To appear in *10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC 2022)*, November 7, 2022.

Liberating TFHE: Programmable Bootstrapping with General Quotient Polynomials

Marc Joye
Zama
Paris, France

Michael Walter
Zama
Paris, France

ABSTRACT

All known instantiations for fully homomorphic encryption (FHE) produce noisy ciphertexts and rely on a technique called bootstrapping to reduce the noise so as to enable an arbitrary number of homomorphic operations. Bootstrapping is the main performance bottleneck and arguably the biggest obstacle to widespread adoption of FHE. Among the FHE schemes, TFHE and its variations present the appealing property of having a bootstrapping procedure—as well as its extension to programmable bootstrapping—that is relatively light-weight. The essential operations consist of a series of multiplications in $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N+1)$. While the NTT is seemingly the natural candidate for evaluating these multiplications in a fast and exact way, it restricts the possible choices for q and N. To the authors' knowledge, all current implementations of TFHE with q a power of two actually employ the FFT over the complex numbers instead. This introduces real numbers to the otherwise purely discrete algorithms, including all the drawbacks of the need to approximate them using finite precision.

This work studies the avenues available to apply the NTT in the context of TFHE-like schemes. In particular, it considers various combinations of coefficient rings and quotient polynomials that are compatible with the requirements of the underlying scheme. Importantly, this work provides methods for adapting the (programmable) bootstrapping to quotient polynomials beyond power-of-two cyclotomics. As a side effect, it also demonstrates how this may enhance the programmability of the bootstrapping.

CCS CONCEPTS

• **Security and privacy → Mathematical foundations of cryptography**.

KEYWORDS

Number-theoretic transform; Polynomial multiplication; Blind rotation; Fully homomorphic encryption; Programmable bootstrapping; Functional circuits

ACM Reference Format:
Marc Joye and Michael Walter. 2022. Liberating TFHE: Programmable Bootstrapping with General Quotient Polynomials. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC '22), November 7, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3560827.3563376

WAHC '22, November 7, 2022, Los Angeles, CA, USA.

ACM ISBN 978-1-4503-9877-0/22/11…$15.00
https://doi.org/10.1145/3560827.3563376

1 INTRODUCTION

Encryption enables the protection of sensitive data while it is stored or when it needs to be transferred. However, standard encryption technologies require data to be decrypted to be processed. On the contrary, there is no necessity to decrypt with *homomorphic encryption* [28]; operations are directly performed on encrypted data. An encryption scheme capable of evaluating any program is said to be *fully* homomorphic.

(Programmable) bootstrapping. Fully homomorphic encryption as introduced by Gentry [15] importantly employs the concept of *bootstrapping*.

Known realizations for fully homomorphic encryption produce noisy ciphertexts. The noise present in the ciphertexts however tends to increase when ciphertexts are homomorphically processed. If the noise grows too much, it can render decryption impossible. This issue is addressed through bootstrapping. In Gentry's original scheme, bootstrapping is done by homomorphically evaluating the decryption circuit, resulting in another ciphertext that encrypts the same plaintext. Since decryption removes noise, the noise present in a bootstrapped ciphertext is reset to a nominal level (i.e., it only contains the noise coming from the bootstrapping).

Building on the FHEW cryptosystem [12], an encryption scheme that achieves a relatively fast bootstrapping (a few tens of milliseconds) is TFHE [8]. Its security is based on the *learning with errors* problem (LWE) [27] and its ring variant (RLWE) [23, 30]. Originally designed for boolean circuits, TFHE can be extended to support further input formats, such as integers [9] (see also [16]). Variants making use of the NTRU assumption are given in [6, 18]. Finally, bootstrapping in TFHE and the likes can be programmed to evaluate a univariate function for free, at the same time as the noise is reduced. This technique is referred to as *programmable bootstrapping* (PBS) and provides a powerful way for homomorphically evaluating non-linear functions such as activation functions in a neural network [10].

Implementation. The costliest operation during a PBS is a series of polynomial multiplications in $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N+1)$, where q and N are powers of two. For concreteness, the reader may think of $q \in \{2^{32}, 2^{64}\}$ and $N \in \{2^{10}, 2^{11}, \ldots, 2^{14}\}$, but other parameters are possible. Due to the large degree of the polynomials, fast Fourier techniques are employed to compute these multiplications. Given that the involved polynomials all have coefficients in $\mathbb{Z}/q\mathbb{Z}$, it would be natural to apply the number-theoretic version of the FFT (the so-called *Number Theoretic Transform*, NTT). Unfortunately, the typical choices of N and q as powers of two, while extremely convenient

from an implementation standpoint, are incompatible with the NTT. Accordingly, to the best of our knowledge, implementations of TFHE with q a power of two, e.g. [7, 9], employ the FFT over the complex numbers for the polynomial multiplication. The same is true for the prototype implementation of FHEW [13]. The drawback in this approach is that it introduces real numbers in the otherwise purely discrete computations of the PBS, which introduces some approximation error. While this approximation error can in theory be controlled to fit the needs of the PBS parametrization by tuning the precision parameter, in practice going beyond a precision that is natively supported in common hardware requires using software libraries for high-precision floating point arithmetic, which usually incurs a significant slowdown and limits the parameter choices for practical purposes. We discuss these issues in more detail in Section 3.2.

Some works consider quotient polynomials that are not power-of-two cyclotomics, but for other techniques and applications. So [4] deals with cyclotomics of prime-power index p^t in the context of FHEW. Their setting enables to homomorphically test equality of an encrypted message with a given plaintext message. The technique in [5] allows lifting elements of $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$, where $p(X)$ is a cyclotomic, to the NTRU ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^p-1)$ (hence they call this technique the "NTRU trick"), while preserving security based on RLWE and (mostly) the geometry of the ring. Corresponding implementations in both cases use the FFT for the multiplication of ring elements.

Our contributions

In this work we explore options to apply the NTT within the TFHE family to obtain a version that operates purely on small discrete data-types without approximation errors. There are two ways to change the parametrization for the NTT to be applicable. The obvious choice is to change the modulus q to some "NTT-friendly" integer. This is the approach taken by the Palisade implementation [24, 26]. In practice, it might however be useful to keep the modulus a power of two. So in this work, we also study the alternative approach of changing the quotient polynomial from X^N+1 to some other cyclotomic polynomial $p(X)$. We focus on trinomials for the choice of $p(X)$ in order for the reductions modulo $p(X)$ to be almost as efficient as modulo a power-of-two cyclotomic, but the approach is more general and works for any cyclotomic polynomial (and in fact an even larger class of polynomials; cf. Section 3.2).

Since the PBS in TFHE-like encryption schemes is specifically tailored to quotient polynomials that are power-of-two cyclotomics, this requires non-trivial adaptations to the operations that we develop in this work. Most notably, a more general quotient polynomial $p(X)$ changes how multiplications with monomials affect the coefficients of the resulting polynomial in $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$. Such a multiplication simply yields a negacyclic rotation when $p(X)$ is a power-of-two cyclotomic and the PBS heavily leverages this property to program the so-called *test polynomial* (cf. Section 3.1). For other quotient polynomials, the impact of a multiplication on an element in $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$ is more complex. The core technical part of this work analyses how to program the test polynomial in such a setting.

Finally, we remark that the size of the fraction of the usable plaintext space in TFHE and the likes is restricted by the ratio N/M, where M is the index and N the degree of the cyclotomic polynomial. In the case of power-of-two cyclotomics this ratio is of $\frac{1}{2}$, so only half of the plaintext space may be used when applying the PBS. An additional advantage of using a different quotient polynomial is that it allows increasing this ratio. For example, in Section 7, we give some concrete parametrization that allow a ratio of $\frac{2}{3}$ with cyclotomic trinomials.

Outline of the paper

The rest of this paper is organized as follows. The next section reviews the number-theoretic transform and its application to the fast multiplication of polynomials. Section 3 recaps the bootstrapping for TFHE-like cryptosystems and its extension to programmable bootstrapping. Section 4 is the core of the paper. It generalizes the programmable bootstrapping so as to accommodate more general quotient polynomials. In particular, this includes programming the so-called test polynomial and extracting the matching LWE ciphertext. Section 5 analyzes which functions may be evaluated without padding in case the quotient polynomial is a trinomial (cf. end of Section 3.1) while Section 6 discusses the noise growth incurred by different quotient polynomials. Section 7 gives certain sets of concrete parameters for the proposed techniques. Finally, Section 8 concludes the paper.

2 FAST POLYNOMIAL MULTIPLICATION

Let $\mathfrak{R}$ be a ring with unity and $M \in \mathbb{Z}_{\geq 1}$. Given two polynomials $f, g \in \mathfrak{R}[X]/(X^M-1)$, their product can be computed thanks to the discrete Fourier transform. This requires the existence of certain roots of unity. Excellent references to the topic are [31, Chapter 8] and [3].

Discrete Fourier transform. Let $\omega \in \mathfrak{R}$ be a principal M-th root of unity. A polynomial $f := f(X) = f_0 + f_1 X + \cdots + f_{M-1} X^{M-1} \in \mathfrak{R}[X]$ of degree $< M$ is identified with its coefficient vector $(f_0, f_1, \ldots, f_{M-1}) \in \mathfrak{R}^M$. The *discrete Fourier transform* (DFT) of polynomial f (viewed as a vector of $\mathfrak{R}^M$) is the vector consisting of the evaluation of f at the successive powers of ω; namely

$$\begin{aligned} \mathrm{DFT}_\omega \colon \mathfrak{R}^M &\xrightarrow{\sim} \mathfrak{R}^M, \\ f &\longmapsto \mathrm{DFT}_\omega(f) = \big(f(\omega^j)\big)_{0 \leq j \leq M-1} \ . \end{aligned}$$

Moreover, for two polynomials $f, g \in \mathfrak{R}[X]$ of degree $< M$, we have

$$\mathrm{DFT}_\omega(f * g) = \mathrm{DFT}_\omega(f) \cdot \mathrm{DFT}_\omega(g)$$

where $*$ denotes the polynomial convolution and $\cdot$ denotes the point-wise multiplication of vectors; see [31, Lemma 8.11].

