction and communication complexity as in [HL09], and apply their scheme to private set membership 2PC.

[PRTY20] gives malicious security for PSI with symmetric set sizes a bit smaller than ours, but the communication is linear and with concrete parameters it is about the same order as our for sets of size 2^{20}.

Verifiable Computation (VC). Introduced by [GGP10], VC schemes are cryptographic systems that enable checking the integrity of results from delegated computations. More recent works [FNP20,BCFK20] have improved the efficiency and the expressivity of VC schemes to work for computations over encrypted data. These schemes, however, require proving the entire FHE circuit evaluation which is very expensive. Moreover, they neither allow using practical parameters for the FHE scheme, nor speedups through classical optimisations such as Residue Number System (RNS).

Polynomial Commitments (PC). First introduced by [KZG10], PCs are commitments for polynomials of maximal degree N and coefficients in a field $\mathbb{F}$ that support an interactive argument of knowledge to prove the correct evaluation of a committed secret polynomial in a given point. However, while this ensures verifiability of the evaluation result, it does not protect the privacy of the evaluation point.

Private Information Retrieval (PIR) Using Somewhat Homomorphic Encryption (SwHE). A long line of PIR articles such as introduced in [TP11] uses Somewhat Homomorphic Encryption with ciphertexts of zero or one multiplied with elements of the sender's database, added up and sent back to the receiver. However, communication is linear in the size of the sender's set, and there is no verifiability.

1.4 Our Contribution

In this work, we study the Oblivious Polynomial Evaluation (OPE) functionality in the malicious setting and show an application to Private Set Intersection (PSI): the untrusted sender is asked to commit an input polynomial and prove consistency of the evaluation with respect to the initial commitment.

More precisely, we introduce MyOPE, a scheme for oblivious evaluation of polynomials of high degree N that achieves $\mathcal{O}(N^{1/d})$ communication costs, where d can be freely chosen and optimized with respect to the ciphertext sizes. This sublinear communication improves on the state-of-the-art. Concretely, in order to avoid malicious attacks, we enforce honest behavior of the sender (the evaluator) by asking an extra proof for the claimed result of the evaluation with

respect to some committed polynomial. This is of main importance when OPE is used in PSI, where the same or related polynomials have to be evaluated on distinct inputs. MyOPE is best adapted for large degree polynomials in the unbalanced setting with computationally powerful senders. In our experiments we consider polynomials with degrees N from 2^{20} to 2^{40}.

MyOPE can be seen as a Verifiable Computation over encrypted data, in the following sense: the sender provides a proof of their honest behavior during the homomorphic polynomial evaluation, to convince the receiver.

Our Techniques. In a nutshell, MyOPE uses FHE to encrypt the receiver's input m under their key, so that the sender can proceed with the homomorphic evaluation over the ciphertext c to obtain an encryption of $\mathsf{f}(\mathsf{m})$ that the receiver can thereafter decrypt. For efficient verifiability, we also need a commitment C_{f} of $\mathsf{f}(X)$ that is compatible with the FHE. At this point we remark that the sender needs to perform an expensive evaluation over the FHE ciphertext c with a multiplicative depth corresponding to the logarithm of the degree of their polynomial $\mathsf{f}(X)$. Practical FHE schemes may still not be efficient enough for this when the degree of $\mathsf{f}(X)$ is large. To overcome this, we can find the best trade-off between communication and computational costs, depending on the polynomial degree N. Namely, the receiver will send $\mathcal{O}(N^{1/d})$ encrypted powers of m for some carefully chosen d. Then the sender will homomorphically compute the ciphertexts c_i of the remaining powers m^i of m in order to evaluate a ciphertext of $\mathsf{f}(\mathsf{m})$ by a simple inner product with the vector of the coefficients. The computation of the ciphertexts c_i will require an only d multiplicative-depth for the FHE circuit.

At this step, a malicious sender could just use arbitrary values c_i instead. Our scheme MyOPE requires the prover/sender to (compactly) commit the vector of ciphertexts $(\mathsf{c}_i)_i$ and prove it indeed coincides with the correct values. To check this, the receiver asks for a random linear combination of the c_i's. Since the vector $(\mathsf{c}_i)_i$ is committed before seeing the random challenge, the sender has negligible probability to compute the expected value if the committed values are incorrect.

The core argument used in this step enables checking that an inner product between a committed vector of ciphertexts and a public vector (here a random challenge chosen by the receiver) has an expected value when decrypted (value unknown to the prover). Similarly, for the final step, the sender has to convince the receiver that the inner product between the vector of the ciphertexts c_i's committed at the previous step and the initial polynomial commitment C_{f} was performed correctly.

While FHE will protect receiver's privacy (the message m), some additional noise will be added by the sender to the final ciphertext to let the receiver learn $\mathsf{f}(\mathsf{m})$ but nothing else about f. The sender will also give guarantees about the added noise.

Applications. The MyOPE checks are useful when we furthermore need to ensure consistency between repetitions. In the context of PSI, in order to determine the intersection of the sets $\mathcal{X} = \{x_1, \ldots, x_N\}$ owned by the prover and $\mathcal{Y} = \{y_1, \ldots, y_K\}$ owned by the verifier, the sender/prover generates the polynomial $\mathsf{f} = \prod_{i=1}^{N}(X - x_i)$, commits it, and the receiver will learn whether $\mathsf{f}(y_j) = 0$ for each of their y_j's, but nothing else, by multiple evaluations of the same polynomial on multiple inputs. To check whether the evaluations are equal to zero or not, the sender first commits f in C_{f} and to avoid information leakage, the polynomial is randomized by a multiplication factor at each evaluation. After the initial commitment C_{f}, the sender/prover draws a random invertible coefficient $\mathfrak{r}$ for each evaluation in a specific point x, commits $\mathfrak{r} \cdot \mathsf{f}$ and proves it is f multiplied with a secret $\mathfrak{r}$ using the zero-knowledge quadratic check from Section 3. MyOPE is then used to evaluate $\mathfrak{r} \cdot \mathsf{f}(x)$, which is equal to zero iff $\mathsf{f}(x) = 0$.

Extensions. We believe that our techniques for lifting OPE from passive security to malicious sender security are generic and of independent interest. The main tool is a proven inner-product that can be applied to lift other 2PC computation protocols to the malicious setting. We illustrate such an extension in the case of Symmetric Private Information Retrieval (SPIR). SPIR

schemes allow a receiver to ask for a secret index of a data in the sender's set, and the receiver then only learns this sole data. Our augmented SPIR protocol allows a receiver to recover the i-th field of a private database owned by the sender, without leaking the index i, even against a malicious sender, once the database has been compactly committed.

Efficiency in Practice. When using the Fan-Vercauteren FHE scheme [FV12], from the plaintext ring $\mathbb{Z}_t[X]/\mathsf{r}(X)$, where $\mathsf{r}(X) = X^n+1$, into the ciphertext ring $\mathbb{Z}_q[X]/\mathsf{r}(X)$, the core parameters are the integers n, q, and t. In order to understand various parameter choices, we provide some rough cost estimates for the FHE: each ciphertext is $2n \log q$ bits long, and the underlying plaintexts $n \log t$ bits long. With a MyOPE instantiation for polynomials of degree $N = 2^{30}$, we set $n = 2^{15}$ and use t on 56 bits and q on 839 bits, to obtain appropriate semantic security for the FHE and decryption correctness. Then, the size of the FHE ciphertexts to be sent is a bit more than 1 GByte. Our proof of the sender's honest behavior has less than 25 MByte size for a soundness of 2^{-30}, with 5 rounds. The non-interactive proof with a soundness in 2^{-80} requires less than 100 MBytes.

2 Preliminaries

2.1 OPE: Oblivious Polynomial Evaluation

To define an OPE protocol as a two party protocol run between a receiver and a sender over a ring R, we first agree on the input and output:

Input: Receiver – an input $\mathsf{x} \in \mathsf{R}$, Sender – a polynomial $\mathsf{f}(X) \in \mathsf{R}[X]$.

Output: Receiver – $\mathsf{f}(\mathsf{x})$, Sender – nothing.

We consider the simplified setting where the sender gets no output. Such a protocol should satisfy correctness, sender privacy and receiver privacy, which are first defined in the honest-but-curious setting. Thereafter, we will deal with malicious senders. Privacy notions can be formulated as indistinguishability games:

Definition 1 (Sender Privacy with an Honest-but-Curious Receiver). *For any* PPT *adversary* $\mathcal{A}$ *executing honestly the receiver's part on input* x*, for any* $\mathsf{f} \neq \mathsf{f}' \in \mathsf{R}[X]$*, such that* $\mathsf{f}(\mathsf{x}) = \mathsf{f}'(\mathsf{x})$*, the views that* $\mathcal{A}$ *sees in case the sender's input is* f *and in case the sender's input is* f' *are computationally indistinguishable.*

Definition 2 (Receiver Privacy with an Honest-but-Curious Sender). *For any* PPT *adversary* $\mathcal{A}$ *executing honestly the sender's part, for any* $\mathsf{x} \neq \mathsf{x}' \in \mathsf{R}$*, the views that* $\mathcal{A}$ *sees in case the receiver's input is* x *and in case the receiver's input is* x' *are computationally indistinguishable.*

Malicious Sender Security. To get receiver privacy and correctness even in the case of a malicious sender, one usually enforces honest sender behavior: the sender first commits the polynomial $\mathsf{f}(X)$ and then proves the answers are consistent with that initial commitment.

2.2 FHE: Fully Homomorphic Encryption

Fully homomorphic encryption has been introduced in [Gen09]. Since the initial construction, major improvements have been made, with now practical and efficient solutions. In this work we will use the Fan-Vercauteren FHE scheme [FV12].

The Fan-Vercauteren FHE. One usually denotes $\mathcal{R}_t = \mathbb{Z}_t[X]/\mathsf{r}(X)$ the plaintext message space and $\mathcal{R}_q = \mathbb{Z}_q[X]/\mathsf{r}(X)$ the ciphertext space.

The FV encryption scheme [FV12] is described by parameters $\Gamma = (q, t, \sigma)$, with $q, N \in \mathbb{N}$ such that $0 < \sigma < 1$, where the size of q and noise parameter σ depend on several constraints, and namely the expected multiplicative depth for the correctness and the security of the Ring-LWE problem for the semantic security.

For efficiency reasons, with the possibility of using FHE with RNS [BEHZ16], we will assume $q = \prod_{i=1}^{\ell} p_i$ for prime factors $t = p_1 \leq \ldots \leq p_\ell$. We additionally take t such that $t|q$. We denote

$\Delta = q/t$ and χ the Gaussian distribution on $\mathbb{Z}$ with standard deviation σ. The FV encryption scheme [FV12] consists in the following algorithms:

$\mathsf{KG}(1^\lambda, \Gamma) \to (\mathsf{sk}, \mathsf{pk})$**:** On input the security parameter λ and the efficiency parameters Γ, sample $\mathsf{a}, \mathsf{s} \leftarrow \mathcal{R}_q$ and set $(\mathsf{p}, \mathsf{p}') = (-(\mathsf{a} \cdot \mathsf{s} + \mathsf{e}), \mathsf{a}) \in \mathcal{R}_q^2$. $\mathsf{sk} = \mathsf{s}$ is the secret key, where the coefficients of e are taken from χ. The public key pk contains $(\mathsf{p}, \mathsf{p}')$ with a relinearization key rlk.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to (\mathsf{c}, \mathsf{c}')$**:** Using $(\mathsf{p}, \mathsf{p}')$ from pk and a message $\mathsf{m} \in \mathcal{R}_t$, compute ciphertext $(\mathsf{c}, \mathsf{c}') = (\mathsf{p} \cdot \mathsf{u} + \mathsf{e}_1 + \Delta \cdot \mathsf{m} \bmod q, \mathsf{p}' \cdot \mathsf{u} + \mathsf{e}_2 \bmod q)$, with coefficients of $\mathsf{u}, \mathsf{e}_1, \mathsf{e}_2$ also taken from χ.

$\mathsf{Dec}(\mathsf{sk}, (\mathsf{c}, \mathsf{c}')) \to \mathsf{m}$**:** Given $\mathsf{sk} = \mathsf{s}$, compute

$$\mathsf{d} = \mathsf{c} + \mathsf{c}' \cdot \mathsf{s} = \Delta \cdot \mathsf{m} - \mathsf{e} \cdot \mathsf{u} + \mathsf{e}_2 \cdot \mathsf{s} + \mathsf{e}_1 = \Delta \cdot \mathsf{m} + \mathsf{v} \bmod q$$
$$\mathsf{m}' = \lfloor \mathsf{d}/\Delta \rceil = \lfloor (\Delta \cdot \mathsf{m} + \mathsf{v})/\Delta \rceil = \mathsf{m} + \lfloor \mathsf{v}/\Delta \rceil \bmod t$$

where $\mathsf{v} = -\mathsf{e} \cdot \mathsf{u} + \mathsf{e}_2 \cdot \mathsf{s} + \mathsf{e}_1$ is the error term: $\mathsf{m}' = \mathsf{m}$ if $\|\mathsf{v}\|_\infty \leq \Delta/2$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{c}_i, \mathsf{c}'_i)_{i=1,\ldots,\ell}) \to (\mathsf{c}_f, \mathsf{c}'_f)$**:** Given pk, an arithmetic circuit f with bounded multiplicative depth and ℓ ciphertexts $(\mathsf{c}_i, \mathsf{c}'_i)_{i=1,\ldots,\ell}$ output the ciphertext $(\mathsf{c}_f, \mathsf{c}'_f)$.

The addition of two ciphertexts is a ciphertext of the sum of the plaintexts. Multiplicative operations have also been shown possible with additional information to relinearize the ciphertext after a product using rlk. For Eval to be correct, $\mathsf{Dec}(\mathsf{sk}, (\mathsf{c}_f, \mathsf{c}'_f))$ should return $f(m_1, \ldots, m_\ell)$ where $\mathsf{Dec}(\mathsf{sk}, (\mathsf{c}_i, \mathsf{c}'_i)) = m_i$ for $i = 1, \ldots, \ell$, with overwhelming probability. More detail on the FV parameters is provided in Appendix E.

Remarks. We stress that for verifiable computations, the evaluator will have to provide proofs of correct operations on the ciphertexts $(\mathsf{c}, \mathsf{c}') \in \mathcal{R}_q^2$. We will thus have to deal with polynomials over rings (and not fields).

Since the verifier can decrypt, we also need to require circuit privacy [SYY99]. This ensures the final ciphertext reveals no information about f. Indeed, not only is the encrypted input impacted by executing f, but also the noise, which might thus leak information about the function f. While several approaches have been proposed to avoid such leakage, we will apply the *noise-flooding* proposed by Gentry, which consists in adding a larger random noise to the final result. This will then make the final noise independent of f.

2.3 SNARKs: Succinct Non-Interactive Arguments of Knowledge

Zero-knowledge proofs/arguments are cryptographic protocols between two parties, a prover Prove and a verifier Ver, in which the prover can convince the verifier about the validity of a statement without leaking any extra information beyond the statement validity. SNARKs are non-interactive such protocols with a proof size independent of the size of the statement to be proven.

Bitansky *et al.* [BCI+13] provide an abstract model to build SNARKs under the concept of Linear PCP (LPCP). LPCPs are a form of interactive proofs where security holds under the assumption that the prover is restricted to computing only linear combinations of their inputs. These proofs can then be transformed into SNARKs by means of extractable linear-only encoding schemes that will restrict the prover to linear-only operations. Our techniques are similar and use generic linear-only encodings adapted for FHE ciphertexts.

Formal definitions of SNARKs and zk-SNARKs are given in Appendix A.

2.4 Secure Encoding Schemes over Rings

As stressed above, we will have to consider polynomials over rings. So we revisit the linear encodings defined in [BCI+13,GGPR13] in order to deal with rings.

Definition 3 (Encoding Scheme). *An encoding scheme over a ring* R *consists in a tuple of algorithms* $(\mathsf{Gen}, \mathsf{E})$.

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$, *a key generation algorithm that takes as input a security parameter and outputs public information* pk*, a secret key* sk*, and a verification key* vk*, that can be either public or private;*
- $E \leftarrow \mathsf{E}_{\mathsf{sk}}(a)$*, a (probabilistic) encoding algorithm mapping a ring element* $a \in \mathsf{R}$ *in the encoding space* $\mathcal{E}$*, using the secret key* sk*.*

Properties. An encoding scheme should satisfy the following properties, with efficient and correct algorithms:

- L-Linearly homomorphic: An algorithm $\mathsf{Eval}_{\mathsf{pk}}(E_1, \ldots, E_L; c_1, \ldots, c_L)$, on input public information pk, encodings $E_1 = \mathsf{E}_{\mathsf{sk}}(a_1)$, $\ldots$, $E_L = \mathsf{E}_{\mathsf{sk}}(a_L)$, and coefficients $c_1, \ldots, c_L \in \mathsf{R}$, outputs an encoding of $\sum_{i=1}^{L} c_i \cdot a_i$;
- L-Quadratic root verification: An algorithm $\mathsf{QCheck}_{\mathsf{vk}}(Q, E_1, \ldots, E_L)$, on input the verification key vk, a quadratic polynomial $Q \in \mathsf{R}[X_1, \ldots, X_L]$ and encodings $E_1 = \mathsf{E}_{\mathsf{sk}}(a_1)$, $\ldots$, $E_L = \mathsf{E}_{\mathsf{sk}}(a_L)$, checks whether or not the relation $Q(a_1, \ldots, a_L) = 0$ is satisfied in R;
- Image verification: An algorithm $\mathsf{Verify}_{\mathsf{vk}}(E)$, on input the verification key vk and an element E, verifies E is an actual encoding of some element in R: this algorithm not only verifies that $E \in \mathcal{E}$, but also that there exists an element $a \in \mathsf{R}$ such that E can be an encoding of a.

According to the verification key that can be either public or private, the verification processes will be either public or private.

Secure Encodings. We now formally define the soundness properties for the above verification algorithms, in terms of knowledge-soundness:

Definition 4 (Linear-Only Extractability). *An encoding scheme* $(\mathsf{Gen}, \mathsf{E})$ *over* R *is extractable if for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that the following probability is negligible in the security parameter:*

$$\Pr\left[\mathsf{QCheck}_{\mathsf{vk}}\left(X - \sum_{i=1}^{n} c_i X_i, E, E_1, \ldots, E_n\right) = \mathsf{false} \;\middle|\; \mathsf{Verify}_{\mathsf{vk}}(E) = \mathsf{true}\right]$$

on the probability space $(\mathsf{pk}, \mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$*,* $a_1, \ldots, a_n \xleftarrow{\$} \mathsf{R}$*,* $E_i \leftarrow \mathsf{E}_{\mathsf{sk}}(a_i)$*, for* $i = 1, \ldots, n$*, and* $(E; c_1, \ldots, c_n) \leftarrow (\mathcal{A} || \mathsf{Ext}_{\mathcal{A}})(\mathsf{pk}, (E_i)_i)$*.*

While the encoding is linearly-homomorphic, the above extractability property requires that it be impossible to derive new valid encodings excepted by linear combinations: any new *valid* encoding E of some $a \in \mathsf{R}$ will necessarily satisfy $a = \sum_i c_i a_i$, for extractable elements $c_i \in \mathsf{R}$. Intuitively, when an encoding E passes the verification test, one can extract the linear combination of the given initial encodings.

Zero-Knowledge Proofs. The above properties will be enough for a binding commitment, but additional blinding factors in an appropriate masking set $\mathcal{M}$ will be required for hiding commitments, which will depend on the ring R (see below for the particular case of $\mathsf{R} = \mathbb{Z}_q$). Then, Zero-Knowledge proofs will be needed for Quadratic root verifications with private linear combinations: $\mathsf{ZKLQCheck}_{\mathsf{vk}}(Q, E_1, \ldots, E_L; E'_1, \ldots, E'_\mu)$, on input a quadratic polynomial $Q \in \mathsf{R}[X_1, \ldots, X_L]$ and encodings $E_1 = \mathsf{E}_{\mathsf{sk}}(a_1)$, $\ldots$, $E_L = \mathsf{E}_{\mathsf{sk}}(a_L)$, $E'_1 = \mathsf{E}_{\mathsf{sk}}(b_1)$, $\ldots$, $E'_\mu = \mathsf{E}_{\mathsf{sk}}(b_\mu)$. The verification key vk on the verifier side and the private coefficients $c_1, \ldots, c_\mu$ on the prover side prove that the relation $Q(a_1, \ldots, a_L) = \sum c_i \cdot b_i$ is satisfied. The c_i's will be the blinding factors in $\mathcal{M}$.

Paillier-Based Secure Encodings. In Appendix B, we recall some constructions from pairing-based techniques (with public verification key vk) and from the linearly-homomorphic Paillier encryption scheme [Pai99] (with private verification key vk), as already explained in [GGPR13]. As we will hereafter exploit it, we just recall the latter:

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$: Run Paillier key generation $\mathsf{KG} \rightarrow (\mathsf{sk}, \mathsf{pk} = N)$, where $\mathcal{N}$ is an RSA modulus and set $\mathsf{vk} = (\mathsf{sk}, \alpha \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}})$.
- $(\mathsf{Enc}(a), \mathsf{Enc}(\alpha \cdot a)) \leftarrow \mathsf{E}_{\mathsf{sk}}(a)$: A Paillier ciphertext of a scalar $x \in \mathbb{Z}_{\mathcal{N}}$ is $E = (1+\mathcal{N})^x \cdot r^{\mathcal{N}} \bmod \mathcal{N}^2$, for some random $r \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}}$. An encoding is thus $4 \log \mathcal{N}$ bit-long.

This is an additively-homomorphic encryption scheme over $\mathbb{Z}_{\mathcal{N}}$, but in order to get the L-linear homomorphism over $\mathbb{Z}_q$, we will need $\mathcal{N} > Lq^2$. The linear-only extractability is reasonable to assume, as this scheme is not known to be multiplicatively homomorphic when $\mathcal{N}$ is hard to factor. Hence, for a quadratic check, the verifier decrypts the ciphertexts modulo $\mathcal{N}$, which is the value in $\mathbb{N}$, and one can then check the quadratic relation in $\mathbb{Z}_q$ on the cleartexts.

