Multiquadratic Rings and Walsh-Hadamard Transforms for Oblivious Linear Function Evaluation

Alberto Pedrouzo-Ulloa[1] Juan Ramón Troncoso-Pastoriza[2] Nicolas Gama[3] Mariya Georgieva[3] Fernando Pérez-González[1]
[1] atlanTTic Research Center, University of Vigo, Vigo, Spain, {apedrouzo,fperez}@gts.uvigo.es
[2] École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, juan.troncoso-pastoriza@epfl.ch
[3] Inpher, Lausanne, Switzerland, {nicolas,mariya}@inpher.io

***Abstract*—The Ring Learning with Errors (RLWE) problem has become one of the most widely used cryptographic assumptions for the construction of modern cryptographic primitives. Most of these solutions make use of power-of-two cyclotomic rings mainly due to its simplicity and efficiency. This work explores the possibility of substituting them for multiquadratic rings and shows that the latter can bring about important efficiency improvements in reducing the cost of the underlying polynomial operations. We introduce a generalized version of the fast Walsh-Hadamard Transform which enables faster degree-n polynomial multiplications by reducing the required elemental products by a factor of $\mathcal{O}(\log n)$. Finally, we showcase how these rings find immediate application in the implementation of OLE (Oblivious Linear Function Evaluation) primitives, which are one of the main building blocks used inside Secure Multiparty Computation (MPC) protocols.**

I. Introduction

The Ring Learning with Errors problem (RLWE) has become a very promising tool for the development and improvement of new cryptographic primitives, notably those belonging to the field of homomorphic encryption.

Most of the efficiency improvements that RLWE has introduced are based on the algebraic structure of the used cyclotomic rings $R = \mathbb{Z}[z]/\Phi_m(z)$. With the adequate choice of modulus q for $R_q = \mathbb{Z}_q[z]/\Phi_m(z)$, the cyclotomic polynomial splits into $\phi(m)$ linear factors, and this enables the use of the CRT (Chinese Remainder Theorem) to efficiently add and multiply the corresponding elements belonging to R_q [1].

Additionally, this property has also been considered for the plaintext ring, as a tool to batch several integers in one encryption (as many as $n = \phi(m)$ values when the modular function fully splits into linear factors), which contributes to a reduction in the cipher expansion.

From a practical perspective, some of the most recent libraries dealing with lattice-based cryptography, such as HElib [2], [3], PALISADE,[1] Lattigo,[2] SEAL[3] and NFLlib [4], take advantage of this fact to optimize the polynomial operations. The first one uses the double-CRT representation, that is, a first CRT splitting the cyclotomic polynomial into linear factors, and a second CRT factoring the coefficients of the polynomials depending on the prime-decomposition of the modulus q. The three latter libraries are specialized for power-of-two cyclotomic rings $\mathbb{Z}_q[z]/(1+z^n)$, so they consider a CRT representation for the coefficients together with an efficient NTT/INTT (Number Theoretic Transform) representation.

In the case that all the involved operations are polynomial multiplications and additions, working in this transformed domain enables polynomial operations with a cost of $\mathcal{O}(n)$ elemental operations between coefficients. However, the current state-of-the-art homomorphic schemes, such as BGV [5], [6] and FV [7], apply a rounding operation over the polynomial coefficients which is not compatible with the double-CRT (or CRT and NTT) representation.

This means that this rounding has to be applied in the coefficient-wise representation, with the corresponding overhead for swapping between representations.

A. Rounding over the RNS (Residue Number System) representation

Bajard *et al.* [8] have studied how to perform a rounding operation without leaving the CRT representation (also called RNS, Residue Number System).

They implement their method using the NFLlib library for the FV cryptosystem and show that for practical parameters, staying in the CRT domain outperforms the results of the usual approach of moving between domains. It is also shown how the asymptotic complexity of decryption is improved by a $\mathcal{O}(\log n)$ factor when staying in the CRT domain.

However, this asymptotic improvement is not preserved for multiplication primitives, as the effect of the NTT/INTT computations is the same for both implementations.

In any case, even when there is no asymptotic improvement for all the primitives, the RNS representation proposed by Bajard *et al.* [8] enables to fit all the used values into the size of a machine word, which in practice helps in considerably

WIFS'2020, December, 6-11, 2020, New York, USA. This is the accepted version. The final published version in WIFS can be found in https://doi.org/10.1109/WIFS49906.2020.9360891.

[1] Available online in https://git.njit.edu/palisade/PALISADE.
[2] Available online in https://github.com/ldsec/lattigo.
[3] Available online in http://sealcrypto.org.

improving the performance when comparing with an implementation requiring the use of multi-precision arithmetic.

In a recent work, Halevi *et al.* [9] propose further optimizations beyond the results of Bajard *et al.* [8], and implement them in the PALISADE library. They achieve a simplification in the procedure and also avoid the additional noise that their method introduces inside the ciphertexts. For a detailed comparison between the methods from [9] and [8] we refer the reader to [10], where CPU and GPU implementation run times are provided for both. Hybrid approaches of [8] and [9] are used in SEAL and Lattigo.

