Fast and Error-Free Negacyclic Integer Convolution using Extended Fourier Transform *

Jakub Klemsa

Czech Technical University in Prague, Czech Republic
jakub.klemsa@fel.cvut.cz

Abstract. With the rise of lattice cryptography, (negacyclic) convolution has received increased attention. E.g., the NTRU scheme internally employs cyclic polynomial multiplication, which is equivalent to the standard convolution, on the other hand, many Ring-LWE-based cryptosystems perform negacyclic polynomial multiplication. A method by Crandall implements an efficient negacyclic convolution over a finite field of prime order using an extended Discrete Galois Transform (DGT) – a finite field analogy to Discrete Fourier Transform (DFT). Compared to DGT, the classical DFT runs faster by an order of magnitude, however, it suffers from inevitable rounding errors due to finite floating-point number representation. In a recent Fully Homomorphic Encryption (FHE) scheme by Chillotti et al. named TFHE, small errors are acceptable (although not welcome), therefore we decided to investigate the application of DFT for negacyclic convolution.
The primary goal of this paper is to suggest a method for fast negacyclic convolution over integer coefficients using an extended DFT. The key contribution is a thorough analysis of error propagation, as a result of which we derive parameter bounds that can guarantee even error-free results. We also suggest a setup that admits rare errors, which allows to increase the degree of the polynomials and/or their maximum norm at a fixed floating-point precision. Finally, we run benchmarks with parameters derived from a practical TFHE setup. We achieve around 24× better times than the generic NTL library (comparable to Crandall's method) and around 4× better times than a naïve approach with DFT, with no errors.

Keywords: Negacyclic Convolution, Fast Fourier Transform, Fully Homomorphic Encryption

1 Introduction

In 1994, Peter Shor discovered efficient quantum algorithms for discrete logarithm and factoring [26], which started the quest to design novel quantum-proof algorithms, aka. *Post-Quantum Cryptography*. Since then, there have emerged

* This is the full version of the paper.

many new schemes, which are based on various problems that are believed to be quantum hard. E.g., supersingular elliptic curve isogeny [18], multivariate cryptography [12], or lattice cryptography [2], in particular Learning With Errors (LWE) and its variants [24,21]. In addition, many *Fully Homomorphic Encryption* (FHE) schemes (e.g. [6,8]) belong to lattice-based ones, including Gentry's first-ever FHE scheme [14]. Most notably, the NIST's Post-Quantum Cryptography Standardization Program entered the third "Selection Round" in July 2020 [23], while lattice-based cryptosystems occur among the selected algorithms.

With the popularity of lattice-based cryptography, the need for its fast implementation has risen. Besides linear algebra, many schemes require a fast algorithm for cyclic (i.e., $\bmod X^N-1$) or negacyclic (i.e., $\bmod X^N+1$) polynomial multiplication. Some schemes work with polynomial coefficients modulo an integer (e.g., NTRU [17]), however, our main interest is in the TFHE scheme [8], where negacyclic multiplication of integer-torus polynomials is performed. Here the *torus* refers to reals modulo 1, i.e., the fractional part of a real number. In practice, torus elements are represented as unsigned integers, which represent the fraction of 1 uniformly in the interval $[0, 1)$. It follows that integer-torus polynomial multiplication can be performed with their integer representation. Also note that TFHE accepts small errors – we prefer to avoid them, but their impact is not fatal for decryption.

Recently, there have emerged efforts to make TFHE work with multivalued plaintexts [7], also applications of TFHE for homomorphic evaluation of neural networks show promising results [5]. In particular, for neural networks, it holds that they are quite error-tolerant (also verified in [5]), which supports the acceptability of errors.

Problem Statement. Our goal is to develop a method for fast negacyclic multiplication of univariate integer polynomials. For this method, we aim to estimate and tune its parameters in order to provide certain guarantees of its correctness. As outlined above, we will not focus solely on an error-free case and we will also accept the scenario, where errors may rarely occur. Last but not least—as we intend our method also for an FPGA implementation—we derive all results in a generic manner, i.e., without sticking to a concrete platform, although we run our tests on an ordinary 64-bit machine.

Related Work. There is a long and rich history of methods for fast multiplication over various rings, ranging from Karatsuba's algorithm [19], through Fast Fourier Transform (FFT; [9]) to Schönhage-Strassen algorithm [25]. Most of these methods are based on a similar principle as Bernstein pointed out in his survey [4].

It was the classical cyclic convolution, which was accelerated by FFT and Convolution Theorem, and which can be employed for polynomial multiplication modulo $X^N - 1$, too. On the contrary, polynomial multiplication modulo

X^N+1 (negacyclic convolution) cannot be directly calculated via FFT. One possible approach was implemented as a part of the TFHE Library [28], although not discussed in the paper [8]. However, this method suffers from a four-tuple redundancy in its intermediate results. An effective (non-redundant) method for negacyclic convolution has been proposed by Crandall [11] and recently improved by Al Badawi et al. [3]. In these methods, polynomials are considered over a finite ring and both authors employed a number-theoretic variant of FFT, named DGT, which operates on the field $\mathsf{GF}(p^2)$. On the one hand, DGT calculates exact results (as opposed to FFT, where rounding errors occur and propagate), on the other hand, it runs significantly slower as it uses modular arithmetics.

Our Contributions. We propose an efficient algorithm for negacyclic convolution over the reals, for which we derive estimates of bounds on the maximum error and its variance. Based on our estimates, we show that our method can be used for an error-free negacyclic convolution over integers. Or—in case we admit errors—we suggest to relax the estimates in order to achieve higher performance: either in terms of shorter number representation (useful in particular for FPGA), longer polynomials, or larger polynomial coefficients that can be processed. Finally, we provide experimental benchmarking results of our implementation as well as we evaluate its rounding error magnitudes and result correctness, even with remarkably underestimated parameters.

Paper Outline. In Section 2, we provide a brief overview of the required mathematical background, i.e., cyclic and negacyclic convolutions, their relation to modular polynomial multiplication, as well as the Discrete Fourier Transform and Convolution Theorem. Next, in Section 3, we revisit a straightforward FFT-based approach for negacyclic polynomial multiplication, and we propose a method that avoids the calculation of redundant intermediates. We analyze error propagation thoroughly in Section 4, where we suggest lower bounds on floating point type bit-precision in order to guarantee certain levels of correctness. In Section 5, we discuss the implementation details and we propose a set of testing parameters with respect to TFHE. Using these parameters, we benchmark our implementation and we also examine the error magnitude and result correctness. Finally, we conclude our paper in Section 6.

2 Preliminaries

In this section, we briefly recall some basic mathematical concepts related to convolution and Discrete Fourier Transform.