DFT and polynomial multiplication. Consider now polynomials $f, g \in \mathfrak{R}[X]/(X^M-1)$ (and thus of degree $< M$). Since multiplication modulo X^M-1 is a convolution (i.e., $f \cdot g \equiv f * g \pmod{X^M-1}$), the product of $h := f \cdot g \in \mathfrak{R}[X]/(X^M-1)$ can be obtained as

$$h = \mathrm{DFT}_\omega^{-1}\big(\mathrm{DFT}_\omega(f) \cdot \mathrm{DFT}_\omega(g)\big)$$

where DFT_ω^{-1} denotes the inverse discrete Fourier transform. Interestingly, it holds that $\mathrm{DFT}_\omega^{-1}(h) = \frac{1}{M}\,\mathrm{DFT}_{\omega^{-1}}(h)$, for any polynomial $h \in \mathfrak{R}[X]$ of degree $< M$; see [31, Theorem 8.13].

FFT trick. The "FFT trick" is apparently due to Gauss and was re-discovered by Cooley and Tukey [11] for the case $\mathfrak{R} = \mathbb{C}$; see also [14].

Assume that M is even. In this case, the Chinese remainder theorem states that we have the mapping

$$\mathfrak{R}[X]/(X^M - 1) \to \left(\mathfrak{R}[X]/(X^{\frac{M}{2}} - 1)\right) \times \left(\mathfrak{R}[X]/(X^{\frac{M}{2}} + 1)\right) \ .$$

So, for $f(X) = f_0 + f_1 X + \cdots + f_{M-1} X^{M-1} \in \mathfrak{R}[X]/(X^M - 1)$, letting $F_0 \coloneqq F_0(X) = f_0 + f_1 X + \cdots + f_{\frac{M}{2}-1} X^{\frac{M}{2}-1}$ and $F_1 \coloneqq F_1(X) = f_{\frac{M}{2}} + f_{\frac{M}{2}+1} X + \cdots + f_{M-1} X^{\frac{M}{2}-1}$, we can write $f(X) = F_0 + F_1 X^{\frac{M}{2}}$ and therefore $f(X) \equiv F_0 + F_1 \pmod{(X^{\frac{M}{2}} - 1)}$ and $f(X) \equiv F_0 - F_1 \pmod{(X^{\frac{M}{2}} + 1)}$. Furthermore, applying $X \mapsto \omega X$ defines $f^*(X) = f(\omega X)$ and so, from $(\omega X)^{\frac{M}{2}} + 1 = -X^{\frac{M}{2}} + 1$, the last equation can be rewritten as

$$f^*(X) \equiv F_0^* - F_1^* \pmod{(X^{\frac{M}{2}} - 1)}$$

with $F_0^* \coloneqq F_0^*(X) = F_0(\omega X)$ and $F_1^* \coloneqq F_1^*(X) = F_1(\omega X)$. Now, ω^2 is a principal $(M/2)$-th root of unity. Hence, if M is a power of two, the process can be iterated. This divide-and-conquer strategy allows one to evaluate f at the powers ω^j for $0 \le j \le M-1$ (i.e., the DFT of f) in $O(M \log(M))$ ring operations—instead of $O(M^2)$ with the naive approach.

Cryptographic applications. For security reasons, RLWE-based cryptosystems consider cyclotomic polynomials $p(X) \mid X^M - 1$ and work in $\mathfrak{R}[X]/(p(X))$. In particular, TFHE-type schemes take M a power of two and define $p(X) = X^N + 1$ with $N = \varphi(M) = M/2$. The coefficient ring is $\mathfrak{R} = \mathbb{Z}/q\mathbb{Z}$ with q a power of two for discretized implementations.

Remark 2.1. When computing modulo $X^N + 1$, it is advantageous to use the *negacyclic* discrete Fourier transform,

$$\overline{\mathrm{DFT}} \colon \mathfrak{R}^N \to \mathfrak{R}^N, f \mapsto \overline{\mathrm{DFT}}_\omega(f) = \left(f(\omega^{2j+1})\right)_{0 \le j \le N-1}$$

where ω is a principal $2N$-th root of unity. If $\hat{f} = \overline{\mathrm{DFT}}_\omega(f)$, the inverse transform is given by $\overline{\mathrm{DFT}}_\omega^{-1}(\hat{f}) = \left(\frac{\omega^{-j}}{N} \hat{f}(\omega^{-2j})\right)_{0 \le j \le N-1}$.

For TFHE-like cryptosystems, polynomial multiplications modulo $X^N + 1$ can always be quickly evaluated through classical FFT (i.e., DFT over $\mathbb{C}$). A principal $2N$-th root of unity is given by $\omega = e^{i\pi/N}$.

Example 2.2. Suppose $N = 2^3$ and $q = 2^4$. Set $\omega = e^{i\pi/8} \in \mathbb{C}$. Given polynomials $f(X) = 5X^7 + 3X^6 + 5X^5 + 8X^4 + 9X^3 + 7X^2 + 9X + 3$ and $g(X) = 2X^7 + 8X^6 + 4X^5 + 5X^4 + 9X^3 + 4X + 5$ with $f, g \in (\mathbb{Z}/16\mathbb{Z})[X]/(X^8 + 1)$, the goal is to compute $h \coloneqq f \, g \in (\mathbb{Z}/16\mathbb{Z})[X]/(X^8 + 1)$.

Lifting the coefficients of f and g to $\mathbb{C}$, we obtain (displaying decimals rounded to 2 digits) $\overline{\mathrm{DFT}}_\omega(f) \approx (11.05 + 33.36i, -1.99 + 6.65i, 2.34 + 8.51i, 0.60 + 3.22i, 0.60 - 3.22i, 2.34 - 8.51i, -1.99 - 6.65i, 11.05 - 33.36i)$ and $\overline{\mathrm{DFT}}_\omega(g) \approx (3.10 + 24.96i, 6.80 + 1.23i, 14.51 - 0.09i, -4.42 + 3.65i, -4.42 - 3.65i, 14.51 + 0.09i, 6.80 - 1.23i, 3.10 - 24.96i)$, which in turn leads to $\overline{\mathrm{DFT}}_\omega(h) \approx (-798.53 + 379.53i, -21.70 + 42.78i, 34.66 + 123.22i, -14.42 - 12.03i, -14.42 + 12.03i, 34.66 - 123.22i, -21.70 - 42.78i, -798.53 - 379.53i)$. We so obtain $h(X) = 260X^7 + 201X^6 + 131X^5 + 118X^4 + 7X^3 - 91X^2 - 113X - 200$ as an element of $\mathbb{Z}[X]/(X^N + 1)$. Reducing modulo q yields the expected result $h(X) = 4X^7 + 9X^6 + 3X^5 + 6X^4 + 7X^3 + 5X^2 + 15X + 8$.

The NTT (i.e., DFT over $\mathbb{Z}/q\mathbb{Z}$) does not readily apply to the typical TFHE-type parameters: the existence of a principal $2N$-th root of unity in $\mathbb{Z}/q\mathbb{Z}$ prohibits q of being a power of two. In this case, when $p(X) = X^N + 1$, the coefficient ring is usually selected as a prime field $\mathbb{F}_p$ with $p \equiv 1 \pmod{2N}$. This ensures the existence of a principal $2N$-th root of unity. We illustrate below an application of NTT in such a setting.

Example 2.3. Suppose $N = 2^3$ and $q = 17$ (a prime). It is easily checked that $\omega = 3$ is a principal 16-th root of unity in $\mathbb{F}_{17}$. Consider the same polynomials f and g as in Example 2.2, but in $\mathbb{F}_{17}[X]/(X^8 + 1)$. We have $\overline{\mathrm{DFT}}_\omega(f) = (4, 13, 14, 10, 1, 15, 1, 0)$, $\overline{\mathrm{DFT}}_\omega(g) = (11, 8, 15, 3, 12, 16, 4, 5)$, and $\overline{\mathrm{DFT}}_\omega(h) = (10, 2, 6, 13, 12, 2, 4, 0)$. The inverse transformation directly gives $h(X) = 5X^7 + 14X^6 + 12X^5 + 16X^4 + 7X^3 + 11X^2 + 6X + 4$ in $\mathbb{F}_{17}[X]/(X^8 + 1)$, as expected.

There exist variants of the NTT that only require $2N$ to be a unit in $\mathbb{Z}/q\mathbb{Z}$. Examples include Schönhage's algorithm [29] (see also [31, Algorithm 8.30]) and Nussbaumer's algorithm [25] (see also [19, Exercise 4.6.4.59]). We follow the presentation of [3, § 9]. In some sense, these methods "manufacture" roots of unity. For instance, assuming that $N = mt$ with $t \mid m$, Nussbaumer's method relies on the correspondence

$$(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1) \cong \left((\mathbb{Z}/q\mathbb{Z})[Y]/(Y^m + 1)\right)[X]/(X^t - Y) \ .$$

The elements of the right-hand side are polynomials of X-degree $< t$ and with coefficients in $\mathfrak{R} \coloneqq (\mathbb{Z}/q\mathbb{Z})[Y]/(Y^m + 1)$. As such, they can be seen as elements in $\mathfrak{R}[X]/(X^{2t} - 1)$. The main observation is that $\omega = Y^{N/t^2}$ is a $2t$-th principal root of unity in $\mathfrak{R}$. The usual DFT machinery can therefore be applied. This is shown in the next example.

