Efficient Protocols for Oblivious Linear Function Evaluation from Ring-LWE[⋆]

Carsten Baum[1], Daniel Escudero[1], Alberto Pedrouzo-Ulloa[2], Peter Scholl[1] ✉, and Juan Ramón Troncoso-Pastoriza[3]

[1] Aarhus University, Aarhus, Denmark
{cbaum,escudero,peter.scholl}@cs.au.dk

[2] University of Vigo, Vigo, Galicia, Spain
apedrouzo@gts.uvigo.es

[3] EPFL, Lausanne, Switzerland
juan.troncoso-pastoriza@epfl.ch

Abstract. An oblivious linear function evaluation protocol, or OLE, is a two-party protocol for the function $f(x) = ax + b$, where a sender inputs the field elements a, b, and a receiver inputs x and learns $f(x)$. OLE can be used to build secret-shared multiplication, and is an essential component of many secure computation applications including general-purpose multi-party computation, private set intersection and more.

In this work, we present several efficient OLE protocols from the ring learning with errors (RLWE) assumption. Technically, we build two new passively secure protocols, which build upon recent advances in homomorphic secret sharing from (R)LWE (Boyle et al., Eurocrypt 2019), with optimizations tailored to the setting of OLE. We upgrade these to active security using efficient amortized zero-knowledge techniques for lattice relations (Baum et al., Crypto 2018), and design new variants of zero-knowledge arguments that are necessary for some of our constructions.

Our protocols offer several advantages over existing constructions. Firstly, they have the lowest communication complexity amongst previous, practical protocols from RLWE and other assumptions; secondly, they are conceptually very simple, and have just one round of interaction for the case of OLE where b is randomly chosen. We demonstrate this with an implementation of one of our passively secure protocols, which can perform more than 1 million OLEs per second over the ring $\mathbb{Z}_m$, for a 120-bit modulus m, on standard hardware.

1 Introduction

Oblivious linear function evaluation, or OLE, is a two-party protocol between a sender, with input $a, b \in \mathbb{F}$, and a receiver, who inputs $x \in \mathbb{F}$ and receives $y = ax + b$. OLE is an arithmetic generalization of oblivious transfer to a larger field $\mathbb{F}$, since OLE over $\mathbb{F}_2$ can be seen as equivalent to oblivious transfer on the messages z_0, z_1 by setting $a = z_0 + z_1$ and $b = z_0$, so the receiver learns $y = z_x$. Similarly to oblivious transfer, OLE can be used in constructions of secure two-party and multi-party computation, and is particularly useful for the setting of securely computing arithmetic circuits over $\mathbb{F}$ [23,20,3,19], where OT tends to be less efficient. As well as general secure computation protocols, OLE can be used to carry out specific tasks like private set intersection [22], secure matrix multiplication and oblivious polynomial evaluation [28,30].

OLE can be constructed from a range of "public-key" type assumptions. In the simplest, folklore construction, the receiver encrypts its input x using a linearly homomorphic encryption scheme and gives this to the sender. Using the homomorphic properties of the scheme, the sender computes an encryption of $y = ax + b$ and sends this back to the receiver to decrypt. This approach can be instantiated with Paillier encryption or lattice-based encryption based on the learning with errors (LWE) [29] or RLWE assumptions [26], and has been implicitly used in several secure multi-party computation protocols [8,24,27]. There are also constructions of OLE from coding-theoretic assumptions [28,23,21] which mostly rely on the hardness of

[⋆] This work is the accepted version of a paper published in the proceedings of SCN 2020. The final published version is: C. Baum, D. Escudero, A. Pedrouzo-Ulloa, P. Scholl, J.R. Troncoso-Pastoriza, "Efficient Protocols for Oblivious Linear Function Evaluation from Ring-LWE," Lecture Notes in Computer Science, vol 12238. Springer, Cham. The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-57990-6_7.

decoding Reed-Solomon codes in certain parameter regimes with a high enough noise rate. These constructions are asymptotically efficient, but so far have not been implemented in practice, to the best of our knowledge. For the special (and easier) case of *vector-OLE*, which is a large batch of many OLEs with the same input x from the receiver, there are efficient constructions from more standard coding-theoretic assumptions over general codes, which also have good performance in practice [3,10,30,11].

Despite the fact there are many existing constructions of OLE, either implicit or explicit in the literature, very few of these works study the practical efficiency of OLE in its own right (except for the special case of vector-OLE). Instead, most of the aforementioned works either focus on the efficiency of higher-level primitives such as secure multi-party computation, or mainly discuss asymptotic efficiency rather than performance in practice. In this work, we advocate for the practical study of OLE as a *standalone primitive*. This has the benefits that it can be plugged into any higher-level application that needs it in a modular way, potentially simplifying analysis and security proofs compared with a more monolithic approach.

1.1 Our Contributions

We present and study new OLE protocols with security based on the ring learning with errors (RLWE) assumption, with passive and active security. Our passively secure protocols are very simple, consisting of just one message per party, and our most efficient variant achieves the lowest communication complexity of any practical (implemented) OLE protocol we are aware of, requiring around half the bandwidth of previous solutions. We add active security using zero-knowledge proofs, which have a low amortized complexity when performing a large number of OLEs, giving only a small communication overhead over the passive protocols. To adapt existing zero-knowledge proof techniques to our protocols, we have to make several modifications, and describe a new amortized proof of knowledge that can be used to show a batch of *secret-key* (R)LWE ciphertexts is well-formed (previous techniques only apply to *public-key* ciphertexts).

We have implemented and benchmarked our most efficient passively secure protocol, and show it can compute more than 1 million OLEs per second on a standard laptop, over a ≈ 120-bit ring $\mathbb{Z}_m$ where m is the product of two CPU word-sized primes. The communication cost per OLE is around 4 elements of $\mathbb{Z}_m$ per party, and the amortized complexity of our actively secure protocol is almost the same, when computing a large enough number of OLEs. This is almost half the communication cost of previous protocols based on RLWE, and less than 25% of the cost of an actively secure protocol based on oblivious transfer and noisy Reed-Solomon encodings [21].

1.2 Outline

In Section 1.3 below, we present an overview of the main techniques in our constructions. We then describe some preliminaries in Section 2. Section 3 contains our OLE protocols based on public-key RLWE encryption, which only require a standard public key infrastructure as a setup assumption. In Section 4, we present more efficient protocols which reduce communication using secret-key encryption, and a more specialized setup assumption. In the full version of this work, we give additional details on the zero-knowledge arguments which are used to make the previous protocols actively secure. Finally, in Section 5, we analyze the concrete efficiency of our solutions, compare this with previous OLE protocols, and present implementation results for our most efficient passively secure protocol.

1.3 Techniques

Our protocols construct a symmetric variant of OLE, where one party, Alice, inputs a field element $u \in \mathbb{F}$, the other party, Bob, inputs $v \in \mathbb{F}$, and the parties receive random values α and β (respectively) such that $\alpha + \beta = u \cdot v$. This can easily be used to construct an OLE by having the sender, say Alice, one-time-pad encrypt her additional input using α, allowing Bob to correct his output accordingly. In this formulation, OLE is also equivalent to producing an additive secret-sharing of the product of two private inputs; this type of secret-shared multiplication is an important building block in multi-party computation protocols, for instance in constructing Beaver multiplication triples [7]. In our protocols, we first create OLEs over a large polynomial ring $\mathcal{R}_m = \mathbb{Z}_m[X]/(X^N + 1)$, which comes from the RLWE assumption, and then convert each OLE over $\mathcal{R}_m$ to a batch of N OLEs over $\mathbb{Z}_m$, for some prime modulus m, using packing techniques from homomorphic encryption [31].

Our point of departure is the recent *homomorphic secret sharing* scheme by Boyle et al. [13], based on LWE or RLWE. Homomorphic secret sharing is a form of secret sharing in which shares can be computed upon non-interactively, such that the parties end up with an *additive* secret sharing of the result of the computation. HSS was first constructed under the DDH assumption [12] and variants of threshold and multi-key fully homomorphic encryption [18], followed by the more efficient lattice-based construction of [13], which supports homomorphic computation of branching programs (or, "restricted multiplication" circuits where every multiplication gate must involve at least one input wire). Note that any "public-key" type two-party HSS scheme that supports multiplication leads to a simple OLE protocol: each party sends a secret-sharing of its input, then both parties multiply the shares to obtain an additive share of the product.