Cyclic & Negacyclic Convolution. Let $\mathbf{f}, \mathbf{g} \in \mathbb{C}^N$ for some $N \in \mathbb{N}$. As opposed to the classical cyclic convolution defined as

$$(\mathbf{f} * \mathbf{g})_k \coloneqq \sum_{j=0}^{N-1} f_j g_{(k-j) \bmod N}, \tag{1}$$

negacyclic convolution adds a factor of -1 with each wrap of the cyclic index at $\mathbf{g}$, i.e.,

$$(\mathbf{f} \bar{*} \mathbf{g})_k \coloneqq \sum_{j=0}^{N-1} (-1)^{\lfloor \frac{k-j}{N} \rfloor} f_j g_{(k-j) \bmod N}. \tag{2}$$

With respect to polynomials, it is easy to verify that the cyclic convolution calculates the coefficients of a product of two polynomials modulo $X^N - 1$. Indeed, their coefficients can be considered cyclic since $X^N = 1$. On the other hand, the *negacyclic* convolution calculates the coefficients of a product of two polynomials modulo $X^N + 1$, since $X^N = -1$ adds a factor of -1 with each wrap.

Convolution Theorem. A relation known as the *Convolution Theorem* (CT) states an equality between the Fourier image of convoluted vectors and an element-wise (dyadic) product of their respective Fourier images (in the discrete variant). CT writes as follows:

$$\mathcal{F}(\mathbf{f} * \mathbf{g}) = \mathcal{F}(\mathbf{f}) \odot \mathcal{F}(\mathbf{g}), \tag{3}$$

where $\mathcal{F}(\cdot)$ stands for the *Discrete Fourier Transform* (DFT) and $\odot$ denotes the dyadic multiplication of two vectors. In fact, DFT is a change of basis, defined as

$$\mathcal{F}(\mathbf{f})_k \coloneqq \sum_{j=0}^{N-1} f_j \exp\Big(-\frac{2\pi \mathrm{i} jk}{N}\Big) = F_k, \tag{4}$$

$$\mathcal{F}^{-1}(\mathbf{F})_j = \frac{1}{N} \sum_{k=0}^{N-1} F_k \exp\Big(\frac{2\pi \mathrm{i} jk}{N}\Big) = f_j. \tag{5}$$

Convolution theorem has gained its practical significance after *Fast Fourier Transform* (FFT) was (re)invented[1] in 1965 by Cooley & Tukey [9]. As opposed to a direct calculation of DFT coefficients, which requires $O(N^2)$ time, FFT runs in $O(N \log N)$. Next, by the convolution theorem, one can calculate the convolution of two vectors as $\mathbf{f} * \mathbf{g} = \mathcal{F}^{-1}\big(\mathcal{F}(\mathbf{f}) \odot \mathcal{F}(\mathbf{g})\big)$, which spends $O(N \log N)$ time, compared to $O(N^2)$ needed for a direct calculation.

3 Efficient Negacyclic Convolution

First, we describe a method for negacyclic convolution that uses the standard cyclic convolution and FFT. We identify its redundancy and briefly comment on possible workarounds. Next, we outline an approach that yields no redundancy and achieves a $4\times$ better performance than the previous method.

[1] Goldstine [15] attributes an FFT-like algorithm to C. F. Gauss dating to around 1805.

3.1 Redundant Approach

Since (negacyclic) convolution is equivalent to (negacyclic) polynomial modular multiplication, we switch to the polynomial point of view for now. Interested in polynomial multiplication modulo $X^N + 1$, we note that $X^{2N} - 1 = (X^N - 1) \cdot (X^N + 1)$. Hence, we can calculate the product first modulo $X^{2N} - 1$ (via cyclic convolution of $2N$ elements) and then only reduce the result modulo $X^N + 1$. This method can be optimized based on the following observations.

Observation 1 (Redundancy of negacyclic extension)**.** *Let $p \in \mathbb{R}[X]$ be a real-valued polynomial of degree $N-1$, $N \in \mathbb{N}$, and let $\bar{p}(X) \coloneqq p(X) - X^N \cdot p(X)$ be a negacyclic extension of $p(X)$. Then the Fourier image of $\mathsf{coeffs}(\bar{p})$ contains zeros at eventh positions (indexed from 0). In addition, the remaining coefficients (at oddth positions) are mirrored and conjugated. I.e.,*

$$\mathcal{F}\big(\mathsf{coeffs}(\bar{p})\big) = (0, P_1, 0, P_3, \ldots, 0, P_{N-1}, 0, \overline{P_{N-1}}, \ldots, 0, \overline{P_3}, 0, \overline{P_1}). \tag{6}$$

Note 1. Given N input (real-valued) polynomial coefficients, $\mathcal{F}\big(\mathsf{coeffs}(\bar{p})\big)$ needs to calculate $2N$ complex values, i.e., $4N$ real values. The redundancy is clearly in the N complex zeros and in the $N/2$ complex conjugates.

Observation 2 (Convolution of negacyclic extensions)**.** *Let $p, q \in \mathbb{R}[X]$ be real-valued polynomials of degree $N-1$ for some $N \in \mathbb{N}$ and let $\bar{p}, \bar{q}$ be their respective negacyclic extensions. Then it holds*

$$\mathsf{coeffs}\big(p \cdot q \bmod (X^N+1)\big) = \frac{1}{2}\,\mathcal{F}^{-1}\Big(\mathcal{F}\big(\mathsf{coeffs}(\bar{p})\big) \odot \mathcal{F}\big(\mathsf{coeffs}(\bar{q})\big)\Big)[0 \ldots N-1]. \tag{7}$$

By Observation 1, it follows that the dyadic multiplication in (7) can only be performed at odd positions of the first half, the rest can be copied (with appropriate sign). Also note that after $\mathcal{F}^{-1}$, the coefficients are negacyclic, hence we can only take the first half of the vector. This method is implemented in the original TFHE Library [28].

Possible Improvements. The clear goal is to omit all calculations leading to redundant values as outlined in Note 1. Digging deeper into FFT, we deduced the same initial step as proposed by Crandall [11] in his method for negacyclic convolution (namely, the folding step). However, without the additional twisting step, we ended up with a bunch of numbers, from which we were not able to recover the original values efficiently. Therefore, we decided to adapt the concept of the method by Crandall.

3.2 Non-Redundant Approach

The method for negacyclic polynomial multiplication by Crandall [11] is intended for polynomials over $\mathbb{Z}_p$ and it employs internally the *Discrete Galois Transform* (DGT). DGT is an analogy to DFT, which operates over the field

$\mathsf{GF}(p^2)$ for a Gaussian prime number p, whereas DFT operates over $\mathbb{C}$. Note that recently Al Badawi et al. [3] extended the Crandall's method for non-Gaussian primes, too. The Crandall's method prepends DGT with two steps: folding and twisting. In the following definition we propose an analogous transformation using DFT.