Example 2.4. Consider the exact same setting as in Example 2.3. Write $N = 2^3 = 4 \cdot 2$ and fix $m = 4$ and $t = 2$. Define $\omega = Y^2$. As elements of $\left((\mathbb{Z}/17\mathbb{Z})[Y]/(Y^4 + 1)\right)[X]/(X^2 - Y)$, polynomials f and g are respectively expressed as $f(X, Y) = (5Y^3 + 5Y^2 + 9Y + 9)X + (3Y^3 + 8Y^2 + 7Y + 3)$ and $g(X, Y) = (2Y^3 + 4Y^2 + 9Y + 4)X + (8Y^3 + 5Y^2 + 5)$. We get $\mathrm{DFT}_\omega(f) = (8Y^3 + 13Y^2 + 16Y + 12, 12Y^3 + 2Y + 15, 15Y^3 + 3Y^2 + 15Y + 11, 11Y^3 + 16Y^2 + 12Y + 8)$ and $\mathrm{DFT}_\omega(g) = (10Y^3 + 9Y^2 + 9Y + 9, 9Y^2 + 15Y + 1, 6Y^3 + Y^2 + 8Y + 1, 16Y^3 + Y^2 + 2Y + 9)$. This leads to $\mathrm{DFT}_\omega(h) = (11Y^3 + 16Y + 14, 13Y^3 + 12Y^2 + 5, Y^3 + 10Y^2 + 2Y + 2, 16Y^3 + 10Y + 12)$ and, in turn, $\mathrm{DFT}_\omega^{-1}\left(\mathrm{DFT}_\omega(h)\right) \equiv (8Y^2 + 2Y + 4)X^2 + (5Y^3 + 12Y^2 + 7Y + 6)X + (6Y^3 + 14Y^2 + 7Y + 4)$. Replacing Y with X^2 and reducing the result modulo $X^8 + 1$ finally yields $h(X) = 5X^7 + 14X^6 + 12X^5 + 16X^4 + 7X^3 + 11X^2 + 6X + 4$.

Remark 2.5. Schönhage's and Nussbaumer's algorithms extend to higher radices. The mappings for q a power of three can be found in [3, § 9]. The 3-adic variant of Schönhage's algorithm is also described in [31, Algorithm 8.30; Exercise 8.30].

3 PROGRAMMABLE BOOTSTRAPPING

Programmable bootstrapping (PBS) is an extension of the bootstrapping technique that allows resetting the noise to a fixed level

while at the same time evaluating a function on the input ciphertext. Given an LWE-type ciphertext $(\boldsymbol{a}, b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + \mu(m) + e) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$—where

- the secret $\boldsymbol{s}$ is a (typically small) vector in $(\mathbb{Z}/q\mathbb{Z})^n$,
- $\boldsymbol{a} \in (\mathbb{Z}/q\mathbb{Z})^n$ is uniformly random,
- $\mu(m) \in \mathbb{Z}/q\mathbb{Z}$ is the encoding of a plaintext $m \in \mathbb{Z}/p\mathbb{Z}$ with $p < q$, and
- the noise $e \in \mathbb{Z}$ is small enough to be able to recover m from $\mu(m) + e$,

the goal is to compute an LWE-type ciphertext $(\boldsymbol{a'}, b' = \langle \boldsymbol{a'}, \boldsymbol{s} \rangle + \mu(f(m))) + e' \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ for some function $f \colon \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ and $|e'| < |e|$. In the following, the reader should keep in mind that PBS is a public operation and needs to be feasible without knowledge of the secret key $\boldsymbol{s}$.

3.1 Negacyclic PBS

We begin with an overview of the PBS, which we call *negacyclic PBS* for reasons that will become apparent at the end of the section. The technical details of this procedure are quite involved, but not important to understand this work. We refer the interested reader to [9, 10, 16] and the references therein for further information. It suffices to note that while this procedure does incur some nominal noise in the output ciphertext, this noise is *independent* of the noise present in the input ciphertext.

The PBS involves a series of four steps:

Modulus Switch Let $N = 2^\beta$ be a parameter defining the ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$. Each component of an input LWE-type ciphertext is scaled by $2N/q$ and rounded; i.e., each component goes through an operation of the form $\lceil 2N(\cdot \bmod q)/q \rfloor$. The result is an LWE-type ciphertext modulo $2N$, $(\bar{\boldsymbol{a}}, \bar{b} = \langle \bar{\boldsymbol{a}}, \boldsymbol{s} \rangle + \bar{\mu}(m) + \bar{e})$. Note that we assume here (and throughout) that the encoding μ is compatible with such a scaling and yields a scaled encoding $\bar{\mu}$ (and a corresponding decoding).

Blind Rotate This is the core operation. Essentially, it is a homomorphic decryption of the input ciphertext and thus requires a *bootstrapping key*, which basically are encryptions of $\boldsymbol{s}$ under some other secret key $\mathfrak{z}'$. The output of the blind rotation is an RLWE-type ciphertext* $(\mathfrak{a}, \mathfrak{b} = \mathfrak{a} \cdot \mathfrak{z}' + \mu + \mathfrak{e}) \in \big((\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)\big)^2$, where polynomial μ contains $\mu(f(m))$ in its constant coefficient.

The blind rotation starts with a so-called *test polynomial* $\mathfrak{v} \in (\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$. This polynomial (as any polynomial) can be viewed as a trivial RLWE-type encryption of itself (under any key); i.e., as $(0, \mathfrak{v})$. The key idea is to rotate it *homomorphically* by $\langle \bar{\boldsymbol{a}}, \boldsymbol{s} \rangle - \bar{b} = -(\bar{\mu}(m) + \bar{e})$ positions by multiplying it with $X^{\langle \bar{\boldsymbol{a}}, \boldsymbol{s} \rangle - \bar{b}}$. The result is an RLWE-type ciphertext encrypting the polynomial $X^{-(\bar{\mu}(m)+\bar{e})} \cdot \mathfrak{v}(X)$ under key $\mathfrak{z}'$. In order for the PBS to work, the coefficients of the test polynomial $\mathfrak{v}$ need to be set up such that after the blind rotation it holds that the underlying plaintext polynomial μ has its constant coefficient equal to $\mu(f(m))$. Since the quotient polynomial is $X^N + 1$, forming the test polynomial is immediate: first a look-up table T is defined by setting $T[i] = \mu(f(m_i)) \in \mathbb{Z}/q\mathbb{Z}$ for each $i \in \{0, \ldots, N-1\}$ if the decoding corresponding to $\bar{\mu}$ applied to i is $m_i \in \mathbb{Z}/p\mathbb{Z}$. The table is then used to program the test polynomial as

$$\mathfrak{v}(X) = v_0 + v_1 X + \cdots + v_{N-1} X^{N-1}$$

with $v_i = T[i]$. It is worth noting that the constant coefficient of $X^{-(\bar{\mu}(m)+\bar{e})} \cdot \mathfrak{v}(X)$ is $v_{\bar{\mu}(m)+\bar{e}}$ and by construction this is equal to $\mu(f(m))$.

Sample Extract To turn the RLWE-type ciphertext back into an LWE-type ciphertext, a sample extraction is applied. This operation uses the coefficients of the RLWE-type ciphertext $(\mathfrak{a}, \mathfrak{b})$ resulting from the blind rotation to get an LWE-type ciphertext of the constant coefficient under the 'same' key $\boldsymbol{s'}$ (where $\boldsymbol{s'}$ is the coefficient vector of the secret key $\mathfrak{z}'$). In the negacyclic setting (i.e., where the quotient polynomial is $X^N + 1$), the LWE sample can essentially be "read off" of the RLWE sample.

Key Switch Finally, to obtain a ciphertext under $\boldsymbol{s}$, a key switch is applied. This uses an encryption of $\boldsymbol{s'}$ under $\boldsymbol{s}$, which is typically called the *key-switching key*. Switching keys is a fairly standard operation and has no impact on this work.

Remark that the parameter N (or more generally the quotient polynomial $p(X) = X^N + 1$) plays a crucial role in the blind rotation: it controls the size of the ciphertext space, and thus also the plaintext space and tolerable noise, and the size of the table T to be evaluated. Note though that the cost of the PBS also depends in large part on N. Finally, remark that the modulus in the first step is switched from q to $2N$ to guarantee the correctness of the blind rotation. Indeed, as an element of $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$, X verifies $X^{2N} = 1$. Accordingly, one needs

- either to ensure that $\bar{\mu}(m) + \bar{e} \in \{0, \ldots, N-1\}$; i.e., only make use of half the plaintext space—this is known as the *padding* technique (see e.g. [9]), or
- require p to be even and the function f to be negacyclic over $\mathbb{Z}/p\mathbb{Z}$; that is, $f(x + p/2) = -f(x) \pmod p$. An important example of a negacyclic function is the sign function.

3.2 Implementation Considerations

As aforementioned, PBS involves performing a number of multiplications in $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$ for some power-of-two N. Since N is typically large ($\geq 2^{10}$) fast Fourier techniques as described in Section 2 seem to be the way to go for performing these multiplications. The obvious choice would be to rely on DFT over $\mathbb{Z}/q\mathbb{Z}$ (so performing an NTT), which would require $\mathbb{Z}/q\mathbb{Z}$ to contain a principal $2N$-th root of unity. Unfortunately, the modulus q is typically chosen to be also a power of two, which is extremely convenient from an implementation perspective since reductions mod q are trivial and highly efficient. But then $M = 2N$ is not a unit in $\mathbb{Z}/q\mathbb{Z}$, which is a necessary condition for an M-th principal root of unity to exist in $\mathbb{Z}/q\mathbb{Z}$.