We stress that for our commitments to be hiding, we will need to add random masks to the plaintexts: a random integer in $\mathcal{M} = [\![0; 2^\lambda L q^2]\!]$, for a security parameter λ, will statistically hide the plaintext x. More details are provided in Appendix B. For correct decryption and for the soundness of the $\mathsf{ZKLQCheck}$ proofs on μ secret scalars, we will need $\mathcal{N} > 2\mu L \cdot 2^\lambda q^3$. As μ will never be larger than 4, we will need $\mathcal{N} > L \cdot 2^{\lambda+3} q^3$.

For our concrete instantiations, in Section 5, we take $\mathcal{N}$ between 2560 and 3072 bits, which for an RSA modulus is consistent with a security parameter λ between 80 and 128.

3 Commitments

A major contribution of this paper is the construction of commitments of multivariate polynomials over rings so that succinct proofs can later be described. This is in the same vein as in [FNP20], but the latter is only defined for secure encodings based in pairings, whereas we describe here the construction from any security encodings. While not publicly verifiable anymore, this will be useful to build compact commitments of polynomials over $\mathbb{Z}_q$, where q may contain small prime factors (but still larger than the degree of the polynomials), for a designated verifier. These commitments may not hide the polynomials but just guarantee the binding property, with a unique opening. We thereafter show how to also make them hiding, so that no information leaks about the committed polynomials (when needed), with zero-knowledge arguments for the relations the polynomials may satisfy, for a designated verifier.

3.1 Binding Commitments

Because of the linear-only combinations, one can limit encodings in sub-spaces. As we can also do quadratic verifications, we will be able to check products of two polynomials. From an encoding scheme $(\mathsf{Gen}, \mathsf{E})$ over a ring R, we can define a compact binding commitment scheme over multivariate polynomials. More precisely, such commitment schemes will be defined by 4 algorithms:

- $\mathsf{Setup}(1^\lambda, \mathcal{R}, (\mathcal{R}_\pi)_\pi)$ generates the public key pk and the verification key vk, according to the polynomial space $\mathcal{R}$, and the authorized subspaces $(\mathcal{R}_\pi)_\pi$;
- $\mathsf{Commit}_{\mathsf{pk}}(\mathsf{u}, \mathcal{R}_\pi)$, for a polynomial $\mathsf{u} \in \mathcal{R}_\pi \subseteq \mathcal{R}$, outputs a commitment C of u;
- $\mathsf{Validity}_{\mathsf{vk}}(C, \mathcal{R}_\pi)$, for a commitment C, verifies whether this is a valid commitment for an (unknown) polynomial u in the appropriate subspace $\mathcal{R}_\pi$;
- $\mathsf{Quadratic}_{\mathsf{vk}}(Q, C_1, \ldots, C_\ell)$, for valid commitments C_i of (unknown) polynomials u_i and a quadratic polynomial Q in ℓ variables, verifies whether $Q(\mathsf{u}_1, \ldots, \mathsf{u}_\ell) = 0$ in $\mathcal{R}$.

For the above definition to make sense, for any adversary $\mathcal{A}$, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ such that for any valid commitment C generated by $\mathcal{A}$, $\mathsf{Ext}_{\mathcal{A}}$ outputs the committed polynomial $\mathsf{u} \in \mathcal{R}_\pi$.

While $\mathcal{R}$ will usually be a global ring of polynomials, such as $\mathsf{R}[X]$ or $\mathsf{R}[X, Y]$, the sub-spaces $\mathcal{R}_\pi$ will only be the module generated by a limited basis, $\mathsf{R}[X^{n-1}]$, $\mathsf{R}[Y^N]$, or $\mathsf{R}[X^{n-1}, Y^N]$, where the explicit exponents limit the degrees.

A Warm-Up with the Ring of Polynomials $\mathcal{R} = \mathsf{R}[X]$. Before dealing with multivariate polynomials, we start with univariate polynomials, to illustrate some requirements and some issues:

- $\mathsf{Setup}(1^\lambda, \mathcal{R} = \mathsf{R}[X], (\mathcal{R}_1 = \mathsf{R}[X^{n-1}]))$ first runs $(\mathsf{pk}', \mathsf{sk}', \mathsf{vk}') \leftarrow \mathsf{Gen}(1^\lambda)$, chooses a random element $s \xleftarrow{\$} \mathsf{R}^*$ and, for $i \in [\![0; n-1]\!]$, sets $E_i \leftarrow \mathsf{E}_{\mathsf{sk}'}(s^i)$. Then, the public key of the commitment scheme is $\mathsf{pk} = (\mathsf{pk}', \{E_i\}_i)$, while the verification key is $\mathsf{vk} = \mathsf{vk}'$;
- $\mathsf{Commit}_{\mathsf{pk}}(\mathsf{u}, \mathcal{R}_1)$, for a polynomial $\mathsf{u} = \sum_{i=0}^{n-1} u_i X^i \in \mathcal{R}_1 \subset \mathcal{R}$ of degree at most $n-1$, outputs $E = \mathsf{Eval}(\{E_i\}_i, \{u_i\}_i) = \mathsf{E}(\sum_i u_i s^i) = \mathsf{E}(\mathsf{u}(s))$;
- $\mathsf{Validity}_{\mathsf{vk}}(E)$ is exactly the Verify of the encoding scheme, as it outputs whether this is a valid encoding or not, and thus a valid commitment or not.

Thanks to the linear-only extractability, when a player generates a valid encoding (or commitment) E, being only given $(E_0, \ldots, E_{n-1})$, one can extract (c_i) such that E is an encoding of $c_0 + c_1 s + \ldots c_{n-1} s^{n-1}$ in R, and thus of the polynomial $\mathsf{c} = \sum_i c_i X^i$ in $\mathcal{R}$. Our above commitment scheme on polynomials is thus extractable under the linear-only extractability of the secure encoding scheme.

In addition, thanks to the quadratic verification on the encodings, if we have four polynomials u, v, m and r such that $\mathsf{m} = \mathsf{u}\cdot\mathsf{v} \bmod \mathsf{r}$, which means that $\mathsf{m} = \mathsf{u}\cdot\mathsf{v}+\mathsf{r}\cdot\mathsf{q}$, where all the polynomials are of degree at most $n-1$, we can check such a product: from valid commitments U and V of u and v, R and Q of r and q, respectively, and M of the polynomial m, all of degree at most $n-1$, as they are all simple encodings, $\mathsf{QCheck}(X_1X_2 + X_3X_4 - X_5, U, V, R, Q, M) = \mathsf{true}$ implies that $\mathsf{m}(s) = \mathsf{u}(s)\cdot\mathsf{v}(s) + \mathsf{r}(s)\cdot\mathsf{q}(s)$. According to the ring R, this might imply the expected relation, or not. If R is a large enough field, this is true.

However, in the following, we will be interested in the particular case of $\mathsf{R} = \mathbb{Z}_q$, with $q = p_1 \cdot \ldots \cdot p_\ell$ a product of ℓ prime integers $p_1 < \ldots < p_\ell$. Having $\mathsf{m}(s) = \mathsf{u}(s)\cdot\mathsf{v}(s) + \mathsf{r}(s)\cdot\mathsf{q}(s) \bmod q$ while $\mathsf{m} \neq \mathsf{u}\cdot\mathsf{v} + \mathsf{r}\cdot\mathsf{q}$ in $\mathbb{Z}_q[X]$ means there is an index $j \in [\![1; \ell]\!]$ such that $\mathsf{m}(s) = \mathsf{u}(s)\cdot\mathsf{v}(s) + \mathsf{r}(s)\cdot\mathsf{q}(s) \bmod p_j$ while $\mathsf{m} \neq \mathsf{u}\cdot\mathsf{v} + \mathsf{r}\cdot\mathsf{q}$ in $\mathbb{Z}_{p_j}[X]$. Under the Schwartz-Zippel lemma [Sch80,Zip79], this probability is bounded by $2n/p_j$ for each j, as the total degree of the relation is at most $2n$. Hence, the probability over s to have a false positive is bounded by $2n\ell/t$. This probability is unfortunately non-negligible for polynomial prime factors. According to the expected soundness parameter ε, to reduce the probability of false positive cases, the natural solution is to iterate K times, with multiple evaluation points s_k, for $k \in [\![1; K]\!]$, so that $(2n\ell/t)^K \leq \varepsilon$. But then, we have to make sure the same polynomials are committed in each point. We now directly deal with bivariate polynomials, and this could be extended to multivariate polynomials, but notations would become hard to follow.

3.2 Commitments on Bivariate Polynomials

We now build commitments on bivariate polynomials over $\mathbb{Z}_q$ for a composite q, providing a way to deal with scalars and univariate polynomials, when they are evaluated in one-fixed coordinate or a fixed point. In addition, a bivariate polynomial is a way to encode multiple univariate polynomials: given N polynomials $\mathsf{u}_j = \sum_i u_{j,i} X^i \in \mathbb{Z}_q[X]$, of degree $n-1$, we can consider the polynomial in $\mathbb{Z}_q[X, Y]$: $\mathsf{u}(X, Y) = \sum_j Y^j \mathsf{u}_j(X) = \sum_{j,i} u_{j,i} X^i Y^j$.

As explained above, in order to reduce the probability of errors with the Schwartz-Zippel lemma, we will use encodings in K multiple points. We will additionally prove they all encode the same polynomial.

$\mathsf{Setup}(1^\lambda, \mathcal{R} = \mathbb{Z}_q[X, Y], \mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N])$ first runs $(\mathsf{pk}', \mathsf{sk}', \mathsf{vk}') \leftarrow \mathsf{Gen}(1^\lambda)$, chooses K tuples of random elements $s_k, t_k \xleftarrow{\$} \mathbb{Z}_q^*$ and, for $k \in [\![1; K]\!]$, $i \in [\![0; n-1]\!]$, and $j \in [\![0; N]\!]$, sets $E_{k,j,i} \leftarrow \mathsf{E}_{\mathsf{sk}'}(s_k^i \cdot t_k^j)$. To explicitly limit to some degrees, such as $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$ or even to univariate polynomials. One also chooses K tuples of random elements $r_k^{(1)}, r_k^{(2)} \xleftarrow{\$} \mathbb{Z}_q^*$. Then, for $k \in [\![1; K]\!]$, $i \in [\![0; n-1]\!]$, and $j \in [\![0; N]\!]$, one sets $E^{(2)}_{k,j,i} \leftarrow \mathsf{E}_{\mathsf{sk}'}(r_k^{(2)} \cdot s_k^i \cdot t_k^j)$ for bivariate polynomials in $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$. The public key of the commitment scheme is composed of the above encodings $\mathsf{pk} = (\mathsf{pk}', \{E_{k,j,i}, E^{(2)}_{k,j,i}\}_{k,j,i})$, and the verification key is $\mathsf{vk} = \mathsf{vk}'$. For our application, other sub-spaces will be added, but in this section, we focus on $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$. More spaces are detailed in Appendix C, with other random values $r_k^{(\iota)}$.

Commit($\mathsf{u}, \mathbb{Z}_q[X^{n-1}, Y^N]$), for a polynomial $\mathsf{u} = \sum_{j=0}^{N} \sum_{i=0}^{n-1} u_{j,i} X^i Y^j$ in $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$, outputs $C = (E_{\mathsf{u}} = (E_k, E_k^{(2)})_k, \Pi_{\mathsf{u}})$, for $k \in [\![1; K]\!]$, with Π_{u} detailed later, where

$$E_k \leftarrow \mathsf{Eval}(\{E_{k,j,i}\}_{j,i}, \{u_{j,i}\}_{j,i}) = \mathsf{E}(\mathsf{u}(s_k, t_k))$$
$$E_k^{(2)} \leftarrow \mathsf{Eval}(\{E_{k,j,i}^{(2)}\}_{j,i}, \{u_{j,i}\}_{j,i}) = \mathsf{E}(r_k^{(2)} \cdot \mathsf{u}(s_k, t_k))$$
$$\text{such that } \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E_k^{(2)}, E_k, E_{k,0,0}^{(2)}) = \mathsf{true}$$

The idea behind the twin encodings $E_{\mathsf{u}} = (E_k, E_k^{(2)})_k$ is that the first element E_k is compatible between all the polynomials in $\mathcal{R} = \mathbb{Z}_q[X, Y]$, independently of the constraints on the allowed monomials, while the second element restrains the polynomial space: the limited basis in $\{E_{k,j,i}^{(2)}\}_{j,i}$ and the relation with $E_{k,0,0}^{(2)}$ limits to $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$. Note there is still a non-negligible probability of false positives when accepting a twin encoding, as remarked above, as the quadratic check might accept the relation on the specific points without the relation holding on the polynomials. We will say an encoding E_k is *valid* when the relation really holds on the extracted polynomials from the twin encodings $(E_k, E_k^{(2)})$. Intuitively, for each index k, the probability of false positive (accepting an invalid twin encoding) is bounded by $\ell\mathcal{D}/t$, where $\mathcal{D}$ will be the maximal total degree of the polynomials that one will be able to generate from the elements in the basis provided as input, as $E_{k,0,0}^{(2)}$ encodes a polynomial of degree 0. An additional proof Π_{u} (either provided from an interactive protocol or built in a non-interactive way) is thus appended to the commitment to make the validity check sound, with error-probability ε_c for each commitment: a huge fraction of the encodings are valid and encode the same polynomial.

In this section, we target the soundness of the commitments, whereas our ultimate goal will be a soundness bound ε_s for our global proof. To achieve this soundness bound, we will first require all the commitments to be correct, excepted an error probability $\varepsilon_s/3$, the relations between the commitments to also hold excepted with error probability $\varepsilon_s/3$, and the bounds on the noise-flooding to be small enough with error probability $\varepsilon_s/3$. Hence, we use a specific soundness parameter $\varepsilon_c \leq \varepsilon_s/3\nu_c$ for commitments, where ν_c will be the total number of commitments.

Validity(C) first verifies the twin encodings, which checks the appropriate sub-spaces for each E_k: the quadratic check with the limited bases in $\{E_{k,j,i}^{(2)}\}_{k,j,i}$ guarantees the limited list of monomials, but with an error probability bounded by $2\ell\mathcal{D}/t$. Note that in this section, $\mathcal{D} = N + n - 1$, because of the limited $\mathcal{R}_2$, but monomials with higher degrees will be required later. With K large enough, we can expect a large number of valid encodings E_k: let us assume more than $K/5$ encodings are not valid, this will remain undetected with probability at most $(\ell\mathcal{D}/t)^{K/5}$. If we impose $t \geq 2^S \times 2\ell\mathcal{D}$ (where S will be seen as a security margin, all along this analysis), the error probability is less than $2^{-(S+1)K/5}$. Hence, the number of valid encodings E_k is greater than $4K/5$ excepted with probability upper-bounded by $2^{-(S+1)K/5}$.

But this is not enough to amplify the above Schwartz-Zippel lemma: one also has to make sure that all the valid E_k encode the same polynomial u to exploit the iterations in the quadratic check between encoded polynomials:

$$\begin{aligned}\mathsf{u}_k(X, Y) - \mathsf{u}(X', Y') &= \mathsf{u}(X, Y) - \mathsf{u}(X', Y') \\ &= \mathsf{u}(X, Y) - \mathsf{u}(X, Y') + \mathsf{u}(X, Y') - \mathsf{u}(X', Y') \\ &= (Y - Y') \cdot \mathsf{v}(X, Y, Y') + (X - X') \cdot \mathsf{w}(X, X', Y').\end{aligned}$$

This can be checked with random $x_m, y_m \xleftarrow{\$} \mathbb{Z}_q$, sent by the verifier:

$$\mathsf{u}(X, Y) - \mathsf{u}(x_m, y_m) = (Y - y_m) \cdot \mathsf{v}_m(X, Y) + (X - x_m) \cdot \mathsf{w}_m(X),$$

from the proof

$$\Pi_{\mathsf{u}} = (u_m \leftarrow \mathsf{u}(x_m, y_m), (V_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(s_k, t_k)), W_{k,m} \leftarrow \mathsf{E}(\mathsf{w}_m(s_k)))_k)_m$$

that can be checked as

$$\mathsf{QCheck}(X_1 - u_m - (X_2 - y_m) \cdot X_3 - (X_4 - x_m) \cdot X_5,$$
$$E_k, E_{k,1,0}, V_{k,m}, E_{k,0,1}, W_{k,m}) = \mathsf{true}$$

For each $k \in [\![1; K]\!]$, if the above relations do not hold at the polynomial level (which means for the polynomials $\mathsf{v}_{k,m}$, $\mathsf{w}_{k,m}$, encoded in $V_{k,m}$, and $W_{k,m}$ respectively, possibly in any sub-space) for more than $1/3$ of the $m \in [\![1; M]\!]$, they will remain undetected with probability at most $(\ell(\mathcal{D}+1)/t)^{M/3} \leq 2^{-(S+1)M/3}$. Otherwise

$$\mathsf{u}_k(X, Y) - u_m = (Y - y_m) \cdot \mathsf{v}_{k,m} + (X - x_m) \cdot \mathsf{w}_{k,m}$$

for more than $2M/3$ indices m. Then, for any $k' \neq k$, that correspond to valid encodings E_k and $E_{k'}$, at least $M/3$ common values (x_m, y_m) satisfy both relations in k and k', hence $\mathsf{u}_k(x_m, y_m) = \mathsf{u}_{k'}(x_m, y_m) = u_m$. As a consequence, as the polynomials u_k and $\mathsf{u}_{k'}$ were committed before seeing (x_m, y_m), there are $M/3$ random points in which the two polynomials are equal. Then $\mathsf{u}_k = \mathsf{u}_{k'}$, excepted with probability upper-bounded by $(\ell\mathcal{D}/t)^{M/3} \leq 2^{-2M}$. A false acceptance for some pair (k, k') of consecutive valid encodings is upper-bounded by $2K \cdot 2^{-(S+1)M/3}$. Globally, the probability of having a false positive among the valid encodings E_k is bounded by $2K \cdot 2^{-(S+1)M/3}$.

We thus complete the commitments in $\mathcal{R}_2$: in addition to the twin encodings $E_{\mathsf{u}} = (E_k, E_k^{(2)})_k$, after having received (or seen) $(x_m, y_m)_m$, one completes the commitment with the proofs Π_{u}, for $k \in [\![1; K]\!]$, $m \in [\![1; M]\!]$

– in $\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[X^{n-1}, Y^N])$, $\Pi_{\mathsf{u}} = (u_m, (V_{k,m}, W_{k,m})_k)_m$;

Then, we can state the following security result, which can be extended to other subspaces:

Theorem 5 (Knowledge-Soundness of Commitments in $\mathcal{R}_2$). *For any commitment $C = (E = (E_k, E_k^{(2)})_k, \Pi = (u_m, (V_{k,m}, W_{k,m})_k)_m)$, if C successfully passes all the validity checks and quadratic root checks, on randomly chosen $(x_m, y_m) \xleftarrow{\$} \mathbb{Z}_q^2$, there exists a polynomial $\mathsf{u} \in \mathcal{R}_2$ such that at least $4K/5$ of the twin encodings actually encode u (which can be extracted from the extractability of valid encodings), excepted with probability less than $2^{-(S+1)K/5} + 2K \cdot 2^{-(S+1)M/3}$, where $q = \prod_{i=1}^{\ell} p_i$, with all $p_i \geq 2^S \times 2\ell\mathcal{D}$, and $\mathcal{D}$ is the maximal degree of the polynomials in the subspace.*

Quadratic Root Checks. According to the above analysis, if we assume $t \geq 2^S \times 2\ell\mathcal{D}$, for all the accepted commitments, we know that we have at least $4K/5$ valid encodings of the same polynomials in all twin encodings:

– if the number of invalid encodings is greater than $K/5$, they will remain undetected with probability less than $2^{-(S+1)K/5}$.
– all these valid encodings contain the same polynomial u, excepted with probability less than $2K \cdot 2^{-(S+1)M/3}$.

As a consequence, when all the verifications succeed for the commitment, there are at least $4K/5$ valid encodings on the same polynomial u, excepted with small error probability. Hence, when 4 commitments are involved in a quadratic check, at least $K/5$ common indices k correspond to valid encodings on the same polynomials in the 4 commitments. Those polynomials then satisfy the relation excepted with probability less than $(2\ell\mathcal{D}/t)^{K/5} \leq 2^{-S \cdot K/5}$.

More Parameters. Commitments in $\mathbb{Z}_q[X^{n-1}]$ thus consist of $2K + KM = K(M+2)$ encodings, and M scalars in $\mathbb{Z}_q$ while commitments in $\mathbb{Z}_q[X^{n-1}, Y^N]$ consist of $2K + 2KM = 2K(M+1)$ encodings, and M scalars in $\mathbb{Z}_q$.

The above analysis was for commitments that appear in quadratic checks involving $c = 4$ commitments, hence one required $4K/5$ valid encodings. When other values of c in $\{2, 3, 4\}$, this is enough to have $cK/(c+1)$ valid encodings in the c commitments to have $K/(c+1)$ common indices, and then the soundness of the equation is $2^{-S \cdot K/(c+1)}$. More detail is provided in Appendix C.

Corollary 6. *When all the tests pass in a quadratic check between at most c prover-generated commitments, that are all valid, the relation is really satisfied on the committed polynomials (which can be extracted from the extractability of valid encodings), excepted with probability less than* $2^{-S\cdot K/(c+1)}$.

Bounds on K and M are detailed in Appendix F.

3.3 Hiding Commitments

These commitments with encodings are strongly binding, because of the knowledge-soundness, but are not hiding, as sometimes expected from commitments. Because of the quadratic verification, if the verifier hesitates between two polynomials in a commitment C, they can commit them and do a simple linear verification, as they know vk.

To statistically hide the content, one has to add random blinding elements from the appropriate masking set $\mathcal{M}$ to every encodings, to get hiding encodings. We illustrate this here with with $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$. In Appendix C, we also detail the case of $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$.