B. Motivation and contributions of our work

A careful examination of the previous results reveals that if we were able to either (a) efficiently compute the rounding operation without reversing the NTT/INTT (or, more generally, the CRT over the cyclotomic polynomial in the double-CRT representation) or (b) speed-up the run times involved in its calculation, then the efficiency of the current RLWE-based schemes could be considerably improved (as almost all the operations in these schemes need to call this NTT/INTT basic block).

This work focuses on improving the cost of the underlying polynomial operations for cryptographic primitives based on RLWE. We show how the well-known asymptotic cost of $\mathcal{O}(n \log n)$ for cyclotomic rings with polynomials of n coefficients can be improved by a factor of $\log n$ in terms of elemental multiplications. To this aim, we propose to work over a multivariate ring which exhibits a convolution property relating the coefficient-wise representation with the transformed domain by means of an $\boldsymbol{\alpha}$-generalized variant of the Walsh-Hadamard transform (over finite rings instead of the usual real numbers). This transform can be very efficiently computed with FFT (Fast Fourier Transform) algorithms (specifically, with a variant of the Fast Walsh-Hadamard transform) whose computational cost is only $\mathcal{O}(n \log n)$ additions, hence being much more amenable for a practical implementation. Finally, we discuss how a basic building block in Secure Multiparty Computation (MPC) as OLE (Oblivious Linear function Evaluation) can benefit from these improvements, hence speeding up the computation of some machine learning applications, as for example the secure evaluation of the convolution layers inside convolutional neural networks.

II. Ring Learning with Errors

We first introduce the notation used in this work. Polynomials are denoted with regular lowercase letters, omitting the polynomial variable (i.e., a instead of $a(x)$) when there is no ambiguity. We follow a recursive definition of multivariate modular rings: $R_q[x_1] = \mathbb{Z}_q[x_1]/f_1(x_1)$ denotes the polynomial ring in the variable x_1 modulo $f_1(x_1)$ with coefficients belonging to $\mathbb{Z}_q$. Analogously, $R_q[x_1, x_2] = (R_q[x_1])[x_2]/(f_2(x_2))$ is the bivariate polynomial ring with coefficients belonging to $\mathbb{Z}_q$ reduced modulo univariate $f_1(x_1)$ and $f_2(x_2)$. In general, $R_q[x_1, \ldots, x_l]$ (resp. $R[x_1, \ldots, x_l]$) represents the multivariate polynomial ring with coefficients in $\mathbb{Z}_q$ (resp. $\mathbb{Z}$) and the l modular functions $f_i(x_i)$ with $1 \leq i \leq l$. The polynomial a can also be denoted by a column vector $\boldsymbol{a}$ whose components are the corresponding polynomial coefficients. Finally, $||a||$ refers to the infinity norm of $\boldsymbol{a}$, the Hadamard product of two vectors is $\boldsymbol{a} \circ \boldsymbol{b}$, and $[l]$ denotes the set $\{1, 2, \ldots, l\}$.

A. Multiquadratic Rings

Let us define the multiquadratic version of RLWE, where all the modular functions have the form $f_i(x_i) = d_i + x_i^2$, as

Definition II.1 (multivariate polynomial RLWE with quadratic modular functions [11])**.** *Given a multivariate polynomial ring $R_q^\vee[x_1, \ldots, x_l]$ with $f_j(x_j) = d_j + x_j^2$ for $j = 1, \ldots, l$ where $l = \log_2 n$ (with n a power of two) and an error distribution $\chi[x_1, \ldots, x_l] \in R_q^\vee[x_1, \ldots, x_l]$ that generates small-norm random multivariate polynomials in $R_q^\vee[x_1, \ldots, x_l]$, the multivariate polynomial RLWE relies upon the computational indistinguishability between samples $(a_i, b_i = a_i \cdot s + e_i)$ and (a_i, u_i), where $a_i \leftarrow R_q[x_1, \ldots, x_l]$, $u_i \leftarrow R_q^\vee[x_1, \ldots, x_l]$ are chosen uniformly at random from the rings $R_q[x_1, \ldots, x_l]$ and $R_q^\vee[x_1, \ldots, x_l]$; and $s, e_i \leftarrow \chi[x_1, \ldots, x_l]$ are drawn from the error distribution.*

The security reduction from [12, Theorem 6.2] applies to this particular version of the RLWE problem (over a multiquadratic number field) whenever all d_i are distinct primes satisfying $-d_i = 1 \bmod 4$. It is also required that the error distribution χ satisfies the condition $r \geq 2$ described in [13, Invulnerability condition].[4]

For more details on the use of this type of rings and the security of multivariate instantiations of RLWE we refer the reader to [11].

B. Ring Expansion Factor

An important measure that we use for performance comparison in Section V is the ring expansion factor. This parameter relates to the underlying noise growth in an RLWE sample after several homomorphic operations.