Definition 1. *Let* $\mathbf{f} \in \mathbb{R}^N$ *for some* $N \in \mathbb{N}$*,* N *even. We define the* Discrete Fourier Negacyclic Transform *(DFNT, denoted* $\bar{\mathcal{F}}$*) as follows:*

$$\bar{\mathcal{F}}(\mathbf{f}) \coloneqq \mathcal{F}\Big(\underbrace{\big(\mathbf{f}[0 \dots {}^{N}\!/_{2} - 1] + \mathrm{i} \cdot \mathbf{f}[{}^{N}\!/_{2} \dots N - 1]\big)}_{folding} \underbrace{\odot \big(\omega_{2N}^{j}\big)_{j=0}^{{}^{N}\!/_{2}-1}}_{twisting}\Big), \tag{8}$$

where $\omega_{2N}^{j} = \exp\big(\frac{2\pi \mathrm{i} j}{2N}\big)$ *and* $\mathcal{F}$ *stands for the ordinary DFT. For the inverse DFNT, we have*

$$\mathbf{t} \coloneqq \mathcal{F}^{-1}(\mathbf{F}) \odot (\omega_{2N}^{-j})_{j=0}^{{}^{N}\!/_{2}-1}, \tag{9}$$

$$\bar{\mathcal{F}}^{-1}(\mathbf{F}) = \big[\Re(\mathbf{t}), \Im(\mathbf{t})\big]. \tag{10}$$

Note 2. We will refer to DFNT, where DFT is internally calculated via FFT, as the *Fast Fourier Negacyclic Transform* (FFNT).

With respect to negacyclic convolution, DFNT has two important properties:

1. given N reals at input, it outputs ${}^{N}\!/_{2}$ complex numbers, i.e., there is *no redundancy*, unlike in the previous approach, and
2. it can be used for negacyclic convolution in the same manner as DFT for cyclic convolution, a theorem follows.

Theorem 1 (Negacyclic Convolution Theorem; NCT). *Let* $\mathbf{f}, \mathbf{g} \in \mathbb{R}^N$ *for some* $N \in \mathbb{N}$*,* N *even. It holds*

$$\bar{\mathcal{F}}(\mathbf{f} \bar{*} \mathbf{g}) = \bar{\mathcal{F}}(\mathbf{f}) \odot \bar{\mathcal{F}}(\mathbf{g}). \tag{11}$$

For a full description of negacyclic convolution over the reals via NCT see Algorithm 1. Next, we analyze this algorithm from the error propagation point of view, which allows us to apply this method for negacyclic convolution over integers, too.

4 Analysis of Error Propagation

Since Algorithm 1 operates implicitly with real numbers (starting $N = 4$, ω_{2N}'s are irrational), there emerge rounding errors provided that we use a standard finite floating-point representation. In this section, we analyze Algorithm 1 from the error propagation point of view and we derive estimates of the bounds of errors as well as their variance. Based on our estimates, we derive a bound for sufficient bit-precision of the employed floating point representation, which guarantees error-free convolution over the ring of integers. We also provide an estimate of the bit-precision based on error variance and the 3σ-rule. In addition and as a byproduct, we derive all bounds for cyclic convolution, too. First of all, we revisit the FFT algorithm, as we will refer to it later.

Algorithm 1 Efficient Negacyclic Convolution over $\mathbb{R}$.

Input: $\mathbf{f}, \mathbf{g} \in \mathbb{R}^N$ for some $N \in \mathbb{N}$, N even.
Precompute: $\omega_{2N}^j \coloneqq \exp\left(\frac{2\pi \mathrm{i} j}{2N}\right)$ for $j = -N/2 + 1 \dots N/2 - 1$.
Output: $\mathbf{h} \in \mathbb{R}^N$, $\mathbf{h} = \mathbf{f} \circledast \mathbf{g}$.

for $j = 0 \dots N/2 - 1$ **do**
  $f'_j = f_j + \mathrm{i} f_{j+N/2}$ // fold
  $g'_j = g_j + \mathrm{i} g_{j+N/2}$
for $j = 0 \dots N/2 - 1$ **do**
  $f''_j = f'_j \cdot \omega_{2N}^j$ // twist
  $g''_j = g'_j \cdot \omega_{2N}^j$
$\mathbf{F} = \mathcal{F}_{N/2}(\mathbf{f}'')$, $\mathbf{G} = \mathcal{F}_{N/2}(\mathbf{g}'')$
for $j = 0 \dots N/2 - 1$ **do**
  $H_j = F_j \cdot G_j$
$\mathbf{h}'' = \mathcal{F}^{-1}{}_{N/2}(\mathbf{H})$
for $j = 0 \dots N/2 - 1$ **do**
  $h'_j = h''_j \cdot \omega_{2N}^{-j}$ // untwist
for $j = 0 \dots N/2 - 1$ **do**
  $h_j = \Re(h'_j)$ // unfold
  $h_{j+N/2} = \Im(h'_j)$
return h

FFT in Brief. FFT [9] is a recursive algorithm, which builds upon the following observation: for $N = n_1 \cdot n_2$ and $k = k_1 + k_2 n_1$, we can write the k-th Fourier coefficient of an $\mathbf{f} \in \mathbb{C}^N$ as

$$\mathcal{F}(\mathbf{f})_{k_1+k_2 n_1} = \sum_{j_2=0}^{n_2-1} \left(\underbrace{\left(\sum_{j_1=0}^{n_1-1} f_{j_2+j_1 n_2} \omega_{n_1}^{j_1 k_1} \right)}_{\mathcal{F}\left((f_{j_2+j_1 n_2})_{j_1=0}^{n_1-1}\right)_{k_1}} \omega_N^{-j_2 k_1} \right) \omega_{n_2}^{-j_2 k_2}, \tag{12}$$

where

$$\omega_N^j = \exp\left(\frac{2\pi \mathrm{i} j}{N}\right), \tag{13}$$

while ω's can be precomputed.

Note 3. There exist two major FFT data paths for N a power of two: the Cooley-Tukey data path [9] (aka. decimation-in-time), and the Gentleman-Sande data path [13] (aka. decimation-in-frequency). At this point, let us describe the decimation-in-time data path, we will discuss their implementation consequences later in Section 5.

For N a power of two, FFT splits its input into two halves and proceeds recursively. Next, it multiplies the results with ω's, and finally it proceeds adequate pairs; see (14) and (15).