Efficient OLE from a public-key setup. Our first construction can be seen as taking the HSS scheme of Boyle et al. and optimizing it for the specific functionality of OLE. When plugging in their scheme to perform OLE, a single share from one party consists of two RLWE ciphertexts: one encrypting the message, and one encrypting a function of the secret key, which is needed to perform the multiplication. Our first observation is that, in the setting of OLE where we have two parties who each have one of the inputs to be multiplied, we can reduce this to just *one ciphertext per party*, where Alice sends an encryption of her input u multiplied by a secret key, and Bob sends an encryption of his input. Both of these ciphertexts, including the one dependent on the secret key, can be created from a standard public-key infrastructure-like setup where Alice and Bob have each others' RLWE public keys, thanks to a weak KDM security property of the scheme. This gives a communication complexity of two $\mathcal{R}_q$ elements per party, for a RLWE ciphertext modulus q, to create a single ring-OLE over $\mathcal{R}_m$. We can also obtain a further saving by sending one party's ciphertext at a smaller modulus $p < q$.

Reducing communication with a dedicated setup. Our second protocol considers a different setup assumption, where the parties are assumed to have access to a single OLE over $\mathcal{R}_q$, which gives them secret shares of the product of two RLWE secret keys. With this, we are able to replace the public-key RLWE ciphertexts from the previous protocol with *secret-key* ciphertexts, which can be of size just one ring element instead of two. This cuts the overall communication in half, and also reduces computational costs.

Achieving active security. To obtain security against active corruptions, we need to ensure that both parties' RLWE ciphertexts are correctly generated, in particular, that the small "error" polynomials used as encryption randomness were generated correctly (and not too large). For a public key RLWE encryption, this boils down to proving knowledge of a short vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, such that $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{c}$ where $\boldsymbol{A}, \boldsymbol{c}$ are public values defined by the RLWE public key and ciphertext, respectively. In practice, we do not know efficient methods of proving the above statement. Instead, we can obtain good *amortized* efficiency when proving knowledge for *many* such relations of the form

$$\boldsymbol{A}\overline{\boldsymbol{s}}_i = \boldsymbol{c}_i \tag{1}$$

for the same matrix $\boldsymbol{A}$, where now the secret $\overline{\boldsymbol{s}}_i$ may have slightly larger coefficients than the original secret $\boldsymbol{s}_i$. This overhead is known as the *soundness slack* parameter, and comes from the fact that a dishonest prover can sometimes make the proof succeed even when $\boldsymbol{s}_i$ is slightly larger than the claimed bound. Efficient amortized proofs for (1) have been given in several works [25,16,5,15], most recently with a communication overhead that is *independent* of the number of relations being proven [4].

Proving correctness of a batch of public-key RLWE ciphertexts can be essentially done by proving a batch of relations of the form in (1), allowing use of these efficient amortized proofs. To achieve active security in our public-key OLE protocol, we use a slightly modified version of the proof from [4], by allowing different size bounds to be proven for different components of $\boldsymbol{s}_i$. This gives us tighter parameters for the encryption scheme.

On the other hand, for our second protocol, things are not so straightforward. To see why, recall that a batch of secret-key RLWE ciphertexts have the form:

$$(a_i, a_i \cdot s + e_i + (q/p) \cdot x_i) \tag{2}$$

Here, a_i is a random element in the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, e_i is a small error value in $\mathcal{R}_q$, and $s \in \mathcal{R}_q$ is the secret key. We want to prove that both s and e_i have small coefficients.

The problem is, since a_i is different for each ciphertext, these cannot be expressed in the form of (1), since they are not linear in a fixed public value. This was *not* the case for the public-key setting, where every ciphertext is linear in the fixed public key; here, by switching to a secret-key encryption scheme to improve efficiency, we can unfortunately no longer apply the amortization techniques of [4].

Furthermore, there is a second obstacle, since we now have a special preprocessing phase which gives out shares of $s_A \cdot s_B$, for the parties' RLWE secret keys s_A and s_B. These must be the *same* secret keys that are used to produce the encryptions, and to ensure this, we also have to tie these together into the ZK proof statement.

To work around these issues, we perform two steps. Firstly, we modify the preprocessing so that each party gets a *commitment* to its secret key, under a suitable homomorphic commitment scheme (which can also be based on lattices [6]). We then design a new proof of knowledge, which proves knowledge of short (s, e_i, x_i) satisfying (2) with similar amortized efficiency to the proof from [4] for (1). Our proof simultaneously guarantees the secret s is the same s that was committed to in the preprocessing, leveraging the homomorphic properties of the commitment scheme.

2 Preliminaries

In this section we introduce the basic notation we will need throughout our work. As basic notation, we write $\boldsymbol{\alpha} \star \boldsymbol{\beta}$ to denote the component-wise product of the vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}$.

Rings & Rounding. Let q be an odd integer and $N = 2^r$ be a power of two. We define the ring $\mathcal{R} := \mathbb{Z}[X]/\langle X^N + 1\rangle$ as well as $\mathcal{R}_q = \mathcal{R}/\langle q\rangle$ as the reduction of the polynomials of $\mathcal{R}$ modulo q. Representing the coefficients of $f \in \mathcal{R}_q$ uniquely by its representatives from $[-(q-1)/2, (q-1)/2]$ we define $\|f\|_\infty$ as the largest norm of any coefficient of f when considered over the above interval. We define by $\mathcal{U}(R)$ the uniform distribution over the finite set R and furthermore let $S_\beta = \{x \in \mathcal{R} \mid \|x\|_\infty \leq \beta\}$.

Let $n, k \in \mathbb{N}^+$. In this work we consider the computational problems Ring-LWE, Module-LWE and Module-SIS. Their full description can be found in the full version of this manuscript.

We define by $\lfloor f \rceil_p$ the scaling of each coefficient of f by p/q over the reals and then rounding to the nearest element in $[-(p-1)/2, (p-1)/2]$ respectively. A simple but useful result we will use throughout our protocols is the following.

Lemma 1. *Let $p|q$, $\boldsymbol{x} \leftarrow \mathcal{R}_q^n$ and $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \bmod q$ for some $\boldsymbol{e} \in \mathcal{R}_q^n$ with $\|\boldsymbol{e}\|_\infty < B < q/p$. Then* $\Pr[\lfloor \boldsymbol{y} \rceil_p \neq \lfloor \boldsymbol{x} \rceil_p \bmod p] \leq \frac{2npNB}{q}$

Gaussian Distributions and Simulatability. We denote by $\rho^m_{\boldsymbol{v},\sigma}(\boldsymbol{x})$ the continuous normal distribution over $\mathbb{R}^m$ centered at $\boldsymbol{v} \in \mathbb{R}^m$ with standard deviation $\sigma \in \mathbb{R}$. If $\boldsymbol{v} = \boldsymbol{0}$ then we just write $\rho^m_\sigma(\boldsymbol{x})$. For a countable set $S \subset \mathbb{R}^m$ we furthermore define $\rho^m_\sigma(S) = \sum_{\boldsymbol{x}\in S} \rho^m_\sigma(\boldsymbol{x})$. Finally, we define the discrete normal distribution over $\mathbb{Z}^m$ centered at $\boldsymbol{v} \in \mathbb{Z}^m$ with standard deviation $\sigma \in \mathbb{R}^m$ as $\mathcal{D}^m_{\boldsymbol{v},\sigma}(\boldsymbol{x}) = \rho^m_{\boldsymbol{v},\sigma}(\boldsymbol{x})/\rho^m_\sigma(\mathbb{Z}^m)$.

Throughout this work we apply $\mathcal{D}$ to vectors from $\mathcal{R}^k$ in which case we mean that $\mathcal{D}_\sigma(\boldsymbol{x})^k = \mathcal{D}^{Nk}_\sigma(\overline{\boldsymbol{x}})$ with $\overline{\boldsymbol{x}} \in \mathbb{Z}^{Nk}$ being the coefficient-wise embedding of $\mathcal{R}^k$ into $\mathbb{Z}^{Nk}$. We consider sampling $\mathcal{R}$-elements from $\mathcal{D}_\sigma$ as sampling each coefficient independently from this distribution.