Definition II.2 (Ring Expansion Factor)**.** *The expansion factor δ_R of a ring R is defined as:*

$$\delta_R = \max_{a,b \in R} \frac{||a \cdot b||}{||a|| \cdot ||b||}.$$

Lemma 1 (Ring Expansion Upper Bound for Power-of-two Cyclomic Rings [14])**.** *For a ring $R = \mathbb{Z}[x]/(1 + z^n)$ with n a power of two, the ring expansion factor δ_R is upper bounded by n.*

Lemma 2 (Ring Expansion Upper Bound for Multiquadratic Rings [11])**.** *For a ring R as the one introduced in Definition II.1, the ring expansion factor δ_R is upper bounded by nD, where $D = \prod_{i \in [\log_2 n]} |d_i|$.*

[4]Being $r = \sigma/\sqrt{2\pi}$ the parameter of the Gaussian error distribution χ defined in the embedding domain.

We can easily see from Lemmas 1 and 2 that multiquadratic rings suffer a higher expansion than power-of-two cyclotomics by a factor of D.

III. Faster polynomial arithmetic over multiquadratic rings

This section focuses on improving the cost of the underlying polynomial operations for cryptographic primitives based on RLWE, especially polynomial products (convolutions).

A. Walsh-Hadamard Transform

The Hadamard transform over real numbers is usually defined by means of the recursion

$$\boldsymbol{H}_i = \frac{1}{\sqrt{2}} \begin{pmatrix} \boldsymbol{H}_{i-1} & \boldsymbol{H}_{i-1} \\ \boldsymbol{H}_{i-1} & -\boldsymbol{H}_{i-1} \end{pmatrix}, \tag{1}$$

where $i \in \mathbb{N}^+$ and $\boldsymbol{H}_0 = 1$.

It can be seen that the matrix $\boldsymbol{H}_i$ with $i \geq 1$ is equivalent to the Kronecker product of i DFT (Discrete Fourier Transform) matrices of size 2 ($\boldsymbol{H}_1$ equals the DFT matrix of size 2); that is, it can be seen as a $\underbrace{2 \times 2 \times \cdots \times 2}_{i \text{ times}}$-DFT transform, defined over i dimensions of length 2 each.

Analogously to the DFT, the Walsh Hadamard Transform (WHT) of size n possesses a particular fast algorithm called FWHT (Fast Walsh Hadamard Transform) which can be very efficiently computed with no products and with a cost of $\mathcal{O}(n \log n)$ additions and subtractions [15]. Hence, when working over rings satisfying a convolution property with the Hadamard transform, it is possible to efficiently compute the multiplication of elements from these rings with a cost of $\mathcal{O}(n)$ elemental multiplications.

Security reasons prevent us from directly working over rings satisfying this convolution property with the Walsh Hadamard transform (that is, multivariate rings whose modular functions are $f(x_i) = x_i^2 - 1$), as they can be easily factored into $(x_i - 1)(x_i + 1)$ over $\mathbb{Z}$. Therefore, we resort to the type of multivariate rings presented in Definition II.1 and apply an ($\boldsymbol{\alpha}$-generalized) variant of the WHT.

B. α-generalized convolutions

An α-generalized convolution[5] corresponds to the ring operation defined over polynomials belonging to $\mathbb{Z}_q[z]/(1-\alpha z^n)$. Figure 1 shows the realization of an α-generalized convolution between vectors of length N (with $l = 0, \ldots, N-1$), by means of a cyclic convolution combined with an element-wise pre/post-processing applied before/after [16], [17].

As the cyclic convolution can be efficiently computed by means of standard fast algorithms, this means that an α-generalized convolution can be implemented with only a light overhead ($\mathcal{O}(n)$ scalar products).

[5]The cyclic convolution is a particular case for $\alpha = 1$.

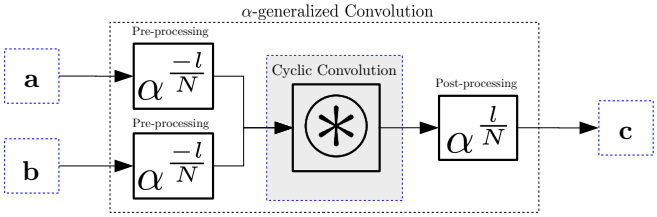

Fig. 1. Block diagram for the α-generalized convolution.

C. $\boldsymbol{\alpha}$-generalized Walsh Hadamard transform

We are mainly interested in modular functions with the form $x_i^2 + d_i$. We can rewrite $1 - \alpha x^n$ as $-\alpha((-\alpha)^{-1} + x^n)$. Hence for $x_i^2 + d_i$ we have $d_i = (-\alpha_i)^{-1} = -\alpha_i^{-1}$. For this particular type of polynomial rings we can define the following direct and inverse transforms:

$$\boldsymbol{W}_1 = \boldsymbol{H}_1 \begin{pmatrix} 1 & 0 \\ 0 & {\alpha_1}^{-1/2} \end{pmatrix},$$

and

$$\boldsymbol{W}_1^{-1} = 2^{-1} \begin{pmatrix} 1 & 0 \\ 0 & {\alpha_1}^{1/2} \end{pmatrix} \boldsymbol{H}_1,$$

where the square-roots ${\alpha_i}^{\frac{1}{2}}$ and ${\alpha_i}^{\frac{-1}{2}}$ have to exist in R_q for all i (see Definition II.1). Equivalently, if q is an odd prime, we can make use of the Legendre symbol $\left(\frac{-d \bmod p}{p}\right)$ to check when the multivariate ring factors into linear terms. For this we need that $\left(\frac{-d_i \bmod q}{q}\right) = 1$ for a prime q and for all i. We also redefine $\boldsymbol{H}_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ without taking into account the normalizing factor $\frac{1}{2}$.