To overcome this problem and keep the modulus q as a power of two we may perform the DFT over the complex numbers instead

*This is a slight simplification. More generally, the output of the blind rotation is a so-called GLWE-type ciphertext of the form $(\mathfrak{a}_1, \ldots, \mathfrak{a}_k, \mathfrak{b} = \sum_i \mathfrak{a}_i \cdot \mathfrak{z}'_i + \mu + \mathfrak{e})$, but for the sake of clarity we focus on the $k = 1$ case here. The generalization of our techniques to larger k is immediate.

of $\mathbb{Z}/q\mathbb{Z}$; i.e., the FFT. This has the drawback of introducing real numbers to the otherwise purely discrete computations in the PBS. Since real numbers need to be approximated, this results in an approximation error for the polynomial multiplication determined by the used precision. One can think of this approximation error as an additional error being introduced during the blind rotation. The problem is exacerbated by the fact that the FFT over the complex numbers cannot perform the reductions mod q in the FFT domain and thus this needs to be done as an extra step after the multiplication. This requires to represent the unreduced numbers and thus even more precision. In theory, we may scale the precision to our needs given a set of PBS parameters (see e.g. [3] and references therein), but in practice this is more complicated. While the FFT over natively supported data-types in common hardware (i.e., precision of 53 to at most 80 bits) is very fast, moving to arbitrary-precision libraries for higher precision slows it down severely. This means that for any practical purpose, there is a hard limit on the precision of the FFT based multiplication, which essentially yields a lower bound on the error introduced during the PBS. For an exploration of this bound, see e.g. [17]. This critically limits the possible range of parameters for TFHE-like schemes and may lead to inefficient implementations for certain applications.

As should be clear from the above, it would be very desirable to be able to apply the NTT in the context of the PBS. An easy solution is to change the modulus q to some "NTT-friendly" integer such that $\mathbb{Z}/q\mathbb{Z}$ contains an M-th principal root of unity, as done in other lattice-based schemes [2, 26]. But this slows down all operations that involve modular reductions. This can be alleviated to some degree by carefully choosing the modulus and designing reduction algorithms specifically for this choice [20].

In this work, we also explore an alternative route to address this problem: changing the quotient polynomial. We consider some (suitable) M that is a unit in $\mathbb{Z}/q\mathbb{Z}$ for a power-of-two q and choose the corresponding M-th cyclotomic polynomial as the quotient polynomial. The fact that M is a unit in $\mathbb{Z}/q\mathbb{Z}$ does not imply that there exists a principal M-th root of unity in $\mathbb{Z}/q\mathbb{Z}$, but it suffices to enable the use of certain variants of the NTT (cf. Remark 2.5).[†]

As a by-product, changing the quotient polynomial also allows one to mitigate the padding issue alluded to in Section 3.1. When M is a power of two, the degree of the M-th cyclotomic polynomial is $\varphi(M) = M/2$, which means that only a $\varphi(M)/M = \frac{1}{2}$ fraction of the plaintext space can be used. Using e.g. a power-of-three M, the M-th cyclotomic polynomial has degree $\varphi(M) = \frac{2}{3}M$ and thus enables to use a $\frac{2}{3}$ fraction of the plaintext space. So even when opting to change the modulus to an NTT-friendly prime, it may be worth to consider using a different quotient polynomial.

4 GENERAL PBS

Of course, changing the polynomial ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$ requires to generalize the operations behind the programmable bootstrapping (PBS), most obviously the computation of the test polynomial for the blind rotation. The sample extraction requires only a slight adaptation.

[†]If q is a power of two, any attempt to find an M such that $\mathbb{Z}/q\mathbb{Z}$ contains a principal M-th root of unity must fail, since one simultaneously needs M to be co-prime to q, but require $\mathbb{Z}/q\mathbb{Z}$ to contain an element of order M. The latter means that $M \mid \varphi(q) = q/2$, which is clearly incompatible with the first requirement.

In this section, we study how to accommodate the PBS with more general quotient polynomials. Recall from Section 3.1 that the core operation of the PBS is a series of homomorphic multiplications by monomials so as to obtain an RLWE-type encryption of polynomial $X^{-(\mu(m)+e)} \cdot v(X)$. The main challenge in moving to other quotient polynomials is to adapt the computation of the test polynomial, since multiplying $v(X)$ by $X^{-(\mu(m)+e)}$ does not simply yield a (negacyclic) rotation of the coefficients.

We begin by describing the effect of multiplications with X^{-1} modulo a monic polynomial $p(X)$ (that divides $X^M - 1$ for some positive integer M) on the coefficient vector of an element in $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$. Next, we show how to compute the test polynomial for a given table T defining the function to be evaluated during the PBS in different settings.

- We consider the case where the blind rotation should encode the result in the constant coefficient of the resulting RLWE-type ciphertext (the typical case) and where the quotient polynomial is arbitrary (except with above restrictions).
- We describe how to program the test polynomial in case the result of the operation should be embedded in the leading coefficient instead of the constant one.
- We specialize to the case where $p(X)$ is a trinomial, in which case the computation of the test polynomial is simpler and more efficient. In fact, in this latter case, we provide efficient formulas to embed the result in an arbitrary coefficient index.

Finally, we explain how to adapt the sample extraction step.

Remark 4.1. The PBS can be trivially adapted to homomorphically compute $X^{\mu(m)+e} \cdot v(X)$ instead of $X^{-(\mu(m)+e)} \cdot v(X)$, so we may also program $v(X)$ accordingly. That would require to consider multiplications with X instead of X^{-1}, which would slightly simplify Section 4.1. On the other hand, the results and proofs from Sections 4.2.1 to 4.2.3 would be symmetrical with no additional simplification, so we opt for consistency with previous works and consider monomials with negative exponent.

4.1 Monomial Multiplication in $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$

Let $\mathfrak{R}$ be a ring with identity. Consider the monic degree-N polynomial

$$p(X) = X^N + p_{N-1} X^{N-1} + p_{N-2} X^{N-2} + \cdots + p_0$$

defined over $\mathfrak{R}$, and such that $p(X)$ divides $X^M - 1$ for a certain given M. For convenience, we let $p_N = 1$ so that we can write $p(X) = \sum_{j=0}^{N} p_j X^j$.

Clearly, in $\mathfrak{R}[X]/(p(X))$, it holds that $\sum_{j=1}^{N} p_j X^j = -p_0$ and, in turn, that

$$(-p_0)^{-1} \sum_{j=1}^{N} p_j X^j = 1 \iff X^{-1} = (-p_0)^{-1} \sum_{j=0}^{N-1} p_{j+1} X^j \quad . \quad (1)$$

This assumes that p_0 is invertible in $\mathfrak{R}$. Note that this condition is also implied by the fact that $p(X)$ divides $X^M - 1$. Indeed, writing $X^M - 1 = p(X) \cdot q(X)$ for some polynomial $q(X) = \sum_{i=0}^{M-N} q_i X^i \in \mathfrak{R}[X]$, we must have $p_0 q_0 = -1$ or, equivalently, $p_0 (-q_0) = 1$;

hence, the inverse of p_0 exists and is given by $-q_0$. For any polynomial $v(X) = \sum_{i=0}^{N-1} v_i X^i$ in $\mathfrak{R}[X]/(p(X))$ we define

$$\left(v(X)\right)_i \coloneqq v_i$$

the degree-i coefficient of $v(X)$ (in particular, $\left(v(X)\right)_0 = v_0$ denotes the constant term), and $v_N = 0$ for convenience. Now using (1), given such a polynomial $v(X) = \sum_{i=0}^{N-1} v_i X^i$, we have

$$\begin{aligned} X^{-1} v(X) &= \sum_{i=0}^{N-1} v_i X^{i-1} \\ &= \left(\textstyle\sum_{i=1}^{N-1} v_i X^{i-1}\right) + v_0 X^{-1} \\ &= \left(\textstyle\sum_{i=0}^{N-2} v_{i+1} X^i\right) + (-p_0)^{-1} v_0 \left(\textstyle\sum_{i=0}^{N-1} p_{i+1} X^i\right) \\ &= \left(\textstyle\sum_{i=0}^{N-2} \left(v_{i+1} + (-p_0)^{-1} p_{i+1} v_0\right) X^i\right) \\ &\qquad + (-p_0)^{-1} p_N v_0 X^{N-1} \\ &= \sum_{i=0}^{N-1} \left(v_{i+1} + (-p_0)^{-1} p_{i+1} v_0\right) X^i \end{aligned}$$

where the last equality holds due to our definition of $v_N = 0$. Remark that this is the same as

$$X^{-1} v(X) = \sum_{i=0}^{N-1} \left(v_{i+1} + (-p_0)^{-1} p_{i+1} \left(v(X)\right)_0\right) X^i \quad . \tag{2}$$

Intuitively, this means that a multiplication with X^{-1} leads to adding a scaled version of the constant term of $v(X)$ to the other coefficients (where the scaling depends on the index of the coefficient) and a shift.

4.2 Programming the Test Polynomial

We show that one can efficiently compute the test polynomial, where a set of values can be embedded either in the constant or in the leading coefficients of $X^{-t} v(X)$. Extensions to general coefficients are also dealt with; in particular, in the case of trinomials.

4.2.1 Constant coefficient. Given a table T with N input/output values $\left\{T[i]\right\}_{i=0}^{N-1}$, we need to be able to compute a polynomial $v(X)$ such that the constant term of $X^{-t} v(X)$ is $K_t \coloneqq T[t]$. We now show how to do that.