Ring-LWE based encryption scheme. In this work we use basic ideas from RLWE-based encryption, specially in our public-key based construction from Section 3. We describe here a simplified version of the public-key encryption scheme from [26], which we refer to as LPR. We consider two distributions $\mathcal{D}_{\mathsf{sk}}$ and $\mathcal{D}$ over $\mathcal{R}_q$, bounded (with overwhelming probability) by B_{sk} and B_{err}, respectively. The key generation, encryption and decryption procedures are defined as follows:

$\mathsf{Gen}(a)$**:** On input a public random $a \in \mathcal{R}_q$, first sample $s \leftarrow \mathcal{D}_{\mathsf{sk}}$ and $e \leftarrow \mathcal{D}$. Output $\mathsf{sk} = (s)$ and $\mathsf{pk} = (a, b)$ where $b = a \cdot s + e$.

$\mathsf{Enc}_{p,q}(\mathsf{pk}, x)$**:** On input $\mathsf{pk} \in \mathcal{R}^2_q$ and $x \in \mathcal{R}_p$, sample $w, e_0, e_1 \leftarrow \mathcal{D}$ and output (c_0, c_1), where $c_1 = -a \cdot w + e_1$ and $c_0 = b \cdot w + e_0 + (q/p) \cdot x$.

$\mathsf{Dec}(\mathsf{sk}, (c_0, c_1))$**:** Compute $x' = c_0 + s \cdot c_1 \bmod q$, and output $x = \lfloor x' \rceil_p \bmod p$.[4] Notice that this works if the total noise $e = s \cdot e_1 + e \cdot w + e_0$ is bounded by $p/2q$.

On top of these standard procedures, we also use an algorithm KDMEnc which produces an encryption of $x \cdot s$, where s is the secret key. As observed in [14,13], this can be done using only the public key by adding the message to the second component of an encryption of zero.

$\mathsf{KDMEnc}_{p,q}(\mathsf{pk}, x)$**:** Sample $w, e_0, e_1 \leftarrow \mathcal{D}$ and output (c_0, c_1), where $c_1 = (q/p) \cdot x - a \cdot w + e_1$ and $c_0 = b \cdot w + e_0$.

Commitments and Zero-Knowledge Arguments. In this work, in order to achieve active security, we make extensive use of commitments schemes and zero knowledge arguments of knowledge. We refer the reader to the full version of this manuscript for full definitions of these cryptographic notions. Here, we only introduce the basic notation.

Commitment Schemes. We consider an additively homomorphic statistically hiding commitment scheme, which we denote by a tuple $C = (\mathsf{KG}, \mathsf{Com}, \mathsf{Open})$. In this work we mainly use two different commitment schemes, namely the somewhat additively homomorphic commitment scheme of Baum et al. [6] (denoted as $C = (\mathsf{KG}, \mathsf{Com}, \mathsf{Open})$) as well as a compressing statistically secure commitment scheme $C_{\mathsf{aux}} = (\mathsf{KG}_{\mathsf{aux}}, \mathsf{KG}_{\mathsf{aux}}, \mathsf{Open}_{\mathsf{aux}})$.

One can easily instantiate C_{aux} either using the Random Oracle or [17]. The scheme of [6] is only somewhat homomorphic, meaning that it only supports a limited number of addition of commitments due to the growth of r. More details on the used commitment scheme can be found in the full version.

Zero-Knowledge Arguments of Knowledge (ZKA). Let $\mathcal{R}$ be an NP relation. For $(pp, x, w) \in \mathcal{R}$ we call pp the public parameter, x the statement and w the witness. A Zero-Knowledge Proof of Knowledge for $\mathcal{R}$ is an interactive protocol Π between a PPT prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ satisfying completeness, soundness against bounded malicious provers and honest-verifier zero-knowledge. The actual zero-knowledge arguments that are used with respect to the commitment scheme C can be found in the full version.

[4] Our protocols do not directly use the decryption algorithm, but our simulator in the proof of Theorem 2 does.

Oblivious Linear Function Evaluation. The functionality we implement in this work is oblivious linear evaluation (OLE), which, in a nutshell, consists of producing an additive sharing of a multiplication. A bit more precisely, $\mathcal{P}_{\mathsf{Alice}}$ and $\mathcal{P}_{\mathsf{Bob}}$ have each one secret input $v \in \mathcal{R}_m$ and $u \in \mathcal{R}_m$, respectively, and their goal is to get additive random shares of the product $u \cdot v$. The formal description of the functionality appears in the full version of this manuscript.

SIMD for Lattice-based Primitives. In this work we exploit plaintext packing techniques used in homomorphic encryption [31] based on the Chinese remainder theorem: We choose $m = 1 \bmod (2N)$ such that the polynomial $X^N + 1$ splits completely into a product of linear factors modulo m. This implies that $\mathcal{R}_m$ is isomorphic to N copies of $\mathbb{Z}_m$, so a single OLE over the ring $\mathcal{R}_m$ can be directly used to obtain a batch of N OLEs over $\mathbb{Z}_m$.

3 OLE from PKI Setup

In this section we present our first OLE construction, which is particularly simple and efficient. Furthermore, the only setup required is a correlated form of public key infrastructure in which $\mathcal{P}_{\mathsf{Alice}}$ and $\mathcal{P}_{\mathsf{Bob}}$ have each a secret/public key pair for the LPR scheme, where the $a \in \mathcal{R}_q$ component of the public key is the same for both. This can be seen as a PKI setup in which the public keys are derived using some public randomness. The precise functionality $\mathcal{F}_{\mathsf{PKI}}$ is given in the full version of this manuscript.

Our protocol, $\Pi_{\mathsf{OLE\text{-}pk}}$, can be found in Fig. 1. The passively secure version $\Pi_{\mathsf{OLE\text{-}pk}}^{\mathsf{passive}}$ is obtained from the active one by removing the zero knowledge arguments, whose steps are framed in the description of the protocol. To provide a high level idea of our construction, we first recall the main techniques from the homomorphic secret-sharing scheme of Boyle et al. [13]. Suppose two parties have additive secret shares of a RLWE secret key $s \in \mathcal{R}_q$, and are also given secret shares modulo q of x, $x \cdot s$ and a public ciphertext $\boldsymbol{c}_y = (c_0, c_1) = \mathsf{Enc}(\mathsf{pk}, y)$, for some messages x, y. Boyle et al. observed that if each party *locally* decrypts $\boldsymbol{c}_y$ using its shares, denoted $[x], [x \cdot s]$, we have:

$$[x] \cdot c_0 + [x \cdot s] \cdot c_1 = [x \cdot (c_0 + c_1 \cdot s)] \approx [(q/p) \cdot x \cdot y].$$

Applying the rounding operation from decryption on the above shares then gives *exact* additive shares of $x \cdot y$, provided the error is much smaller than q/p.

To create the initial shares of x and $x \cdot s$, it is enough to start with shares of s and ciphertexts encrypting $x, x \cdot s$, since each ciphertext can then be locally decrypted to obtain shares of these values. Boyle et al. also described a variant which removes the need for encryptions of $x \cdot s$, but at the cost of an additional setup assumption involving shares of s^2.

Our OLE protocol from this section builds upon this blueprint, with some optimizations. First, we observe that in the two-party OLE setting, it is not necessary to give out $\mathsf{Enc}(\mathsf{pk}, x)$ to obtain shares of x, since one of the parties always knows x so they can simply choose these shares to be x and 0. (This is in contrast to the homomorphic secret-sharing setting, where the evaluating parties may be a set of servers who did not provide inputs.) Since we only do one multiplication, it's therefore enough to give out the two ciphertexts $\boldsymbol{c}_x = \mathsf{Enc}(\mathsf{pk}, x \cdot s)$ and $\boldsymbol{c}_y = \mathsf{Enc}(\mathsf{pk}, y)$, compared with four ciphertexts used in the HSS scheme from [13]. Since both ciphertexts can be easily generated from the public-key setup, this leads to a very simple protocol where each party (in parallel) sends a single message that is either an encryption of its input, or its input times s.

As an additional optimization, we show that the second ciphertext encrypting y can be defined at a smaller modulus p instead of q, since we only care about obtaining the result modulo $m < p$, which saves further on bandwidth.[5]

[5] This optimization is possible since we skip the "modulus lifting" step from [13], which is only needed when doing several repeated multiplications.

Protocol $\Pi_{\mathsf{OLE\text{-}pk}}$

We use moduli $q > p > m$, where $m|p$ and $p|q$, and m is the final modulus of inputs and outputs.