Commit*$(\mathsf{u}, \mathbb{Z}_q[\boldsymbol{X^{n-1}, Y^N}])$, the hiding commitment for a bivariate polynomial $\mathsf{u}(X,Y) = \sum_{j=0}^{N}\sum_{i=0}^{n-1} u_{j,i}X^iY^j \in \mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$, outputs the tuple $C^* = (E^*_{\mathsf{u}} = (E^*_k, E^{(2*)}_k, \pi_k)_k, \Pi^*_{\mathsf{u}})$, where for all indices $k \in [\![1; K]\!]$ and $m \in [\![1; M]\!]$, with $\rho_k, \rho'_k \xleftarrow{\$} \mathcal{M}$:

$$\begin{aligned} E^*_k &\leftarrow \mathsf{Eval}(\{E_{k,j,i}\}_{j,i}, \{E_{k,0,0}\}, \{u_{j,i}\}_{j,i}, \{\rho_k\}) = \mathsf{E}(\mathsf{u}(s_k, t_k) + \rho_k)\\ E^{(2*)}_k &\leftarrow \mathsf{Eval}(\{E^{(2)}_{k,j,i}\}_{j,i}, \{E_{k,0,0}\}, \{u_{j,i}\}_{j,i}, \{\rho'_k\}) = \mathsf{E}(r^{(2)}_k \cdot \mathsf{u}(s_k, t_k) + \rho'_k)\\ \text{with } \pi_k &= \{\mathsf{ZKLQCheck}(X_1 - X_2 \cdot X_3, E^{(2*)}_k, E^*_k, E^{(2)}_{k,0,0}; E_{k,0,0}, E^{(2)}_{k,0,0}) = \mathsf{true}\} \end{aligned}$$

as the quadratique relation is equal to $\rho'_k \times 1 - \rho_k \times r^{(2)}_k$, with private scalars ρ_k and ρ'_k, and a proof, using random $x_m, y_m \xleftarrow{\$} \mathbb{Z}_q$ sent by the verifier, of

$$\begin{aligned} \mathsf{u}(X,Y) - \mathsf{u}(x_m, y_m) &= (Y - y_m)\cdot \mathsf{v}_m(X,Y) + (X - x_m)\cdot \mathsf{w}_m(X)\\ &= (Y - y_m)\cdot \mathsf{v}_m + (X - x_m)\cdot \mathsf{w}_m \end{aligned}$$

which can be verified with

$$\Pi^*_{\mathsf{u}} = (V^*_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(s_k, t_k) + \rho_{k,m}), W^*_{k,m} \leftarrow \mathsf{E}(\mathsf{w}_m(s_k) + \rho'_{k,m}), \pi_{k,m})_{k,m}$$

for random $\rho_{k,m}, \rho'_{k,m} \xleftarrow{\$} \mathcal{M}$, chosen by the prover for their privacy, and

$$\begin{aligned} \pi_{k,m} = \{&\mathsf{ZKLQCheck}(X_1 - (X_2 - y_m)\cdot X_3 - (X_4 - x_m)\cdot X_5,\\ &E^*_k, E_{k,1,0}, V^*_{k,m}, E_{k,0,1}, W^*_{k,m}; E_{k,0,0}, E_{k,0,0}, E^{(m)}_{k,1,0}, E^{(m)}_{k,0,1}) = \mathsf{true}\} \end{aligned}$$

where anybody can compute

$$\begin{aligned} E^{(m)}_{k,1,0} &= \mathsf{Eval}(\{E_{k,1,0}, E_{k,0,0}\}, \{1, -y_m\}) = \mathsf{E}(t_k - y_m)\\ E^{(m)}_{k,0,1} &= \mathsf{Eval}(\{E_{k,0,1}, E_{k,0,0}\}, \{1, -x_m\}) = \mathsf{E}(s_k - x_m) \end{aligned}$$

as the quadratic relation is equal to $\rho_k + \mathsf{u}(x_m, y_m) - \rho_{k,m}(t_k - y_m) - \rho'_{k,m}(s_k - x_m)$. One thus proves their knowledge of the 4 private scalars, $\rho_k \in \mathcal{M}$ (the same as above), $u_m = \mathsf{u}(x_m, y_m) \in \mathsf{R}$ (the same for all the k's), and $\rho_{k,m}, \rho'_{k,m} \in \mathcal{M}$.

Let us assume that a reasonable fraction of the twin encodings $(E^*_k, E^{(2*)}_k)$ are valid for a polynomial u^*_k in $\mathbb{Z}_q[X^{n-1}, Y^N]$. For each k, the quadratic check guarantees that

$$\begin{aligned} \mathsf{u}^*_k(X,Y) = \mathsf{u}_k(X,Y) + \rho_k = (Y - y_m)\cdot \mathsf{v}_{k,m} + (X - x_m)\cdot \mathsf{w}_{k,m}\\ + \rho_k + u_m - \rho_{k,m}\cdot(Y - y_m) - \rho'_{k,m}\cdot(X - x_m) \end{aligned}$$

excepted with the same error probability as in Theorem 5. On the other hand, one can state:

Theorem 7 (Hiding Property of Commitments in $\mathcal{R}_2$). *For any commitment $C^* = (E^* = (E_k^*, E_k^{(2*)}, \pi_k)_k, \Pi^* = (V_{k,m}^*, W_{k,m}^*, \pi_{k,m})_{k,m})$, thanks to the random masks, and the zero-knowledge proofs, all the encodings are statistically indistinguishable from random encodings.*

Note that for a hiding commitment, we have zero-knowledge proofs on $K(M+1)$ equations involving globally $2K(M+1)+M$ private scalars (but at most 4 in each equation). In Appendix B.3, we detail the zero-knowledge proofs when the encoding relies on the Paillier's encryption scheme: soundness is $2^{-\lambda'}$, for $\lambda' < \log_2 t$, then one needs to iterate $-\log_2(\varepsilon_s/3\nu_c)/\lambda'$ times. Each proof consists of $K(M+1)$ Paillier ciphertexts and $2K(M+1)+M$ scalars in $\mathbb{Z}_q$, iterated $\log(3\nu_c/\varepsilon_s)/\log t$ times. Soundness also implies the RSA modulus needs to be larger than $L \cdot 2^{\lambda+3} q^3$.

3.4 Quadratic Root Check

The check $\mathsf{Quadratic}(Q, C_1, \ldots, C_\ell)$, on non-hiding commitments, for a quadratic polynomial Q, that verifies whether the committed polynomials P_i's in the C_i's satisfy the relation $Q(P_1, \ldots, P_\ell) = 0$, is performed as a QCheck on the encodings for each index k. Since a huge fraction of the indices k encode the same polynomials, for the prover-generated commitments, iterations amplify the soundness.

When some hiding commitments are involved, one additionally has to prove the appropriate blinding factors. More concretely, let us consider the binding commitment D of d, and the hiding commitments C^* and H^* of c^* and h^* respectively, with blinding factors $\rho_k, \sigma_k \in \mathcal{M}$ respectively. The relation $\mathsf{c}^* = \mathsf{d} \times \mathsf{h}^*$ can be checked as:

$$\begin{aligned}&\mathsf{Quadratic}(X_1 - X_2 \cdot X_3, C^*, D, H^*)\\ &\qquad = \mathsf{ZKLQCheck}(X_1 - X_2 \cdot X_3, C_k^*, D_k, H_k^*; E_{k,0,0}, D_k) \text{ for all } k\end{aligned}$$

as the quadratic relation is equal to $\rho_k \cdot 1 - \sigma_k \cdot \mathsf{d}$, where ρ_k and σ_k are the same private values as the ones used in the validity verification of the hiding commitments C^* and H^*.

We stress that we only allow quadratic verifications where at most one polynomial is committed in a hiding way in each quadratic product. In the end, we know that the quadratic equations among polynomials are satisfied in at least $K/5$ random points. They are thus satisfied by the polynomials excepted with probability less than $2^{-S \cdot K/5}$, which is chosen to be much less than $\varepsilon_s/3\nu_e$, where ν_e is the global number of equations to be checked.

3.5 Complete Construction of the Commitment

As already explained, in our application, we are considering $q = p_1 \cdot \ldots \cdot p_\ell$, a composite modulus q, with ℓ prime factors $p_1 < \ldots < p_\ell$. We will work in subspaces of $\mathcal{R} = \mathbb{Z}_q[X^{n-1}, Y^{2N}]$. We will consider the subspaces $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$, $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$, $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, and $\mathbb{Z}_q[Y^{2N}]$ with no term in Y^N, denoted $\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N \setminus N}]$, hence $\mathcal{D} = 2N$, so we will assume $t \geq 2^S \times 4N\ell$. Details on the complete construction of the commitments with system setup and detailed contents and checks are given in Appendix C.

4 Verifiable Inner Product

We now have the tools to allow the receiver/verifier with their private input message m to learn in a verifiable way the inner product of the vector $\Phi(\mathsf{m}) = (\phi(\mathsf{m}, 0), \phi(\mathsf{m}, 1), \ldots, \phi(\mathsf{m}, N))$ with a private vector $\mathsf{F} = (\mathsf{f}_j)_{j \in [\![0;N]\!]}$, committed by the sender/prover, where ϕ is a function known by them both, depending on the application. If $\phi(\mathsf{m}, j) = \mathsf{m}^j$, the inner product corresponds to the Oblivious Polynomial Evaluation (OPE) of the polynomial F with coefficients $(\mathsf{f}_j)_j$ on the message m; if $\phi(\mathsf{m}, j) = \delta_{\mu,j}$, with δ the Kronecker symbol and $\mu \in [\![0; N]\!]$ such that m is the representation of μ, it provides the μ-th coefficient of f, which would coincide with a Symmetric Private Information Retrieval (SPIR) application. We now show how the sender can provide this evaluation with fully homomorphic encryption in a provable way.

More precisely, we consider the above FV scheme that encrypts messages from $\mathcal{R}_t = \mathbb{Z}_t[X]/\mathsf{r}(X)$, where $\mathsf{r} = X^n + 1$, into $\mathcal{R}_q = \mathbb{Z}_q[X]/\mathsf{r}(X)$ for $q = p_1 \times \ldots \times p_\ell$, with $2^S \times 4N\ell \leq t = p_1 < \ldots < p_\ell$.

The receiver encrypts the input $\mathsf{m} \in \mathcal{R}_t$ under their own key, and sends $\mathsf{Enc}(\mathsf{m}) = (\mathsf{c}, \mathsf{c}') \in \mathcal{R}_q^2$, in order to get back the homomorphic inner product of $\Phi(\mathsf{m}) = (\phi(\mathsf{m}, j))_{j \in [\![0;N]\!]} \in \mathcal{R}_t^{N+1}$ with $\mathsf{F} = (\mathsf{f}_j)_{j \in [\![0;N]\!]} \in \mathcal{R}_t^{N+1}$, committed by the sender, with a proof of correct value $(\mathsf{d}, \mathsf{d}') = \mathsf{Enc}(\langle \Phi(\mathsf{m}), \mathsf{F} \rangle) \in \mathcal{R}_q^2$, that thereafter decrypts to $\langle \Phi(\mathsf{m}), \mathsf{F} \rangle \in \mathcal{R}_t$, using the receiver's decryption key.

From $(\mathsf{c}, \mathsf{c}')$, and possible additional intermediate ciphertexts, sent by the receiver, the sender is able to compute $(\mathsf{u}_j, \mathsf{u}'_j) = \mathsf{Enc}(\phi(\mathsf{m}, j))$, for $j \in [\![0; N]\!]$, in any way they want, using the homomorphic properties of the encryption scheme:

$$(\mathsf{u}_j, \mathsf{u}'_j) = \mathsf{Enc}(\phi(\mathsf{m}, j)) = \left(\sum_{i=0}^{n} u_{j,i} \cdot X^i, \sum_{i=0}^{n} u'_{j,i} \cdot X^i \right) \in \mathcal{R}_q^2.$$

Thereafter, the goal is to evaluate the inner products:

$$\begin{aligned}(\mathsf{d}, \mathsf{d}') = \mathsf{Enc}(\langle \Phi(\mathsf{m}), \mathsf{F} \rangle) &= \mathsf{Enc}\left(\sum_{j=0}^{N} \mathsf{f}_j \cdot \phi(\mathsf{m}, j) \right) = \sum_{j=0}^{N} \mathsf{f}_j \cdot \mathsf{Enc}(\phi(\mathsf{m}, j)) \\ &= \sum_{j=0}^{N} \mathsf{f}_j \cdot (\mathsf{u}_j, \mathsf{u}'_j) = \left(\sum_{j=0}^{N} \mathsf{f}_j \cdot \mathsf{u}_j, \sum_{j=0}^{N} \mathsf{f}_j \cdot \mathsf{u}'_j \right).\end{aligned}$$

The sender can add noise to hide F in the evaluation and prove the correct evaluation of $\tilde{\mathsf{d}} = \mathsf{d} + \mathsf{z}^*$ and $\tilde{\mathsf{d}}' = \mathsf{d}' + \mathsf{z}'^*$ so that $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$ decrypts to $\langle \Phi(\mathsf{m}), \mathsf{F} \rangle$. The pair $(\mathsf{z}^*, \mathsf{z}'^*)$ will be committed in a hidden way, with a proof of correct noise, in the appropriate range, to ensure the correct decryption.

This verifiable evaluation will be done in two steps: first, the sender will prove the correct evaluation of all the $(\mathsf{u}_j, \mathsf{u}'_j)$, while committed in a very compact way. Then, a proof of the inner product is provided. We will later explain how everything remains succinct.

For the following proof and analysis to hold, we need the property, in the particular case where $L = N + 1$:

Definition 8 ((L, d)-$\mathcal{R}_t$-Linear-Homomorphism). *For any $\mathsf{a}_i, \mathsf{u}_i \in \mathcal{R}_t$, for $i = 1, \ldots, L$,*

$$\mathsf{Dec}\left(\sum_{i=1}^{L} \mathsf{a}_i \cdot (\mathsf{c}_i, \mathsf{c}'_i) \right) = \sum_{i=1}^{L} \mathsf{a}_i \cdot \mathsf{u}_i \in \mathcal{R}_t$$

where $(\mathsf{c}_i, \mathsf{c}'_i) \in \mathcal{R}_q^2$ is an encryption of u_i obtained from a circuit of multiplicative depth bounded by d.

4.1 Commitment of F

One can first note that ring elements in $\mathcal{R}_t$ and $\mathcal{R}_q$ are polynomials of degree at most $n - 1$, and can be encoded into $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$. The sender's vector $\mathsf{F} \in \mathcal{R}_t^{N+1}$ can be committed using the polynomial $\mathsf{f} = \sum_{i=0}^{N} \mathsf{f}_j \cdot Y^j$ in $\mathcal{R}_t[Y]$, where $\mathsf{f}_j = \sum_{i=0}^{n-1} f_{j,i} \cdot X^i$, can be committed in a hidden way in $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$ as $\mathsf{f}^* = \sum_{j=0}^{N} \sum_{i=0}^{n-1} f_{j,i} \cdot X^i Y^{N-j}$ (with reverse order coefficients) into F^*.

The sender can also compute and publish the noisy inner products $\tilde{\mathsf{d}}$ and $\tilde{\mathsf{d}}'$ in $\mathcal{R}_q$, the expected ciphertext of the result, for some private $\mathsf{q}^*, \mathsf{q}'^*, \mathsf{z}^*, \mathsf{z}'^* \in \mathcal{R}_q$:

$$\tilde{\mathsf{d}} = \sum_{j=0}^{N} \mathsf{f}_j \cdot \mathsf{u}_j + \mathsf{z}^* - \mathsf{q}^* \cdot \mathsf{r} \qquad \tilde{\mathsf{d}}' = \sum_{j=0}^{N} \mathsf{f}_j \cdot \mathsf{u}'_j + \mathsf{z}'^* - \mathsf{q}'^* \cdot \mathsf{r}.$$

The polynomials $\mathsf{z}^*, \mathsf{z}'^*, \mathsf{q}^*, \mathsf{q}'^*$ are committed in a hidden way in Z^*, Z'^*, Q^*, Q'^*, in $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$. The inner products must be proven, which is the goal of the second part, after we have proven the correct computation of the $(\mathsf{u}_j, \mathsf{u}'_j)$'s, as both together will prove correctness of the ciphertext $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$.

4.2 Validity of the $(\mathbf{u}_j, \mathbf{u}'_j)$'s

We thus start by proving the validity of the encryptions of the $\phi(\mathsf{m}, j)$. From the ciphertexts $(\mathsf{u}_j, \mathsf{u}'_j)$ that the prover computed on their own, they then define the polynomials:

$$\mathsf{u}(X,Y) = \sum_{j=0}^{N} \mathsf{u}_j(X) \cdot Y^j = \sum_{j=0}^{N} \sum_{i=0}^{n-1} u_{j,i} X^i Y^j \qquad \mathsf{u}'(X,Y) = \sum_{j=0}^{N} \mathsf{u}'_j(X) \cdot Y^j$$

and commit them from the sub-space $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$ into U and U' respectively. Our first step is to prove that the polynomials committed in U and U' by the sender indeed satisfy, for any exponents $j \in [\![0; N]\!]$, $(\mathsf{u}_j, \mathsf{u}'_j) = \mathsf{Enc}(\phi(\mathsf{m}, j))$, or more precisely that the decryptions $\mathsf{Dec}(\mathsf{u}_j, \mathsf{u}'_j)$ that we denote $\varphi_{\mathsf{m},j}$ are indeed $\phi(\mathsf{m}, j) \in \mathcal{R}_t$, for m initially encrypted by the verifier in $(\mathsf{c}, \mathsf{c}')$.

The verifier first chooses and sends a random polynomial $\mathsf{n} \xleftarrow{\$} \mathcal{R}_t$ (or it can be generated by a hash function on the previous data to remove interaction). Both parties can compute, $\mathsf{n}_j = \mathsf{n}^j = \sum_{i=0}^{n-1} n_{j,i} \cdot X^i$ in $\mathcal{R}_t$, for all $j \in [\![0; N]\!]$, and commit them in $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$ as $\mathsf{t} = \sum_{j=0}^{N} \sum_{i=0}^{n-1} n_{j,i} \cdot X^i Y^{N-j}$ (with reverse order coefficients) into T. Then, the prover computes and sends $(\mathsf{b}, \mathsf{b}') = \sum_{j=0}^{N} \mathsf{n}_j \cdot (\mathsf{u}_j, \mathsf{u}'_j)$ in $\mathcal{R}_q^2$.

Since this is symmetric, we now focus on the first component (without $'$), and similar analysis will have to be done on the second component (with $'$): the prover also computes and sends l so that $\sum_{j=0}^{N} \mathsf{n}_j \cdot \mathsf{u}_j = \mathsf{b} + \mathsf{l} \cdot \mathsf{r}$, that they both commit as L in $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$.

The verifier chooses and sends a sequence of random scalars $\beta_\kappa \xleftarrow{\$} \mathbb{Z}_q$, for $\kappa \in [\![1; \Lambda]\!]$. They will define $v_{\kappa,j} = \mathsf{u}_j(\beta_\kappa) \in \mathbb{Z}_q$ for $j \in [\![0; N]\!]$. We thus have the polynomial:

$$\mathsf{v}_\kappa(Y) = \sum_{j=0}^{N} v_{\kappa,j} \cdot Y^j = \sum_{j=0}^{N} \mathsf{u}_j(\beta_\kappa) \cdot Y^j = \sum_{j=0}^{N} \sum_{i=0}^{n-1} u_{j,i} \beta_\kappa^i \cdot Y^j = \mathsf{u}(\beta_\kappa, Y)$$

that can be committed as V_κ in $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, and proven correct with the quadratic check

$$\mathsf{u}(X,Y) - \mathsf{v}_\kappa(Y) = \mathsf{u}(X,Y) - \mathsf{u}(\beta_\kappa, Y) = (X - \beta_\kappa) \cdot \mathsf{w}_\kappa(X,Y) \tag{1}$$

on the commitment W_κ of $\mathsf{w}_\kappa \in \mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$ and the public commitment C_κ of $X - \beta_\kappa$, and thus only 3 prover-generated commitments (U, V_κ, and W_κ, as C_κ is public). With $\mathsf{t}_\kappa = \mathsf{t}(\beta_\kappa, Y) = \sum_{j=0}^{N} \mathsf{n}_j(\beta_\kappa) \cdot Y^j$ publicly computed and committed in T_κ, we can note that

$$\begin{aligned} \mathsf{v}_\kappa(Y) \cdot \mathsf{t}_\kappa(Y) &= \sum_{0 \le i,j \le N} v_{\kappa,i} \cdot \mathsf{n}_j(\beta_\kappa) \cdot Y^{N+i-j} \\ &= \sum_{0 \le i \le N} v_{\kappa,i} \cdot \mathsf{n}_i(\beta_\kappa) \cdot Y^N + \sum_{0 \le i \ne j \le N} v_{\kappa,i} \cdot \mathsf{n}_j(\beta_\kappa) \cdot Y^{N+i-j} \end{aligned}$$

and

$$\sum_{0 \le i \le N} v_{\kappa,i} \cdot \mathsf{n}_i(\beta_\kappa) = \sum_{0 \le i \le N} \mathsf{u}_i(\beta_\kappa) \cdot \mathsf{n}_i(\beta_\kappa) = b_\kappa + l_\kappa \cdot r_\kappa$$

where $b_\kappa = \mathsf{b}(\beta_\kappa)$, $l_\kappa = \mathsf{l}_\kappa(\beta_\kappa)$, and $r_\kappa = \mathsf{r}(\beta_\kappa)$ can be publicly computed in $\mathbb{Z}_q$, so we can check:

$$\mathsf{v}_\kappa(Y) \cdot \mathsf{t}_\kappa(Y) - (b_\kappa + l_\kappa \cdot r_\kappa) \cdot Y^N = \sum_{0 \le i \ne j \le N} v_{\kappa,i} \cdot \mathsf{n}_j(\beta_\kappa) \cdot Y^{N+i-j} = \mathsf{y}_\kappa(Y) \tag{2}$$

with a commitment Y_κ of y_κ in $\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N\backslash N}]$ and C_N a public commitment of Y^N, which makes only 2 prover-generated commitments (V_κ and Y_κ).

We stress that because of the repetitions in each commitment (up to K, according to the kind of relations they are involved in, and the number of prover-generated commitments), we know (see Corollary 6) that when all the tests pass, the above equations (1) and (2) are all satisfied with error probability less than $2\Lambda \cdot 2^{-(S+1)K/4}$.

They altogether prove that, for each $\kappa \in [\![1;\Lambda]\!]$, $b_\kappa + l_\kappa \cdot r_\kappa$ is the coefficient of Y^N in $\mathsf{v}_\kappa(Y) \cdot \mathsf{t}_\kappa(Y)$, and so $\sum_j \mathsf{u}_j(\beta_\kappa) \cdot \mathsf{n}_j(\beta_\kappa) = \mathsf{b}(\beta_\kappa) + \mathsf{l}(\beta_\kappa) \cdot \mathsf{r}(\beta_\kappa) \bmod q$ for the random β_κ, on polynomials committed beforehand, of degree n. Hence, $\sum_j \mathsf{u}_j \cdot \mathsf{n}_j = \mathsf{b} + \mathsf{l} \cdot \mathsf{r}$, excepted with error probability bounded by $2\ell n/t \le n/2^{S+1}N$, from the Schwartz-Zippel lemma.