Proposition 4.2. *With the previous notations, define $v(X) = \sum_{i=0}^{N-1} v_i X^i \in \mathfrak{R}[X]/(p(X))$ be such that*

$$v_i = \sum_{j=0}^{i} \frac{p_{i-j}}{p_0} K_j$$

for any $\left\{K_i \in \mathfrak{R}\right\}_{i=0}^{N-1}$. Then $\left(X^{-t} v(X)\right)_0 = K_t$ for all $t \in \{0, \ldots, N-1\}$.

Proof. The proof is by induction over t. Clearly, the result is true for $t = 0$ since $\left(X^{-0} v(X)\right)_0 = v_0 = K_0$.

From (2), for any polynomial $w(X) \in \mathfrak{R}[X]/(p(X))$, we note that $\left(X^{-1} w(X)\right)_0 = \left(w(X)\right)_1 - \frac{p_1}{p_0} \left(w(X)\right)_0$ and, more generally, that

$$\left(X^{-1} w(X)\right)_i = \left(w(X)\right)_{i+1} - \frac{p_{i+1}}{p_0} \left(w(X)\right)_0 \quad .$$

Repeatedly using this observation, we see that

$$\begin{aligned} &\left(X^{-t} v(X)\right)_0 \\ &\quad = \left(X^{-1} (X^{-t+1} v(X))\right)_0 \\ &\quad = \left(X^{-t+1} v(X)\right)_1 - \tfrac{p_1}{p_0} \left(X^{-t+1} v(X)\right)_0 \\ &\quad = \left(X^{-t+2} v(X)\right)_2 - \tfrac{p_2}{p_0} \left(X^{-t+2} v(X)\right)_0 - \tfrac{p_1}{p_0} \left(X^{-t+1} v(X)\right)_0 \\ &\quad \vdots \\ &\quad = \left(X^{-t+t} v(X)\right)_t - \tfrac{p_t}{p_0} \left(X^{-(t-t)} v(X)\right)_0 - \ldots \\ &\qquad\qquad - \tfrac{p_1}{p_0} \left(X^{-t+1} v(X)\right)_0 \\ &\quad = v_t - \textstyle\sum_{j=0}^{t-1} \frac{p_{t-j}}{p_0} \left(X^{-j} v(X)\right)_0 \quad . \end{aligned}$$

To conclude, we simply invoke the induction hypothesis for all $0 \le j < t$:

$$\begin{aligned} \left(X^{-t} v(X)\right)_0 &= v_t - \textstyle\sum_{j=0}^{t-1} \frac{p_{t-j}}{p_0} \left(X^{-j} v(X)\right)_0 \\ &= v_t - \textstyle\sum_{j=0}^{t-1} \frac{p_{t-j}}{p_0} K_j \\ &= \textstyle\sum_{j=0}^{t} \frac{p_{t-j}}{p_0} K_j - \sum_{j=0}^{t-1} \frac{p_{t-j}}{p_0} K_j \\ &= K_t \quad . \end{aligned}$$

□

4.2.2 Leading coefficient. Given a table T with N input/output values $\left\{T[i]\right\}_{i=0}^{N-1}$, we can also compute a polynomial $v(X)$ such that the leading coefficient of $X^{-t} v(X)$ is $K_t \coloneqq T[t]$.

Proposition 4.3. *With the previous notations, define $v(X) = \sum_{i=0}^{N-1} v_i X^i \in \mathfrak{R}[X]/(p(X))$ be such that*

$$v_i = \begin{cases} K_0 & \text{if } i = N-1 \\ -\sum_{j=0}^{i} p_{i-j} K_{j+1} & \text{otherwise} \end{cases}$$

for any $\left\{K_i \in \mathfrak{R}\right\}_{i=0}^{N-1}$. Then $\left(X^{-t} v(X)\right)_{N-1} = K_t$ for all $t \in \{0, \ldots, N-1\}$.

Proof. The proof is similar to the one of Proposition 4.2, but we need an additional equation. In particular, we note that from (2) we obtain

$$\left(v(X)\right)_0 = -p_0 \left(X^{-1} v(X)\right)_{N-1}$$

and, more generally,

$$\begin{aligned} \left(X^{-j} v(X)\right)_0 &= -p_0 \left(X^{-1} X^{-j} v(X)\right)_{N-1} \\ &= -p_0 \left(X^{-j-1} v(X)\right)_{N-1} \end{aligned} \tag{3}$$

for any j. Now recall from the proof of Proposition 4.2 that we have

$$\left(X^{-i} v(X)\right)_0 = v_i - \textstyle\sum_{j=0}^{i-1} \frac{p_{i-j}}{p_0} \left(X^{-j} v(X)\right)_0 \tag{4}$$

for all $0 \le i < N$. Using (3) and (4), we can write

$$\begin{aligned} -p_0 \left(X^{-t} v(X)\right)_{N-1} &= \left(X^{-t+1} v(X)\right)_0 \\ &= v_{t-1} - \textstyle\sum_{j=0}^{t-2} \frac{p_{t-1-j}}{p_0} \left(X^{-j} v(X)\right)_0 \\ &= v_{t-1} + \textstyle\sum_{j=0}^{t-2} p_{t-1-j} \left(X^{-j-1} v(X)\right)_{N-1} \quad . \end{aligned}$$

The case $t = 0$ is trivial. We conclude again using induction for $t > 1$, where the base case $t = 1$ is straightforward to verify. Then,

invoking the induction hypothesis for all $1 \le j < t$:

$$\begin{aligned}-p_0 \left(X^{-t}\, v(X)\right)_{N-1} &= v_{t-1} + \textstyle\sum_{j=0}^{t-2} p_{t-1-j} \left(X^{-(j+1)}\, v(X)\right)_{N-1} \\ &= -\textstyle\sum_{j=0}^{t-1} p_{t-1-j}\, K_{j+1} + \sum_{j=0}^{t-2} p_{t-1-j}\, K_{j+1} \\ &= -p_0\, K_t \quad . \end{aligned}$$ □

4.2.3 Trinomials. This section deals with the special case where $p(X)$ is a trinomial of the form $p(X) = X^N + \epsilon X^{N/2} + 1$ for some even N and $\epsilon \in \{-1, 1\}$. In particular, this means that $p_0 = p_N = 1$, $p_{N/2} = \pm 1$, and $p_i = 0$ for all $i \in \{0, \ldots, N\} \setminus \{0, N/2, N\}$. We demonstrate how to program the test polynomial $v(X)$ such that $K_t = (X^{-t}\, v(X))_s$ for arbitrary s. For this, we need to consider different cases of s.

Proposition 4.4. *Let N be an even positive integer, $p(X) = X^N + \epsilon X^{N/2} + 1$ with $\epsilon \in \{-1, 1\}$, and $s < N/2$ a non-negative integer. If $v(X) = \sum_{i=0}^{N-1} v_i X^i \in \mathfrak{R}[X]/(p(X))$ is such that*

$$v_i = \begin{cases} -\epsilon K_{N/2+(i-s)} - K_{N+(i-s)} & \text{if } 0 \le i < s \\ K_{i-s} & \text{if } s \le i < \frac{N}{2} \\ -\epsilon K_{N/2+(i-s)} & \text{if } \frac{N}{2} \le i < \frac{N}{2} + s \\ \epsilon K_{(i-s)-N/2} + K_{i-s} & \text{if } \frac{N}{2} + s \le i < N \end{cases}$$

for any $\left\{K_i \in \mathfrak{R}\right\}_{i=0}^{N-1}$, then $\left(X^{-t}\, v(X)\right)_s = K_t$ for all $t \in \{0, \ldots, N-1\}$.

Proof. We may specialize Equation (2) to our case as

$$X^{-1}\, v(X) = -\epsilon \left(v(X)\right)_0 X^{\frac{N}{2}-1} - \left(v(X)\right)_0 X^{N-1} + \sum_{i=0}^{N-1} v_{i+1} X^i \quad .$$

From this we deduce

$$\left(X^{-t}\, v(X)\right)_i = \begin{cases} \left(X^{-t+1}\, v(X)\right)_{N/2} & \\ \qquad - \epsilon \left(X^{-t+1}\, v(X)\right)_0 & \text{if } i = \frac{N}{2} - 1 \\ -\left(X^{-t+1}\, v(X)\right)_0 & \text{if } i = N - 1 \\ \left(X^{-t+1}\, v(X)\right)_{i+1} & \text{otherwise} \end{cases} \quad . \tag{5}$$

We may now derive the following expressions for $\left(X^{-t}\, v(X)\right)_s$. For $0 \le t < \frac{N}{2} - s$, we have

$$\left(X^{-t}\, v(X)\right)_s = \left(v(X)\right)_{t+s} \quad .$$

For $\frac{N}{2} - s \le t < N - s$, we reduce it to the case above before applying (5):

$$\begin{aligned}\left(X^{-t}\, v(X)\right)_s &= \left(X^{-(N/2-s-1)} X^{-(t-N/2+s+1)}\, v(X)\right)_s \\ &= \left(X^{-(t-N/2+s+1)}\, v(X)\right)_{N/2-1} \\ &= \left(X^{-(t-N/2+s)}\, v(X)\right)_{N/2} - \epsilon \left(X^{-(t-N/2+s)}\, v(X)\right)_0 \\ &= \left(v(X)\right)_{t+s} - \epsilon \left(v(X)\right)_{t+s-N/2} \quad . \end{aligned}$$

And finally, for $N - s \le t < N$, again by reducing to the previous case:

$$\begin{aligned}\left(X^{-t}\, v(X)\right)_s &= \left(X^{-(N-s-1)} X^{-(t-N+s+1)}\, v(X)\right)_s \\ &= \left(X^{-(t-N+s+1)}\, v(X)\right)_{N-1} \\ &\qquad - \epsilon \left(X^{-(t-N+s+1)}\, v(X)\right)_{N/2-1} \\ &= -\left(X^{-(t-N+s)}\, v(X)\right)_0 - \epsilon \Big(\left(X^{-(t-N+s)}\, v(X)\right)_{N/2} \\ &\qquad - \epsilon \left(X^{-(t-N+s)}\, v(X)\right)_0 \Big) \\ &= -\epsilon \left(X^{-(t-N+s)}\, v(X)\right)_{N/2} \\ &= -\epsilon \left(v(X)\right)_{t+s-N/2} \end{aligned}$$

where we used that $\epsilon^2 = 1$. To sum up,

$$\left(X^{-t}\, v(X)\right)_s = \begin{cases} \left(v(X)\right)_{t+s} & \text{if } 0 \le t < \frac{N}{2} - s \\ \left(v(X)\right)_{t+s} - \epsilon \left(v(X)\right)_{t+s-N/2} & \text{if } \frac{N}{2} - s \le t < N - s \\ -\epsilon \left(v(X)\right)_{t+s-N/2} & \text{if } N - s \le t < N \end{cases} \quad . \tag{6}$$

We now verify the solution for all t.