1. *Setup.* The parties call $\mathcal{F}_{\mathsf{PKI}}$, so that both parties obtain $\mathsf{pk} = (a, b) \in \mathcal{R}_q^2$, while $\mathcal{P}_{\mathsf{Alice}}$ gets $s_{\mathsf{Alice}} \in \mathcal{R}_q$ and $\mathcal{P}_{\mathsf{Bob}}$ gets $s_{\mathsf{Bob}} \in \mathcal{R}_q$.
2. *First Message.* On input $\boldsymbol{u} \in \mathcal{R}_m^n$ from $\mathcal{P}_{\mathsf{Bob}}$:
   (a) $\mathcal{P}_{\mathsf{Bob}}$ sends $(\boldsymbol{c}_0, \boldsymbol{c}_1) = \mathsf{KDMEnc}_{p,q}(\mathsf{pk}, \boldsymbol{u})$ to $\mathcal{P}_{\mathsf{Alice}}$:
   (b) The parties engage in a zero-knowledge argument for the relation $\mathcal{R}_{\mathsf{Bob}}^{\mathsf{pk}}$ with $\mathcal{P}_{\mathsf{Alice}}$ as the verifier and $\mathcal{P}_{\mathsf{Bob}}$ as the prover. If this fails then the parties abort.
   (c) $\mathcal{P}_{\mathsf{Alice}}$ computes $\boldsymbol{\rho}_{\mathsf{Alice}} = \lfloor s_{\mathsf{Alice}} \cdot \boldsymbol{c}_1 \rceil_p$ and $\mathcal{P}_{\mathsf{Bob}}$ computes $\boldsymbol{\rho}_{\mathsf{Bob}} = \lfloor \boldsymbol{c}_0 + s_{\mathsf{Bob}} \cdot \boldsymbol{c}_1 \rceil_p$ (it should hold that $\boldsymbol{\rho}_{\mathsf{Alice}} + \boldsymbol{\rho}_{\mathsf{Bob}} = s \cdot \boldsymbol{u} \bmod p$)
3. *Second Message.* On input $\boldsymbol{v} \in \mathcal{R}_m^n$ from $\mathcal{P}_{\mathsf{Alice}}$:
   (a) $\mathcal{P}_{\mathsf{Alice}}$ sends $(\boldsymbol{d}_0, \boldsymbol{d}_1) = \mathsf{Enc}_{m,p}(\mathsf{pk}, \boldsymbol{v})$ to $\mathcal{P}_{\mathsf{Bob}}$
   (b) The parties engage in a zero-knowledge argument for the relation $\mathcal{R}_{\mathsf{Alice}}^{\mathsf{pk}}$ with $\mathcal{P}_{\mathsf{Bob}}$ as the verifier and $\mathcal{P}_{\mathsf{Alice}}$ as the prover. If this fails then the parties abort.
   (c) $\mathcal{P}_{\mathsf{Alice}}$ outputs $\boldsymbol{\alpha} = \lfloor \boldsymbol{d}_1 \star \boldsymbol{\rho}_{\mathsf{Alice}} \rceil_m$.
   (d) $\mathcal{P}_{\mathsf{Bob}}$ outputs $\boldsymbol{\beta} = \lfloor \boldsymbol{d}_0 \star \boldsymbol{u} + \boldsymbol{d}_1 \star \boldsymbol{\rho}_{\mathsf{Bob}} \rceil_m$.
   We should now have $\boldsymbol{\alpha} + \boldsymbol{\beta} = \boldsymbol{u} \star \boldsymbol{v} \bmod m$.

Fig. 1: Actively secure OLE protocol from a PKI setup. The passively secure version of the protocol is obtained by removing the framed steps.

The protocol described above is passively secure, but an active adversary can break the security of this construction by sending incorrectly-formed ciphertexts. Due to our simple communication pattern this turns out to be the only potential source of attack, which we rule out by having the parties prove, in zero knowledge, that their ciphertexts are correctly formed.

3.1 Passive Security

We now proceed to the security proof of our protocol $\Pi_{\mathsf{OLE\text{-}pk}}^{\mathsf{passive}}$, which consists of protocol $\Pi_{\mathsf{OLE\text{-}pk}}$ in Fig. 1 without the zero knowledge arguments framed in the protocol description.

Our proof requires that a random element of $\mathcal{R}_q$ is invertible with high probability. As we will see, this technicality allows the simulator to "solve equations", matching real and ideal views. For our choice of parameters this is always the case, and for this we make use of the following lemma whose proof appears in the full version.

Lemma 2. *Let $q = \prod_{i=1}^k p_i$, where each p_i is an ℓ-bit prime. If the polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree N used to define $\mathcal{R}_q$ splits completely mod p_i as $f(x) = \prod_{j=1}^N f_{ij}(x) \bmod p_i$, where each $f_{ij}(x)$ is linear, then the probability that a uniformly random element from $\mathcal{R}_q$ is* not *invertible is upper bounded by $\frac{N \cdot k}{2^\ell}$.*

Given the above, the probability that at least one component of a vector in $\mathcal{R}_q^n$ is not invertible is upper bounded by $n \cdot N \cdot k \cdot 2^{-\ell}$. For all our parameter sets in Section 5, this quantity is below $2^{-\lambda}$ for $\lambda \approx 36$, which is good enough for our purposes since we need it only as an artefact for the proof and it does not lead to any concrete attack or leakage.[6] We also use invertibility to argue correctness of the protocol, as it is required for being able to use Lemma 1 in our protocols. If this probability is not good enough for a certain application, the parties could use a PRF to rerandomize their shares so that this lemma can be applied without invertibility. However, in order to keep our exposition simple we do not discuss such extension.

Another simple but useful lemma for our construction is the following.

Lemma 3. *Assume that $p|q$. Given $y \in \mathcal{R}_p$, the set of $x \in \mathcal{R}_q$ such that $y = \lfloor x \rceil_p$ is given by $x = \left(\frac{q}{p}\right) \cdot y + e$ for $e \in \mathbb{Z} \cap (-q/2p, q/2p]$. In particular, the mapping $\mathcal{R}_q \to \mathcal{R}_p$ given by*

[6] This restriction can be easily overcome by modifying the definition of security against passive adversaries, allowing the adversary to choose its output. However, we prefer to stick to more standard security definitions.

$x \mapsto \lfloor x \rceil_p$ is a surjective regular mapping, meaning that every element in the codomain has an equal number of preimages.

Finally, we have the following proposition, concerning correctness of our construction. It follows as a corollary of Proposition 2 by setting the soundness slack parameter τ to be 1, so we defer the proof to that section.

Proposition 1. *Assume that $3 \cdot 2^{\kappa+1} \cdot n \cdot (mN)^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}} \leq p \leq \frac{q}{3 \cdot 2^{\kappa+1} \cdot n \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. Let $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{R}_m^n$ be the inputs to Protocol $\Pi_{\mathsf{OLE\text{-}pk}}^{\mathsf{passive}}$, and let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}_m^n$ be the outputs. Then, with probability at least $1 - 2^{-\kappa}$, $\boldsymbol{u} \star \boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta}$.*

With these tools at hand we proceed with the main result from this section.

Theorem 1. *Assume that $3 \cdot 2^{\kappa+1} \cdot n \cdot (mN)^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}} \leq p \leq \frac{q}{3 \cdot 2^{\kappa+1} \cdot n \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. Then protocol $\Pi_{\mathsf{OLE\text{-}pk}}^{\mathsf{passive}}$, which consists of protocol $\Pi_{\mathsf{OLE\text{-}pk}}$ without the underlined steps, realizes functionality $\mathcal{F}_{\mathsf{OLE}}$ in the $\mathcal{F}_{\mathsf{PKI}}$-hybrid model under the RLWE assumption.*

The proof of this Theorem is presented in the full version.

3.2 Active Security

As we saw in the previous section, the correctness of our construction relies on the different terms involved having a certain bound: The input $\boldsymbol{u}$ must be smaller than m, the noise terms used for the encryption have to be upper bounded by B_{err}, and the randomness $\boldsymbol{w}$ and $\boldsymbol{w}'$ used for the encryption must be less than B_{sk}. An actively corrupted party who chooses randomness outside these bounds can easily distinguish between the real and ideal executions.