Now we can extend this definition to multivariate rings with modular functions of the form $x_i^2 + d_i$: we consider the Kronecker product of the matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_1^{-1}$ as $\boldsymbol{W}_i = \bigotimes_{j \in [i]} \boldsymbol{W}_1$ and $\boldsymbol{W}_i^{-1} = \bigotimes_{j \in [i]} \boldsymbol{W}_1^{-1}$, arriving at the following expression:

$$\boldsymbol{W}_i = \boldsymbol{H}_i \left(\bigotimes_{j \in [i]} \begin{pmatrix} 1 & 0 \\ 0 & {\alpha_j}^{-1/2} \end{pmatrix} \right),$$

and

$$\boldsymbol{W}_i^{-1} = 2^{-i} \left(\bigotimes_{j \in [i]} \begin{pmatrix} 1 & 0 \\ 0 & {\alpha_j}^{1/2} \end{pmatrix} \right) \boldsymbol{H}_i,$$

where the normalizing factors are again left out of $\boldsymbol{H}_i$.

Consequently, if we define the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)^T$, when working over the multivariate ring $R_q[x_1, \ldots, x_l]$ with $f_j(x_j) = d_j + x_j^2$ for $j = 1, \ldots, l$ we can use the transforms $\boldsymbol{W}_l$ and $\boldsymbol{W}_l^{-1}$ analogously to the use of negacyclic NTTs in the univariate RLWE [17]. Both $\boldsymbol{W}_l$ and $\boldsymbol{W}_l^{-1}$ transforms can be efficiently computed in $\mathcal{O}(n)$ (where $n = 2^l$) elemental multiplications thanks to the FWHT. This enables the computation of the $\boldsymbol{H}_l$ matrix multiplications with only $\mathcal{O}(n \log n)$ additions and subtractions and no products, which brings a net improvement with respect to the analogous and wide-spread radix implementation of the NTT.

IV. Implementation of the Fast Walsh-Hadamard Transform (FWHT)

Algorithm 1 shows a pseudocode implementation of the (cyclic) FWHT (Fast Walsh-Hadamard Transform) implementation [15], computing the Hadamard transform of a length-n vector $\boldsymbol{a}$. It can be easily seen that this algorithm requires a total of $n \log_2 n$ additions (specifically, $\frac{n \log_2 n}{2}$ additions and $\frac{n \log_2 n}{2}$ subtractions), instead of the n^2 additions/subtractions required when directly applying the matrix multiplication.

Algorithm 1 Fast Walsh-Hadamard Transform ($\boldsymbol{H}_i\boldsymbol{a}$ with $i \geq 1$)

procedure FastWalsh-HadamardTransform($\boldsymbol{a}$)
Input:
  $\boldsymbol{a}$ such that $\texttt{length}(\boldsymbol{a}) = n = 2^i$ and $i \geq 1$
Algorithm for $\texttt{FWHT}(\boldsymbol{a})$ *(computing* $\boldsymbol{H}_i\boldsymbol{a}$*)*:
  $\texttt{depth} = 1$;
  for $j = 0$ until $\log_2 n - 1$ **do**
    $\texttt{scale} = 2 * \texttt{depth}$;
    for $k = 0$ until $\left\lfloor \frac{\texttt{length}(\boldsymbol{a})-1}{\texttt{scale}} \right\rfloor$ **do**
      for $l = \texttt{scale} * k$ until $\texttt{scale} * k + \texttt{depth} - 1$ **do**
        $\texttt{ac} = \boldsymbol{a}[l]$;
        $\boldsymbol{a}[l] = \boldsymbol{a}[l] + \boldsymbol{a}[l + \texttt{depth}]$;
        $\boldsymbol{a}[l + \texttt{depth}] = \texttt{ac} - \boldsymbol{a}[l + \texttt{depth}]$;
    $\texttt{depth} = 2 * \texttt{depth}$;
Output:
  $\boldsymbol{a} \leftarrow \boldsymbol{H}_i\boldsymbol{a}$

Finally, the $\boldsymbol{\alpha}$-generalized version of the direct (inverse) FWHT can be defined by adding a right (left) product by a diagonal matrix, so that the total cost for the $\boldsymbol{\alpha}$-generalized FWHT on a length-n vector is n elemental multiplications and $n \log_2 n$ additions.