Case $0 \le t < \frac{N}{2} - s$**:** This implies that $s \le t + s \le \frac{N}{2}$ and thus

$$\left(X^{-t}\, v(X)\right)_s = \left(v(X)\right)_{s+t} = K_{(s+t)-s} = K_t \quad .$$

Case $\frac{N}{2} - s \le t < \frac{N}{2}$**:** This corresponds to the second case of (6) and so we have

$$\left(X^{-t}\, v(X)\right)_s = \left(v(X)\right)_{t+s} - \epsilon \left(v(X)\right)_{t+s-N/2} \quad .$$

The condition implies $\frac{N}{2} \le s+t \le \frac{N}{2}+s$ and $0 \le s+t-\frac{N}{2} \le s$ and so

$$\left(X^{-t}\, v(X)\right)_s = -\epsilon K_{N/2+t} - \epsilon(-K_{N/2+t} - \epsilon K_t) = K_t \quad .$$

Case $\frac{N}{2} \le t < N - s$**:** Again, this corresponds to the second case of (6), but this time we have $\frac{N}{2} + s \le t + s < N$ and $s \le t + s - \frac{N}{2} < \frac{N}{2}$, which yields

$$\left(X^{-t}\, v(X)\right)_s = \epsilon K_{t-N/2} + K_t - \epsilon K_{t-N/2} = K_t \quad .$$

Case $N - s \le t < N$**:** In this case we have $\frac{N}{2} \le t + s - \frac{N}{2} < \frac{N}{2} + s$, and thus

$$\begin{aligned}\left(X^{-t}\, v(X)\right)_s &= -\epsilon \left(v(X)\right)_{t+s-N/2} \\ &= \epsilon^2 K_{N/2+(t+s-N/2-s)} = K_t \quad . \end{aligned}$$ □

We now consider the case $s \ge N/2$.

Proposition 4.5. *Let N be an even positive integer, $p(X) = X^N + \epsilon X^{N/2} + 1$ with $\epsilon \in \{-1, 1\}$ and $N/2 \le s < N$ an integer. If $v(X) = \sum_{i=0}^{N-1} v_i X^i \in \mathfrak{R}[X]/(p(X))$ is such that*

$$v_i = \begin{cases} -K_{N-(s-i)} & \text{if } 0 \le i < \frac{N}{2} \\ -\epsilon K_{N/2-(s-i)} - K_{N-(s-i)} & \text{if } \frac{N}{2} \le i < s \\ K_{-(s-i)} & \text{if } s \le i < N \end{cases}$$

for any $\left\{K_i \in \mathfrak{R}\right\}_{i=0}^{N-1}$, then $\left(X^{-t}\, v(X)\right)_s = K_t$ for all $t \in \{0, \ldots, N-1\}$. □

The same steps from the proof of Proposition 4.4 can be mechanically applied to this case, so we omit the proof of this proposition.

4.2.4 Other polynomials. The approach with trinomials can be adapted to other sparse polynomials by writing out the explicit formulas for $\big((X^{-t}\, v(X))\big)_s$ for each value of t and solving the resulting system. The complexity of the solution will then depend on the sparsity of the quotient polynomial $p(X)$.

For dense polynomials, programming the test polynomial for other coefficients than the leading or the constant one is also possible by viewing the multiplication of a polynomial $v(X)$ by X^{-1} in $\mathfrak{R}[X]/(p(X))$ as a matrix multiplication with the coefficient vector of $v(X)$. This enables to set up a system of N linear equations and N variables, with the solution corresponding to the coefficients of the test polynomial. Of course, this requires that the corresponding linear system indeed has a solution, which is not always the case. This means that for certain choices of $\mathfrak{R}$, $p(X)$ and index of coefficient, programming the test polynomial is not possible.

4.3 Sample Extraction

Assume we are given an RLWE-type encryption

$$(a, b) \in \big((\mathbb{Z}/q\mathbb{Z}[X])/(p[X])\big)^2$$

of a plaintext $\mu = \sum_i \mu_i X^i$ encoding $m(X)$ such that $b - a \cdot s \approx \mu$ for some secret key s. The goal of sample extraction (for a fixed index j) is to compute a vector $\boldsymbol{a} \in (\mathbb{Z}/q\mathbb{Z})^N$ such that $b_j - \langle \boldsymbol{a}, \boldsymbol{s} \rangle \approx \mu_j$, where $\boldsymbol{s} \in (\mathbb{Z}/q\mathbb{Z})^N$ is the vector consisting of the coefficients of s. This is easily done from the observation that $a(X) \cdot s(X) = a(X) \sum s_i X^i = \sum s_i X^i \cdot a(X)$. Since addition is component-wise in $(\mathbb{Z}/q\mathbb{Z})[X]/(p[X])$, this allows one to compute the correct $\boldsymbol{a}$ as $a_i = \big(X^i \cdot a(X)\big)_j$ for all i.

5 PBS-FRIENDLY FUNCTIONS

Recall that the PBS in TFHE is able to evaluate negacyclic functions over $\mathbb{Z}/p\mathbb{Z}$ without the need for a padding bit when using a power-of-two cyclotomic as the quotient polynomial. This is important as it allows the evaluation of the sign function, which can be used to implement boolean gates, like the NAND gate (after a suitable linear combination of the input ciphertexts, see e.g. [8]). Thus we call negacyclic functions *PBS-friendly* for power-of-two cyclotomics. Note that by definition these functions have a function table of the form $K = [a \mid -a]$ for some sub-table a of size $M/2 = N$.

Clearly, the class of PBS-friendly functions is closely related to the quotient polynomial and so it stands to reason that there are different PBS-friendly functions also for other quotient polynomials. In this section, we derive the classes of PBS-friendly functions for trinomials.

5.1 Negative Trinomials

Let $M = 2^\alpha 3^\beta$ with $\alpha, \beta \geq 1$, in which case the quotient polynomial is $\Phi_M(X) = X^N - X^{N/2} + 1$ for $N = M/3$. For simplicity we focus on the case $s = 0$; i.e., we program the table into the constant coefficient of the test polynomial.

Proposition 5.1. *A function over $\mathbb{Z}/p\mathbb{Z}$ is PBS-friendly w.r.t. $\Phi_M(X) = X^N - X^{N/2} + 1$ if its function table K satisfies*

$$K_t = \begin{cases} -K_{t-N} + K_{t-\frac{N}{2}} & \text{if } N \leq t < \frac{3N}{2} \\ -K_{t-\frac{3N}{2}} & \text{if } \frac{3N}{2} \leq t < 2N \\ -K_{t-\frac{3N}{2}} & \text{if } 2N \leq t < \frac{5N}{2} \\ K_{t-\frac{5N}{2}} - K_{t-2N} & \text{if } \frac{3N}{2} \leq t < 3N \end{cases} .$$

In other words, letting $a = \big(K_i \in \mathbb{Z}/q\mathbb{Z}\big)_{i=0}^{N/2-1}$ and $b = \big(K_i \in \mathbb{Z}/q\mathbb{Z}\big)_{i=N/2}^{N-1}$ for some function f with function table K, then f is PBS-friendly for $\Phi_M(X)$ if

$$K = [a \mid b \mid b - a \mid -a \mid -b \mid a - b] .$$

Proof. Using (6) and its counterpart for $s \geq \frac{N}{2}$ we obtain

$$\big(X^{-(N-1)}\, v(X)\big)_s = \begin{cases} \big(v(X)\big)_{N-1} + \big(v(X)\big)_{\frac{N}{2}-1} & \text{if } s = 0 \\ \big(v(X)\big)_{\frac{N}{2}-1+s} & \text{if } 0 < s < \frac{N}{2} \\ -\big(v(X)\big)_{\frac{N}{2}-1} & \text{if } s = \frac{N}{2} \\ -\big[\big(v(X)\big)_{s-1} + \big(v(X)\big)_{s-1-\frac{N}{2}}\big] & \text{if } \frac{N}{2} < s < N \end{cases} \tag{7}$$

and

$$\big(X^{-t}\, v(X)\big)_0 = \begin{cases} \big(v(X)\big)_t & \text{if } 0 \leq t < \frac{N}{2} \\ \big(v(X)\big)_t + \big(v(X)\big)_{t-\frac{N}{2}} & \text{if } \frac{N}{2} \leq s < N \end{cases} . \tag{8}$$

Combining (5) and (7), we get

$$\big(X^{-N}\, v(X)\big)_s = \begin{cases} \big(v(X)\big)_{\frac{N}{2}+s} & \text{if } 0 \leq s \leq \frac{N}{2} \\ -\big(v(X)\big)_s - \big(v(X)\big)_{s-\frac{N}{2}} & \text{if } \frac{N}{2} \leq s < N \end{cases} . \tag{9}$$