At the end of the recursion we have for $N = 2$:

$$\mathsf{FFT}_2 \, |f_0 \quad f_1| = |f_0 + f_1 \quad f_0 - f_1|. \tag{14}$$

Next, for $N \geq 4$ we have

$$\mathsf{FFT}_N(\mathbf{f}): \left|\begin{matrix} f_0 & f_1 \\ f_2 & f_3 \\ \vdots & \vdots \\ f_{N-2} & f_{N-1} \end{matrix}\right|_{n_1 \times n_2 = N/2 \times 2} \xrightarrow[\text{(recursively)}]{\mathsf{FFT}_{N/2} \text{ columns}} \left|\begin{matrix} f'_0 & f'_1 \\ f'_2 & f'_3 \\ \vdots & \vdots \\ f'_{N-2} & f'_{N-1} \end{matrix}\right| \odot \left|\begin{matrix} 1 & 1 \\ 1 & \omega_N^{-1\cdot 1} \\ \vdots & \vdots \\ 1 & \omega_N^{-1\cdot(N/2-1)} \end{matrix}\right|_{\omega_N^{-j_2 k_1}} \longrightarrow$$

$$\rightarrow \left|\begin{matrix} f''_0 & f''_1 \\ f''_2 & f''_3 \\ \vdots & \vdots \\ f''_{N-2} & f''_{N-1} \end{matrix}\right| \xrightarrow{\mathsf{FFT}_2 \text{ rows}} \left|\begin{matrix} f''_0 + f''_1 & f''_0 - f''_1 \\ f''_2 + f''_3 & f''_2 - f''_3 \\ \vdots & \vdots \\ f''_{N-2} + f''_{N-1} & f''_{N-2} - f''_{N-1} \end{matrix}\right| = \left|\begin{matrix} F_0 & F_{N/2} \\ F_1 & F_{N/2+1} \\ \vdots & \vdots \\ F_{N/2-1} & F_{N-1} \end{matrix}\right|. \tag{15}$$

FFT^{-1} proceeds similarly to the direct transformation with the following exceptions:

1. in the second step, it multiplies by $\omega_N^{j_2 k_1}$ (i.e., with a positive exponent), and
2. the final result is multiplied by $1/N$ (only once at the top level).

4.1 Error Propagation through FFT and FFNT

Let us begin with two lemmas, which provide bounds on the error and variance of complex multiplication and FFT, respectively. Note that we will assume for our estimates of variance bounds that the rounding errors are uniformly random and independent.

Note 4. We will distinguish two types of the maximum norm $\|\cdot\|_\infty$ over $\mathbb{C}^N$. For 1. error vectors, and for 2. other complex vectors, we consider:

1. the maximum of real and imaginary parts (i.e., rectangular), and
2. the maximum of absolute values (i.e., circular), respectively.

Lemma 1. *Let $a, b \in \mathbb{C}$, $|a| \leq A_0$ and $|b| \leq B_0$ for some $A_0, B_0 \in \mathbb{R}^+$. Then*

$$|a \cdot b| \leq A_0 \cdot B_0, \tag{16}$$

$$\|\mathsf{Err}(a \cdot b)\|_\infty \lessapprox \sqrt{2} \cdot \big(A_0 \cdot \|\mathsf{Err}(b)\|_\infty + B_0 \cdot \|\mathsf{Err}(a)\|_\infty\big), \quad \textit{and} \tag{17}$$

$$\mathsf{Var}\big(\mathsf{Err}(a \cdot b)\big) \lessapprox 2 \cdot \big(A_0^2 \cdot \mathsf{Var}\big(\mathsf{Err}(b)\big) + B_0^2 \cdot \mathsf{Var}\big(\mathsf{Err}(a)\big)\big), \tag{18}$$

where we neglected second-order error terms and for (18)*, we further assumed that the errors of a and b are independent.*

Proof. Let $a = (p + E_p) + \mathrm{i}(q + E_q)$ and $b = (r + E_r) + \mathrm{i}(s + E_s)$, where we denote the parts' bounds as $|p| \leq P_0$ etc. According to Note 4, we split the complex error into parts – we write for the real part (similarly for the complex part)

$$\mathsf{Err}\big(\Re(a \cdot b)\big) = pE_r + rE_p - (qE_s + sE_q) + negl., \tag{19}$$

which can be bounded as

$$\begin{aligned}\big|\mathsf{Err}\big(\Re(a\cdot b)\big)\big| &\lessapprox P_0\|\mathsf{Err}(b)\|_\infty + R_0\|\mathsf{Err}(a)\|_\infty + Q_0\|\mathsf{Err}(b)\|_\infty + S_0\|\mathsf{Err}(a)\|_\infty \lessapprox \\ &\lessapprox (P_0+Q_0)\|\mathsf{Err}(b)\|_\infty + (S_0+R_0)\|\mathsf{Err}(a)\|_\infty. \end{aligned} \tag{20}$$

Since $|p+\mathrm{i}q| \lessapprox A_0$, we can bound $P_0+Q_0 \lessapprox \sqrt{2}A_0$ and the result (17) follows, similarly for (18). □

Lemma 2. *Let $\mathbf{f}\in\mathbb{C}^N$, where $N=2^\nu$ for some $\nu\in\mathbb{N}$, $\|\mathbf{f}\|_\infty \le 2^{\varphi_0}$ for some $\varphi_0\in\mathbb{N}$, and let χ denote the bit-precision of ω's as well as all intermediate values during the calculation of $\mathsf{FFT}_N(\mathbf{f}) =: \mathbf{F}$, represented as a floating point type. Then*

$$\|\mathbf{F}\|_\infty \le 2^{\varphi_0+\nu}, \tag{21}$$

$$\|\mathsf{Err}(\mathbf{F})\|_\infty \lessapprox c_H\cdot\big(\sqrt{2}+1\big)^\nu + c_N\cdot 2^\nu \quad (\textit{for } \nu\ge 2), \quad \textit{and} \tag{22}$$

$$\mathsf{Var}\big(\mathsf{Err}(\mathbf{F})\big) \lessapprox d_H\cdot 3^\nu + d_N\cdot 4^\nu \quad (\textit{for } \nu\ge 2), \tag{23}$$

where

$$\begin{aligned} c_H &= 2(\sqrt{2}-1)\cdot\|\mathsf{Err}(\mathbf{f})\|_\infty + (2-\sqrt{2})\cdot 2^{\varphi_0-\chi+1}, & c_N &= -(2+\sqrt{2})\cdot 2^{\varphi_0-\chi-1},\\ d_H &= {}^2\!/_3\ \mathsf{Var}\big(\mathsf{Err}(\mathbf{f})\big) - {}^8\!/_{27}\,2^{2\varphi_0-2\chi}, & d_N &= {}^1\!/_6\,2^{2\varphi_0-2\chi}. \end{aligned} \tag{24}$$

Proof. We write

$$\mathsf{FFT}_N\colon\ \mathsf{FFT}_2\circ(\odot\,\omega_N)\circ\mathsf{FFT}_{N/2}, \tag{25}$$

from where we derive recurrence relations for the bounds on absolute value, error and variance.