A. Implementation and evaluation

Polynomial multiplications are the main bottleneck of lattice cryptography, as they are the most time-consuming basic blocks of any cryptographic algorithm, from encryption/decryption to relinearization and bootstrapping. The replacement of the traditional NTTs by FWHTs by transitioning from cryptographic constructions built on univariate RLWE to the proposed multiquadratic version [11] can bring a considerable improvement in terms of computational efficiency. To showcase the achieved gains, we implemented Algorithm 1 in C++ and compared it with one of the currently most efficient radix-2 implementations of the NTT [18]; this is the algorithm featured in the NFLlib, one of the fastest lattice-based cryptographic libraries available for homomorphic encryption. NFL also off-loads the complexity of the bit-reversal operation to the INTT, in order to speed up the NTT, and makes use of SSE and AVX2 optimizations to further enhance the obtained performance. Figure 2 shows the comparison of the obtained run times for a wide range of practical values of n (vector size or polynomial degree), when using our FWHT implementations, including an SSE/AVX2 vectorized version. It can be seen that we obtain a reduction to about 45-50% of the time of the NTT (38-43% of the INTT) in the non-vectorized implementation of the FWHT with respect to the fast NTT of NFLlib, while the vectorized one further reduces this figure to 22-24% (19-22% of the INTT).

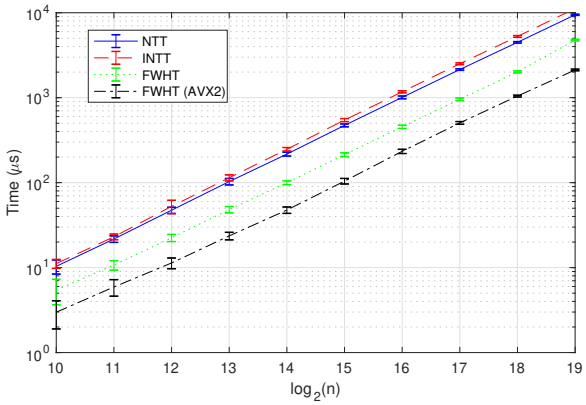

Fig. 2. Runtimes of the proposed FWHT compared to the NTT/INTT [18].

V. Application: Oblivious Linear Function Evaluation

Multiquadratic rings find an immediate application on cryptographic primitives which make an extensive use of direct and inverse transforms for polynomial operations. One example is the case of the Oblivious Linear Evaluation (OLE) primitive which, when instantiated based on RLWE [14], requires many FFT transform calls to efficiently perform encryptions, decryptions, and also when switching ciphertexts' moduli due to division and rounding.

OLE is not only very useful as a building block for MPC [19], [20], but also for more specific machine learning applications: for example, OLE enables the secure computation of the convolution layers of a neural network [14].

In this section, we use the RLWE-based OLE protocol from [14], and study how multiquadratic rings would behave in comparison with the power-of-two cyclotomic counterpart.

A. OLE functionality

OLE can be seen as a particular case of the more general OPE (Oblivious Polynomial Evaluation) primitive [21], in which a polynomial $p(x)$ is obliviously computed. The range of possible scenarios covers, among others, function evaluation by means of their polynomial approximation, secure set intersection [22], and the generation of correlated randomness in the offline phase of SPDZ-type protocols [23].

In contrast, OLE securely computes affine transformations [24], [20] $f(x) = ax + b$, hence simplifying OPE to the case of linear functions. Even so, it is known that OPE can be securely computed for a polynomial $p(x)$ of degree d with only $\mathcal{O}(d)$ calls to OLE, by decomposing $p(x)$ into d consecutive linear functions [24], [25].

In particular, an OLE protocol π_f can be seen as the solution of a two-party computation (2PC) problem on which π_f must implement the ideal functionality $f(x) = ax + b$ and also preserve the input privacy [26] of the two involved parties.

Definition V.1 (2-party OLE computation problem)**.** *Let* $\mathcal{P} = \{P_\mathsf{R}, P_\mathsf{S}\}$ *be a set of* 2 *parties composed by the receiver* P_R *and*

TABLE I
OLE PROTOCOL π_f (SEE DEFINITION V.1) BASED ON AHE

Public input: the ideal functionality f to be computed is an affine function **Private inputs**: x for P_{R} and $\{a, b\}$ for P_{S} **Output** of P_{R}: $f(x) = ax + b$	
Setup	Party P_{R} instantiates the additively homomorphic scheme $\mathsf{E}_{\mathsf{LinHom}}$ $\mathsf{sk} = \mathsf{E}_{\mathsf{LinHom}}.\mathsf{SecKeyGen}(\lambda)$ $\mathsf{pk} = \mathsf{E}_{\mathsf{LinHom}}.\mathsf{PubKeyGen}(\mathsf{sk})$ which is made public
Input	P_{R} encrypts its input x and provides it to P_{S} $c = \mathsf{E}_{\mathsf{LinHom}}.\mathsf{Enc}(\mathsf{pk}, x)$
Evaluation	P_{S} computes the encrypted output c' for the ideal functionality f $c_b = \mathsf{E}_{\mathsf{LinHom}}.\mathsf{Enc}(\mathsf{pk}, b)$ $c' = \mathsf{E}_{\mathsf{LinHom}}.\mathsf{Add}(\mathsf{E}_{\mathsf{LinHom}}.\mathsf{LinMult}(a, c), c_b)$
Output	The party P_{R} executes the decryption $f(x) = \mathsf{E}_{\mathsf{LinHom}}.\mathsf{Dec}(\mathsf{sk}, c')$

the sender P_{S}, where P_{R} holds input x and P_{S} holds inputs a and b. Let $f(x) = ax + b$ be an affine function over the parties' inputs. *Let $\mathcal{A}$ be a static semi-honest adversary that can corrupt either P_{R} or P_{S}. Then, the* secure two-party OLE computation problem *consists in providing P_{R} with $ax+b$, yet when*