To achieve active security, each party proves in zero-knowledge that the ciphertexts they send are correctly formed. We begin by analyzing the case of a corrupt $\mathcal{P}_{\mathsf{Bob}}$. Consider the message from $\mathcal{P}_{\mathsf{Bob}}$, which consists of a batch of ciphertexts

$$(\boldsymbol{c}_0, \boldsymbol{c}_1) = (b \cdot \boldsymbol{w} + \boldsymbol{e}_0, (q/p) \cdot \boldsymbol{u} - a \cdot \boldsymbol{w} + \boldsymbol{e}_1)$$

Rewriting this, $\mathcal{P}_{\mathsf{Bob}}$ has to prove knowledge of vectors (over $\mathcal{R}_q$) $\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{e}_0, \boldsymbol{e}_1$ satisfying

$$\underbrace{\begin{pmatrix} b & 1 & 0 & 0 \\ -a & 0 & 1 & q/p \end{pmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{pmatrix} \boldsymbol{w} & \boldsymbol{e}_0 & \boldsymbol{e}_1 & \boldsymbol{u} \end{pmatrix}^\top}_{\boldsymbol{S}} = \underbrace{\begin{pmatrix} \boldsymbol{c}_0 \\ \boldsymbol{c}_1 \end{pmatrix}}_{\boldsymbol{T}} \tag{3}$$

and $\|\boldsymbol{w}\|_\infty \leq B_{\mathsf{sk}}$, $\|\boldsymbol{u}\|_\infty \leq m$, $\|\boldsymbol{e}_0\|_\infty \leq B_{\mathsf{err}}$ and $\|\boldsymbol{e}_1\|_\infty \leq B_{\mathsf{err}}$. This can be written in matrix form as follows

$$\mathcal{R}_{\mathsf{Bob}}^{\mathsf{pk}} = \left\{ \begin{array}{r} (pp, u, w) = ((\mathcal{R}, q, n, \beta, \boldsymbol{A}), \boldsymbol{T}, \boldsymbol{S}) \mid (\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{T}) \in \mathcal{R}_q^{2\times 4} \times \mathcal{R}^{4\times n} \times \mathcal{R}_q^{2\times n} \\ \wedge \boldsymbol{AS} = \boldsymbol{T} \wedge \|\boldsymbol{s}_i\|_\infty \leq \beta_i \end{array} \right\}$$

where $\boldsymbol{s}_i$ is the i-th row of $\boldsymbol{S}$ and the bound vector is $\beta = (B_{\mathsf{sk}}, B_{\mathsf{err}}, B_{\mathsf{err}}, B_{\mathsf{msg}})$. Such type of statements can be proven efficiently using the amortized proof from [4], as we discuss more thoroughly in the full version.

We can similarly define a relation for the message $(\boldsymbol{d}_0, \boldsymbol{d}_1)$ that $\mathcal{P}_{\mathsf{Alice}}$ sends, and we call this relation $\mathcal{R}_{\mathsf{Alice}}^{\mathsf{pk}}$. We note however that in the proof of Theorem 1 we did not actually use any bound on the message $\boldsymbol{v}$, so we may exclude the bound $\|\boldsymbol{v}\|_\infty \leq m$ from this relation.

For the rest of this section we assume the existence of zero knowledge arguments of knowledge for the relations $\mathcal{R}_{\mathsf{Alice}}^{\mathsf{pk}}$ and $\mathcal{R}_{\mathsf{Bob}}^{\mathsf{pk}}$. Note that when proving knowledge of the relation $\mathcal{R}_{\mathsf{Bob}}^{\mathsf{pk}}$ or $\mathcal{R}_{\mathsf{Alice}}^{\mathsf{pk}}$ above, if the prover is malicious then our proof actually only guarantees that $\|\boldsymbol{s}_i\|_2 \leq \tau \cdot \beta_i$, where τ is the soundness slack parameter of the zero knowledge argument. We therefore need to choose our parameters with respect to the larger bounds, to ensure correctness of the protocol.

We begin with the following proposition, which shows that, under an appropriate choice of parameters, our protocol guarantees correctness. Its proof appears in the full version.

Proposition 2. *Assume that* $3 \cdot 2^{\kappa+1} \cdot n \cdot \tau \cdot (mN)^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}} \leq p \leq \frac{q}{3 \cdot 2^{\kappa+1} \cdot n \cdot \tau \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. *Let* $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{R}_m^n$ *be the inputs to Protocol* $\Pi_{\mathsf{OLE\text{-}pk}}$, *and let* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}_m^n$ *be the outputs. Assume that the relations* $\mathcal{R}^{\mathsf{pk}}_{\mathsf{Alice}}$ *and* $\mathcal{R}^{\mathsf{pk}}_{\mathsf{Bob}}$ *defined in Section 3.2 hold, but that at most one of them has slack parameter* τ.[7] *Then, with probability at least* $1 - 2^{-\kappa}$, $\boldsymbol{u} \star \boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta}$.

With this tool at hand, the security of the actively secure version of our protocol can be proven. The proof appears in the full version.

Theorem 2. *Assume that* $3 \cdot 2^{\kappa+1} \cdot n \cdot \tau \cdot (mN)^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}} \leq p \leq \frac{q}{3 \cdot 2^{\kappa+1} \cdot n \cdot \tau \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. *Protocol* $\Pi_{\mathsf{OLE\text{-}pk}}$ *realizes functionality* $\mathcal{F}_{\mathsf{OLE}}$ *under the RLWE assumption.*

4 OLE from Correlated Setup

Ciphertexts in the public key version of the LPR cryptosystem consist of two ring elements. However, in the secret key variant, we can reduce this to one element, since the first element is uniformly random so can be compressed using, for example, a PRG. Given this, a natural way of shaving off a factor of two in the communication complexity of our protocol from Section 3 would be to use secret key encryption instead of public key.

In this section we present an OLE protocol that instantiates precisely this idea. The communication pattern is very similar to the one from Protocol $\Pi_{\mathsf{OLE\text{-}pk}}$, in which there is a setup phase, then $\mathcal{P}_{\mathsf{Bob}}$ sends an encryption of his input $\boldsymbol{u}$ to $\mathcal{P}_{\mathsf{Alice}}$ (and proves in zero-knowledge its correctness for the actively secure version), and then $\mathcal{P}_{\mathsf{Alice}}$ does the same. The challenge, here, is that now, as we are using secret-key encryption to obtain his ciphertext in the first message, there is no way for Bob to encrypt $\boldsymbol{u}$ multiplied by the (combined) secret key.

To make this work, we replace the PKI setup functionality from the previous section with a more specialized setup, where $\mathcal{P}_{\mathsf{Bob}}$ gets $\sigma_{\mathsf{Bob}} \in \mathcal{R}_q$ and $\mathcal{P}_{\mathsf{Alice}}$ gets $\sigma_{\mathsf{Alice}} \in \mathcal{R}_q$ such that $s_{\mathsf{Alice}} \cdot s_{\mathsf{Bob}} = \sigma_{\mathsf{Alice}} + \sigma_{\mathsf{Bob}} \bmod q$. This can be seen as an OLE itself, where the values being multiplied are small RLWE secret keys; under this interpretation, our protocol can be seen as a form of "OLE extension" protocol. The intuition for why this setup is useful, is that Bob's secret-key ciphertext can now be distributively "decrypted" using the shares of $s_{\mathsf{Alice}} \cdot s_{\mathsf{Bob}}$, which (after rounding) leads to shares of $\boldsymbol{u} \cdot s_{\mathsf{Alice}}$. In the second phase, these shares are then used to "decrypt" Alice's ciphertext, giving shares of the product $\boldsymbol{u} \star \boldsymbol{v}$.

The setup functionality is described in the full version of this manuscript, where we present both the passive and active versions of the functionality, with the main difference being that in the active setting we must ensure that the corrupt party uses the same secret key for encrypting its input as the secret key distributed in the setup phase. Thus, in this case, when the corrupted party proves in zero knowledge the correctness of its encryption, it also proves that the secret key is the same as in the setup phase. This requires the setup functionality in the active case to output some extra information that allows us to "bind" the key from the setup with the key from the encryption sent, for which we use commitments. We discuss this in more detail when we look at active security in Section 4.2.

Our protocol is described in full detail in Fig. 2. As in Section 3, we present the full, actively secure version, but outline in a box those steps that are only necessary for active security.

4.1 Passive Security

The following proposition states that our construction satisfies correctness when the parties are honest, and follows from Proposition 4 in Section 4.2, which analyzes the case where the bounds satisfied by the values from one of the parties may not be sharp.

[7] That is, the bounds in one of the two relations have an extra factor of τ. This corresponds to what can be guaranteed for a corrupt party via the zero knowledge argument.