In each recursion level, the values propagate to a lower level, then they are multiplied by a complex unit and two such values are added, or subtracted. Firstly, note that in every level the initial bound on the absolute value is doubled, hence (21) follows.

Regarding the errors, it is important to note that the final FFT_2 acts on two values, each of which has been previously multiplied by $\omega_N^{j_2k_1}$, where j_2 ranges in $\{0,1\}$. I.e., one value is multiplied by 1 and only the other is multiplied by a (mostly) non-trivial complex unit, which is rounded to χ bits of precision, i.e., $\|\mathsf{Err}(\omega)\|_\infty \le 2^{-\chi-1}$. Putting things together, we get the following recurrence relations for the bounds on the error and its variance after ν levels, respectively:

$$\begin{aligned} E_\nu &= \sqrt{2}\cdot\big(1\cdot E_{\nu-1} + 2^{\varphi_0+\nu-1}\cdot 2^{-\chi-1}\big) + E_{\nu-1} = \\ &= \big(\sqrt{2}+1\big)\cdot E_{\nu-1} + \sqrt{2}\cdot 2^{\varphi_0+\nu-\chi-2}, \end{aligned} \tag{26}$$

$$E_2 = (E_1 + 2^{\varphi_0+1}\cdot\underbrace{E_{\omega_4}}_{=0})\cdot\sqrt{2} + E_1 = (\sqrt{2}+1)E_1 = 2(\sqrt{2}+1)E_0, \quad \text{and} \tag{27}$$

$$\begin{aligned} V_\nu &= 2\cdot\big(1^2\cdot V_{\nu-1} + (2^{\varphi_0+\nu-1})^2\cdot {}^1\!/_{12}\,(2^{-\chi})^2\big) + V_{\nu-1} = \\ &= 3V_{\nu-1} + {}^1\!/_3\,2^{2\varphi_0+2\nu-2\chi-3}, \end{aligned} \tag{28}$$

$$V_2 = 3V_1 = 6V_0, \tag{29}$$

where in (27), we applied the fact that ω_4 is error-free; cf. (13). Also note that the error more than doubles in each step (while the bound only doubles), therefore the χ bits of precision are sufficient and rounding errors can be neglected. The results follow by solving (26) and (27), and (28) and (29), respectively. □

In the following proposition, we bound the error and variance of the result of cyclic and negacyclic convolution via FFT / FFNT, respectively. For a quick reference, we provide an overview of these methods in (30) and (31), respectively:

$$\begin{aligned} \mathbf{f} &\xrightarrow{\mathsf{FFT}_N} \mathbf{F} \\ \mathbf{g} &\xrightarrow{\mathsf{FFT}_N} \mathbf{G} \end{aligned} \xrightarrow{\odot} \mathbf{H} \xrightarrow{\mathsf{FFT}_N^{-1}} \mathbf{h} = \mathbf{f} * \mathbf{g}, \tag{30}$$

$$\begin{aligned} \mathbf{f} &\xrightarrow{\text{fold}} \mathbf{f}' \xrightarrow{\text{twist}} \mathbf{f}'' \xrightarrow{\mathsf{FFT}_{N/2}} \bar{\mathbf{F}} \\ \mathbf{g} &\xrightarrow{\text{fold}} \mathbf{g}' \xrightarrow{\text{twist}} \mathbf{g}'' \xrightarrow{\mathsf{FFT}_{N/2}} \bar{\mathbf{G}} \end{aligned} \xrightarrow{\odot} \bar{\mathbf{H}} \xrightarrow{\mathsf{FFT}_{N/2}^{-1}} \mathbf{h}'' \xrightarrow{\text{untwist}} \mathbf{h}' \xrightarrow{\text{unfold}} \bar{\mathbf{h}} = \mathbf{f} \mathbin{\bar{*}} \mathbf{g}. \tag{31}$$

Proposition 1. *Let $\mathbf{f}, \mathbf{g} \in \mathbb{R}^N$, where $N = 2^\nu$ for some $\nu \in \mathbb{N}$, $\|\mathbf{f}\|_\infty \leq 2^{\varphi_0}$ and $\|\mathbf{g}\|_\infty \leq 2^{\gamma_0}$ for some $\varphi_0, \gamma_0 \in \mathbb{N}$, and let χ denote the bit-precision of ω's as well as all intermediate values during the calculation of $\mathsf{FFT}_N(\cdot)$ and its inverse, represented as a floating point type. We denote $\mathbf{h} \coloneqq \mathsf{FFT}_N^{-1}\big(\mathsf{FFT}_N(\mathbf{f}) \odot \mathsf{FFT}_N(\mathbf{g})\big)$ and $\bar{\mathbf{h}} \coloneqq \mathsf{FFNT}_N^{-1}\big(\mathsf{FFNT}_N(\mathbf{f}) \odot \mathsf{FFNT}_N(\mathbf{g})\big)$, while we consider the errors as $\|\mathsf{Err}(\mathbf{h})\|_\infty = \|\mathbf{h} - \mathbf{f} * \mathbf{g}\|_\infty$ and $\|\mathsf{Err}(\bar{\mathbf{h}})\|_\infty = \|\bar{\mathbf{h}} - \mathbf{f} \mathbin{\bar{*}} \mathbf{g}\|_\infty$, respectively. Then*

$$\log\|\mathsf{Err}(\mathbf{h})\|_\infty \lessapprox (2\nu - 2) \cdot \log\big(\sqrt{2}+1\big) + \varphi_0 + \gamma_0 - \chi + 4, \tag{32}$$

$$\log \mathsf{Var}\big(\mathsf{Err}(\mathbf{h})\big) \lessapprox 4\nu + 2\varphi_0 + 2\gamma_0 - 2\chi - 1 - \log(3), \quad \textit{and} \tag{33}$$

$$\log\|\mathsf{Err}(\bar{\mathbf{h}})\|_\infty \lessapprox (2\nu - 4) \cdot \log\big(\sqrt{2}+1\big) + \varphi_0 + \gamma_0 - \chi + 4 + \log(3) + 1/2, \tag{34}$$

$$\log \mathsf{Var}\big(\mathsf{Err}(\bar{\mathbf{h}})\big) \lessapprox 4\nu + 2\varphi_0 + 2\gamma_0 - 2\chi - 3. \tag{35}$$

Proof. Find the proof in Appendix A. □

We apply our estimates of the error and variance bounds in order to derive two basic parameter setups for convolution over integers: an error-free setup and a setup with rare errors based on the 3σ-rule; see the following corollary.