- *$\mathcal{A} = P_{\mathsf{S}}$, $\mathcal{A}$ must learn nothing about receiver's input x.*
- *$\mathcal{A} = P_{\mathsf{R}}$, $\mathcal{A}$ must learn nothing more about a and b than what can be deduced from the input x and output $ax+b$ it controls.*

Although the previous two-party computation problem considers a scenario dealing with passive corruptions, it can be upgraded to withstand active adversaries by roughly doubling the number of semi-honest OLE calls [14], [19].

B. OLE based on Additively Homomorphic Encryption (AHE)

A natural approach to implement the OLE functionality (see Definition V.1) is to make use of an additive homomorphic cryptosystem (e.g., Paillier [27]):

Definition V.2. *Let $\mathsf{E} = (\mathsf{SecKeyGen}, \mathsf{PubKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an asymmetric encryption scheme, whose security is parameterized by λ. When considering additively homomorphic encryption, we can extend E with* Add *and* LinMult *procedures, having $\mathsf{E}_{\mathsf{LinHom}} = \{\mathsf{E.SecKeyGen}, \mathsf{E.PubKeyGen}, \mathsf{E.Enc}, \mathsf{E.Dec}, \mathsf{Add}, \mathsf{LinMult}\}$, where*

- *Let c_x and c_y be two ciphertexts encrypting, respectively, x and y.* Add *homomorphically adds them:* $\mathsf{E}_{\mathsf{LinHom}}.\mathsf{Dec}(\mathsf{E}_{\mathsf{LinHom}}.\mathsf{Add}(c_x, c_y)) = x + y$.
- *Let c_x be a ciphertext encrypting x, and y be a plaintext.* LinMult *homomorphically multiplies a ciphertext and a plaintext:* $\mathsf{E}_{\mathsf{LinHom}}.\mathsf{Dec}(\mathsf{E}_{\mathsf{LinHom}}.\mathsf{LinMult}(y, c_x)) = x \cdot y$.

This standard AHE-based approach is described in Table I.

C. Batch OLE based on Lattices

The use of RLWE to instantiate the $\mathsf{E}_{\mathsf{LinHom}}$ scheme in Definition V.2 brings about some important improvements with respect to other schemes such as Paillier [27]:

- $\mathsf{E}_{\mathsf{LinHom}}$ can encrypt a batch of elements (batch OLE, BOLE) [14], [20] by packing them into slots. Thus, one execution of the Protocol from Table I can directly calculate $f(\boldsymbol{x}) = \boldsymbol{a} \circ \boldsymbol{x} + \boldsymbol{b}$ with $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x} \in \mathbb{Z}_p^n$; i.e., a batch of n OLEs in parallel for each encryption sent by P_{R}.
- The homomorphic procedures Add and LinMult are more efficient, as homomorphic addition and multiplication correspond to polynomial addition and multiplication.

Following the structure of typical B/FV schemes [6], [7], fresh ciphertexts are composed of two polynomial elements belonging to $\mathbb{Z}_q[x]/(x^n + 1)$. Let $(c_0, c_1) \in R_q^2$ be an encryption of $\boldsymbol{x}$; $\tilde{a}, \tilde{b} \in R_p$ be two polynomials encoding into slots, respectively, the vectors $\boldsymbol{a}, \boldsymbol{b}$ and let (d_0, d_1) be an encryption of zero generated by P_{S}. Then, by doing

$$(c_0', c_1') = (\tilde{a}c_0 + \tilde{b} + d_0 \bmod q, \tilde{a}c_1 + d_1 \bmod q),$$

P_{S} obtains (c_0', c_1') which encrypts a polynomial $\tilde{x}$ encoding into slots the result $f(\boldsymbol{x})$.

D. Circuit Privacy and Correctness

RLWE-based cryptosystems do not provide circuit privacy by default. This means that, when decrypting, the receiver P_{R} can obtain an error polynomial $e \in R_q$ which leaks some information of both $\boldsymbol{a}$ and $\boldsymbol{b}$. This issue is usually solved by adding a flooding noise with a variance which is high enough to hide this information. However, sampling this noise is usually very inefficient.

Recently, an alternative method avoiding the use of flooding noise while still guaranteeing circuit privacy is proposed in [14]. It basically applies a division and rounding step $(\lceil \frac{c_0'}{q_0^*} \rfloor, \lceil \frac{c_1'}{q_0^*} \rfloor) \bmod q_0$ (where q is equal to $q_0 q_0^*$) which is able to remove the sender's inputs leakage.