Then, we can use (8) and (9) to obtain

$$\big(X^{-t}\, v(X)\big)_0 = \big(X^{-(t-N)} X^{-N}\, v(X)\big)_0 = \begin{cases} \big(v(X)\big)_{t-\frac{N}{2}} & \text{if } N \leq t < \frac{3N}{2} \\ -\big(v(X)\big)_{t-\frac{3N}{2}} & \text{if } \frac{3N}{2} \leq t < 2N \end{cases} .$$

Finally, by applying Proposition 4.4, we see that

$$\big(X^{-t}\, v(X)\big)_0 = \begin{cases} -K_{t-N} + K_{t-\frac{N}{2}} & \text{if } N \leq t < \frac{3N}{2} \\ -K_{t-\frac{3N}{2}} & \text{if } \frac{3N}{2} \leq t < 2N \end{cases} .$$

So, to be able to compute a function without padding, it is necessary that the function table satisfies

$$K_t = \begin{cases} -K_{t-N} + K_{t-\frac{N}{2}} & \text{if } N \leq t < \frac{3N}{2} \\ -K_{t-\frac{3N}{2}} & \text{if } \frac{3N}{2} \leq t < 2N \end{cases} . \tag{10}$$

Applying (10) again, we also see that we need

$$K_t = \begin{cases} -K_{t-\frac{3N}{2}} & \text{if } 2N \leq t < \frac{5N}{2} \\ K_{t-\frac{5N}{2}} - K_{t-2N} & \text{if } \frac{3N}{2} \leq t < 2N \end{cases} .$$

□

5.2 Positive Trinomials

Let $M = 3^\beta$ with $\beta \geq 1$, in which case the quotient polynomial is $\Phi_M(X) = X^N + X^{N/2} + 1$ for $N = 2M/3$. Again, for simplicity we focus on the case $s = 0$; i.e., we program the table into the constant coefficient of the test polynomial.

Proposition 5.2. *A function over $\mathbb{Z}/p\mathbb{Z}$ is PBS-friendly w.r.t. $\Phi_M(X) = X^N + X^{N/2} + 1$ if its function table K satisfies*

$$K_t = -\left(K_{t-N} + K_{t-N/2}\right) \ .$$

In other words, letting $a = \left(K_i \in \mathbb{Z}/q\mathbb{Z}\right)_{i=0}^{N/2-1}$ and $b = \left(K_i \in \mathbb{Z}/q\mathbb{Z}\right)_{i=N/2}^{N-1}$ for some function f with function table K, then f is PBS-friendly for $\Phi_M(X)$ if

$$K = [a \mid b \mid -(a+b)] \ .$$

We omit the proof since it is very similar to the one of Proposition 5.1. It is noteworthy that the class of PBS-friendly functions for positive trinomials includes a function that is similar to a step activation function with two thresholds; e.g., with function table $K = [\mathbf{1}, \mathbf{0}, \mathbf{-1}]$.

6 NOISE GROWTH

During the blind rotation (cf. Section 3.1), an RLWE ciphertext of the form $(a, b = a\,s' + \mu + e)$ is multiplied homomorphically by powers of X. Note that the absolute size of the coefficients of e is not affected by these multiplications if we work in the ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1)$, since these multiplications are simply negacyclic rotations. A potential concern is the noise growth when considering more general rings $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$.

In order to analyze the impact of the blind rotation on the noise, recall that we assume that $p(X)\,q(X) = X^M - 1$ for some polynomial $q(X)$ and $M \in \mathbb{Z}$. Accordingly, any computation in the ring $(\mathbb{Z}/q\mathbb{Z})[X]/(p(X))$ may be viewed as being performed in $(\mathbb{Z}/q\mathbb{Z})[X]/(X^M - 1)$ with a final reduction modulo $p(X)$. It is clear that multiplications with powers of X in $(\mathbb{Z}/q\mathbb{Z})[X]/(X^M - 1)$ also do not affect the noise term as, again, these multiplications simply yield rotations on the coefficients.

It remains to analyze how a reduction$\bmod p(X)$ of a degree $M-1$ polynomial affects the size of its coefficients. This is characterized by the *expansion factor* of the quotient polynomial [21, 22]. We focus on the important case of trinomials. For trinomials, we can show tighter bounds than the ones given in [21] for more general polynomials. Specifically, we show below that the size of the coefficients may grow by at most a factor of

- 2 in case $p(X) = X^N + X^{N/2} + 1$, i.e., a power-of-three-index cyclotomic, and
- 4 in case $p(X) = X^N - X^{N/2} + 1$, i.e., the index of $p(X)$ is of the form $M = 2^\alpha 3^\beta$ with $\alpha, \beta \geq 1$.

Proof. Consider the degree $M-1$ polynomial $f = \sum_{i=0}^{M-1} f_i X^i$. We have:

(1) Suppose $M = 3^\beta$ with $\beta \geq 1$. Then $N = 2M/3$ and $p(X) = X^N + X^{N/2} + 1$. We so have $f(X) \equiv \sum_{i=0}^{3N/2-1} f_i X^i \equiv \sum_{i=0}^{N-1} f_i X^i + \sum_{i=0}^{N/2-1} f_{i+N} X^i(-X^{N/2} - 1) \equiv \sum_{i=0}^{N/2-1}(f_i - f_{i+N})X^i + \sum_{i=N/2}^{N-1}(f_i - f_{i+N/2})X^i \pmod{p(X)}$.

(2) Suppose now $M = 2^\alpha 3^\beta$ with $\alpha, \beta \geq 1$. Then $N = M/3$ and $p(X) = X^N - X^{N/2} + 1$. We get

$$\begin{aligned}
f(X) &\equiv \textstyle\sum_{i=0}^{3N-1} f_i X^i \\
&\equiv \textstyle\sum_{i=0}^{N-1} \left(f_i X^i + f_{i+N} X^i (X^{N/2} - 1) + f_{i+2N} X^i (-X^{N/2})\right) \\
&\equiv \textstyle\sum_{i=0}^{N-1} \left((f_i - f_{i+N})X^i + (f_{i+N} - f_{i+2N})X^{i+N/2}\right) \\
&\equiv \textstyle\sum_{i=0}^{N-1} (f_i - f_{i+N})X^i + \sum_{i=N/2}^{N-1} (f_{i+N/2} - f_{i+3N/2})X^i \\
&\quad + \textstyle\sum_{i=0}^{N/2-1} (f_{i+3N/2} - f_{i+5N/2})X^i (X^{N/2} - 1) \\
&\equiv \textstyle\sum_{i=0}^{N/2-1} (f_i - f_{i+N} - f_{i+3N/2} + f_{i+5N/2})X^i \\
&\quad + \textstyle\sum_{i=N/2}^{N-1} (f_i - f_{i+N} + f_{i+N/2} - f_{i+3N/2} + f_{i+N} - f_{i+2N})X^i \\
&\equiv \textstyle\sum_{i=0}^{N/2-1} (f_i - f_{i+N} - f_{i+3N/2} + f_{i+5N/2})X^i \\
&\quad + \textstyle\sum_{i=N/2}^{N-1} (f_i + f_{i+N/2} - f_{i+3N/2} - f_{i+2N})X^i \pmod{p(X)}
\end{aligned}$$

which concludes the proof. □

In conclusion, while the noise may increase in contrast to the negacyclic PBS, at least in the cases that we target in this work, this increase is very moderate and can be easily compensated by adjusting the parameters of the scheme.

7 EXAMPLE PARAMETERS

The (external) product in the TFHE family and similar cryptosystems boils down to a series of polynomial multiplications modulo $X^M - 1$, or a divisor thereof. In particular, the programmable bootstrapping as described in Section 3 corresponds to the case $M = 2N$ and polynomial multiplications modulo $p(X) = X^N + 1$ where N is a power of two. Using our results from Section 4, we may now change the quotient polynomial.

For efficiency reasons it makes sense to focus on sparse polynomials in order to allow for efficient modular reductions, so we will consider the case of trinomials as quotient polynomials.

We give below example parameters that admit the application of the NTT for multiplying polynomials. We select a power of three for M, concretely $M = 3^7 = 2187$, and the M-th cyclotomic polynomial $p(X) = X^N + X^{N/2} + 1$ as the quotient polynomial, where $N = 2 \cdot 3^6 = 1458$. Note that $N/M = 2/3 > 1/2$, which yields a larger plaintext space. Given $\boldsymbol{K} = (K_0, \ldots, K_{N-1}) \in (\mathbb{Z}/q\mathbb{Z})^N$ we choose to embed the values of $\boldsymbol{K}$ in the constant coefficients of $X^{-t}\,v(X)$. Then Proposition 4.4 shows that we can compute the coefficients of $v(X) = \sum_{i=0}^{N-1} v_i X^i$ as

$$v_i = \begin{cases} K_i & \text{if } i < \frac{N}{2} \\ K_i + K_{i-N/2} & \text{otherwise} \end{cases} \ .$$

We now describe possible choices of the modulus q.

Power-of-two modulus. As previously noted, it is often very convenient to choose a power-of-two modulus. Common choices are $q = 2^{32}$ or $q = 2^{64}$. Since we choose M to be a power of three, clearly M and q are co-prime. Still, we cannot apply the [plain] NTT directly in this case, since $\mathbb{Z}/q\mathbb{Z}$ does not contain a principal M-th root of unity. However, since M is a unit in $\mathbb{Z}/q\mathbb{Z}$, variants like Schönhage's algorithm and Nussbaumer's algorithm are still applicable (cf. Remark 2.5) so this choice is still compatible with some variants of the NTT.