Corollary 1. *Provided that*

$$\chi_0^{(c.)} \gtrsim \underbrace{2\log\big(\sqrt{2}+1\big)}_{\approx 2.54} \cdot \nu + \varphi_0 + \gamma_0 + \underbrace{5 - 2\log\big(\sqrt{2}+1\big)}_{\approx 2.46}, \quad \textit{or} \tag{36}$$

$$\chi_0^{(nc.)} \gtrsim \underbrace{2\log\big(\sqrt{2}+1\big)}_{\approx 2.54} \cdot \nu + \varphi_0 + \gamma_0 + \underbrace{5 + \log(3) + 1/2 - 4\log\big(\sqrt{2}+1\big)}_{\approx 2.00}, \tag{37}$$

we have $\|\mathsf{Err}(\mathbf{h})\|_\infty \lesssim 1/2$, or $\|\mathsf{Err}(\bar{\mathbf{h}})\|_\infty \lesssim 1/2$, which means an error-free cyclic, or negacyclic convolution on integers via FFT_N, or FFNT_N, respectively. I.e.,

for $\mathbf{f}, \mathbf{g} \in \mathbb{Z}^N$*, we have*

$$\left\lfloor \mathsf{FFT}_N^{-1}\big(\mathsf{FFT}_N(\mathbf{f}) \odot \mathsf{FFT}_N(\mathbf{g})\big) \right\rceil = \mathbf{f} * \mathbf{g}, \quad \textit{or} \tag{38}$$

$$\left\lfloor \mathsf{FFNT}_N^{-1}\big(\mathsf{FFNT}_N(\mathbf{f}) \odot \mathsf{FFNT}_N(\mathbf{g})\big) \right\rceil = \mathbf{f} \bar{*} \mathbf{g}, \tag{39}$$

respectively, up to negligible probability.

Next, if

$$\chi_{3\sigma}^{(c.)} \gtrsim 2\nu + \varphi_0 + \gamma_0 + \underbrace{1/2 \log(6)}_{\approx 1.29}, \quad \textit{or} \tag{40}$$

$$\chi_{3\sigma}^{(nc.)} \gtrsim 2\nu + \varphi_0 + \gamma_0 + \underbrace{\log(3) - 1/2}_{\approx 1.08}, \tag{41}$$

we have $3\sqrt{\mathsf{Var}\big(\mathsf{Err}(\mathbf{h})\big)} \lesssim 1/2$*, or* $3\sqrt{\mathsf{Var}\big(\mathsf{Err}(\bar{\mathbf{h}})\big)} \lesssim 1/2$*, which estimates the required floating point type precision for the respective convolution variant based on the* 3σ*-rule.*

Note 5. In the most common practical setting with the `binary64` type as per IEEE 754 standard [1] (aka. `double`), we have $\chi = 53$ bits of precision. For the 80-bit variant of the extended precision format (aka. `long double`), we have $\chi = 64$ bits of precision.

5 Implementation & Experimental Results

In this section, we briefly comment on how we use the data paths in our implementation (as outlined in Note 3), we discuss the choice of parameters with respect to TFHE, and then we focus on the following:

1. benchmarking with other implementations using chosen parameters,
2. performance on long polynomials using both 64-bit `double` and 80-bit `long double` floating point number representations, and
3. error magnitude and correctness of the results.

Implementation Remarks. In our implementation of the Cooley-Tukey data path [9], we adapted the 4-vector approach from the Nayuki Project [22], which optimizes the RAM access for the most common 64-bit architectures. In a similar manner, we implemented the Gentleman-Sande data path [13]. To calculate FFT properly, both data paths require a specific reordering of their input or output, respectively. The reordering is based on bit-reversal of position indexes, counting from 0. E.g., for 16 elements (4 bits), we exchange the elements at positions $5 \leftrightarrow 10$, since $5 =$ `0b0101` and $10 =$ `0b1010`.

Since our goal is solely convolution, i.e., we do not care about the exact order of the FFT coefficients, the bit-reverse reordering can be omitted, as pointed out

by Crandall and Pomerance [10]. By construction, it follows that the Gentleman-Sande data path must be used for the direct transformation and the Cooley-Tukey data path for the inverse.

For benchmarking purposes, we also adopted some code from the TFHE Library [28] to compare the redundant and non-redundant approaches; cf. Sections 3.1 and 3.2, respectively.

Relation to the TFHE Parameters. The main (cryptographic) motivation of our algorithm for negacyclic convolution over integers is the negacyclic polynomial multiplication in the TFHE scheme [8]. Below we outline a relation of the TFHE parameters to the parameters of negacyclic convolution via FFNT. As a result, we suggest a reasonable parameter setup for benchmarking.

In TFHE, negacyclic polynomial multiplication occurs in the bootstrapping procedure (namely, in the calculation of the external product), where an integer polynomial is multiplied by a torus polynomial. The coefficients of the right-hand side (torus) polynomial can be represented as integers scaled to $[0, 1)$ and bounded by 2 to the power of their bit-precision, denoted by τ. In the left-hand side (integer) polynomial, the coefficients are bounded by 2^γ, where γ is one of the fundamental TFHE parameters. By construction, the parameter γ is smaller than τ, namely, $\gamma \leq \tau/l$, where l is another TFHE parameter. In a corner case, it can be $\gamma = 1$ and the bound can be hence as low as 2^0.

Based on our preliminary calculations for multivalue TFHE, we need the degree of TFHE polynomials to be at least $N = 2^{14}$ for 8-bit plaintexts with 128-bit security, and the torus precision to be at least $\tau = 34$ (both can be smaller for shorter plaintexts). Finally, we suggest to run the tests using polynomials with $\varphi_0 = \gamma_0 = \tau/2 = 17$ and $N = 2^{10}, \ldots, 2^{14}$.

5.1 Benchmarking Results

As a reference for benchmarking of our implementation [20] of negacyclic convolution, we have chosen the NTL Library [27] and the redundant method (as used in the original TFHE Library [28]; cf. Section 3.1), for which we used the same implementation of FFT as for our non-redundant method. Note that the implementation by Al Badawi et al. [3] shows similar results to the popular NTL (only about 1.01–1.2× faster) and they also show that NTL is faster than the concurrent FLINT Library [16]. For NTL, we tested both `ZZ_pX` and `ZZ_pE` classes, while the latter shows slightly better performance, hence we used that for benchmarking. Find the results of our benchmarks in Table 1.

Note 6. During the parameter setup, we silently passed over the fact that $\chi = 53$ (bit-precision of `double`) is lower than our 3σ-rule estimates for all tested ν's, as per (41) in Corollary 1. Indeed, they dictate $\chi_{3\sigma}^{(nc.)} \gtrsim 2\nu + \varphi_0 + \gamma_0 + 1.08 = 55.08 \ldots 63.08$. For this reason, we reran the scenario with $\nu = 14$ for 1 000-times, we checked the results for correctness, and we did not detect *any* error across all tested polynomials.