We make use of this last variant (see Algorithms $1, 3$ and 4 from [14] for more details) to compare the behaviour of multiquadratic rings with respect to the power-of-two cyclotomics.

To this aim, we need to redefine the bounds for circuit privacy and correctness presented in [14]:

Lemma 3 (OLE Circuit Privacy [14])**.** *For a B-bounded error distribution χ, plaintext modulo p, ciphertext modulo $q = q_0 q_0^*$ defined on multiquadratic rings with $\log_2 n$ independent variables, and security parameter λ, the homomorphic evaluation of [14, algorithm 4] can be done with $2^{-\lambda}$-circuit privacy if parameters satisfy the following bound:*

$$2n\frac{\delta_R B^2 + B + \delta_R pB}{q_0^*} \leq 2^{-\lambda}$$

Lemma 4 (OLE Correctness [14])**.** *For a B-bounded distribution χ, plaintext modulo p, ciphertext modulo $q = q_0 q_0^*$ defined on multiquadratic rings with $\log_2 n$ independent variables. The OLE protocol implemented by means of [14, Algortihms 4 and 5] achieves the OLE functionality (Definition V.1) if q_0^* is chosen according to Lemma 3 and $q_0 > 2\delta_R pB + 2p + (q_0 \bmod p)$.*

TABLE II
EXAMPLE PARAMETER SET FOR A TOTAL OF $32768 \cdot l$ OLES WITH A PLAINTEXT'S MODULUS p OF 120 BITS (2 LIMBS) AND $\sigma = 3.2$, $\lambda = 80$.

Parameter	Par. set 1	Par. set 2	Par. set 3
$\{n, \#\text{batches}, B\}$	$\{32768, l, 32 \cdot D\}$	$\{32768, l, 32\}$	$\{16384, 2l, 32\}$
q	720 bits (12 limbs)	420 bits (7 limbs)	420 bits (7 limbs)
bit security	> 128	> 256	≈ 128

TABLE III
COMPUTATIONAL COST AND CIPHER EXPANSION

Parameters	Asymptotic mult. cost	Cipher Exp. (limbs)
Par. Set 1	$\mathcal{O}(l \cdot (\mathsf{Limbs} + \mathsf{ExtraLimbs}) \cdot n)$	$\mathsf{Limbs} + \mathsf{ExtraLimbs}$
Par. Sets 2 & 3	$\mathcal{O}(l \cdot \mathsf{Limbs} \cdot n \log n)$	Limbs

E. Performance Comparison

Taking into account the runtimes provided in Section IV, we can see from Fig. 2 that the n-length FWHT transform is around 5 times (resp. 2 times) faster than the conventional negacyclic NTT with length n (resp. with length $n/2$). Multiquadratic rings cause an increase on the number of limbs because δ_R and B are D times higher than the ones considered with power-of-two cyclotomics (see Lemmas 3 and 4).

Table II compares three concrete OLE instantiations,[6] where Par. set 1 corresponds to multiquadratic rings and the two last ones to the use of the conventional $\mathbb{Z}[x]/(x^n + 1)$ ring. We can see that for $n = \{2^{14}, 2^{15}\}$, and considering limbs of 60 bits, a cyclotomic ring requires 7 limbs. However, when considering multiquadratic rings we need additional $5 \log_2 D \approx 337$ bits, where each different d_i comes from $\{3, 7, 11, -13, -17, 19, 23, -29, 31, -37, -41, 43, 47, -53, 59\}$. If we assume a fast enough network, the estimated runtimes for the Par. set 1 in Table II are roughly at least 2 times (resp. 2.5 times) faster than the ones obtained with Par. set 3 (resp. Par. set 2).

Table III shows the existing asymptotic tradeoffs when working with multiquadratics. Although both Limbs and ExtraLimbs also grow with n, for practical parameters they are smaller than the $\log n$ factor.

VI. CONCLUSIONS

This work studies the use of multiquadratic rings in the Ring Learning with Errors (RLWE) problem. We show that by substituting the conventional power-of-two cyclotomic polynomial rings by multiquadratics, the multiplicative cost of the involved polynomial operations is decreased by a logarithmic factor. To this aim, we introduce a generalization of the Walsh-Hadamard Transform over finite rings which can be computed with a cost of $\mathcal{O}(n)$ multiplications and $\mathcal{O}(n \log n)$ additions for a signal of length n. Finally, we also show the advantages that these rings can introduce when implementing a basic building block in Secure Multiparty Computation (MPC) as the case of the OLE (Oblivious Linear Function Evaluation) protocol.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their helpful and constructive comments. GPSC is funded by the Agencia Estatal de Investigación (Spain) and the European Regional Development Fund (ERDF) under project RODIN (PID2019-105717RB-C21). Also funded by the Xunta de Galicia and the European Union (European Regional Development Fund - ERDF) under projects ED431G2019/08 and Grupo de Referencia ED431C2017/53. EPFL is funded in part by the grant #2017-201 (DPPH) of the Swiss PHRT.