For Λ large enough (to be set later), after all these steps, on both b and b', one gets the proof that, in $\mathcal{R}_q$,

$$\mathsf{b} = \sum_{j=0}^{N} \mathsf{n}_j \cdot \mathsf{u}_j \quad \mathsf{b}' = \sum_{j=0}^{N} \mathsf{n}_j \cdot \mathsf{u}'_j : \qquad (\mathsf{b}, \mathsf{b}') = \sum_{j=0}^{N} \mathsf{n}_j \cdot (\mathsf{u}_j, \mathsf{u}'_j).$$

These relations hold for both b and b' excepted with error probability bounded by $2 \times (2\Lambda \cdot 2^{-(S+1)K/4} + (n/2^{S+1}N)^\Lambda)$.

The verifier now checks that $\mathsf{Dec}(\mathsf{b}, \mathsf{b}') = \sum_{j=0}^{N} \mathsf{n}^j \cdot \phi(\mathsf{m}, j)$, which thus leads to $\sum_{j=0}^{N} \mathsf{n}^j \cdot \varphi_{\mathsf{m},j} = \sum_{j=0}^{N} \mathsf{n}^j \cdot \phi(\mathsf{m}, j)$ in $\mathcal{R}_t$. We can thus consider the polynomial $\sum_j (\varphi_{\mathsf{m},j} - \phi(\mathsf{m}, j)) \cdot Y^j$ in $\mathcal{R}_t$, which evaluates to zero in the random point $\mathsf{n} \xleftarrow{\$} \mathcal{R}_t$. Note that $\mathcal{R}_t$ is a ring that is the product of large fields only: $\mathsf{r} = X^{2^k} + 1$ is not irreducible in any $\mathbb{Z}_p[X]$ for a prime p, but for well-chosen prime, all the factors have large degrees: according to [BGM93], with $t + 1 = 2^a(2p + 1)$, for any integer p, $a \ge 2$, and $n = 2^k$, then all the factors have degree 2^{k+1-a}. If one chooses $t = 3 \bmod 8$, $a = 2$, and there are just two irreducible factors of degree $2^{k-1} = n/2$: the above polynomial thus has all zero coefficients by the Schwartz-Zippel lemma, excepted with probability $2 \times N/t^{n/2}$, as the polynomial is of degree N and n is randomly chosen among $t^{n/2}$ possible values in each of the two fields. Hence, $\varphi_{\mathsf{m},j} = \phi(\mathsf{m}, j)$ in $\mathcal{R}_t$, excepted with probability bounded by $2N/t^{n/2}$, which is clearly negligible.

4.3 Validity of $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$

As above, we focus on $\tilde{\mathsf{d}}$ with respect to all $(\mathsf{u}_j)_j$ and $\mathsf{z}^*, \mathsf{q}^*$. The same should be done for $\tilde{\mathsf{d}}'$ with respect to all $(\mathsf{u}'_j)_j$ and $\mathsf{z}'^*, \mathsf{q}'^*$, but for the same $(\mathsf{f}_j)_j$ and z, r:

$$\tilde{\mathsf{d}} = \sum_{j=0}^{N} \mathsf{f}_j \cdot \mathsf{u}_j + \mathsf{z}^* - \mathsf{q}^* \cdot \mathsf{r}.$$

This following analysis is quite similar to the previous one, considering f^*, $(\mathsf{z}^*, \mathsf{q}^*)$, $\tilde{\mathsf{d}}$ instead of $\mathsf{t}, \mathsf{p}, \mathsf{b}$, and with hidden intermediate values, as $(\mathsf{f}_j)_j$ is private to the prover, contrarily to n which was publicly chosen by the verifier. One first proves the quadratic relation on the hidden commitment of $\mathsf{g}^*_\kappa(Y) = \mathsf{f}^*(\beta_\kappa, Y) = \sum_{j=0}^{N} g_{\kappa,j} \cdot Y^{N-j}$ in $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, where $g_{\kappa,j} = \mathsf{f}_j(\beta_\kappa)$, and the hiding commitment of $\mathsf{h}^*_\kappa(X, Y)$ in $\mathcal{R}_2 = \mathbb{Z}_q[X^n, Y^N]$, so that

$$\mathsf{f}^*(X, Y) - \mathsf{g}^*_\kappa(Y) = \mathsf{f}^*(X, Y) - \mathsf{f}^*(\beta_\kappa, Y) = (X - \beta_\kappa) \cdot \mathsf{h}^*_\kappa(X, Y) \tag{3}$$

with only 3 prover-generated commitments. And similarly, one proves in a hidden way:

$$\mathsf{q}^*(X) - q^*_\kappa = \mathsf{q}^*(X) - \mathsf{q}^*(\beta_\kappa) = (X - \beta_\kappa) \cdot \mathsf{q}^*_\kappa(X) \tag{4}$$

$$\mathsf{z}^*(X) - z^*_\kappa = \mathsf{z}^*(X) - \mathsf{z}^*(\beta_\kappa) = (X - \beta_\kappa) \cdot \mathsf{z}^*_\kappa(X) \tag{5}$$

#variables	Binding or Hiding	Polynomials	#commitments
$1v$	B	$\mathsf{v}_\kappa, \mathsf{v}'_\kappa, \mathsf{y}_\kappa, \mathsf{y}'_\kappa$	6Λ
	H	$\mathsf{q}^*, \mathsf{q}'^*, \mathsf{q}^*_\kappa, \mathsf{q}'^*_\kappa, \mathsf{u}^*_k, \mathsf{e}^*_{1,k}, \mathsf{e}^*_{2,k}, \mathsf{z}^*_\kappa, \mathsf{z}'^*_\kappa, \mathsf{g}^*_\kappa$	$5\Lambda + 2 + 3Z$
$2v$	B	$\mathsf{u}, \mathsf{u}', \mathsf{w}_\kappa, \mathsf{w}'_\kappa$	$2(\Lambda + 1)$
	H	$\mathsf{f}^*, \mathsf{h}^*_\kappa$	$\Lambda + 1$
Total			$14\Lambda + 5 + 3Z$

Fig. 1. Number of Commitments in the Global Proof, and Size in Number of Encodings, with $\Lambda = \lceil(\log_2(24/\varepsilon_s))/(S + 1 + \log_2(N/n))\rceil$

with the private $q^*_\kappa = \mathsf{q}^*_\kappa(\beta_\kappa)$ and $z^*_\kappa = \mathsf{z}^*_\kappa(\beta_\kappa)$ in $\mathbb{Z}_q$ that will be also used below. There are only 2 prover-generated commitments in each relation. We can note

$$\begin{aligned}\mathsf{v}_\kappa(Y) \cdot \mathsf{g}^*_\kappa(Y) &= \sum_{0\le i,j\le N} v_{\kappa,i} \cdot g_{\kappa,j} \cdot Y^{N+i-j} \\ &= \sum_{0\le i\le N} v_{\kappa,i} \cdot g_{\kappa,i} \cdot Y^N + \sum_{0\le i\ne j\le N} v_{\kappa,i} \cdot g_{\kappa,j} \cdot Y^{N+i-j}\end{aligned}$$

and

$$\begin{aligned}\sum_{0\le i\le N} v_{\kappa,i} \cdot g_{\kappa,i} &= \sum_{0\le i\le N} \mathsf{u}_i(\beta_\kappa) \cdot \mathsf{f}_i(\beta_\kappa) \qquad\qquad = \tilde{\mathsf{d}}(\beta_\kappa) - \mathsf{z}^*(\beta_\kappa) + \mathsf{q}^*(\beta_\kappa) \cdot \mathsf{r}(\beta_\kappa) \\ &= \tilde{d}_\kappa - z^*_\kappa + q^*_\kappa \cdot r_\kappa\end{aligned}$$

where $\tilde{d}_\kappa = \tilde{\mathsf{d}}(\beta_\kappa)$ and $r_\kappa = \mathsf{r}(\beta_\kappa)$ can be publicly computed, and so one proves

$$\mathsf{v}_\kappa(Y) \cdot \mathsf{g}^*_\kappa(Y) - (\tilde{d}_\kappa - z^*_\kappa + q^*_\kappa \cdot r_\kappa) \cdot Y^N = \mathsf{s}_\kappa(Y) \tag{6}$$

with a commitment of s_κ in $\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N\setminus N}]$ and a public commitment of Y^N.

Again, from Corollary 6, when all the tests pass, the above equations (3), (4), (5), and (6) are all satisfied with error probability less than $3\Lambda \cdot 2^{-(S+1)K/4}$.

They altogether prove that, for each $\kappa \in [\![1; \Lambda]\!]$, $d_\kappa - z^*_\kappa + q^*_\kappa \cdot r_\kappa$ is the coefficient of Y^N in $\mathsf{v}_\kappa(Y) \cdot \mathsf{g}^*_\kappa(Y)$, so

$$\tilde{\mathsf{d}}(\beta_\kappa) = \sum_j \mathsf{u}_j(\beta_\kappa) \cdot \mathsf{f}_j(\beta_\kappa) + \mathsf{z}^*(\beta_\kappa) - \mathsf{q}^*(\beta_\kappa) \cdot \mathsf{r}(\beta_\kappa) \bmod q$$

for the random β_κ, on polynomials committed beforehand. Hence,

$$\tilde{\mathsf{d}} = \sum_j \mathsf{u}_j \cdot \mathsf{f}_j + \mathsf{z}^* - \mathsf{q}^* \cdot \mathsf{r},$$

excepted with error probability bounded by $2\ell n/t \le n/2^{S+1}N$, for each κ. Again, these relations hold for both $\tilde{\mathsf{d}}$ and $\tilde{\mathsf{d}}'$ excepted with error probability bounded by $2 \times (3\Lambda \cdot 2^{-(S+1)K/4} + (n/2^{S+1}N)^\Lambda)$.

Globally, the error probability is bounded by $2\times(5\Lambda\cdot 2^{-(S+1)K/4}+2\times(n/2^{S+1}N)^\Lambda)$. We thus need to take $\Lambda = \lceil \log_2(24/\varepsilon_s)/\log_2(2^{S+1}N/n)\rceil$ different values for β_κ so that $4\times(n/2^{S+1}N)^\Lambda \le \varepsilon_s/6$, and $K > 4\log_2(60\Lambda/\varepsilon_s)/(S+1)$ so that $2 \times 5\Lambda \cdot 2^{-(S+1)K/4} \le \varepsilon_s/6$ too, so that the global error on the equations be bounded by $\varepsilon_s/3$.

On the other hand, as shown on Figure 1, the global number of commitments is bounded by $14\Lambda+5+3Z$. We explain the $3Z$ commitments in the next section to build $(\mathsf{z}^*, \mathsf{z}'^*)$. We thus need to take $\varepsilon_c \le \varepsilon_s/(3 \times \nu_c) = \varepsilon_s/(3(14\Lambda + 5 + 3Z))$.

Indeed, after all these steps, on both $\tilde{\mathsf{d}}$ and $\tilde{\mathsf{d}}'$, one gets the proof that, in $\mathcal{R}_q$, $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$ is a ciphertext of $\langle \Phi(\mathsf{m}), \mathsf{F}\rangle$, if the noise $(\mathsf{z}^*, \mathsf{z}'*)$ is correct.

4.4 Proofs on the Noise $(\mathsf{z}^*, \mathsf{z}'^*)$

We now have to prove that the polynomials $(\mathsf{z}^*, \mathsf{z}'^*)$ are indeed of the form

$$(\mathsf{z}^*, \mathsf{z}'^*) = \left(\mathsf{p} \cdot \mathsf{u} + \mathsf{e}_1 \bmod q, \mathsf{p}' \cdot \mathsf{u} + \mathsf{e}_2 \bmod q\right), \tag{7}$$

with coefficients of $\mathsf{u}, \mathsf{e}_1, \mathsf{e}_2$ small enough for the correctness of the decryption. The sender needs to make them follow the appropriate distribution for their own privacy. Using noise-flooding, the sender will thus choose $\mathsf{u}, \mathsf{e}_1, \mathsf{e}_2$, commit them in a hidden way, and prove their relationship with $(\mathsf{z}^*, \mathsf{z}'^*)$, as a simple public linear combination of commitments.

Then, the sender has to prove the bounds on the coefficients of $\mathsf{u}, \mathsf{e}_1, \mathsf{e}_2$. Let us focus on u (the same should be done for e_1 and e_2), with the generation of multiple random polynomials r_k, for $k = 1 \ldots, Z$, according to a Gaussian distribution on $\mathbb{Z}$ with standard deviation σ'. For random scalars $\delta_k \xleftarrow{\$} [\![-\eta; \eta]\!]$, one asks to see the linear combination $\mathsf{r} = \sum_k \delta_k \mathsf{r}_k$, with coefficients $r_i = \sum_k \delta_k r_{k,i}$ that follow a Gaussian distribution on $\mathbb{Z}$ with standard deviation $\sigma' \cdot \sum_k \delta_k$. They should thus be smaller than $10 \cdot \sigma' \cdot \sum_k \delta_k$ (excepted with probability 2^{-128}). One can thus check that. When true, this at least proves that $|r_{k,i}| \leq 10 \cdot \sigma' \cdot \sum_k \delta_k \leq 10Z \cdot \sigma'\eta$, with overwhelming probability. We can thus define $\mathsf{u} = \sum_k \delta'_k \mathsf{r}_k$, for random scalars $\delta'_k \xleftarrow{\$} \{-1, 1\}$. We then know that $\|\mathsf{u}\|_\infty \leq 10Z^2 \cdot \sigma'\eta$. We will then have to consider this bound for the correctness of the FHE decryption, the bound B'' on the final ciphertext error.

Bounds are explained in detail in Appendix D. To reduce the probability of having false negatives, θ linear combinations will be published. In the end we find that with η of the order of $(1/\varepsilon_s)^{1/5}$, we have $\theta = 5$. We can then take $Z = 8$, $\sigma' = 2^\lambda B'/80$, and $q \approx 2^{\lambda+3}\eta \times 2tB'(2nB+1)$. This increases the initial size of q by $\lambda + 3 + \log \eta$ more bits. With $\lambda = 128$, $\varepsilon_s = 2^{-128}$, and $\eta = 2^{21}$, this is 152 more bits.

4.5 Protocol Overview

Here are all the steps in the interactive version of the protocol, that can be run in 5 rounds:

1. The public information for the encoding and the FHE are provided by the receiver to the sender;
2. The **receiver** encrypts their message $\mathsf{m} \in \mathcal{R}_t$ with possible additional intermediate ciphertexts. They send these ciphertexts;
3. The **sender** commits their vector F, as f^*;
4. The sender computes all the $\mathsf{Enc}(\phi(\mathsf{m}, i)) = (\mathsf{u}_i, \mathsf{u}'_i)$, for $i \in [\![0; N-1]\!]$ from the ciphertexts sent by the receiver, and sends a compact commitment (U, U');
5. The sender generates but sends hiding commitments of $(\mathsf{u}^*_k, \mathsf{e}^*_{1,k}, \mathsf{e}^*_{2,k})$;
6. from a hash on these commitments, both parties can generate the scalars $\delta'_k \in \{-1, +1\}$. The sender can generate $(\mathsf{z}^*, \mathsf{z}'^*) = \sum_k \delta'_k(\mathsf{p} \cdot \mathsf{u}^*_k + \mathsf{e}^*_{1,k}, \mathsf{p}' \cdot \mathsf{u}^*_k + \mathsf{e}^*_{2,k} \mod q)$, and send the resulting ciphertext $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$. The receiver can derive the commitments (Z^*, Z'^*) on $(\mathsf{z}^*, \mathsf{z}'^*)$;
7. The **receiver** can ask for (or derive from a hash) scalars $\delta_k \in [\![-\eta; \eta]\!]$, θ times, and the **sender** sends the receiver the $\sum_k \delta_k(\mathsf{p} \cdot \mathsf{u}^*_k + \mathsf{e}^*_{1,k} \mod q, \mathsf{p}' \cdot \mathsf{u}^*_k + \mathsf{e}^*_{2,k} \mod q)$ and the aggregation of the random values for the related hidden commitment, so they can check the coefficients are small enough;
8. from a hash on the (U, U'), both parties generate a challenge $\mathsf{n} \in \mathcal{R}_t$: the sender uses it to generate and send $(\mathsf{b}, \mathsf{b}')$ and $(\mathsf{l}, \mathsf{l}')$;
9. The **receiver** can ask for (or derive from a hash) $\{\beta_\kappa \in \mathbb{Z}_q\}$, for $\kappa \in [\![1; \Lambda]\!]$;
10. The **sender** generates $\mathsf{v}_\kappa, \mathsf{w}_\kappa, \mathsf{y}_\kappa, \mathsf{s}_\kappa$, as well as $\mathsf{g}^*_\kappa, \mathsf{h}^*_\kappa, \mathsf{q}^*_\kappa, \mathsf{z}^*_\kappa, \mathsf{z}_\kappa$, and the twin polynomials with $'$, and sends the twin encodings of these polynomials to the receiver;
11. The **receiver** can ask for (or derive from a hash) $\{(x_m, y_m) \in \mathbb{Z}_q^2\}$, for $m \in [\![0; M_{\max}]\!]$;
12. The **sender** generates the proofs of all their previous commitments, and the zero-knowledge proofs on the quadratic relations, with additional random challenges either from the **receiver** or derived from a hash, and the **sender** sends the answers to the receiver;

13. The receiver verifies all the proofs with respect to the twin encodings and the required quadratic checks to prove the correctness of $(\mathsf{b}, \mathsf{b}')$ and $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$.

In the above protocol, some of the scalars can either be sent by the verifier or taken from a hash function, using the Fiat-Shamir paradigm [FS87].

In the latter setting, the challenges $(\delta_k)_k$, $(\beta_\kappa)_\kappa$ and $(x_m, y_m)_m$, and challenges in the zero-knowledge proofs are derived from a hash function on the values that should not be modified by the prover. This leads to a non-interactive protocol. When they are generated by the verifier, leading to a 5-round protocol, the soundness can be reduced to 2^{-30}, as the prover should then cheat on-line.

4.6 Security Claims

The security of such 2PC protocols is two-folds, with privacy and correctness. The former guarantees that no information leaks about the parties' inputs beyond the obtained result, and the latter ensures the result is correct.

Honest-but-Curious Security. In the honest-but-curious case described in definitions 1 and 2, where the players play honestly, thanks to the correctness of the FHE scheme, the result obtained after decryption by the receiver is the correct one.

The receiver's privacy is ensured as they send only ciphertexts of their input message: under the semantic security of the FHE, no information leaks about m.

The sender's privacy is also ensured thanks to the hidden commitment on their vector F, that is statistically hiding, but also with the noise-flooding in the final ciphertext $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$ that hides the evaluated circuit too, and thus the vector F, but only reveals $\langle \mathsf{F}, \Phi(\mathsf{m}) \rangle$: the sender's privacy is statistical.

Security against a Malicious Sender. The goal of our work was to enhance security a against malicious sender, which can only impact the correctness, as the sender does not receive any information: thanks to the additional proofs, the receiver has a guarantee of correct output $(\tilde{\mathsf{d}}, \tilde{\mathsf{d}}')$ and correct decryption.

Contributions. During all the descriptions, we have shown the correctness and soundness of the commitments and proofs. Soundness differs for interactive proofs (with $\varepsilon_s = 2^{-30}$) and non-interactive proofs (with $\varepsilon_s = 2^{-\lambda}$).

Knowledge-soundness comes from the linear-only extractability of the encoding scheme: the above soundness proves the extracted polynomials are valid witnesses, and satisfy all the equations.

Thanks to the hiding commitments and the zero-knowledge proofs, the above global proof is also zero-knowledge from the prover's point of view: no information leaks on their private inputs.

Succinctness. Beside the security, this is our main goal: the proof essentially consists of a few millions encodings, plus a similar number of zero-knowledge proofs: communication complexity only depends on the security parameter. Concrete sizes for the proofs are provided on Figure 2, but we would first like to show that our construction is asymptotically succinct in the case of OPE: we want to evaluate $\mathsf{f}(\mathsf{m})$ for a polynomial f of degree N, using the FV FHE scheme, and the above SNARK.

Ciphertexts to be sent: Let us consider any constant $d \in \mathbb{N}$, and define the basis $b = \lceil (N+1)^{1/d+1} \rceil$. Then, any $k \in [\![0; N]\!]$ can be written in basis b as $\sum_{i=0}^{d} x_i b^i$, with $x_i \in [\![0; b-1]\!]$. Hence, from the ciphertexts of all the powers $\mathsf{m}^{x_i b^i}$ for $x_i \in [\![0; b-1]\!]$ and $i \in [\![0; d]\!]$, one can compute ciphertexts of all the powers. The verifier will thus have to send $(d+1)b$ ciphertexts, where $b^d \sim N$ so about $dN^{1/d}$ ciphertexts, and the prover will be able to rebuild all the ciphertexts with circuits of maximal multiplicative depth d.

The Receiver's Privacy: As recalled in Appendix E, the semantic security of the FV scheme relies on the decision Ring-LWE problem, which is related to the SIVP problem with some approximation factor $\gamma(n)$ [PRS17]. If we take $\sigma = \alpha q = \sqrt{n}$, for the Gaussian distribution of the secret key and the noise, then we have $\gamma(n) \leq q\sqrt{\log n}$, which will be polynomial in n, in our case, as shown below.

The Sender's Privacy: In order to hide the evaluated circuit, we add some noise to the final result: in order to make the global noise statistically independent of the circuit, we add a noise 2^λ times larger than the original noise. We discuss below the original noise for the correctness, and we will then multiply it by 2^λ. The proof of correct noise does not exclude a noise that is a bit larger, which even improves the sender's privacy.