Degree (N)	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
NTL [ms]	0.617	1.258	2.643	6.132	12.771
FFT_{2N} [ms]	0.122	0.230	0.458	0.982	2.277
FFNT_N [ms]	0.036	0.069	0.120	0.243	0.541
FFNT_N over FFT_{2N}	3.35×	3.33×	3.82×	4.04×	4.21×
FFNT_N avg. error [‰]	0.06	0.08	0.12	0.18	0.27
FFNT_N max. error [‰]	0.37	0.55	0.98	1.47	1.95

Table 1: Mean time per negacyclic multiplication of uniformly random polynomials with $\|p\|_\infty \leq 2^{17}$ using NTL (similar times as FLINT), FFT_{2N} on negacyclic extension (implemented in [28]), and FFNT_N, both using 64-bit `double`. Speedup of FFNT_N over FFT_{2N}. Average and maximum rounding errors of FFNT_N. 1 000 runs per degree and method on an Intel Core i7-8550U CPU @ 1.80GHz.

5.2 Performance on Long Polynomials

As a reference for other prospective applications of our method, we tested our code on longer polynomials, too. We provide the performance results using both 64-bit `double` and 80-bit `long double` in Figure 1.

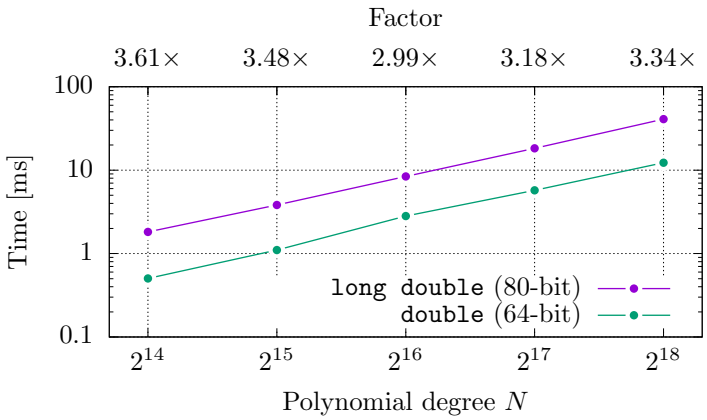

Fig. 1: Mean time per polynomial multiplication $\bmod X^N+1$ and speedup factor of `double` over `long double`. Uniformly random polynomials with $\|p\|_\infty \leq 2^{17}$, 1 000 measurements.

5.3 Error Magnitude & Correctness on Long Polynomials

As outlined in Note 6, our experimental setup exceeds the derived theoretical bounds, even for lower-degree polynomials. Hence, our next goal is to evaluate the error magnitude as well as to check the correctness of the results. We tested the following input polynomial scenarios:

1. uniformly random coefficients (bounded as $\|p\|_\infty \leq 2^{\varphi_0}$), and
2. all coefficients equal to the bound 2^{φ_0}.

Find the results of the random polynomial setup in Figure 2, where we tested both 64-bit `double` and 80-bit `long double` implementations.

Regarding the setup with all coefficients equal to the bound, we ran the same scenarios as for random polynomials, cf. Figure 2. The only correct results were obtained for the setup with $\|p\|_\infty \leq 2^{17}$ and $N = 2^{14}$, regardless the floating point type in use (either 64-bit, or 80-bit).

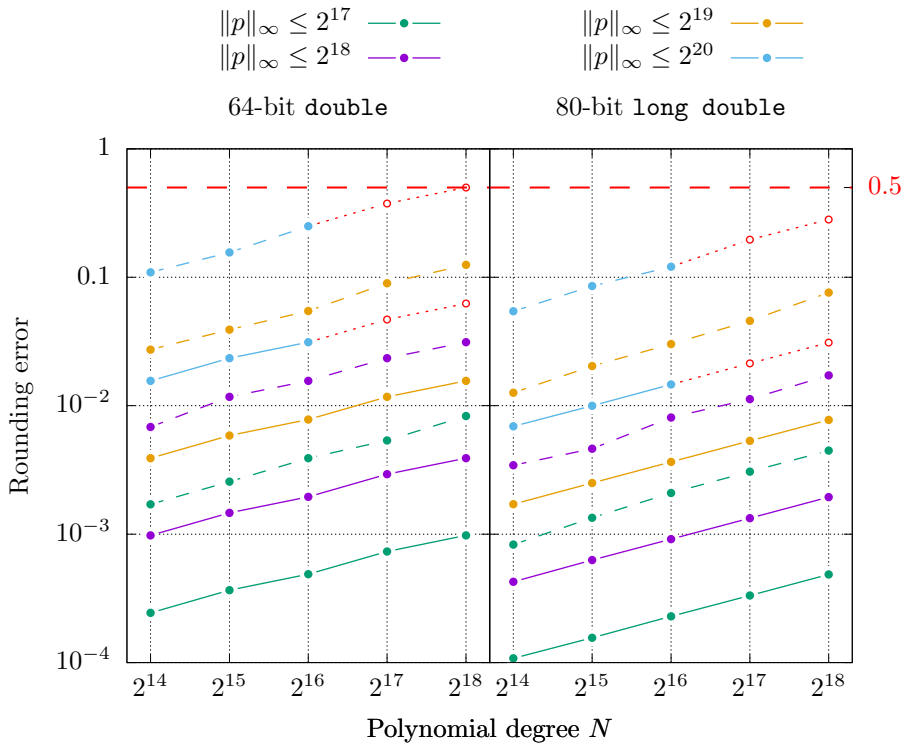

Fig. 2: Median (solid) and Maximum (dashed) rounding errors for uniformly random polynomials. Erroneous results emphasized by empty red circles. 10 measurements per degree, bound and floating point type.

Discussion. We observed a factor $\sim 4\times$ speedup of FFNT_N (i.e., the non-redundant approach) over FFT_{2N} (i.e., the redundant approach). Compared to NTL, which calculates the coefficients precisely using a number-theoretic transform, our FFT-based method shows by more than an order of magnitude better results. Even though we ran our tests with underestimated precision, we obtained correct results for much larger polynomials with uniformly random coefficients. Note that random-like polynomials occur in TFHE, hence our benchmarking scenario with random polynomials is representative for the usage with TFHE.

However and to our surprise, the 80-bit floating point type did not calculate correctly any extra scenario over the 64-bit type. We assume that this is because the steps in the polynomial degree or in the maximum norm bound are too big to make the difference between the 53 and 64 bits of their mantissa precision, respectively.