Protocol $\Pi_{\mathsf{OLE\text{-}sk}}$

We use moduli $q > p > m$, where m is the final modulus of inputs and outputs. We assume that m divides p and that p divides q.

1. *Setup phase.*
   (a) *Passive case.* $\mathcal{P}_{\mathsf{Alice}}, \mathcal{P}_{\mathsf{Bob}}$ each send (Sample, sid) to $\mathcal{F}_{\mathsf{setup}}$. $\mathcal{P}_{\mathsf{Alice}}$ obtains $s_{\mathsf{Alice}}, \sigma_{\mathsf{Alice}}$ while $\mathcal{P}_{\mathsf{Bob}}$ obtains $s_{\mathsf{Bob}}, \sigma_{\mathsf{Bob}}$.
   Active case. $\mathcal{P}_{\mathsf{Alice}}, \mathcal{P}_{\mathsf{Bob}}$ each send (Sample, sid) to $\mathcal{F}_{\mathsf{setup}}$. $\mathcal{P}_{\mathsf{Alice}}$ obtains $s_{\mathsf{Alice}}, \sigma_{\mathsf{Alice}}, r_{\mathsf{Alice}}, c_{\mathsf{Alice}}, c_{\mathsf{Bob}}$ and $\mathcal{P}_{\mathsf{Bob}}$ obtains $s_{\mathsf{Bob}}, \sigma_{\mathsf{Bob}}, r_{\mathsf{Bob}}, c_{\mathsf{Alice}}, c_{\mathsf{Bob}}$.
   (b) The parties sample two public random values $\boldsymbol{a}, \boldsymbol{a}' \in \mathcal{R}_q^n$.[a]
2. *First Message.* On input $\boldsymbol{u} \in \mathcal{R}_m^n$ from $\mathcal{P}_{\mathsf{Bob}}$:
   (a) $\mathcal{P}_{\mathsf{Bob}}$ samples a noise vector $\boldsymbol{e}_{\mathsf{Bob}} \leftarrow \mathcal{D}^n$ and sends $\boldsymbol{c} = \left(\frac{q}{p}\right) \cdot \boldsymbol{u} + (\boldsymbol{a} \cdot s_{\mathsf{Bob}} + \boldsymbol{e}_{\mathsf{Bob}}) \mod q$ to $\mathcal{P}_{\mathsf{Alice}}$.
   (b) The parties engage in a zero-knowledge argument for the relation $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Bob}}$ with $\mathcal{P}_{\mathsf{Alice}}$ as the verifier and $\mathcal{P}_{\mathsf{Bob}}$ as the prover with witness $(\boldsymbol{u}, \boldsymbol{e}_{\mathsf{Bob}}, s_{\mathsf{Bob}}, r_{\mathsf{Bob}})$. If this fails then the parties abort.
   (c) $\mathcal{P}_{\mathsf{Alice}}$ computes $\boldsymbol{\rho}_{\mathsf{Alice}} = \lfloor s_{\mathsf{Alice}} \cdot \boldsymbol{c} - \boldsymbol{a} \cdot \sigma_{\mathsf{Alice}} \rceil_p$.
   (d) $\mathcal{P}_{\mathsf{Bob}}$ computes $\boldsymbol{\rho}_{\mathsf{Bob}} = - \lfloor \boldsymbol{a} \cdot \sigma_{\mathsf{Bob}} \rceil_p$. It should now hold that $\boldsymbol{u} \cdot s_{\mathsf{Alice}} = \boldsymbol{\rho}_{\mathsf{Alice}} + \boldsymbol{\rho}_{\mathsf{Bob}} \mod p$.
3. *Second Message.* On input $\boldsymbol{v} \in \mathcal{R}_m^n$ from $\mathcal{P}_{\mathsf{Alice}}$.
   (a) $\mathcal{P}_{\mathsf{Alice}}$ samples a noise vector $\boldsymbol{e}_{\mathsf{Alice}} \leftarrow \mathcal{D}^n$ and sends $\boldsymbol{d} = \left(\frac{p}{m}\right) \cdot \boldsymbol{v} + (\boldsymbol{a}' \cdot s_{\mathsf{Alice}} + \boldsymbol{e}_{\mathsf{Alice}}) \mod p$ to $\mathcal{P}_{\mathsf{Bob}}$.
   (b) The parties engage in a zero-knowledge argument for the relation $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Alice}}$ with $\mathcal{P}_{\mathsf{Bob}}$ as the verifier and $\mathcal{P}_{\mathsf{Alice}}$ as the prover, with witness $(\boldsymbol{v}, \boldsymbol{e}_{\mathsf{Alice}}, s_{\mathsf{Alice}}, r_{\mathsf{Alice}})$. If this fails then the parties abort.
   (c) $\mathcal{P}_{\mathsf{Bob}}$ outputs, $\boldsymbol{\beta} = \lfloor \boldsymbol{u} \star \boldsymbol{d} - \boldsymbol{a}' \star \boldsymbol{\rho}_{\mathsf{Bob}} \rceil_m \mod m$.
   (d) $\mathcal{P}_{\mathsf{Alice}}$ outputs, $\boldsymbol{\alpha} = - \lfloor \boldsymbol{a}' \star \boldsymbol{\rho}_{\mathsf{Alice}} \rceil_m \mod m$. It should hold that $\boldsymbol{u} \star \boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta} \mod m$

[a] In practice this can be done by using a PRF with some pre-shared key. In our proofs we use a random oracle that can be programmed by the simulator.

Fig. 2: Actively secure OLE protocol based on RLWE. The passively secure version of the protocol is obtained by removing the framed steps.

Proposition 3. *Assume that* $2^{\kappa+1} \cdot n \cdot (mN)^2 \cdot B_{\mathsf{err}} \leq p \leq \frac{q}{2^{\kappa+1} \cdot n \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. *Let* $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{R}_m^n$ *be the inputs to Protocol* $\Pi_{\mathsf{OLE\text{-}sk}}$, *and let* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}_m^n$ *be the outputs. Then, with probability at least* $1 - 2^{-\kappa}$, $\boldsymbol{u} \star \boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta}$.

With this proposition, we proceed to the proof of security of our passively secure protocol.

Theorem 3. *Assume that* $m^2 \cdot B_{\mathsf{err}} \cdot 2^{\kappa+1} \cdot n \cdot N^2 \leq p \leq \frac{q}{2^{\kappa+1} \cdot n \cdot N^2 \cdot B_{\mathsf{sk}} \cdot B_{\mathsf{err}}}$. *Then protocol* $\Pi^{\mathsf{passive}}_{\mathsf{OLE\text{-}sk}}$, *which consists of protocol* $\Pi_{\mathsf{OLE\text{-}sk}}$ *without the underlined steps, realizes functionality* $\mathcal{F}_{\mathsf{OLE}}$ *in the* $\mathcal{F}_{\mathsf{setup}}$*-hybrid model under the RLWE assumption.*

The proof bears similarity with the proof of Theorem 1, and we defer it to the full version.

4.2 Active Security

An active adversary in the protocol $\Pi_{\mathsf{OLE\text{-}sk}}$ can cheat by sending incorrect messages. For example, a corrupt $\mathcal{P}_{\mathsf{Bob}}$ may send an incorrectly formed $\boldsymbol{c}$, and one can show that, in fact, by choosing $\boldsymbol{c}$ appropriately a corrupt $\mathcal{P}_{\mathsf{Bob}}$ may learn some information about $\mathcal{P}_{\mathsf{Alice}}$'s input $\boldsymbol{v}$. A similar attack can be carried out by a corrupt $\mathcal{P}_{\mathsf{Alice}}$. Hence, to achieve active security, we must ensure that the message $\boldsymbol{c}$ sent by $\mathcal{P}_{\mathsf{Bob}}$ and the message $\boldsymbol{d}$ sent by $\mathcal{P}_{\mathsf{Alice}}$ are computed honestly.

We implement zero knowledge arguments to show precisely these statements. $\mathcal{P}_{\mathsf{Bob}}$ proves that he knows $\boldsymbol{u}, \boldsymbol{e}$ and s_{Bob} of the appropriate sizes such that $\boldsymbol{c} = \left(\frac{q}{p}\right) \cdot \boldsymbol{u} + (\boldsymbol{a} \cdot s_{\mathsf{Bob}} + \boldsymbol{e}_{\mathsf{Bob}}) \mod q$, and $\mathcal{P}_{\mathsf{Alice}}$ proceeds similarly.