Correctness: As recalled in Appendix E, using the [LN14] noise derivation formula, the error noise growth after having evaluated a sequence of d multiplications with a fresh ciphertext is bounded by $C_1^d V + dC_1^{d-1}C_2 = C_1^{d-1} \cdot (C_1 V + dC_2)$, where:

$$C_1 \leq 11tn^{5/2} \qquad C_2 \leq 20n^{\frac{3}{2}}t^2(n+\ell) \qquad V \leq 201n^2$$

With reasonable assumptions on our parameters we find the correctness result on this noise, and also when it is greatened with the noise flooding, as detailed in Appendix E.

Communication: With theses parameters, for a fixed constant d, with $N = n^d$,

$$\log t \approx (d+1)\log n \qquad \log q \approx 3(d+1)\log t \approx 3(d+1)^2 \log n.$$

A ciphertext thus consists of $2n$ elements in $\mathbb{Z}_q$: $6(d+1)^2 n \log n$ bits each. The verifier first sends $(d+1)\lceil (N+1)^{1/d+1} \rceil \approx dn^{d/d+1}$ ciphertexts, so approximately $6d(d+1)^2 n^{2-1/d+1} \log n$ bits, which is in $\mathcal{O}(n^2 \log n) = \tilde{\mathcal{O}}(N^{2/d})$.

The size of the encoding being only linearly impacted by the size of q, we have a proof size in $\tilde{\mathcal{O}}(1)$ when N grows. This provides the succinctness of the proof.

5 Applications

The protocol described in this paper can be deployed for several applications, depending on the function ϕ defining the receiver vector $(\phi(\mathsf{m}, 0), \ldots, \phi(\mathsf{m}, N))$ from which we get the scalar product with the sender's vector. Hereafter we detail the cases of PSI (using OPE), and of SPIR.

5.1 Application to Private Set Intersection

As already explained, we can apply this technique to PSI, where the receiver owns a set $\mathcal{X} = \{x_1, \ldots, x_a\}$ of cardinality $\#\mathcal{X}$, the sender a set $\mathcal{Y} = \{y_1, \ldots, y_b\}$ of cardinality $\#\mathcal{Y}$, and the receiver wants to learn the intersection. We consider the case where $\#\mathcal{Y}$ is much larger than $\#\mathcal{X}$, and hope to get a protocol essentially linear in $\#\mathcal{X}$ only. To this aim, we follow the basic approach from [FNP04], with the polynomial $P(Y) = \prod_{i=1}^{N}(Y - \mathsf{h}_i)$ committed once for all in P^*, where the values $\mathsf{h}_i \in \mathcal{R}_t$ encode the elements y_i in the sender's set, and the degree $N = \#\mathcal{Y}$. Then the receiver wants to check whether $P(x_i) = 0$ on every input x_i.

Thanks to the verifiability of our proof, and even the knowledge-soundness of our commitments, our PSI protocol is secure against malicious senders, as they cannot use distinct polynomials as P in each execution.

The protocol can be adapted to support adding elements to the sender's set, with a new check to make sure the old sender's polynomial divides the new one which only has additional roots.

$\lvert N\rvert$	$\lvert n\rvert$	d	ℓ	S	$\lvert t\rvert$	$\lvert q\rvert$	Receiver Commun. FHE Security	Receiver Commun. Total Size	$\lvert\mathcal{N}\rvert$	Sender Communications $\varepsilon_s : 2^{-30}$	Sender Communications $;2^{-80}$	Sender Communications $;2^{-128}$
20	14	4	14	21	47	656	100	218 MB	2133	8 MB	74 MB	267 MB
25	15	5	16	16	48	753	251	636 MB	2430	13 MB	137 MB	503 MB
30	14	5	16	11	48	753	81	583 MB	2434	24 MB	271 MB	1.03 GB
30	15	5	15	20	56	839	209	1.3 GB	2693	9 MB	90 MB	324 MB
35	15	5	15	15	56	839	209	2.28 GB	2698	14 MB	149 MB	550 MB
35	15	5	14	25	66	922	179	2.51 GB	2947	6 MB	54 MB	207 MB
40	15	5	15	10	56	839	209	4.01 GB	2703	26 MB	317 MB	1.2 GB

Fig. 2. Parameters and Security Bounds for the Fully Homomorphic Encryption (FHE) and the Proof, with $t \geq 2^S \cdot 4N\ell$. FHE size encompasses all the ciphertexts sent by an honest-but-curious receiver, and the proof sizes columns encompass all the elements the sender (with enforced honest behavior) sends back, including cleartexts. $|x|$ is the bit-length of x.

5.2 Application to Symmetric Private Information Retrieval

In the Symmetric PIR setting, the sender owns a set $\mathcal{Y} = \{y_1, \ldots, y_N\}$ of cardinality N, and the receiver wants to retrieve an element of the sender's set with a private index μ. To this aim, the sender defines F as a vector of representations of their set elements in $\mathcal{R}_t$, the receiver a representation $\mathsf{m} = \tau(\mu)$ of their index μ in $\mathcal{R}_t$ (optimizations could be made using a smaller space than $\mathcal{R}_t$), and we apply our scheme with $\phi(\mathsf{m}, j) = \delta_{\mu,j}$, where δ is the Kroenecker symbol. Writing μ and j in the basis b with $\mu = \sum_{k=0}^{d} \mu_k b^k$ and $j = \sum_{k=0}^{d} j_k b^k$, where the $\mu_k, j_k \in [\![0; b-1]\!]$:

$$\phi(\tau(\mu), j) = \delta_{\mu,j} = \delta_{\sum_{k=0}^{d} \mu_k b^k, \sum_{k=0}^{d} j_k b^k} = \prod_{k=0}^{d} \delta_{\mu_k, j_k} = \prod_{k=0}^{d} \phi(\tau(\mu_k), j_k)$$

So sending the encryptions of $\psi(\mathsf{m}, k, i) = \phi(\tau(\mu_i), k) = \delta_{\mu_i,k}$ for $i \in [\![0; d]\!], k \in [\![0; b-1]\!]$ allows the sender to calculate all the required ciphertexts with circuits of multiplicative depth d. Because $b^d \sim N$, the receiver sends $(d+1)b \approx dN^{1/d}$ ciphertexts. This is the same number of ciphertexts as in the PSI application.

This can also be adapted to growing sender databases, adding new elements as coefficients of the sender polynomial and making sure the difference of the new one and the old one only has monomials of a specified degree range.

5.3 Experiments

Parameters and Sizes. Concrete sizes are provided in Figure 2, for one execution of our OPE protocol on a polynomial of degree N. Security bounds on the privacy of FHE use the LWE estimator [APS15], for N between 2^{20} and 2^{40}, for a variable security margin $S \geq 6$ in $t \geq 2^S \times 2\ell\mathcal{D}$. Indeed, the size of the proof depends a lot on Λ. The larger N/n is, the shorter the proof, but S also has a huge impact, as shown on Figure 2, when S makes Λ decrease, with the number of encodings. However, when S grows, so does t, which can lower FHE security and make the FHE ciphertexts heavier. In Figure 2, parameter sets either in favor of lowering the weight of the receiver or the sender's communications are given for a particular N.

In this figure, the ϕ parameter was not toggled, as our choice seemed a good compromise between FHE ciphertexts size and proof sizes, but of course it could be.

One can note these results are quite practical, in particular with the interactive version, where only a few MBytes have to be sent for the proof, which is much less than for the FHE ciphertexts, even for high security levels. A python script to test more parameters is provided in Appendix F, along with a description of all the optimizations performed in this script.

References

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

BCFK20. Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. Cryptology ePrint Archive, Report 2020/1526, 2020. https://eprint.iacr.org/2020/1526.

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

BEHZ16. Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 423–442. Springer, Heidelberg, August 2016.

BGM93. Ian F. Blake, Shuhong Gao, and Ronald C. Mullin. Explicit factorization of $x^{2^k} + 1$ over $\mathbb{F}_p$ with prime $p = 3 \bmod 4$. *Appl. Algebra Eng. Commun. Comput.*, 4:89–94, 1993.

BN00. Daniel Bleichenbacher and Phong Q. Nguyen. Noisy polynomial interpolation and noisy Chinese remaindering. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 53–69. Springer, Heidelberg, May 2000.

CHLR18. Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018.

CL01. Yan-Cheng Chang and Chi-Jen Lu. Oblivious polynomial evaluation and oblivious neural learning. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 369–384. Springer, Heidelberg, December 2001.

CLR17. Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

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.

FNP04. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

FNP20. Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Heidelberg, May 2020.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.

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.

Gil99. Niv Gilboa. Two party RSA key generation. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 116–129. Springer, Heidelberg, August 1999.

GNN17. Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. Cryptology ePrint Archive, Report 2017/409, 2017. http://eprint.iacr.org/2017/409.

Haz18. Carmit Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. *Journal of Cryptology*, 31(2):537–586, April 2018.

HL09. Carmit Hazay and Yehuda Lindell. Efficient oblivious polynomial evaluation with simulation-based security. Cryptology ePrint Archive, Report 2009/459, 2009. http://eprint.iacr.org/2009/459.

HN12. Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. *Journal of Cryptology*, 25(3):383–433, July 2012.

HT10. Carmit Hazay and Tomas Toft. Computationally secure pattern matching in the presence of malicious adversaries. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 195–212. Springer, Heidelberg, December 2010.

JL09. Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, Heidelberg, March 2009.

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.

LN14. Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 318–335. Springer, Heidelberg, May 2014.

LP00. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 36–54. Springer, Heidelberg, August 2000.

NP99. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *31st ACM STOC*, pages 245–254. ACM Press, May 1999.

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.

PRS17. Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 461–473. ACM Press, June 2017.

PRTY20. Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

Sch80. Jack T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of ACM*, 27(4):701–717, 1980.

SYY99. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC1. In *40th FOCS*, pages 554–567. IEEE Computer Society Press, October 1999.

TP11. Jonathan T. Trostle and Andy Parrish. Efficient computationally private information retrieval from anonymity or trapdoor groups. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC 2010*, volume 6531 of *LNCS*, pages 114–128. Springer, Heidelberg, October 2011.

Ver11. Damien Vergnaud. Efficient and secure generalized pattern matching via fast Fourier transform. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 41–58. Springer, Heidelberg, July 2011.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

ZB05. Huafei Zhu and Feng Bao. Augmented oblivious polynomial evaluation protocol and its applications. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 222–230. Springer, Heidelberg, September 2005.

Zip79. Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79*, volume 72 of *LNCS*, pages 216–226. Springer, 1979.

Supplementary Material

A SNARKs and zk-SNARKs

Let us recall the formal definition of (zero-knowledge) succinct non-interactive arguments of knowledge (zk-SNARKs), for a Boolean relation $\mathcal{R}$ on words or statements u and witnesses w, of an NP language.

Definition 9 (SNARK). *A SNARK is defined by three algorithms,*

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

$\mathsf{Prove}(\mathsf{crs}, u, w)$**:** *given a common reference string crs, an instance u and a witness w such that $\mathcal{R}(u, w) = \mathsf{true}$, this algorithm produces a proof π;*

$\mathsf{Ver}(\mathsf{crs}, u, \pi)$**:** *on input the common reference string crs, an instance u, and a proof π, the verifier algorithm outputs false (reject) or true (accept);*

satisfying the following properties:

Correctness. *For all valid statement u (i.e., it exists a witness w such that $\mathcal{R}(u, w) = \mathsf{true}$),*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, u, \pi) = \mathsf{false} \,\middle|\, \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{R}), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, u, w)\right] = \mathsf{negl}(\lambda);$$

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. *for the class $\mathcal{Z}$ of auxiliary input generators: for any PPT adversary $\mathcal{A}^{\mathsf{KS}}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ such that:*

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

Intuitively, this means that for any prover able to produce a valid proof for a statement in the language, there exists an efficient extractor that outputs a witness for the given statement;

Zero Knowledge. *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:*

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

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

B Encodings

In this section we illustrate the definition of secure encodings with two distinct constructions. the first provides public verifiability whereas the second will be for designated verifier only.

B.1 Encodings over $\mathbf{R} = \mathbb{Z}_q$ from Pairings

The Knowledge of Exponent Assumption, introduced by Damgard [Dam92] states that given g and g^α in a group $\mathbb{G}$, it is hard to create c and $\hat{c}$ in that group $\mathbb{G}$ so that $\hat{c} = c^\alpha$, without knowing a such that $c = g^a$. The only way to compute $\hat{c}$ being to generate $(g^\alpha)^a$.

In a pairing-friendly setting $\mathsf{pk} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, \mathfrak{g}, e)$, one can check the appropriate relation between c and $\hat{c}$, with $\mathfrak{g}^\alpha$. We can thus consider such a pairing-friendly setting with $q = p_1 \cdot \ldots \cdot p_\ell$ a product of ℓ prime integers $p_1 < \ldots < p_\ell$ (possibly with $\ell \neq 1$, so that $\mathbb{Z}_q$ is a ring, but not necessarily a field), we denote $G = e(g, \mathfrak{g})$. The verification key is $\mathsf{vk} = \mathfrak{g}^\alpha$ for the secret key $\alpha \xleftarrow{\$} \mathbb{Z}_q$: the encoding function $\mathsf{E_{sk}}$ is $\mathsf{E_{sk}}(a) = (g^a, g^{\alpha \cdot a}, \mathfrak{g}^a) \in \mathbb{G}_1^2 \times \mathbb{G}_2$. It is clearly L-linearly-homomorphic for any L. The bilinear map e allows public quadratic root verification, on the elements g^a and $\mathfrak{g}^a$: for example, $Q = X_1 \cdot X_2 - X_3$ on encodings of a_1, a_2 and a_3, it can be done with $e(g^{a_1}, \mathfrak{g}^{a_2}) \cdot e(g^{-a_3}, \mathfrak{g}) = e(g, \mathfrak{g})^{Q(a_1,a_2,a_3)}$. This must be done after image verification of any individual encoding with $e(g^a, \mathfrak{g}) = e(g, \mathfrak{g}^a)$ and $e(g^a, \mathsf{vk}) = e(g^{\alpha \cdot a}, \mathfrak{g})$. They are both public, as vk is public.

To hide the content of an encoding, one just needs the encoding $E' = (g, g^\alpha, \mathfrak{g})$ of 1, multiplied by a private random factor in $\mathcal{M} = \mathbb{Z}_q$. To prove the existence of μ such private coefficients c_i such that $Q(a_1, \ldots, a_L) = \sum c_i b_i \bmod q$ is satisfied, one has to prove the knowledge of $(c_i)_i$ such that

$$V = e(g, \mathfrak{g})^{Q(a_1,\ldots,a_L)} = \prod e(g^{b_i}, \mathfrak{g})^{c_i}$$

which can be done with a classical Schnorr-proof, from the encodings $\mathsf{E_{sk}}(b_i)$, as $V = e(g, \mathfrak{g})^{Q(a_1,\ldots,a_L)}$ can be computed thanks to the pairing. Using the Fiat-Shamir paradigm[FS87], this proof can be non-interactive: with random exponents $k_i \xleftarrow{\$} \mathbb{Z}_q$, the prover sets $D = \prod e(g^{b_i}, \mathfrak{g})^{k_i}$, gets a random challenge $e = \mathcal{H}(V, \{\mathsf{E_{sk}}(b_i)\}, D) \in [\![0; 2^\lambda]\!]$, with $2^\lambda < p_1$, and generates the proof $\Pi = (\{s_i = k_i - ec_i \bmod q\}, e) \in \mathbb{Z}_q^\mu \times [\![0; 2^\lambda]\!]$. The verifier can compute

$$D' = \prod e(g^{b_i}, \mathfrak{g})^{s_i} \times V^e$$

and check whether $e \stackrel{?}{=} \mathcal{H}(V, \{\mathsf{E_{sk}}(b_i)\}, D')$. Indeed, if the statement is true

$$\begin{aligned}\prod e(g^{b_i}, \mathfrak{g})^{s_i} \times V^e &= \prod e(g^{b_i}, \mathfrak{g})^{s_i} \times (\prod e(g^{b_i}, \mathfrak{g})^{c_i})^e \\ &= \prod e(g^{b_i}, \mathfrak{g})^{s_i + ec_i} = \prod e(g^{b_i}, \mathfrak{g})^{k_i} = D.\end{aligned}$$

The linear-only extractability relies on the above Knowledge of Exponent Assumption, that requires the hardness of the discrete logarithm in the bilinear generic group model, which additionally requires all prime factors p_k to be large enough (at least 2λ bits, for a security parameter λ). An encoding is a tuple of elements in $\mathcal{E} = \mathbb{G}_1^2 \times \mathbb{G}_2$, and a zero-knowledge proof of μ scalars contains μ elements from $\mathbb{Z}_q$ and the challenge in $[\![0; 2^\lambda]\!]$. If case of multiple proofs, one can use a unique global challenge e, and the same (k_i, s_i) can be used for the same private scalars c_i. Hence, globally, the size if thus μ' elements from $\mathbb{Z}_q$ where μ' is the total number of private scalars, independently of the number of equations, plus one challenge. The requirement of large prime factors will be prohibitive when used with Fully Homomorphic Encryption, as this will make quite large q, the modulus of the ciphertext space. We thus do not detail more.

B.2 Encodings over $\mathsf{R} = \mathbb{Z}_q$ from a Linear-Only Encryption Scheme

Given a linear-only encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ from $\mathbb{Z}_q$ onto $\mathcal{C}$, for any integer q, one chooses a random private element $\alpha \xleftarrow{\$} \mathbb{Z}_q^*$, to be the secret key of the encoding. Then, pk is the public key of the encryption scheme, while vk is α together with the secret decryption key of the encryption scheme. It is thus private. Then, the encoding function $\mathsf{E_{sk}}$ is $\mathsf{E_{sk}}(a) = (\mathsf{Enc}(a), \mathsf{Enc}(\alpha \cdot a)) \in \mathcal{C} \times \mathcal{C}$. It is clearly linearly-homomorphic from an additively-homomorphic encryption scheme. The decryption algorithm allows any root verification using the decryption key in vk, while the image verification tests whether the two decryption values verify the secret ratio α. The linear-only extractability depends on the specific encryption scheme: but either the encryption scheme is fully/somewhat homomorphic, or this property is satisfied. An encoding is a pair of elements in $\mathcal{E} = \mathcal{C} \times \mathcal{C}$.

More concretely, one can use Paillier encryption scheme [Pai99] with a large RSA modulus $\mathcal{N}$, the encryption of $x \in \mathbb{Z}_{\mathcal{N}}$ is $E = (1+\mathcal{N})^x \cdot r^{\mathcal{N}} \bmod \mathcal{N}^2$, for $r \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}}$. The ciphertext space is thus $\mathcal{C} = \mathbb{Z}^*_{\mathcal{N}^2}$. For the decryption, one needs the value $\varphi = \varphi(\mathcal{N})$ which is equivalent of the knowledge of the factorisation of $\mathcal{N}$: as $\varphi(\mathcal{N}^2) = \mathcal{N}\varphi$, $E^{\varphi} = (1+\mathcal{N})^{x\varphi} = 1 + x\varphi \cdot \mathcal{N} \bmod \mathcal{N}^2$. If φ is invertible modulo $\mathcal{N}$, one can extract x modulo $\mathcal{N}$.

For the encoding, the public key pk is thus the modulus $\mathcal{N}$, the secret key sk is a random element $\alpha \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}}$, and the verification key vk is the decryption key $\varphi(\mathcal{N})$ and the secret value α:

- $\mathsf{E}_{\mathsf{sk}}(a) = ((1+\mathcal{N})^a \cdot r_0^{\mathcal{N}} \bmod \mathcal{N}^2, (1+\mathcal{N})^{\alpha \cdot a} \cdot r_1^{\mathcal{N}} \bmod \mathcal{N}^2)$, for $r_0, r_1 \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}}$;
- $\mathsf{Verify}_{\mathsf{vk}}(C_0, C_1)$ first decrypts both ciphertexts using $\varphi(\mathcal{N})$ to get $x_0, x_1 \bmod \mathcal{N}$, and checks whether $x_1 = \alpha \cdot x_0 \bmod \mathcal{N}$.

An encoding is thus $4 \log \mathcal{N}$ bit-long.

We want an encoding on $\mathbb{Z}_q$, one can thus take $\mathcal{N} > q$ to encode elements $x \in [\![0; q-1]\!]$. Decoding first decrypts both ciphertexts modulo $\mathcal{N}$, with the elements in $[\![-\mathcal{N}/2; \mathcal{N}/2]\!]$, checks the relation with α modulo $\mathcal{N}$, and reduces it again modulo q in $[\![0; q-1]\!]$ to extract the encoded value in $\mathbb{Z}_q$. For further verifications (quadratic checks), one just considers the decryption of the first ciphertext in $[\![-\mathcal{N}/2; \mathcal{N}/2]\!]$, and relations among the cleartexts.

From L ciphertexts $E_i = (1+\mathcal{N})^{x_i} \cdot {r_i}^{\mathcal{N}} \bmod \mathcal{N}^2$, for $r_i \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}}$, one can compute the linear combination with coefficients smaller than q, $\prod E_i^{c_i} = (1+\mathcal{N})^{\sum c_i x_i} \cdot (\prod r_i^{c_i})^{\mathcal{N}}$, which is an encryption of $\sum c_i x_i \bmod \mathcal{N}$. It decodes to $\sum c_i x_i$ if $|\sum c_i x_i| < \mathcal{N}/2$: we thus have to take $\mathcal{N} > 2L \cdot q^2$ if the encodings are fresh encodings that encrypt elements in $[\![0; q-1]\!]$. For a quadratic check, using vk, the verifier can decrypt all the encodings modulo $\mathcal{N}$ and reduce them modulo q to check the relation modulo q. There is no more constraint on $\mathcal{N}$.

To hide the content of an encoding, even with respect to the owner of the secret key, one uses a random encoding of a random private mask in $[\![0; 2^\lambda L q^2]\!]$, to act as a statistically hiding one-time pad, furthermore randomized with $\mathcal{N}$-th powers to remove any information in the initial random coins. In this case, one needs $\mathcal{N} > 2L \cdot 2^\lambda q^2$ for correct decryption modulo $\mathcal{N}$, without wrapping around modulo N before the reduction modulo q. Hence $\mathcal{M} = [\![0; 2^\lambda L q^2]\!]$.