[6]We have followed the recommendations included in [28] to give bit security estimates.

REFERENCES

[1] V. Lyubashevsky, C. Peikert, and O. Regev, "A Toolkit for Ring-LWE Cryptography," in *EUROCRYPT*, ser. LNCS, vol. 7881. Springer, 2013, pp. 35–54.
[2] S. Halevi and V. Shoup, "Algorithms in helib," in *CRYPTO*, 2014, pp. 554–571.
[3] ——, "Bootstrapping for helib," in *EUROCRYPT*, ser. LNCS, vol. 9056, 2015, pp. 641–670.
[4] C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. Lepoint, *NFLlib: NTT-Based Fast Lattice Library*. Springer, 2016, pp. 341–356.
[5] Z. Brakerski and V. Vaikuntanathan, "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages," in *CRYPTO*, ser. LNCS. Springer, 2011, vol. 6841.
[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption without Bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 13:1–13:36, Jul. 2014.
[7] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," Crypt. ePrint Archive, Report 2012/144, 2012.
[8] J. Bajard, J. Eynard, M. A. Hasan, and V. Zucca, "A full RNS variant of FV like somewhat homomorphic encryption schemes," in *SAC*, 2016, pp. 423–442.
[9] S. Halevi, Y. Polyakov, and V. Shoup, "An improved RNS variant of the BFV homomorphic encryption scheme," *Crypt. ePrint Archive, Report 2018/117*, 2018.
[10] A. A. Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, and K. Rohloff, "Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme," Crypt. ePrint Archive, Report 2018/589, 2018.
[11] A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza, N. Gama, M. Georgieva, and F. Pérez-González, "Revisiting Multivariate Ring Learning with Errors and its Applications on Lattice-based Cryptography," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1109, 2019.
[12] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of ring-LWE for Any Ring and Modulus," in *ACM STOC*, 2017, pp. 461–473.
[13] C. Peikert, "How (Not) to Instantiate Ring-LWE," in *SCN*, 2016, pp. 411–430.
[14] L. de Castro, C. Juvekar, and V. Vaikuntanathan, "Fast Vector Oblivious Linear Evaluation from Ring Learning with Errors," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 685, 2020.
[15] B. J. Fino and V. R. Algazi, "Unified Matrix Treatment of the Fast Walsh-Hadamard Transform," *IEEE Transactions on Computers*, vol. C-25, no. 11, pp. 1142–1146, Nov 1976.
[16] H. Murakami, "Generalization of the cyclic convolution system and its applications," in *IEEE ICASSP*, vol. 6, 2000, pp. 3351–3353.
[17] A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza, and F. Pérez-González, "Number Theoretic Transforms for Secure Signal Processing," *IEEE Trans. Inf. Forensics and Sec.*, vol. 12, no. 5, pp. 1125–1140, May 2017.
[18] D. Harvey, "Faster arithmetic for number-theoretic transforms," *J. Symb. Comput.*, vol. 60, pp. 113–119, 2014.
[19] C. Hazay, Y. Ishai, A. Marcedone, and M. Venkitasubramaniam, "LevioSA: Lightweight Secure Arithmetic Computation," in *ACM CCS*. ACM, 2019, pp. 327–344.
[20] C. Baum, D. Escudero, A. Pedrouzo-Ulloa, P. Scholl, and J. R. Troncoso-Pastoriza, "Efficient Protocols for Oblivious Linear Function Evaluation from Ring-LWE," in *SCN 2020*, ser. LNCS, vol. 12238. Springer, 2020, pp. 130–149.
[21] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation," in *STOC*. ACM, 1999, pp. 245–254.
[22] C. Hazay, "Oblivious Polynomial Evaluation and Secure Set-Intersection from Algebraic PRFs," *J. Cryptology*, vol. 31, no. 2, pp. 537–586, 2018.

[23] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty Computation from Somewhat Homomorphic Encryption," in *CRYPTO*, vol. 7417. Springer, 2012, pp. 643–662.
[24] S. Ghosh, J. B. Nielsen, and T. Nilges, "Maliciously Secure Oblivious Linear Function Evaluation with Constant Overhead," in *ASIACRYPT*, vol. 10624. Springer, 2017, pp. 629–659.
[25] M. Naor and B. Pinkas, "Oblivious Polynomial Evaluation," *SIAM J. Comput.*, vol. 35, no. 5, pp. 1254–1281, 2006.
[26] Y. Lindell, "How to Simulate It - A Tutorial on the Simulation Proof Technique," in *Tutorials on the Foundations of Cryptography*. Springer, 2017, pp. 277–346.
[27] P. Paillier, "Public-key Cryptosystems Based on Composite Degree Residuosity Classes," in *EUROCRYPT*, ser. LNCS, vol. 1592. Springer, 1999, pp. 223–238.
[28] M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, J. Hoffstein, K. Lauter, S. Lokam, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Security of homomorphic encryption," HomomorphicEncryption.org, Redmond WA, Tech. Rep., July 2017.