An additional technicality, however, is that s_{Bob} (and respectively s_{Alice}) has to be exactly the same value that was distributed during the setup phase. To enforce this, we consider a modified setup functionality for the actively secure setting that, on top of distributing $s_{\mathsf{Bob}} \cdot s_{\mathsf{Alice}} = \sigma_{\mathsf{Bob}} + \sigma_{\mathsf{Alice}}$, also distributes commitments to s_{Bob} and s_{Alice} that can be used in the relation of the zero knowledge argument.

Given that the protocol is essentially symmetric with respect to the roles of $\mathcal{P}_{\mathsf{Alice}}$ and $\mathcal{P}_{\mathsf{Bob}}$, from now on we focus on discussing the case of a corrupt $\mathcal{P}_{\mathsf{Bob}}$. A similar argument applies for the case of corrupt $\mathcal{P}_{\mathsf{Alice}}$. The message $\boldsymbol{c}$ that $\mathcal{P}_{\mathsf{Bob}}$ sends is formed by adding n RLWE samples to $\left(\frac{q}{p}\right) \cdot \boldsymbol{u}$, which is a scaled version of its input $\boldsymbol{u}$. Furthermore, the RLWE samples must be generated using the secret s_{Bob} distributed in the setup phase. As a result, the relation that $\mathcal{P}_{\mathsf{Bob}}$ will prove is

$$\mathcal{R}^{\mathsf{sk}}_{\mathsf{Bob}}(\tau) = \left\{ \begin{array}{c|l} (pp, u, w) = & \boldsymbol{c} = \left(\frac{q}{p}\right) \cdot \boldsymbol{u} + \boldsymbol{a} \cdot s + \boldsymbol{e} \bmod q \wedge \\ \left(\begin{array}{c} (\mathcal{R}, q, p, m, \beta, \mathsf{pk}, \boldsymbol{a}, \mathsf{com}_{\mathsf{Bob}}), \\ \boldsymbol{c}, (\boldsymbol{u}, \boldsymbol{e}, s, r) \end{array} \right) & \|\boldsymbol{u}\|_\infty \le \tau \cdot \beta_1 \wedge \|\boldsymbol{e}\|_\infty \le \tau \cdot \beta_2 \wedge \\ & \mathsf{Open}_{\mathsf{pk}}(\mathsf{com}_{\mathsf{Bob}}, s, r) = 1 \end{array} \right\}$$

and $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Alice}}$ can be defined similarly.[8] Here in the honest case $\mathcal{P}_{\mathsf{Bob}}$ starts with $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Bob}}$, but the guarantee given by the zero-knowledge argument will be for a substantially larger factor τ (see the full version). The relation essentially shows that the message that $\mathcal{P}_{\mathsf{Bob}}$ sends is well formed, and furthermore, that the s_{Bob} used for constructing this message is exactly the same as the one provided in the setup phase.

For the purpose of this section we assume the existence of zero knowledge arguments for the relations $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Alice}}$ and $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Bob}}$. We develop such results in the full version.

Now, to proceed with the security proof of our protocol, we first present the following proposition, which states that our construction satisfies correctness even when the bound on the parameters may have some slack. This is similar to Proposition 2 and its proof is deferred to the full version.

Proposition 4. *Assume that $2^{\kappa+1} \cdot n \cdot \tau \cdot (mN)^2 \cdot B_{\mathsf{err}} \le p \le \frac{q}{2^{\kappa+1} \cdot n \cdot \tau \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. Let $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{R}^n_m$ be the inputs to Protocol $\Pi_{\mathsf{OLE\text{-}sk}}$, and let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{R}^n_m$ be the outputs. Assume that the relations $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Alice}}$ and $\mathcal{R}^{\mathsf{sk}}_{\mathsf{Bob}}$ hold, but that at most one of them has slack parameter τ. Then, with probability at least $1 - 2^{-\kappa}$, $\boldsymbol{u} \star \boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta}$.*

Given this, we can prove the security of our actively secure OLE protocol, as stated in the following theorem. The proof appears in the full version.

Theorem 4. *Assume that $2^{\kappa+1} \cdot n \cdot \tau \cdot (mN)^2 \cdot B_{\mathsf{err}} \le p \le \frac{q}{2^{\kappa+1} \cdot n \cdot \tau \cdot N^2 \cdot B_{\mathsf{err}} \cdot B_{\mathsf{sk}}}$. Then protocol $\Pi_{\mathsf{OLE\text{-}sk}}$ realizes functionality $\mathcal{F}_{\mathsf{OLE}}$ in the $\mathcal{F}_{\mathsf{setup}}$-hybrid model under the RLWE assumption.*

5 Evaluation

In this section, we evaluate the efficiency of our OLE protocols, and compare this with protocols based on previous techniques. Firstly, we look at the communication complexity and compare this with other protocols. Then, in Section 5.2, we present implementation results for our passively secure secret-key protocol to demonstrate its practicality.

Choosing Parameters. We estimate parameters for our OLE protocols according to the correctness requirement in Proposition 2. For RLWE we use a ternary secret distribution (so, $B_{\mathsf{sk}} = 1$) a Gaussian error distribution with $\sigma = 3.19$ and $B_{\mathsf{err}} = 6\sigma$; the soundness slack parameter is $\tau = 1$ for passive protocols and $\tau \approx 24\sqrt{8Nn\kappa}$ otherwise. The statistical security parameter is $\kappa = 40$.

5.1 Comparison to Previous Protocols

Table 1 presents the communication complexity, measured from the protocol specifications, of our two public-key and secret-key OLE protocols, and compares this with two other protocols based on RLWE-based homomorphic encryption, either additively homomorphic (AHE) or somewhat

[8] As in public-key protocol from Section 3.2, $\mathcal{P}_{\mathsf{Alice}}$ does not need to prove the bound on her input $\boldsymbol{v}$.

Protocol	Security		Rounds[*]	Total comm. (bits)			
	passive	active		$\log m \approx 128$		$\log m \approx 64$	
				passive	active	passive	active
PK-OLE	RLWE	+ FS[†]	1	1516	1630	1004	1120
SK-OLE	RLWE	+ FS	1	758	815	502	560
AHE	RLWE	+ FS + LOE[‡]	2	1320	1476	800	956
SHE	RLWE	+ FS	2	3682	3682	2310	2310
RS	noisy encodings	–	8	4096	4096	2048	2048

[†] FS is Fiat-Shamir
[‡] LOE is linear-only encryption [9]
[*] 1 round means that each party sends one message simultaneously. 2 rounds either means that each party sequentially sends one message (for AHE), or one simultaneous message, twice in succession (for SHE).

Table 1: Comparison of the complexity of our OLE protocols with previous works based on homomorphic encryption

homomorphic (SHE), as well as a protocol based on noisy Reed-Solomon encodings (RS). As can be seen from the table, ours is the only protocol with just a single round of communication, where each party simultaneously sends just one message (as in a non-interactive key exchange), whereas both other protocols require two rounds. Our secret-key protocol, which requires some special preprocessing, has the lowest communication cost of all the protocols, with both passive and active security. Furthermore, compared with the previously most efficient protocol based on AHE with active security, our active protocols avoid the need for assuming linear-only encryption, which is a relatively strong and un-studied assumption, compared with standard RLWE.

A full description of these protocols can be found in the full version.

5.2 Experimental Results

We have implemented the passive version of the secret-key protocol (see in Fig. 2) in Go language, making use of the ring package provided by the lattigo library [1]. Our implementation features a full-RNS (Residue Number System) realization of all the protocol operations, using a moduli of 60-bit limbs. For comparison purposes, we have also implemented an AHE-based OLE protocol (we refer the reader to the full version for more details).

The execution times of the protocol steps were tested on a laptop with an Intel Core i7-8550U processor with 16GB RAM, running Arch Linux with kernel 5.6.4 and Go 1.14.2 The latency is not simulated, as it is highly dependent on the particular deployment; we include instead the communication complexity of the involved messages, from which the latency can be derived.

Parameter	Par. set 1	Par. set 2
q	360 bits (6 limbs)	480 bits (8 limbs)
p	240 bits (4 limbs)	360 bits (6 limbs)
m	60 bits (1 limb)	120 bits (2 limbs)
bit security	≈ 159	≈ 116
# OLEs	2097152	2097152

(a) Example parameter sets ($n = 128$ and $N = 16384$) and global run times for the passive case of Fig. 2 (uniformly random ternary secret keys $\{-1, 0, 1\}$ and Gaussian noise with $\sigma = 3.19$).