About the zero-knowledge verification $\mathsf{ZKLQCheck}(Q, (E_i)_i; (E'_i)_i)$, to prove the existence of μ coefficients $c_i \in \mathcal{M}$ such that $Q(a_1, \ldots, a_L) = \sum c_i b_i \bmod q$ is satisfied, on the encodings $E_1 = \mathsf{E}_{\mathsf{sk}}(a_1)$, $\ldots$, $E_L = \mathsf{E}_{\mathsf{sk}}(a_L)$, $E'_1 = \mathsf{E}_{\mathsf{sk}}(b_1)$, $\ldots$, $E'_\mu = \mathsf{E}_{\mathsf{sk}}(b_\mu)$, one has to prove the knowledge of (c_i) in $V = \mathsf{Eval}(\{E'_i\}, \{c_i\})$, for $i \in [\![1; \mu]\!]$, where the verifier knows, after decryption of the encodings $(E_i)_i$ and quadratic computations modulo q, $V' = (1+\mathcal{N})^{Q(a_1,\ldots,a_L) \bmod q} \bmod \mathcal{N}^2$. One wants to prove that V and V' decrypt to the same value modulo q: $V = V' \times (1+\mathcal{N})^{qr_1} \cdot r_0^{\mathcal{N}} \bmod \mathcal{N}^2$, but for unknown values r_0, r_1. We stress that we only consider the first ciphertext in the encodings. One thus proves the knowledge of $c_i \in \mathcal{M}$ for $i \in [\![1; \mu]\!]$ such that $V = \prod {E'_i}^{c_i} = V' \times (1+\mathcal{N})^{qr_1} \cdot r_0^{\mathcal{N}} \bmod \mathcal{N}^2$. The c_i's are the masks in $[\![0; 2^\lambda L q^2]\!]$, but one can use their q-reduction. Then, as the encodings E'_i can encrypt elements in $[\![-Lq^2; Lq^2]\!]$, $|r_1 q| \le \mu L q^3$.

For the latter zero-knowledge proof, the prover chooses random $k_i \xleftarrow{\$} \mathbb{Z}_q$, with additional noise $\nu' \xleftarrow{\$} [\![0; 2^\lambda L q^2]\!]$ and $\nu \xleftarrow{\$} \mathbb{Z}^*_N$, to hide any extra information beyond the modulo q relation, and sets $D = \prod {E'_i}^{k_i} \cdot (1+\mathcal{N})^{q\nu'} \nu^{\mathcal{N}} \bmod \mathcal{N}^2$, gets a random challenge e (possibly derived from $(Q, (E_i)_i, (E'_i)_i, D)$ in $[\![0; 2^{\lambda'} - 1]\!]$ and outputs the proof $\Pi = (D, (s_i = k_i - e c_i \bmod q)_i) \in \mathbb{Z}_{\mathcal{N}^2} \times \mathbb{Z}_q^\mu$.

For the moment, we use a different space $[\![0; 2^{\lambda'} - 1]\!]$ for the challenge, with λ' possibly smaller than λ or $-\log \varepsilon_s$, in which case $-\log \varepsilon_s / \lambda'$ parallel repetitions should be performed for correct soundness. One can check

$$e \leftarrow \mathcal{H}(Q, (E_i)_i, (E'_i)_i, D) \qquad D' \leftarrow \prod {E'_i}^{s_i} \cdot (V')^e \bmod \mathcal{N}^2 \qquad \mathsf{Dec}(D/D') \stackrel{?}{=} 0 \bmod q$$

Indeed,

$$\begin{aligned} D' &= \prod {E'_i}^{s_i} \times (V')^e = \prod {E'_i}^{s_i} \times V^e \cdot (1+\mathcal{N})^{-eqr_1} \cdot r_0^{-e\mathcal{N}} \\ &= \prod {E'_i}^{k_i - ec_i} \times \prod {E'_i}^{ec_i} \times (1+\mathcal{N})^{-eqr_1} \cdot r_0^{-e\mathcal{N}} \\ &= \prod {E'_i}^{k_i} \cdot (1+\mathcal{N})^{-eqr_1} \cdot r_0^{-e\mathcal{N}} = D \cdot (1+\mathcal{N})^{-q(er_1+\nu')} \cdot (r_0^e \nu)^{-\mathcal{N}} \bmod \mathcal{N}^2. \end{aligned}$$

One must make sure that $2eqr_1 \leq \mathcal{N}$: with $\mathcal{N} \geq 2\mu L \cdot tq^3$ (where t is the smallest prime factor of q and $2^{\lambda'} < t$), there is no reduction modulo $\mathcal{N}$ before the reduction modulo q. The zero-knowledge proof Π of μ scalars thus contains $2\log\mathcal{N} + \mu \times \log q$ bits.

In case of multiple equations involving the same secret c_i, the same challenge is used, and the same (k_i, s_i), reducing the bit-size to $2\log\mathcal{N}\times$#Equations + $\log q\times$#Secrets.

B.3 Proofs for a Hiding Commitment in $\mathcal{R}_2$

Let us illustrate on the proof of validity of a hiding commitment in $\mathcal{R}_2$, as presented in Section 3.3. $\mathsf{Commit}^*(\mathsf{u}, \mathbb{Z}_q[X^{n-1}, Y^N])$, outputs the tuple $C^* = (E^*_\mathsf{u} = (E^*_k, E^{(2*)}_k)_k, \Pi^*_\mathsf{u})$, where for all indices $k \in [\![1; K]\!]$ and $m \in [\![1; M]\!]$, with $\rho_k, \rho'_k \xleftarrow{\$} \mathcal{M}$ and $\sigma_k, \sigma'_k \in \mathbb{Z}^*_\mathcal{N}$:

$$\begin{aligned} E^*_k &\leftarrow \prod_{j,i} E_{k,j,i}^{u_{j,i}} \times E_{k,0,0}^{\rho_k} \times \sigma_k^\mathcal{N} \bmod \mathcal{N}^2 \\ E^{(2*)}_k &\leftarrow \prod_{j,i} E^{(2)\ u_{j,i}}_{k,j,i} \times E_{k,0,0}^{\rho'_k} \times {\sigma'_k}^\mathcal{N} \bmod \mathcal{N}^2 \end{aligned}$$

with $\mathsf{ZKLQCheck}(X_1 - X_2 \cdot X_3, E^{(2*)}_k, E^*_k, E^{(2)}_{k,0,0}; E_{k,0,0}, E^{(2)}_{k,0,0}) = \mathsf{true}$ for which the verifier can compute the plaintexts in $E^{(2*)}_k$ and E^*_k, and then the quadratic relation a_k, that should be $a_k = \rho'_k \times 1 - \rho_k \times r^{(2)}_k \bmod q$:

$$V'_k = (1+\mathcal{N})^{a_k} = (1+\mathcal{N})^{\rho'_k \times 1 - \rho_k \times r^{(2)}_k \bmod q} \bmod \mathcal{N}^2$$

and the encodings in the proof Π^*_u:

$$\begin{aligned} V^*_{k,m} &\leftarrow (1+\mathcal{N})^{\mathsf{v}_m(s_k,t_k)+\rho_{k,m} \bmod q} \cdot \sigma_{k,m}^\mathcal{N} \bmod \mathcal{N}^2 \\ W^*_{k,m} &\leftarrow (1+\mathcal{N})^{\mathsf{w}_m(s_k)+\rho'_{k,m} \bmod q} \cdot {\sigma'_{k,m}}^\mathcal{N} \bmod \mathcal{N}^2 \end{aligned}$$

for random $\rho_{k,m}, \rho'_{k,m} \xleftarrow{\$} \mathcal{M}$, chosen by the prover for their privacy, and some unknown $\sigma_{k,m}, \sigma'_{k,m} \in \mathbb{Z}^*_\mathcal{N}$, using random $x_m, y_m \xleftarrow{\$} \mathbb{Z}_q$ obtained by a hash function, and

$$\begin{aligned} &\mathsf{ZKLQCheck}(X_1 - (X_2 - y_m) \cdot X_3 - (X_4 - x_m) \cdot X_5, \\ &\qquad E^*_k, E_{k,1,0}, V^*_{k,m}, E_{k,0,1}, W^*_{k,m}; E_{k,0,0}, E_{k,0,0}, E^{(m)}_{k,1,0}, E^{(m)}_{k,0,1}) = \mathsf{true} \end{aligned}$$

where

$$E^{(m)}_{k,1,0} = E_{k,1,0} \cdot E_{k,0,0}^{-y_m} \bmod \mathcal{N}^2 \qquad E^{(m)}_{k,0,1} = E_{k,0,1} \cdot E_{k,0,0}^{-x_m} \bmod \mathcal{N}^2$$

Again, the verifier can compute the plaintexts in the input encodings, and the plaintext $b_{k,m}$ to be proven, that should be $b_{k,m} = \rho_k + \mathsf{u}(x_m, y_m) - \rho_{k,m}(t_k - y_m) - \rho'_{k,m}(s_k - x_m) \bmod q$. They build $V'_{k,m}$:

$$V'_{k,m} = (1+\mathcal{N})^{b_{k,m}} = (1+\mathcal{N})^{\rho_k + \mathsf{u}(x_m,y_m) - \rho_{k,m}(t_k - y_m) - \rho'_{k,m}(s_k - x_m) \bmod q} \bmod \mathcal{N}^2$$

As the same ρ_k and the same $u_m = \mathsf{u}(x_m, y_m)$ are used many times, the prover first randomly chooses $T_k, T'_k, v_m, T_{k,m}, T'_{k,m} \xleftarrow{\$} [\![0; q-1]\!]$, $\nu'_k, \nu'_{k,m} \xleftarrow{\$} [\![0; 2^\lambda Lq^2 - 1]\!]$, $\nu_k, \nu_{k,m} \xleftarrow{\$} \mathbb{Z}^*_{\mathcal{N}}$ and sets

$$D_k = E^{T'_k}_{k,0,0} \cdot (E^{(2)}_{k,0,0})^{-T_k} (1+\mathcal{N})^{q\nu'_k} \nu_k^{\mathcal{N}} \bmod \mathcal{N}^2$$
$$D_{k,m} = E^{T_k}_{k,0,0} \times E^{v_m}_{k,0,0} \times {E^{(m)}_{k,1,0}}^{-T_{k,m}} \times {E^{(m)}_{k,0,1}}^{-T'_{k,m}} (1+\mathcal{N})^{q\nu'_{k,m}} \nu_{k,m}^{\mathcal{N}} \bmod \mathcal{N}^2$$

The huge range for $\nu'_k, \nu'_{k,m} \xleftarrow{\$} [\![0; 2^\lambda Lq^2 - 1]\!]$ is to hide the random values even if then encodings come from L-linear combinations (which is not the case in this specific proof, but will be for the relation (6), with v_κ).

A challenge $e \in [\![0; 2^{\lambda'} - 1]\!]$ is provided by the verifier or drawn from a hash function evaluated on the statement to be proven and all the $(D_k)_k$ and $(D_{k,m})_{k,m}$, and the proof eventually consists of $\Pi = ((D_k)_k, (D_{k,m})_{k,m}, (S_k, S'_k)_k, (w_m)_m, (S_{k,m}, S'_{k,m})_{k,m})$, where

$$S_k = T_k - e\rho_k \bmod q \qquad S'_k = T'_k - e\rho'_k \bmod q$$
$$w_m = v_m - eu_m \bmod q$$
$$S_{k,m} = T_{k,m} - e\rho_{k,m} \bmod q \qquad S'_{k,m} = T'_{k,m} - e\rho'_{k,m} \bmod q$$

It thus contains $K(M+1)$ ciphertexts (the number of equations), of $2\log\mathcal{N}$ bits, and $2K(M+1)+M$ scalars in $[\![0; q-1]\!]$ (the number of private masks), of less than $\log q$ bits.

The verifier can first compute the challenge e, and

$$D'_k = E^{S'_k}_{k,0,0} \cdot (E^{(2)}_{k,0,0})^{-S_k} \times (V'_k)^e \bmod \mathcal{N}^2$$
$$D'_{k,m} = E^{S_k}_{k,0,0} \times E^{w_m}_{k,0,0} \times {E^{(m)}_{k,1,0}}^{-S_{k,m}} \times {E^{(m)}_{k,0,1}}^{-S'_{k,m}} \times (V'_{k,m})^e \bmod \mathcal{N}^2$$

They then decrypt all the D_k/D'_k and $D_{k,m}/D'_{k,m}$, that should be 0 modulo q, as there is no reduction modulo $\mathcal{N}$ before the reduction modulo q, thanks to the constraint on $\mathcal{N} > 2\mu L \cdot 2^\lambda q^3$.

Soundness. After a rewinding, with a different challenge $\tilde{e} \neq e$, such that D'_k and $\tilde{D}'_k$ decrypt to the same value modulo q, and $D'_{k,m}$ and $\tilde{D}'_{k,m}$ decrypt to the same value modulo q, where

$$D'_k = E^{S'_k}_{k,0,0} \cdot (E^{(2)}_{k,0,0})^{-S_k} \times (V'_k)^e \bmod \mathcal{N}^2$$
$$\tilde{D}'_k = E^{\tilde{S}'_k}_{k,0,0} \cdot (E^{(2)}_{k,0,0})^{-\tilde{S}_k} \times (V'_k)^{\tilde{e}} \bmod \mathcal{N}^2$$
$$D'_{k,m} = E^{S_k}_{k,0,0} \times E^{w_m}_{k,0,0} \times {E^{(m)}_{k,1,0}}^{-S_{k,m}} \times {E^{(m)}_{k,0,1}}^{-S'_{k,m}} \times (V'_{k,m})^e \bmod \mathcal{N}^2$$
$$\tilde{D}'_{k,m} = E^{\tilde{S}_k}_{k,0,0} \times E^{\tilde{w}_m}_{k,0,0} \times {E^{(m)}_{k,1,0}}^{-\tilde{S}_{k,m}} \times {E^{(m)}_{k,0,1}}^{-\tilde{S}'_{k,m}} \times (V'_{k,m})^{\tilde{e}} \bmod \mathcal{N}^2$$

we have both

$$A_k = E^{S'_k - \tilde{S}'_k}_{k,0,0} \cdot (E^{(2)}_{k,0,0})^{\tilde{S}_k - S_k} \times (V'_k)^{e-\tilde{e}} \bmod \mathcal{N}^2$$
$$A_{k,m} = E^{S_k - \tilde{S}_k}_{k,0,0} \times E^{w_m - \tilde{w}_m}_{k,0,0} \times {E^{(m)}_{k,1,0}}^{\tilde{S}_{k,m} - S_{k,m}} \times {E^{(m)}_{k,0,1}}^{\tilde{S}'_{k,m} - S'_{k,m}} \times (V'_{k,m})^{e-\tilde{e}} \bmod \mathcal{N}^2$$

decrypt to 0 modulo q, while the small size of the answers and the evaluation of V'_k and $V'_{k,m}$ by the verifier on small scalars guarantees no wrap-up modulo $\mathcal{N}$: $q\alpha_k, q\alpha_{k,m} < 2\mu Lq^3 + 2^{\lambda'} q < \mathcal{N}$, even with encodings generated from L-linear combinations in

$$A_k = (1+\mathcal{N})^{q\alpha_k} \beta_k^{\mathcal{N}} \bmod \mathcal{N}^2 \qquad A_{k,m} = (1+\mathcal{N})^{q\alpha_{k,m}} \beta_{k,m}^{\mathcal{N}} \bmod \mathcal{N}^2$$

If we assume $\tilde{e} - e$ invertible modulo q, and set $\varepsilon = (\tilde{e} - e)^{-1} \bmod q$, we can compute

$$\rho'_k \leftarrow (S'_k - \tilde{S}'_k) \cdot \varepsilon \bmod q \qquad \rho_k \leftarrow (S_k - \tilde{S}_k) \cdot \varepsilon \bmod q$$
$$u_m \leftarrow (w_m - \tilde{w}_m) \cdot \varepsilon \bmod q$$
$$\rho_{k,m} \leftarrow (S_{k,m} - \tilde{S}_{k,m}) \cdot \varepsilon \bmod q \qquad \rho'_{k,m} \leftarrow (S'_{k,m} - \tilde{S}'_{k,m}) \cdot \varepsilon \bmod q$$

all smaller than q

$$E_{k,0,0}^{\rho'_k} \cdot (E_{k,0,0}^{(2)})^{-\rho_k} = V'_k \cdot (1+\mathcal{N})^{q\alpha'_k} \gamma_k^{\mathcal{N}} \bmod \mathcal{N}^2$$
$$E_{k,0,0}^{\rho_k} \times E_{k,0,0}^{u_m} \times E_{k,1,0}^{(m)\ -\rho_{k,m}} \times E_{k,0,1}^{(m)\ -\rho'_{k,m}} = V'_{k,m} \cdot (1+\mathcal{N})^{q\alpha'_{k,m}} \gamma_{k,m}^{\mathcal{N}} \bmod \mathcal{N}^2$$

Eventually, if $\mathcal{N} > 2\mu L q^3$, all the exponents to $(1+\mathcal{N})$ remain smaller than $\mathcal{N}$, which proves the soundness, until $\tilde{e} - e$ is invertible: one can take $2^{\lambda'}$ smaller than the smallest prime factor of q, so that any non-trivial difference will always be invertible, and iterate several times in parallel with multiple challenges e, to increase soundness. For $\mathcal{N}$, this is safe to take it larger than $2\mu L 2^{\lambda} q^3$. As μ will always be smaller than 4, in each individual equations, we can take $\mathcal{N} > L 2^{\lambda+3} q^3$.

Zero-Knowledge. Thanks to the random masks in $\mathcal{M}$, the plaintexts are statistically hidden. The proofs contain tuples $((D_k)_k, (D_{k,m})_{k,m}, (S_k, S'_k)_k, (w_m)_m, (S_{k,m}, S'_{k,m})_{k,m})$, where the ciphertexts statistically hide their representations modulo q, and the scalars are uniformly distributed in $[\![0; q-1]\!]$. Hence, a statistical zero-knowledge proof that guarantees the hiding property of the commitments.

C Commitments

C.1 Commitments in Additional Subspaces

In our proof of inner product, we will use polynomials in multiple subspaces of $\mathcal{R} = \mathbb{Z}_q[X, Y]$. In addition to $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$, we will use $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$, $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, and $\mathbb{Z}_q[Y^{2N}]$ with no term in Y^N, denoted $\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N \setminus N}]$. For this, we draw additional random elements $r_k^{(1)}, r_k^{(3)}, r_k^{(4)} \xleftarrow{\$} \mathbb{Z}_q^*$ for $k \in [\![1; K]\!]$, and set:

$$E_{k,i}^{(1)} \leftarrow \mathsf{E}(r_k^{(1)} \cdot s_k^i) \qquad i \in [\![0; n-1]\!]$$
$$E_{k,j}^{(3)} \leftarrow \mathsf{E}(r_k^{(3)} \cdot t_k^j) \qquad j \in [\![0; N]\!]$$
$$E_{k,j}^{(4)} \leftarrow \mathsf{E}(r_k^{(4)} \cdot t_k^j) \qquad j \in [\![0; 2N]\!] \setminus \{N\}$$

and we extend

$$E_{k,j,i} \leftarrow \mathsf{E}(s_k^i \cdot t_k^j) \qquad i \in [\![0; n-1]\!], j \in [\![0; 2N]\!]$$

Note that $\mathcal{D}$ becomes $\max\{N + n - 1, 2N\} = 2N$, for $N \geq n$. We then commit the polynomials $\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[Y^N])$ or $\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[Y^{2N \setminus N}])$ with appropriate twin encodings, that can be verified as above: for each pair $k \neq k'$ of valid encodings:

$$\mathsf{u}_k(Y) - \mathsf{u}_{k'}(Y') = \mathsf{u}(Y) - \mathsf{u}(Y') = (Y - Y') \cdot \mathsf{v}(Y, Y')$$

and so for a random $y_m \xleftarrow{\$} \mathbb{Z}_q$ chosen by the prover:

$$\mathsf{u}(Y) - \mathsf{u}(y_m) = (Y - y_m) \cdot \mathsf{v}(Y, y_m) = (Y - y_m) \cdot \mathsf{v}_m,$$

from

$$\Pi_{\mathsf{u}} = (u_m \leftarrow \mathsf{u}(y_m), (V_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(t_k)))_k)_m$$

as

$$\mathsf{QCheck}(X_1 - u_m - (X_2 - y_m) \cdot X_3, E_k, E_{k,1,0}, V_{k,m}) = \mathsf{true}$$

and the rest of the proof follows as in Section 3.2, with thus $\Pi_{\mathsf{u}} = (u_m, (V_{k,m})_k)_m$, for $k \in [\![1; K]\!]$, $m \in [\![1; M]\!]$.

C.2 Univariate Hiding Commitments

Commit*$(\mathsf{u}, \mathbb{Z}_q[Y^N])$, the hiding commitment for a univariate polynomial $\mathsf{u}(Y) = \sum_{j=0}^{N} u_j Y^j \in \mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, outputs the tuple $C = (E_{\mathsf{u}}, (E_k, E_k^{(3*)})_k, \Pi_{\mathsf{u}})$, where for all indices $k \in [\![1; K]\!]$ and $m \in [\![1; M]\!]$ with $\rho_k \xleftarrow{\$} \mathcal{M}$:

$$\begin{aligned} E_k &\leftarrow \mathsf{Eval}(\{E_{k,j,0}\}_j, \{F_k\}, \{u_j\}_j, \{\rho_k\}) = \mathsf{E}(\mathsf{u}(t_k) + \rho_k \cdot r_k), \\ E_k^{(3*)} &\leftarrow \mathsf{Eval}(\{E_{k,j,0}^{(3*)}\}_j, \{F_k^{(3*)}\}, \{u_j\}_j, \{\rho_k\}) = \mathsf{E}(r_k^{(3*)}(\mathsf{u}(t_k) + \rho_k \cdot r_k)) \\ &\text{with } \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E_k^{(3*)}, E_k, E_{k,0}^{(3*)}) = \mathsf{true} \end{aligned}$$

and a proof, using random $y_m \xleftarrow{\$} \mathbb{Z}_q$ sent by the verifier, of $\mathsf{u}(Y) - \mathsf{u}(y_m) = (Y - y_m) \cdot \mathsf{v}_m(Y)$ which can be checked with $\Pi_{\mathsf{u}} = (V_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(t_k) + \rho_{k,m} \cdot r_k))_{k,m}$ for random $\rho_{k,m} \xleftarrow{\$} \mathbb{Z}_q^*$, chosen by the prover for their privacy, and

$$\mathsf{ZKLQCheck}(X_1 - (X_2 - y_m) \cdot X_3, E_k^*, E_{k,1,0}, V_{k,m}; E_{k,0,0}, F_k^{(t)}, F_k) = \mathsf{true}$$

where one proves the knowledge of the 3 scalars $u_m = \mathsf{u}(y_m)$, $\rho'_{k,m} = \rho_k - \rho_{k,m} \cdot y_m$ and $\rho_{k,m}$ so that the above relation is equal to

$$u_m \times 1 + \rho_{k,m} \times r_k \cdot t_k - \rho'_{k,m} \times r_k \bmod q.$$

The same analysis as for the $\mathcal{R}_2$ case can be done for the soundness of the proof, but with less strict bounds, as degrees are lower.