Prime modulus. A power-of-two modulus q restricts the type of NTT that is applicable independently of the quotient polynomial. As a consequence, even when using a trinomial as quotient polynomial, it may make sense to choose an NTT-friendly integer for modulus q in order to enable the plain NTT for efficiency reasons. One may now wonder why one would then go through the trouble of using a trinomial since a power-of-two quotient polynomial would also work, but recall that in this case we still retain the advantage of an increased plaintext space in comparison to a power-of-two cyclotomic.

For [plain] NTT, we need to choose the modulus q such that $\mathbb{Z}/q\mathbb{Z}$ contains a principal M-th root of unity. One way of ensuring this is by setting q to be a prime such that M divides $q-1$. We give examples of such primes of size 32 bits and 64 bits to match common data types:

- $2^{31} < q = 10 \cdot 3^{18} + 1 < 2^{32}$, and
- $2^{63} < q = 4 \cdot 3^{39} + 1 < 2^{64}$.

Corresponding principal M-th roots of unity are easy to obtain. For example, one may set $\omega = (1024)^{3^{18-7}} \bmod q$ for $q = 10 \cdot 3^{18} + 1$, and $\omega = (625)^{3^{39-7}} \bmod q$ for $q = 4 \cdot 3^{39} + 1$. They are both respective principal 3^7-roots of unity.

TFHE-like parameters. We note that finding and optimizing parameters for TFHE-like schemes (and FHE schemes in general) is an involved task and outside of the scope of this work. But to give an idea that our choice of parameters above can give significant advantages, we adjust and compare with a typical parameter set for TFHE-like schemes, e.g. as used in (some instantiations of) Concrete [9]. In this setting, it is common to parameterize the RLWE scheme with a secret from a uniform binary distribution, a ring dimension of $N = 1024$ and discrete or discretized Gaussian error distribution with (normalized by q) standard deviation of 2^{-25}. First note that our ring dimension N is somewhat larger, which results in an increase of security, but also a decrease in performance. On the other hand, we need to reduce the error in the bootstrapping key to accommodate for the larger dimension and for the additional noise growth due to the use of a non-power-of-two cyclotomic (cf. Section 6), which reduces security. Using the lattice estimator [1][‡] suggests that the increase in dimension easily compensates for the reduced error rate and, in fact, is likely to admit even smaller error. This might allow to optimize other parameters, e.g. the basis for the decomposition, which could improve performance.

As stated, fully optimizing parameters is a complex task, but we conclude that at the very least, our approach allows to find parameters that provide a new trade-off between performance and available plaintext space that is not possible using only power-of-two cyclotomics. Depending on the application and the required plaintext space, this can significantly improve the performance of the scheme.

8 CONCLUSION

This paper generalized the programmable bootstrapping for TFHE-like cryptosystems so as to make it compliant with the number-theoretic transform (NTT) or its variants. Formulas for programming the test polynomial in the extended settings were provided, including for the important case of q a power of two. Earlier implementations relied on the FFT for the polynomial multiplications (with the limitations that were pointed out).

[‡]https://github.com/malb/lattice-estimator

REFERENCES

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology* 9, 3 (2015), 169–203. https://doi.org/10.1515/jmc-2015-0016

[2] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2021. CRYSTALS-Kyber (version 3.02) – Submission to round 3 of the NIST post-quantum project. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf

[3] Daniel J. Bernstein. 2001. Multidigit multiplication for mathematicians. (Aug. 2001). Unpublished manuscript, available at https://cr.yp.to/papers.html#m3.

[4] Jean-François Biasse and Luis Ruiz. 2015. FHEW with efficient multibit bootstrapping. In *Progress in Cryptology – LATINCRYPT 2015 (Lecture Notes in Computer Science, Vol. 9230)*, K. Lauter and F. Rodríguez-Henríquez (Eds.). Springer, 119–135. https://doi.org/10.1007/978-3-319-22174-8_7

[5] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. 2018. Large FHE gates from tensored homomorphic accumulator. In *Progress in Cryptology – AFRICACRYPT 2018 (Lecture Notes in Computer Science, Vol. 10831)*, A. Joux, A. Nitaj, and T. Rachidi (Eds.). Springer, 217–251. https://doi.org/10.1007/978-3-319-89339-6_13

[6] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. 2022. FINAL: Faster FHE instantiated with NTRU and LWE. Cryptology ePrint Archive, Report 2022/074. https://ia.cr/2022/074.

[7] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2016. TFHE: Fast Fully Homomorphic Encryption Library. https://tfhe.github.io/tfhe/

[8] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology* 33, 1 (2020), 34–91. https://doi.org/10.1007/s00145-019-09319-x

[9] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 2020. CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE. In *8th Workshop on Encrypted Computing and Applied Homomorphic Cryptography (WAHC 2020)*, M. Brenner and T. Lepoint (Eds.). Leibniz Universität IT Services, 57–63. https://doi.org/10.25835/0072999

[10] Ilaria Chillotti, Marc Joye, and Pascal Paillier. 2021. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In *Cyber Security Cryptography and Machine Learning (CSCML 2021) (Lecture Notes in Computer Science, Vol. 12716)*, S. Dolev et al. (Eds.). Springer, 1–19. https://doi.org/10.1007/978-3-030-78086-9_1

[11] James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.* 19, 90 (1965), 297–301. https://doi.org/10.2307/2003354

[12] Léo Ducas and Daniele Micciancio. 2015. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology – EUROCRYPT 2015, Part I (Lecture Notes in Computer Science, Vol. 9056)*, E. Oswald and M. Fischlin (Eds.). Springer, 617–640. https://doi.org/10.1007/978-3-662-46800-5_24

[13] Léo Ducas and Daniele Micciancio. 2017. FHEW – A Fully Homomorphic Encryption library (v2.0 alpha). https://github.com/lducas/FHEW

[14] W. M. Gentleman and G. Sande. 1966. Fast Fourier transforms: For fun and profit. In *Proc. of AFIPS '66*. ACM Press, 563–578. https://doi.org/10.1145/1464291.1464352

[15] Craig Gentry. 2010. Computing arbitrary functions of encrypted data. *Commun. ACM* 53, 3 (2010), 97–105. https://doi.org/10.1145/1666420.1666444 Earlier version in STOC 2009.

[16] Marc Joye. 2022. SoK: Fully homomorphic encryption over the [discretized] torus. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022, 4 (2022), 661–692. https://doi.org/10.46586/tches.v2022.i4.661-692

[17] Jakub Klemsa. 2021. Fast and error-free negacyclic integer convolution using extended Fourier transform. In *Cyber Security Cryptography and Machine Learning (CSCML 2021) (Lecture Notes in Computer Science, Vol. 12716)*, S. Dolev et al. (Eds.). Springer, 282–300. https://doi.org/10.1007/978-3-030-78086-9_22

[18] Kamil Kluczniak. 2022. NTRU-v-um: Secure fully homomorphic encryption from NTRU. Cryptology ePrint Archive, Report 2022/089. https://ia.cr/2022/089.

[19] Donald E. Knuth. 1998. *The Art of Computer Programming: Seminumerical Algorithms* (3rd ed.). Vol. 2. Addison-Wesley.

[20] Patrick Longa and Michael Naehrig. 2016. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *Cryptology and Network Security (CANS 2016) (Lecture Notes in Computer Science, Vol. 10052)*, S. Foresti and G. Persiano (Eds.). Springer, 124–139. https://doi.org/10.1007/978-3-319-48965-0_8

[21] Vadim Lyubashevsky and Daniele Micciancio. 2005. Generalized compact knapsacks are collision resistant. Electron. Colloquium Comput. Complex., TR05-142. https://eccc.weizmann.ac.il/eccc-reports/2005/TR05-142/.

[22] Vadim Lyubashevsky and Daniele Micciancio. 2006. Generalized compact knapsacks are collision resistant. In *Automata, Languages and Programming (ICALP 2006), Part II (Lecture Notes in Computer Science, Vol. 4052)*, M. Bugliesi et al. (Eds.). Springer, 144–155. https://doi.org/0.1007/11787006_13

[23] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On ideal lattices and learning with errors over rings. *J. ACM* 60, 6 (2013), 43:1–43:35. https://doi.org/10.1145/2535925

[24] Daniele Micciancio and Yuriy Polyakov. 2021. Bootstrapping in FHEW-like cryptosystems. In *9th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC 2021)*, M. Brenner et al. (Eds.). ACM Press, 17–28. https://doi.org/10.1145/3474366.3486924

[25] Henri J. Nussbaumer. 1980. Fast polynomial transform algorithms for digital convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28, 2 (1980), 205–215. https://doi.org/10.1109/TASSP.1980.1163372

[26] Yuriy Polyakov, Kurt Rohloff, Gerard W. Ryan, and Dave Cousins. 2021. PALISADE Lattice Cryptography Library, User Manual (v1.11.2). https://palisade-crypto.org/

[27] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009), 34:1–34:40. https://doi.org/10.1145/1568318.1568324

[28] Ronald L. Rivest, Len Adleman, and Michael L. Detouzos. 1978. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, R. A. DeMillo et al. (Eds.). Academic Press, 165–179. Available at https://people.csail.mit.edu/rivest/pubs.html#RAD78.

[29] Arnold Schönhage. 1977. Schnelle Multiplication von Polynomen über Körpen der Charakteristik 2. *Acta Informatica* 7 (1977), 395–398. https://doi.org/10.1007/BF00289470

[30] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. 2009. Efficient public key encryption based on ideal lattices. In *Advances in Cryptology – ASIACRYPT 2009 (Lecture Notes in Computer Science, Vol. 5912)*, M. Matsui (Ed.). Springer, 617–635. https://doi.org/10.1007/978-3-642-10366-7_36

[31] Joachim von zur Gathen and Jürgen Gerhard. 2013. *Modern Computer Algebra* (3rd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9781139856065