Bob	Par. set 1	Par. set 2	Alice	Par. set 1	Par. set 2
Step 2.(a)	462 ms	601 ms	Step 2.(c)	564 ms	817 ms
Step 2.(d)	533 ms	772 ms	Step 3.(a)	350 ms	479 ms
Step 3.(c)	263 ms	438 ms	Step 3.(d)	242 ms	412 ms
1st msg.	995 ms	1373 ms	1st msg.	564 ms	817 ms
2nd msg.	263 ms	438 ms	2nd msg.	591 ms	890 ms

(b) Run times in the passive case of Fig. 2 for the example parameter sets of Table 2a ($n = 128$ and $N = 16384$, uniformly random ternary secret keys $\{-1, 0, 1\}$ and Gaussian noise with $\sigma = 3.19$).

Table 2: Parameter sets and run times in the passive case of Fig. 2

Run time expressions for Bob and Alice
$T_{\mathsf{Bob}} = \max(T_{2.a} + T_{2.d}, T_{3.a} + T_{\boldsymbol{d}}) + T_{3.c}$
$T_{\mathsf{Alice}} = \max(T_{3.a}, T_{2.a} + T_{\boldsymbol{c}}) + T_{2.c} + T_{3.d}$

(a) Total run time expressions for Bob (T_{Bob}) and Alice (T_{Alice}).

Total time	600Mbit/s	1Gbit/s	10Gbit/s
Par. Set 1	2526 ms	2023 ms	1344 ms
Par. Set 2	3508 ms	2837 ms	1931 ms

(b) Extrapolated run times ($\max(T_{\mathsf{Bob}}, T_{\mathsf{Alice}})$) in the passive case of Fig. 2.

Table 3: Total run times expressions and extrapolated run times in the passive case of Fig. 2

Parameter	Par. set 1	Par. set 2
$\{n, N\}$	$\{256, 8192\}$	$\{128, 16384\}$
p	240 bits (4 limbs)	360 bits (6 limbs)
m	60 bits (1 limb)	120 bits (2 limbs)
bit security	≈ 115	≈ 159
# OLEs	2097152	2097152
Alice time	1441 ms	2129 ms
Bob time	1024 ms	1375 ms
Total time	2465 ms	3504 ms

(a) Example parameter sets and global run times for the passively secure OLE based on AHE from the full version (uniformly random ternary secret keys $\{-1, 0, 1\}$ and Gaussian noise with $\sigma = 3.19$).

Proposed protocol of Fig. 2		
{Bob | Alice}	Par. set 1	Par. set 2
1st msg. {2.(a) | —}	{94.37 | —} MB	{125.83 | —} MB
2nd msg. {— | 3.(a)}	{— | 62.91} MB	{— | 94.37} MB
AHE-based protocol (see full version)		
1st round	{— | 62.91} MB	{— | 94.37} MB
2nd round	{125.83 | —} MB	{188.74 | —} MB

(b) Communication cost in the passive case of Fig. 2 and the passively secure OLE based on AHE from the full version .

Table 4: Parameter sets and communication costs for the passive case of Fig. 2 and AHE-based protocol (see full version)

We have chosen two practical parameter sets for both protocols (see Tables 2a and 4a), both featuring more than 110 bits of security,[9] and achieving more than 2 million scalar OLEs per protocol run. Table 2b includes the run times corresponding to each party (Alice and Bob) and Table 4b (see full version) shows the communication costs.

It is worth noting that the public key version from Fig. 1 is not explicitly tested, but it incurs in a similar computational complexity as the one from Fig. 2; it presents, though, an increase on the communication complexity, as the interchanged messages are composed of two polynomials instead of one.

As the latency is not simulated, in order to compare with other protocols, we must consider that the total run time of ours would be $\max(T_{\mathsf{Bob}}, T_{\mathsf{Alice}})$, being T_{Bob} (resp. T_{Alice}) the corresponding run time for Bob (resp. Alice). Tables 3a and 3b include the corresponding expressions and also extrapolate total protocol run times for some specific values of network bandwidth $\{600\mathsf{Mbit/s}, 1\mathsf{Gbit/s}, 10\mathsf{Gbit/s}\}$. T_{step} corresponds to the time of each step included in Table 2b, and $T_{\boldsymbol{d}}$ (resp. $T_{\boldsymbol{c}}$) is the time needed to transmit ciphertext $\boldsymbol{d}$ (resp. $\boldsymbol{c}$)

Extrapolated runtimes are approximately equal or lower than those obtained with the protocol based on AHE (see full version); note that for the last one we are not taking into account transmission runtimes. Consequently, we can see that the proposed protocols in this paper achieve both a better efficiency and lower communication cost than the one based on AHE described in the full version.

Acknowledgements. We thank the anonymous reviewers for comments which helped to improve the paper. This work has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO), the Danish Independent Research Council under Grant-ID DFF–6108-00169 (FoCC), an Aarhus University Research Foundation starting grant, the Xunta de Galicia & ERDF under projects ED431G2019/08 and Grupo de Referencia ED431C2017/53,

[9] We have used the LWE security estimator of Albrecht et al. [2] (available online in `https://bitbucket.org/malb/lwe-estimator`.) to give bit-security estimates.

and by the grant #2017-201 (DPPH) of the Strategic Focal Area "Personalized Health and Related Technologies (PHRT)" of the ETH Domain.

References

1. Lattigo 1.3.1. Online: http://github.com/ldsec/lattigo, Feb. 2020. EPFL-LDS.
2. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.
3. B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. Secure arithmetic computation with constant computational overhead. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 223–254. Springer, Heidelberg, Aug. 2017.
4. C. Baum, J. Bootle, A. Cerulli, R. del Pino, J. Groth, and V. Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699. Springer, Heidelberg, Aug. 2018.
5. C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 478–498. Springer, Heidelberg, Aug. 2016.
6. C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In D. Catalano and R. De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385. Springer, Heidelberg, Sept. 2018.
7. D. Beaver. Efficient multiparty protocols using circuit randomization. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, Aug. 1992.
8. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, Heidelberg, May 2011.
9. D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, Apr. / May 2017.
10. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In D. Lie, M. Mannan, M. Backes, and X. Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, Oct. 2018.
11. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, Nov. 2019.
12. E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, Aug. 2016.
13. E. Boyle, L. Kohl, and P. Scholl. Homomorphic secret sharing from lattices without FHE. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.
14. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, Aug. 2011.
15. R. Cramer, I. Damgård, C. Xing, and C. Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 479–500. Springer, Heidelberg, Apr. / May 2017.
16. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, Aug. 2012.
17. I. Damgård, T. P. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 250–265. Springer, Heidelberg, Aug. 1994.
18. Y. Dodis, S. Halevi, R. D. Rothblum, and D. Wichs. Spooky encryption and its applications. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, Aug. 2016.
19. N. Döttling, S. Ghosh, J. B. Nielsen, T. Nilges, and R. Trifiletti. TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 2263–2276. ACM Press, Oct. / Nov. 2017.
20. D. Genkin, Y. Ishai, M. Prabhakaran, A. Sahai, and E. Tromer. Circuits resilient to additive attacks with applications to secure computation. In D. B. Shmoys, editor, *46th ACM STOC*, pages 495–504. ACM Press, May / June 2014.
21. S. Ghosh, J. B. Nielsen, and T. Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 629–659. Springer, Heidelberg, Dec. 2017.

22. S. Ghosh and T. Nilges. An algebraic approach to maliciously secure private set intersection. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 154–185. Springer, Heidelberg, May 2019.
23. Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 294–314. Springer, Heidelberg, Mar. 2009.
24. M. Keller, V. Pastro, and D. Rotaru. Overdrive: Making SPDZ great again. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 158–189. Springer, Heidelberg, Apr. / May 2018.
25. V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, Apr. 2012.
26. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg, May 2013.
27. P. Mohassel and Y. Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy*, pages 19–38. IEEE Computer Society Press, May 2017.
28. M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *31st ACM STOC*, pages 245–254. ACM Press, May 1999.
29. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.
30. P. Schoppmann, A. Gascón, L. Reichert, and M. Raykova. Distributed vector-OLE: Improved constructions and implementation. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 1055–1072. ACM Press, Nov. 2019.
31. N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, Apr. 2014.