C.3 Complete Construction of the Commitment

In the following, we are considering $q = p_1 \cdot \ldots \cdot p_\ell$, a composite modulus q, with ℓ prime factors $p_1 < \ldots < p_\ell$. We will work in subspaces of $\mathcal{R} = \mathbb{Z}_q[X^{n-1}, Y^{2N}]$. We will consider the subspaces $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$, $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$, $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, and $\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N \setminus N}]$, hence $\mathcal{D} = 2N$, so we will assume $t \geq 128 N \ell$.

Setup of the System: $\mathsf{Setup}(1^\lambda, \mathcal{R}, (\mathcal{R}_i)_i)$ first runs $(\mathsf{pk}', \mathsf{vk}') \leftarrow \mathsf{Gen}(1^\lambda)$, chooses K tuples of random elements $s_k, t_k \xleftarrow{\$} \mathbb{Z}_q^*$, as well as K tuples of random elements $r_k^{(1)}, r_k^{(2)}, r_k^{(3)}, r_k^{(4)}, \xleftarrow{\$} \mathbb{Z}_q^*$, to limit the combinations of the bases. Then, for $k \in [\![1; K]\!]$, one sets

$\mathcal{R} = \mathbb{Z}_q[X^{n-1}, Y^N] + \mathbb{Z}_q[Y^{2N}]$	$E_{k,j,i} \leftarrow \mathsf{E}(s_k^i \cdot t_k^j)$	$(i,j) \in ([\![0; n-1]\!] \times [\![0; N]\!]) \cup (\{0\} \times [\![N+1; 2N]\!])$
$\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$	$E_{k,i}^{(1)} \leftarrow \mathsf{E}(r_k^{(1)} \cdot s_k^i)$	$i \in [\![0; n-1]\!]$
$\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$	$E_{k,j,i}^{(2)} \leftarrow \mathsf{E}(r_k^{(2)} \cdot s_k^i \cdot t_k^j)$	$i \in [\![0; n-1]\!], j \in [\![0; N]\!]$
$\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$	$E_{k,j}^{(3)} \leftarrow \mathsf{E}(r_k^{(3)} \cdot t_k^j)$	$j \in [\![0; N]\!]$
$\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N \setminus N}]$	$E_{k,j}^{(4)} \leftarrow \mathsf{E}(r_k^{(4)} \cdot t_k^j)$	$j \in [\![0; 2N]\!] \setminus \{N\}$

Then, the public key of the commitment scheme is set to pk' with all these encodings, while the verification key is vk'. For $\mathcal{R}$, we can limit to $\mathbb{Z}_q[X^{n-1}, Y^N] + \mathbb{Z}_q[Y^{2N}]$, with $(n+1) \times N + n$ encodings in the public key, sent once for many proofs.

Commitment Generation: there are two commitment algorithm, with or without the hiding property.

Non-Hiding Commitment. We have defined commitments on specific subspaces: $\mathcal{R}_1 = \mathbb{Z}_q[X^{n-1}]$, $\mathcal{R}_2 = \mathbb{Z}_q[X^{n-1}, Y^N]$, $\mathcal{R}_3 = \mathbb{Z}_q[Y^N]$, and $\mathcal{R}_4 = \mathbb{Z}_q[Y^{2N\setminus N}]$:

$\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[X^{n-1}])$: it outputs $C = (E_\mathsf{u} = (E_k, E_k^{(1)})_k, \Pi_\mathsf{u})$, for $k \in [\![1; K]\!]$, where

$$E_k \leftarrow \mathsf{E}(\mathsf{u}(s_k)) \qquad\qquad E_k^{(1)} \leftarrow \mathsf{E}(r_k^{(1)} \cdot \mathsf{u}(s_k)),$$

and for all $m \in [\![1; M]\!]$: $x_m \xleftarrow{\$} \mathbb{Z}_q$ chosen by the verifier (or from a hash function for a non-interactive proof), and then

$$\Pi_\mathsf{u} = (u_m \leftarrow \mathsf{u}(x_m), (W_{k,m} \leftarrow \mathsf{E}(\mathsf{w}_m(s_k)))_k)_m$$

where w_m is such that $\mathsf{u}(X) - u_m = (X - x_m) \cdot \mathsf{w}_m(X)$: in total, these are $2K$ encodings, plus M scalars and KM encodings for the proof.

$\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[X^{n-1}, Y^N])$: it outputs $C = (E_\mathsf{u} = (E_k, E_k^{(2)})_k, \Pi_\mathsf{u})$, for $k \in [\![1; K]\!]$, where

$$E_k \leftarrow \mathsf{E}(\mathsf{u}(s_k, t_k)) \qquad\qquad E_k^{(2)} \leftarrow \mathsf{E}(r_k^{(2)} \cdot \mathsf{u}(s_k, t_k)),$$

and for all $m \in [\![1; M]\!]$: $(x_m, y_m) \xleftarrow{\$} \mathbb{Z}_q^2$ chosen by the verifier (or from a hash function for a non-interactive proof), and then

$$\Pi_\mathsf{u} = (u_m \leftarrow \mathsf{u}(x_m, y_m), (V_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(s_k, t_k)), W_{k,m} \leftarrow \mathsf{E}(\mathsf{w}_m(s_k)))_k)_m$$

where v_m and w_m are such that

$$\mathsf{u}(X, Y) - u_m = (Y - y_m) \cdot \mathsf{v}_m(X, Y) + (X - x_m) \cdot \mathsf{w}_m(X).$$

In total, these are $2K$ encodings, plus M scalars and $2KM$ encodings for the proof.

$\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[Y^N])$: it outputs $C = (E_\mathsf{u} = (E_k, E_k^{(3)})_k, \Pi_\mathsf{u})$, for $k \in [\![1; K]\!]$, where:

$$E_k \leftarrow \mathsf{E}(\mathsf{u}(t_k)) \qquad\qquad E_k^{(3)} \leftarrow \mathsf{E}(r_k^{(3)} \cdot \mathsf{u}(t_k)),$$

and for all $m \in [\![1; M]\!]$: $y_m \xleftarrow{\$} \mathbb{Z}_q$ chosen by the verifier (or from a hash function for a non-interactive proof), and then

$$\Pi_\mathsf{u} = (u_m \leftarrow \mathsf{u}(y_m), (V_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(t_k)))_k)_m$$

where v_m is such that $\mathsf{u}(Y) - u_m = (Y - y_m) \cdot \mathsf{v}_m(Y)$. In total, these are $2K$ encodings, plus M scalars and KM encodings for the proof.

$\mathsf{Commit}(\mathsf{u}, \mathbb{Z}_q[Y^{2N\setminus N}])$: as the previous case but replacing $r^{(3)}$ with $r^{(4)}$ and analogously (3) indices with (4) indices.

Hiding Commitment. We only focus on $\mathcal{R}_2$ and $\mathcal{R}_3$:

$\mathsf{Commit}^*(\mathsf{u}, \mathbb{Z}_q[X^{n-1}, Y^N])$: it outputs $C^* = (E_\mathsf{u}^* = (E_k^*, E_k^{(2*)}, \pi_k)_k, \Pi_\mathsf{u}^*)$, for $k \in [\![1; K]\!]$, where, with $\rho_k, \rho'_k \xleftarrow{\$} \mathcal{M}$, where $\mathcal{M}$ is the appropriate masking set:

$$E_k^* \leftarrow \mathsf{Eval}(\{E_{k,j,i}\}_{j,i}, \{E_{k,0,0}\}, \{u_{j,i}\}_{j,i}, \{\rho_k\}) = \mathsf{E}(\mathsf{u}(s_k, t_k) + \rho_k)$$
$$E_k^{(2*)} \leftarrow \mathsf{Eval}(\{E_{k,j,i}^{(2)}\}_{j,i}, \{E_{k,0,0}\}, \{u_{j,i}\}_{j,i}, \{\rho'_k\}) = \mathsf{E}(r_k^{(2)} \cdot \mathsf{u}(s_k, t_k) + \rho'_k)$$
$$\text{with } \pi_k = \{\mathsf{ZKLQCheck}(X_1 - X_2 \cdot X_3, E_k^{(2*)}, E_k^*, E_{k,0,0}^{(2)}; E_{k,0,0}, E_{k,0,0}^{(2)}) = \mathsf{true}\}$$

and for all $m \in [\![1; M]\!]$: $(x_m, y_m) \xleftarrow{\$} \mathbb{Z}_q^2$ chosen by the verifier (or from a hash function for a non-interactive proof), and then for random $\rho_{k,m}, \rho'_{k,m} \xleftarrow{\$} \mathbb{Z}_q^*$ chosen by the prover for their privacy:

$$\Pi^*_{\mathsf{u}} = (V^*_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(s_k, t_k) + \rho_{k,m}), W^*_{k,m} \leftarrow \mathsf{E}(\mathsf{w}_m(s_k) + \rho'_{k,m})_{k,m}$$

where v_m and w_m are such that

$$\mathsf{u}(X, Y) - \mathsf{u}(x_m, y_m) = (Y - y_m) \cdot \mathsf{v}_m + (X - x_m) \cdot \mathsf{w}_m$$

with

$$\begin{aligned}\pi_{k,m} = \{&\mathsf{ZKLQCheck}(X_1 - (X_2 - y_m) \cdot X_3 - (X_4 - x_m) \cdot X_5,\\ &E^*_k, E_{k,1,0}, V^*_{k,m}, E_{k,0,1}, W^*_{k,m}; E_{k,0,0}, E_{k,0,0}, E^{(m)}_{k,1,0}, E^{(m)}_{k,0,1}) = \mathsf{true}\}\end{aligned}$$

KM additional zero-knowledge proofs of knowledge of 4 scalars $\rho_{k,m}$, $\rho'_{k,m}$, ρ_k, and $u_m = \mathsf{u}(x_m, y_m)$. In total, this is $2K$ encodings, plus KM encodings for the proof, and KM zero-knowledge proofs of 4 scalars.

$\mathsf{Commit}^*(\mathsf{u}, \mathbb{Z}_q[Y^N])$: it outputs $C^* = (E^*_{\mathsf{u}} = (E^*_k, E^{(3*)}_k, \pi_k)_k, \Pi^*_{\mathsf{u}})$, for $k \in [\![1; K]\!]$, where, with $\rho_k, \rho'_k \xleftarrow{\$} \mathcal{M}$:

$$E_k \leftarrow \mathsf{E}(\mathsf{u}(t_k) + \rho_k) \qquad\qquad E^{(3*)}_k \leftarrow \mathsf{E}(r^{(3)}_k \cdot \mathsf{u}(t_k) + \rho'_k)$$

with $\pi_k = \{\mathsf{ZKLQCheck}(X_1 - X_2 \cdot X_3, E^{(3*)}_k, E^*_k, E^{(3)}_{k,0,0}; E_{k,0,0}, E^{(3)}_{k,0,0}) = \mathsf{true}\}$, and for all $m \in [\![1; M]\!]$: $y_m \xleftarrow{\$} \mathbb{Z}_q$ chosen by the verifier (or from a hash function for a non-interactive proof), and then for random $\rho_{k,m} \xleftarrow{\$} \mathbb{Z}_q^*$ chosen by the prover for their privacy:

$$\Pi^*_{\mathsf{u}} = (V^*_{k,m} \leftarrow \mathsf{E}(\mathsf{v}_m(t_k) + \rho_{k,m}))_{k,m}$$

for v_m such that $\mathsf{u}(Y) - \mathsf{u}(y_m) = (Y - y_m) \cdot \mathsf{v}_m(Y)$ with KM additional zero-knowledge proofs of knowledge of 3 scalars $\rho_{k,m}$, ρ_k, and $u_m = \mathsf{u}(y_m)$. In total, this is $2K$ encodings, plus KM encodings for the proof, and KM zero-knowledge proofs of 3 scalars.

Validity Check: it also depends on hiding or non-hiding commitments and on the space $\mathcal{R}_\pi$.

Non-Hiding Commitment. $\mathsf{Validity}(C, \mathcal{R}_\pi)$ first checks the twin encodings, for $k \in [\![1; K]\!]$:

$$\begin{aligned}&\mathcal{R}_1 && \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E^{(1)}_k, E_k, E^{(1)}_{k,0}) = \mathsf{true}\\ &\mathcal{R}_2 && \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E^{(2)}_k, E_k, E^{(2)}_{k,0,0}) = \mathsf{true}\\ &\mathcal{R}_3, \mathcal{R}_4 && \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E^{(3/4)}_k, E_k, E^{(3/4)}_{k,0}) = \mathsf{true}\end{aligned}$$

and then, for $k \in [\![1; K]\!]$ and $m \in [\![1; M]\!]$, either

$$\begin{aligned}&\mathcal{R}_1 && \mathsf{QCheck}(X_1 - u_m - (X_2 - x_m) \cdot X_3, E_k, E_{k,0,1}, W_{k,m}) = \mathsf{true}\\ &\mathcal{R}_2 && \mathsf{QCheck}(X_1 - u_m - (X_2 - y_m) \cdot X_3 - (X_4 - x_m) \cdot X_5,\\ &&& \qquad E_k, E_{k,1,0}, V_{k,m}, E_{k,0,1}, W_{k,m}) = \mathsf{true}\\ &\mathcal{R}_3, \mathcal{R}_4 && \mathsf{QCheck}(X_1 - u_m - (X_2 - y_m) \cdot X_3, E_k, E_{k,1,0}, V_{k,m}) = \mathsf{true}\end{aligned}$$

Hiding Commitment. $\mathsf{Validity}^*(C, \mathcal{R}_\pi)$ first checks the twin encodings, for $k \in [\![1; K]\!]$:

$$\mathcal{R}_2 \qquad \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E_k^{(2*)}, E_k, E_{k,0,0}^{(2)}) = \mathsf{true}$$
$$\mathcal{R}_3 \qquad \mathsf{QCheck}(X_1 - X_2 \cdot X_3, E_k^{(3*)}, E_k, E_{k,0}^{(3)}) = \mathsf{true}$$

and then, for $k \in [\![1; K]\!]$ and $m \in [\![1; M]\!]$, the zero-knowledge proofs

$$\mathcal{R}_2 \qquad \mathsf{ZKLQCheck}(X_1 - (X_2 - y_m) \cdot X_3 - (X_4 - x_m) \cdot X_5,$$
$$E_k^*, E_{k,1,0}, V_{k,m}^*, E_{k,0,1}, W_{k,m}^*; E_{k,0,0}, E_{k,0,0}, E_{k,1,0}^{(m)}, E_{k,0,1}^{(m)}) = \mathsf{true}$$
$$\mathcal{R}_3 \qquad \mathsf{ZKLQCheck}(X_1 - (X_2 - y_m) \cdot X_3, E_k, E_{k,1,0}, V_{k,m}^*; E_{k,0,0}, E_{k,1,0}^{(m)}, E_{k,0,1}^{(m)}) = \mathsf{true}$$

D Proving Correct Noise

In the case of an honest choice of the r_k's by the sender, coefficients of u follow a Gaussian distribution on $\mathbb{Z}$ with standard deviation $Z \cdot \sigma'$. For their privacy, the sender needs $10Z \cdot \sigma' \geq 2^\lambda \cdot B'$, where B' is the maximal error in the clean ciphertext $(\mathsf{d}, \mathsf{d}')$. This bound is analysed later.

Eventually, the publication of the linear combination r reveals information about the r_k's, and the larger η is, the more information leaks. For the check on r to detect large coefficients, one should avoid similar coefficients to compensate, and so with similar random coefficients in the linear combination: false negative will happen with probability bounded by $1/2\eta$. We can reduce it to $\varepsilon_s/3$ with θ iterations: $(1/2\eta)^\theta \leq \varepsilon_s/3$, with $\theta \geq \log_2(3/\varepsilon_s)/(1+\log_2 \eta)$. On the other hand, we need to make sure that enough entropy remains: entropy of Gaussian random variables $r_{k,i}$ is $\log_2(\sigma'\sqrt{2\pi e})$, so globally there are $Z \log_2(\sigma'\sqrt{2\pi e})$ bits of entropy revealed. With θ linear combinations, one reveals $\theta \log q$ bits for each index i. One would like to still have $\log_2(Z \cdot \sigma'\sqrt{2\pi e})$ bits remaining for true randomness in the noise: $Z \log_2(\sigma'\sqrt{2\pi e}) \geq \theta \log_2 q + \log_2(Z \cdot \sigma'\sqrt{2\pi e})$:

$$\theta = \log_2(3/\varepsilon_s)/(1 + \log_2 \eta) \qquad Z \cdot \sigma' \geq 2^\lambda \cdot B'/10$$
$$(Z-1)\log_2(\sigma'\sqrt{2\pi e}) \geq \theta \log_2 q + \log_2 Z$$

On the other hand, for the correctness during the decryption, if u, e_1, and e_2 follow the same process, we will need $q > 2t \cdot 10Z^2 \cdot \sigma'\eta \cdot (2nB + 1)$, as the randomness in the decryption key is likely bounded by $B = 10\sigma$. Hence, $q \geq 2^\lambda \eta Z \times 2tB'(2nB + 1)$, where B' is the error bound in the clean ciphertext $(\mathsf{d}, \mathsf{d}')$, after deterministic evaluation.

E FHE Parameters

As explained in the preliminaries, we consider the Fan-Vercauteren Fully Homomorphic Encryption scheme [FV12].

Notations. Let $\mathcal{R}$ be the ring $\mathbb{Z}[X]/\mathsf{r}(X)$, where $\mathsf{r}(X) = X^n + 1$. Given a polynomial $\mathsf{p} \in \mathcal{R}$, we denote $\|\mathsf{p}\|_\infty$ the infinity norm, *i.e.* the max of its coefficients. We also define the polynomial multiplication expansion factor δ as

$$\delta = \max_{\mathsf{c},\mathsf{d} \in \mathcal{R}} \{\|\mathsf{c} \cdot \mathsf{d}\|_\infty / (\|\mathsf{c}\|_\infty \cdot \|\mathsf{d}\|_\infty)\}.$$

By taking the cyclotomic polynomial $\mathsf{r}(X) = X^n + 1$, the worst-case bound for this expansion factor is $\delta = n$.

The Fan-Vercauteren FHE. We denote $\mathcal{R}_t = \mathbb{Z}_t[X]/\mathsf{r}(X)$ (with $\mathsf{r} \in \mathbb{Z}_t[X]$ an irreducible polynomial of degree n) the plaintext message space and $\mathcal{R}_q = \mathbb{Z}_q[X]/\mathsf{r}(X)$ the ciphertext space, and we denote $\Delta = q/t$. We will denote χ the Gaussian distribution on $\mathbb{Z}$ with standard deviation σ.

The FV encryption scheme [FV12] defines $(\mathsf{p}, \mathsf{p}') = (-(\mathsf{a}\cdot\mathsf{s}+\mathsf{e}), \mathsf{a}) \in \mathcal{R}_q^2$ as the public key, with the coefficients of the secret key s and the error e taken from χ, thereafter reduced modulo q. A fresh ciphertext of $\mathsf{m} \in \mathcal{R}_t$ can be written as

$$(\mathsf{c}, \mathsf{c}') = \left(\mathsf{p}\cdot\mathsf{u} + \mathsf{e}_1 + \Delta\cdot\mathsf{m} \bmod q, \mathsf{p}'\cdot\mathsf{u} + \mathsf{e}_2 \bmod q\right),$$

with coefficients of $\mathsf{u}, \mathsf{e}_1, \mathsf{e}_2$ also taken from χ. In order to decrypt, using the secret key s, one computes

$$\begin{aligned}\mathsf{d} &= \mathsf{c} + \mathsf{c}'\cdot\mathsf{s} = \Delta\cdot\mathsf{m} - \mathsf{e}\cdot\mathsf{u} + \mathsf{e}_2\cdot\mathsf{s} + \mathsf{e}_1 = \Delta\cdot\mathsf{m} + \mathsf{v} \bmod q\\ \mathsf{m}' &= \lfloor \mathsf{d}/\Delta \rceil = \lfloor (\Delta\cdot\mathsf{m} + \mathsf{v})/\Delta \rceil = \mathsf{m} + \lfloor \mathsf{v}/\Delta \rceil \bmod t\end{aligned}$$

where $\mathsf{v} = -\mathsf{e}\cdot\mathsf{u} + \mathsf{e}_2\cdot\mathsf{s} + \mathsf{e}_1$ is the error term: $\mathsf{m}' = \mathsf{m}$ if $\|\mathsf{v}\|_\infty \le \Delta/2$.

Assuming the coefficients of the error polynomials and the secret key polynomial are bounded by B, we just require $B\cdot(2nB+1) \le \Delta/2$ for correct decryption. In particular, $\Pr[\|\mathsf{e}\|_\infty > 10\cdot\sigma, \mathsf{e} \xleftarrow{\$} \chi] \le 2^{-128}$.

Semantic Security. From [PRS17]], if $0 \le \alpha < 1$ be some real, and the modulus q is such that $\sigma = \alpha q = \omega(1)$, then there is a polynomial quantum reduction from the SIVP problem with approximation factor $\gamma(n)$ to the average-case decision Ring-LWE where $\gamma(n) \le \max\{\omega(\sqrt{n\log n}/\alpha), \sqrt{2}n\}$. For the SIVP problem to be hard, we need $\gamma(n)$ to be polynomial in n. And the semantic security of the FV scheme relies on the decision Ring-LWE problem. We can take $\sigma = \alpha q = \sqrt{n}$, then we have $\gamma(n) \le q\sqrt{\log n}$, which will be polynomial in n, in our case.

Correctness Let $\mathsf{a}_i \in \mathcal{R}_t$ be L polynomials and $(\mathsf{c}_i, \mathsf{c}_i')$ be L ciphertexts generated with a circuit of depth d, encrypting $\mathsf{m}_i \in \mathcal{R}_t$. In order to decrypt the linear combination, we compute:

$$\begin{aligned}\sum_{i=1}^{L}\mathsf{a}_i\cdot\mathsf{c}_i + \mathsf{s}\cdot\sum_{i=1}^{L}\mathsf{a}_i\cdot\mathsf{c}_i' &= \sum_{i=1}^{L}\mathsf{a}_i\left(\mathsf{c}_i + \mathsf{s}\cdot\mathsf{c}_i'\right)\\ &= \sum_{i=1}^{L}\mathsf{a}_i(\Delta\cdot\mathsf{m}_i + \mathsf{v}_i) \bmod q = \Delta\cdot\sum_{i=1}^{L}\mathsf{a}_i\cdot\mathsf{m}_i + \sum_{i=1}^{L}\mathsf{a}_i\cdot\mathsf{v}_i \bmod q\end{aligned}$$

We have the required (L,d)-$\mathcal{R}_t$ linear homomorphism property if

$$\left\|\sum_{i=1}^{L}\mathsf{a}_i\cdot\mathsf{v}_i\right\|_\infty \le \sum_{i=1}^{L} n\|\mathsf{a}_i\|_\infty\cdot\|\mathsf{v}_i\|_\infty \le nt\sum_{i=1}^{L}\|\mathsf{v}_i\|_\infty \le \Delta/2.$$

Using [LN14] noise derivation formula, the error noise growth after having evaluated a sequence of d multiplications with a fresh ciphertext is bounded by $C_1^d V + dC_1^{d-1}C_2 = C_1^{d-1}\cdot(C_1V + dC_2)$, where

$$C_1 = n^2tB + 4nt \qquad C_2 = n^2B(B+t^2) + \frac{n\ell q^{2/\ell}B}{2} \qquad V = B(2nB+1)$$

where

$$C_1 \le 11tn^{5/2} \qquad C_2 \le 20n^{\frac{3}{2}}t^2(n+\ell) \qquad V \le 201n^2$$

assuming $n \geq 5$, $\sigma = \sqrt{n}$, $B = 10 \cdot \sqrt{n} \leq t^2$, and the basis for relinearization $w = q^{2/\ell} \leq 4t^2$. With the additional assumption $\ell \leq n$, the initial error in the result is bounded by

$$Lnt \cdot C_1^{d-1} \cdot \big(C_1 V + dC_2\big) \leq Lnt \cdot (11tn^{5/2})^d \cdot \big(201n^2 + 4dt\big).$$

With our additional noise-flooding, the error is still bounded by $2^{2\lambda+4} Lnt \cdot (11tn^{5/2})^d \cdot \big(201n^2 + 4dt\big)$. For this bound, we take the pessimistic situation of $\eta = 2^\lambda$ and $Z = 16$.

With $dt \geq 201n^2$, it is sufficient to have the condition: $5 \cdot 2^{2\lambda+4} \cdot t^2 Lnd \cdot (11tn^{5/2})^d \leq q/2t$. As $t = p_1 < \ldots < p_\ell$, $q \geq t^\ell$. We can thus take ℓ such that

$$160 \cdot 2^{2\lambda} \cdot Lnd \cdot (11n^{5/2})^d \leq t^{\ell-3-d}.$$

With $\ell = 3(d+1)$, we need $t^2 \geq 11n^{5/2} \cdot (160 \cdot 2^{2\lambda} \cdot Lnd)^{1/d}$ to have the (L,d)-$\mathcal{R}_t$ linear homomorphism property and noise-flooding. In particular, we will need the (N,d)-$\mathcal{R}_t$ linear homomorphism property: $t^2 \geq 11n^{5/2} \cdot (160 \cdot 2^{2\lambda} \cdot Nnd)^{1/d}$, which will be satisfied as we already need $t \geq 64 \times 24\ell N \geq 128\ell n^d$, with $S = 5$, $N = n^d$, and $11n^{5/2} \cdot (160 \cdot 2^{2\lambda} \cdot Nnd)^{1/d} = 11n^{7/2} \cdot (160 \cdot 2^{2\lambda} \cdot nd)^{1/d} \leq 22n^{7/2+1/d} \cdot 2^{(2\lambda+8)/d} \leq 2^{5+(2\lambda+8)/d} n^4 \leq 2^{58} n^4$ when $d \geq 5$ and $\lambda = 128$. This is less than $(128)^2 \cdot 2^{44} \cdot n^{2d} \cdot n^{-6} \leq (128)^2 \cdot n^{2d} \leq t^2$ when $n \geq 2^8$.

F Python Script for Parameters

F.1 Parameters Choice

In the application, we performed optimizations on the parameters that were not necessarily detailed in the main paper description, but will be in this section to make all parameter choices clearer.

- The security parameter λ is set as 128, to have an 128-bit security;
- The soundness parameter ε_s is chosen among standard values at 2^{-30} for interactive proofs and 2^{-80} and 2^{-128} for non-interactive proofs;
- The specific soundness parameter for commitments ε_c is taken to be less than $\varepsilon_s/3\nu_c$, where $\nu_c = 14\Lambda + 5 + 3Z$, so that all the commitments are valid excepted with probability less than $\varepsilon_s/3$, making up for a third of the possible soundness error.
- The number Λ of points, committed polynomials have to be evaluated in, is taken as $\Lambda = \left\lceil \frac{\log_2(24/\varepsilon_s)}{S+1+\log_2(N/n)} \right\rceil$, where S is the margin length parameter given by $t \geq 2^S \cdot 2\mathcal{D}\ell = 2^S \cdot 4N\ell$. This ensures that Λ times the probability a quadratic relationship with polynomials in β_κ is wrong, $2\ell n/t \leq n/(2^{S+1}N)$, is less than $\varepsilon_s/24$, and as there are 8 relationships to check, the error coming from these verifications will not exceed $\varepsilon_s/3$, making up for a second third of the possible soundness error;
- We choose $\sigma = 4$ and $B = 10\sigma$ as the corresponding FHE parameters;
- η is the bound of the distribution of the scalars $\delta_k \in [\![-\eta;\eta]\!]$ being multiplied with small errors in the noise checks to prove $(\mathsf{z}^*, \mathsf{z}'^*)$ is correct, where soundness error is also bounded by $\varepsilon_s/3$, making up for the last third of the possible soundness error. Because of the noise flooding constraints, there are $Z = 2 + \left\lceil \frac{\lambda}{\log_2(\eta)+1} \right\rceil$ of these scalars at each draw, $\theta = \lceil 2^{\lambda/(1+\log_2 \eta)} \rceil$ draws, and the noise flooding constraints add $\left\lceil \lambda + \log_2(\eta) + \log_2(2 + \frac{\lambda}{\log_2(\eta)+1}) \right\rceil$ bits to q;
- Getting the FHE parameters:
  - t is taken to be the next prime above $2^S \cdot 2\mathcal{D}\ell = 2^S \cdot 4N\ell$ such that $t \equiv 8 \mod 3$, so that $\mathcal{R}_t$ be a ring product of two large fields of degree $n/2$ each, and the probability of having a false $(\mathsf{b}, \mathsf{b}')$ decoding to the right value be negligible;
  - the $\ell - 1$ next primes are then multiplied with t to obtain q;
  - the maximal error in FHE ciphertexts after d multiplications is then calculated, and we check it remains small enough to ensure correctness;
  - the FHE security with these parameters is calculated using the external reference estimator [APS15].

– After these FHE parameters were calculated, a lower bound on the Paillier modulus (2^{RSAM}), $8nNq^3 2^\lambda$ is given, so that we can be sure the Paillier encodings are never reduced modulo $\mathcal{N}$. We take $|\mathcal{N}|$ (RSAM) as the maximum of this bound and 2048 to ensure satisfactory security (above 100-bit security).

Then we use the following numbers:

– FHE ciphertexts are on $2n \log q$ bits;
– The receiver sends $(d+1)\lceil (N+1)^{1/(d+1)} \rceil$ of these, and this is how the Global FHE Ciphertexts Size is calculated;
– The number Λ of points is $\Lambda = \left\lceil \frac{\log_2(24/\varepsilon_s)}{S+1+\log_2(N/n)} \right\rceil$;
– The number of commitments, $\nu_c = 14\Lambda + 3Z + 5$;
– The encoding parameters are then calculated for the commitments, depending on c, the maximal number of prover-generated commitments they will be involved with in a single quadratic check (see Figure 3);
  – The number of times a Zero-Knowledge Proof has to be repeated is taken as $\left\lceil \frac{\log_2(3\nu_c/\varepsilon_s)}{\log_2 t} \right\rceil$;
  – We need for commitments and equations involving c prover-generated commitments

$$K \geq \log_2(3\nu_e/\varepsilon_s)(c+1)/S \qquad K \geq (\log_2(3\nu_c/\varepsilon_s)+1)(c+1)/(S+1)$$
$$M \geq 3(\log_2(3K\nu_c/\varepsilon_s)+2)/(S+1)$$

  So we take $K = \max\{\left\lceil (c+1) \times \frac{\log_2(\varepsilon_c)+1}{S+1} \right\rceil; \left\lceil (c+1) \times \frac{\log_2(3\nu_e/\varepsilon_s)}{S} \right\rceil\}$ and $M = \left\lceil 3 \times \frac{2+\log_2(\varepsilon_c K)}{S+1} \right\rceil$.
  – In the binding case, there are M scalars, with $K(M+2)$ encodings for the commitment of a one variable polynomial and $2K(M+1)$ encodings for a two variables polynomial.
  – In the hiding case, there are always $K(M+1)$ encodings and $2K(M+1)+M$ scalars, times the number of repetitions.
– Each Paillier encoding holds on $4|\mathcal{N}|$ bits;
– Each proof holds on $2|\mathcal{N}| + 4\log_2 q$ bits.
– All the sizes of the commitments given in figure 4 are then added up and the final size of the sender's communications encompasses this and the size of the $6(1+\theta)$ elements of $\mathcal{R}_q$ directly sent to the receiver.

#Prover-Generated Commitments: c		2	3	4
$\varepsilon_c = 2^{-38}$	K	26	35	43
	M (and binding case #Scalars)	20	20	20
	#Equations (hiding case)	2184	2940	3612
	#Secrets (hiding case)	4448	5960	7304
	#Encodings for $1v$ Pol.	572	770	946
	#Encodings for $2v$ Pol.	1092	1470	1806
$\varepsilon_c = 2^{-136}$	K	75	100	125
	M (and binding case #Scalars)	62	62	63
	#Equations (hiding case)	18900	25200	32000
	#Secrets (hiding case)	38048	50648	64252
	#Encodings for $1v$ Pol.	4800	6400	8125
	#Encodings for $2v$ Pol.	9450	12600	16000

Fig. 3. Parameters for Commitments with $t \geq 64\ell\mathcal{D}$, four repetitions in the zero-knowledge proofs, $\varepsilon_s = 2^8 \cdot \varepsilon_c$, and $\nu_e = 11$.

#variables	c	Binding or Hiding	Polynomials	Domain	#commitments
$1v$	2	B	$\mathsf{y}_\kappa, \mathsf{y}'_\kappa$	$\mathcal{R}_4$	2Λ
		H	$\mathsf{q}^*, \mathsf{q}'^*, \mathsf{q}^*_\kappa, \mathsf{q}'^*_\kappa$ $\mathsf{z}^*_\kappa, \mathsf{z}'^*_\kappa$	$\mathcal{R}_1$	$2(2\Lambda+1)$
	3	H	$\mathsf{u}^*_k, \mathsf{e}^*_{1,k}, \mathsf{e}^*_{2,k}$	$\mathcal{R}_1$	$3Z$
	4	B	$\mathsf{v}_\kappa, \mathsf{v}'_\kappa$ $\mathsf{s}_\kappa, \mathsf{s}'_\kappa$	$\mathcal{R}_3$ $\mathcal{R}_4$	2Λ 2Λ
		H	g^*_κ	$\mathcal{R}_3$	Λ
$2v$	3	B	$\mathsf{u}, \mathsf{u}', \mathsf{w}_\kappa, \mathsf{w}'_\kappa$	$\mathcal{R}_2$	$2(\Lambda+1)$
		H	$\mathsf{f}^*, \mathsf{h}^*_\kappa$	$\mathcal{R}_2$	$\Lambda+1$
Total					$14\Lambda+5+3Z$

Fig. 4. Number of Commitments in the Global Proof, and Size in Number of Encodings, with $\Lambda = \lceil (3 + \log_2(3/\varepsilon_s))/(S+1+\log_2(N/n)) \rceil$

F.2 Script

Here is the python script used to generate our figures, whose parameters are described above.

```
from sage.all import *
import math,ssl
try:
    load("estimator.py") # use the local script if you downloaded it because it gains a lot of time
except:
    ssl._create_default_https_context = ssl._create_unverified_context
    load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")
    # in case ssl does not work this file can be downloaded.
    # load("estimator.py")
    logging.getLogger("estimator").setLevel(logging.WARNING)

# Parameters to be tested
# MarginLength for t: t > 2**MarginLength * D * ell
Values = [ # lN, ln, d, ell, MarginLengh
    [20 , 14 , 4 , 14, 21],
# BEST 20

    [25 , 15 , 5 , 16, 16],
# BEST 25

    [30 , 14 , 5 , 16, 11],
    [30 , 15 , 5 , 15, 20],
# BEST 30

    [35 , 15 , 5 , 15, 15],
    [35 , 15, 5, 14, 25],
# BEST 35

    [40 , 15 , 5 , 15, 10],
# BEST 40
]


secpar = 128 # lambda
# Security bounds for commitments
Eps = [ 30, 80, 128 ]
EpsC = [ 38, 136 ]
log2_eta = 21
Z = 2 + ceil(secpar/(log2_eta+1)) # number of encodings of zero committed by the sender
theta = ceil(2**(secpar/(log2_eta+1))) # number of times we draw scalars to prove the z^*, z^*'
addpar = log2_eta + math.log2(2+secpar/(log2_eta+1)) # because of noise-flooding
MB = 8*1024*1024 # MBytes
MinMarginLength = 5 # t > 32 * 4N ell
sigma = 4
B = 10*sigma

# calculating the number of commitments: nuc = AnbCmt*Lambda + BnbCmt
AnbCmt = 14
BnbCmt = 5+3*Z
nue = 11 # number of quadratic relations between commitments the receiver has to check
```

```
def bitSizeOf(x):
    try: return math.ceil(math.log2(x))
    except: return 0
def security(n, alpha, q):
    cost = estimate_lwe(n, alpha, q)
    min = 1000
    for algo in cost.keys():
        if min > bitSizeOf(cost[algo]['rop']):
            min = bitSizeOf(cost[algo]['rop'])
    return min

# Security Evaluation of FHE Parameters
def FHE(lN, ln, d, ell, MarginLength):
    N = 2**lN; n = 2**ln; D = 2*N
    parameters = { 'logN': lN, 'n': n, 'd':d}
    parameters['ell'] = ell
    # Correctness FHE
    # Selection of the prime numbers
    p=[0]*ell
    pmin = 2**MarginLength * 2 * D * ell
    p[0]=next_prime(pmin)
    while (p[0] % 8) != 3: p[0] = next_prime(p[0]+1)
    t = p[0]
    q = t
    for i in range(ell-1):
        p[i+1] = next_prime(p[i]+1); q = q*p[i+1]
    # Analysis of the noise
    C1 = n*n*t*B + 4*n*t
    w = q**(2/ell); lw = math.floor(ell/2)+1
    C2 = n*n*B*(B+t*t)+n*lw*w*B; V = B*(2*n*B+1)
    error = N*n*t*C1**(d-1)*(C1*V+d*C2)*(2**(secpar+addpar)); ratio = 2*t*error/q # added noise flooding error
    parameters['ratio'] = ratio
    # Privacy FHE
    sec = security(n, sigma/q, q)
    parameters['logMargin'] = MarginLength
    parameters['logpl'] = bitSizeOf(p[-1])
    parameters['logt'] = bitSizeOf(t)
    parameters['logq'] = bitSizeOf(q)
    parameters['sec'] = sec
    parameters['RSAM'] = max(2048,3+ln+lN+3*bitSizeOf(q)+secpar) # greater than 2048 for security
    return parameters

# Number of Encodings
def encodings(epsc, MarginLength, ZKrepetitions, epss, nue):
    C = [2, 3, 4] # number of prover-generated encodings in equation
    SizeE = [{}]*5
    for c in C:
        gamma = c+1
        K = math.ceil(gamma * max((epsc+ 1)/(MarginLength+1), (math.log2(3*nue)+epss)/MarginLength))
        M = math.ceil(3 * (epsc +2 + math.log2(K) ) / (MarginLength+1))
        SizeE[c] = { 'K': K, 'M': M,
            '1vEB': K*(M+2), '2vEB': 2*K*(M+1), 'EH': ZKrepetitions*K*(M+1),
            'SB': M, 'SH': ZKrepetitions*(2*K*(M+1)+M) }
    return SizeE

def get_application_results():
    print("Z: "+str(Z))
    print("additional bits on q: "+ str(ceil(addpar+secpar)))
    for val in Values:
        print("\n**********")
        param = FHE(val[0], val[1], val[2], val[3], val[4])
        print("\n----------\n RSA Modulus: "+str(param['RSAM']))
        print("\n N = 2^"+str(param['logN']))
        print(" n = 2^"+str(bitSizeOf(param['n'])))
        print(" d = "+str(param['d']))
        print(" ell = "+str(param['ell']))
        print(" S = "+str(param['logMargin'])+" >=? "+str(MinMarginLength))
        print(" t on "+str(param['logt'])+" bits")
        print(" p_ell on "+str(param['logpl'])+" bits")
        print(" q on "+str(param['logq'])+" bits")
        print(" ratio = "+str(param['ratio']))
        if param['ratio'] > 1:
```

```
            print("Correctness error! Parameters not compatible")
        else:
            print(" Best attack against FHE: "+str(param['sec']))
            RSAM = param['RSAM']
            SizeFHE = 2*param['n']*param['logq']
            GlobalFHECiphersSize = ceil((2**(param['logN'])+1)**(1/(param['d']+1)))*(param['d']+1)*SizeFHE
            print(" FHE Size: "+str(math.ceil(GlobalFHECiphersSize/MB))+" MB")

            secInd=0
            for epss in Eps:
                secInd=secInd+1
                Lambda = math.ceil((3+epss+math.log2(3))/(param['logN']+param['logMargin']+1-math.log2(param
                    ↪ ['n'])))
                nuc = AnbCmt*Lambda + BnbCmt # total number of commitments
                ZKrepetitions = ceil((math.log2(3*nuc)+epss)/param['logt'])
                epsc = epss + math.ceil(math.log2(3*nuc))
                SizeE = encodings(epsc, param['logMargin'], ZKrepetitions, epss, nue)
                SizeLOE = 4*RSAM; SizeScalar = param['logq']
                SizeProof = 2*RSAM + 4*param['logq'] # Appendix B2, mu = 4
                v1c2B = 2*Lambda*(SizeLOE*SizeE[2]['1vEB'] + SizeScalar*SizeE[2]['SB']) # binding with 1v and c=2
                v1c2H = 2*(2*Lambda+1)*(SizeLOE*SizeE[2]['EH'] + SizeProof*SizeE[2]['SH']) # 1v c=2 hiding
                v1c3H = 3*Z*(SizeLOE*SizeE[3]['EH'] + SizeProof*SizeE[3]['SH']) # 1v c=3 hiding
                v1c4B = 4*Lambda*(SizeLOE*SizeE[4]['1vEB'] + SizeScalar*SizeE[4]['SB']) # 1v c=4 binding
                v1c4H = Lambda*(SizeLOE*SizeE[4]['EH'] + SizeProof*SizeE[4]['SH']) # 1v c=4 hiding
                v2c3B = 2*(Lambda+1)*(SizeLOE*SizeE[3]['2vEB'] + SizeScalar*SizeE[3]['SB']) # 2v c=3 binding
                v2c3H = (Lambda+1)*(SizeLOE*SizeE[3]['EH'] + SizeProof*SizeE[3]['SH']) # 2v c=3 hiding
                GlobalEncodingsSize = v1c2B + v1c2H + v1c3H + v1c4B + v1c4H + v2c3B + v2c3H
                GlobalBobComsSize = GlobalEncodingsSize + 6*(1+theta)*(bitSizeOf(param['n'])+param['logq']) #
                    ↪ adding size of the b,b',l,l',d,d', and the theta u*, e1*, e2* and rho cleartexts in \Rq sent by
                    ↪ Bob
                print(" EPS_s: 2^{-"+str(epss)+"} -- EPS_c: 2^{-"+str(epsc)+"} -- Lambda: "+str(Lambda) + "
                    ↪  -- #Commitments: "+ str(nuc)
                   +" -- Bob Communic. Size: "+str(math.ceil(GlobalBobComsSize/MB))+" MB")

def get_encodings_fig():
    # figure with number of encodings and scalars per commitment
    print("\n**********\n Figure 1 data: \n")
    for epsc in EpsC:
        print("epsc = "+str(epsc))
        SizeE = encodings(epsc, 6, 4, epsc+8, 11)
        print("K: "+str(SizeE[2]["K"])+" & "+str(SizeE[3]["K"])+" & "+str(SizeE[4]["K"])+" \\\\ ")
        print("M: "+str(SizeE[2]["M"])+" & "+str(SizeE[3]["M"])+" & "+str(SizeE[4]["M"])+" \\\\ ")
        print("& \#Equations (hiding case) & "+ str(SizeE[2]["EH"]) +" & "+ str(SizeE[3]["EH"]) +" & "+ str(
            ↪ SizeE[4]["EH"]) +" \\\\")
        print("& \#Secrets (hiding case) & "+ str(SizeE[2]["SH"]) +" & "+ str(SizeE[3]["SH"]) +" & " + str(SizeE
            ↪ [4]["SH"])+ " \\\\")
        print("& \#Encodings for $1v$ Pol. & "+ str(SizeE[2]["1vEB"]) +" & "+ str(SizeE[3]["1vEB"]) +" & "+ str(
            ↪ SizeE[4]["1vEB"]) + " \\\\")
        print("& \#Encodings for $2v$ Pol. & "+ str(SizeE[2]["2vEB"]) +" & "+ str(SizeE[3]["2vEB"]) +" & " + str
            ↪ (SizeE[4]["2vEB"]) + " \\\\")

get_application_results()
# get_encodings_fig()
```