6 Conclusion

We showed that FFT-based convolution algorithms can significantly outperform similar algorithms based on number-theoretic transforms, and they can still guarantee error-free results in the integer domain. We derived estimates of the lower bound of the employed floating point type for error-free cyclic and negacyclic convolutions, as well as we suggested the bounds based on the 3σ-rule.

We suggested a set of testing parameters for negacyclic convolution with particular respect to the usage with the TFHE Scheme on a multivalue plaintext space. We ran a benchmark that compares the popular NTL Library, the approach that is used in the TFHE Library, and our approach. Compared to the generic NTL Library, which employs a number-theoretic transform, and to the TFHE Library approach, which calculates redundant intermediate values, we achieved a speedup of around $24\times$ and $4\times$, respectively.

Finally, our experiments have shown approximate bounds for practical error-free results. Namely, we could multiply polynomials without errors up to degree $N = 2^{16}$ and norm $\|p\|_\infty \leq 2^{20}$ with uniformly random coefficients, and up to degree $N = 2^{14}$ with coefficients equal to 2^{17}. To conclude, we find our approach particularly useful for negacyclic integer polynomial multiplication, not only in TFHE.

Future Directions. Our aim is to implement a version based on the 64-bit signed integer type instead of `double`, where we would keep the exponent at one place for the entire array. Such an approach requires less demanding arithmetics and it would serve as a proof-of-concept for a propective FPGA implementation.

Acknowledgments. We would like to thank Ahmad Al Badawi for useful comments and remarks.

References

1. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, 2019.
2. Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.
3. Ahmad Al Badawi, Bharadwaj Veeravalli, and Khin Mi Mi Aung. Efficient polynomial multiplication via modified discrete galois transform and negacyclic convolution. In *Future of Information and Communication Conference*, pages 666–682. Springer, 2018.
4. Daniel J Bernstein. Multidigit multiplication for mathematicians. 2001.
5. Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *Annual International Cryptology Conference*, pages 483–512. Springer, 2018.
6. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.
7. Sergiu Carpov, Malika Izabachène, and Victor Mollimard. New techniques for multi-value input homomorphic evaluation and applications. In *Cryptographers' Track at the RSA Conference*, pages 106–126. Springer, 2019.
8. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.
9. James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.
10. Richard Crandall and Carl B Pomerance. *Prime numbers: a computational perspective*, volume 182. Springer Science & Business Media, 2006.
11. Richard E Crandall. Integer convolution via split-radix fast galois transform. *Center for Advanced Computation Reed College*, 1999.
12. Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.
13. W Morven Gentleman and Gordon Sande. Fast fourier transforms: for fun and profit. In *Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 563–578, 1966.
14. Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford University, 2009.
15. Herman H Goldstine. A history of numerical analysis from the 16th through the 19th century. *Bull. Amer. Math. Soc*, 1:388–390, 1979.
16. William Hart, Fredrik Johansson, and Sebastian Pancratz. FLINT: Fast Library for Number Theory. https://www.flintlib.org/, 2011.
17. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.
18. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.
19. Anatolii Alekseevich Karatsuba and Yu P Ofman. Multiplication of many-digital numbers by automatic computers. In *Doklady Akademii Nauk*, volume 145, pages 293–294. Russian Academy of Sciences, 1962.

20. Jakub Klemsa. Benchmarking FFNT. https://gitlab.fit.cvut.cz/klemsjak/ffnt-benchmark, 2021.
21. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.
22. Fast Fourier transform in x86 assembly. https://www.nayuki.io/page/fast-fourier-transform-in-x86-assembly, 2021. Accessed: 2021-01-30.
23. NIST. NIST's Post-Quantum Cryptography Program Enters "Selection Round". https://www.nist.gov/news-events/news/2020/07/nists-post-quantum-cryptography-program-enters-selection-round, 2020.
24. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.
25. Arnold Schönhage and Volker Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7(3):281–292, 1971.
26. Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
27. Victor Shoup et al. NTL: A library for doing number theory. https://libntl.org/, 2001.
28. TFHE: Fast Fully Homomorphic Encryption Library over the Torus. https://github.com/tfhe/tfhe, 2016.

Appendix

A Proof of Proposition 1

Proof. Let us begin with the cyclic convolution. By (30) and Lemma 1 and 2, we have

$$\begin{aligned}\|\mathsf{Err}(\mathbf{F}\odot\mathbf{G})\|_\infty &\lessapprox \Big(\underbrace{c_H^{(\mathbf{f})}\cdot\big(\sqrt{2}+1\big)^\nu}_{\gtrapprox\|\mathsf{Err}(\mathbf{F})\|_\infty}\cdot\underbrace{2^{\gamma_0+\nu}}_{\geq\|\mathbf{G}\|_\infty}+c_H^{(\mathbf{g})}\cdot\big(\sqrt{2}+1\big)^\nu\cdot 2^{\varphi_0+\nu}\Big)\cdot\sqrt{2}=\\ &=\big(\sqrt{2}+1\big)^\nu\cdot 2^{\nu+\varphi_0+\gamma_0-\chi+2}\cdot\big(2-\sqrt{2}\big)\cdot\sqrt{2}=:E_{\mathbf{H}},\quad\text{and}\end{aligned}\tag{42}$$

$$\begin{aligned}\mathsf{Var}\big(\mathsf{Err}(\mathbf{F}\odot\mathbf{G})\big) &\lessapprox \Big(\underbrace{d_N^{(\mathbf{f})}\cdot 2^{2\nu}}_{\gtrapprox\mathsf{Var}\left(\mathsf{Err}(\mathbf{F})\right)}\cdot\underbrace{2^{2\gamma_0+2\nu}}_{\geq\|\mathbf{G}\|_\infty^2}+d_N^{(\mathbf{g})}\cdot 2^{2\nu}\cdot 2^{2\varphi_0+2\nu}\Big)\cdot 2=\\ &={}^2\!/_3\cdot 2^{4\nu+2\varphi_0+2\gamma_0-2\chi}=:V_{\mathbf{H}},\end{aligned}\tag{43}$$

which we apply as the initial error and variance bound to (22) and (23), respectively, together with multiplication by ${}^1\!/_N=2^{-\nu}$, which poses the only difference between FFT^{-1} and FFT from the error point of view. We neglect other than leading terms and we get

$$\begin{aligned}\|\mathsf{Err}(\mathbf{h})\|_\infty &\lessapprox 2^{-\nu}\cdot\underbrace{2(\sqrt{2}-1)\cdot E_{\mathbf{H}}}_{\approx c_H^{(\mathbf{H})}}\cdot\big(\sqrt{2}+1\big)^\nu\lessapprox\\ &\lessapprox\big(\sqrt{2}+1\big)^{2\nu-2}\cdot 2^{\varphi_0+\gamma_0-\chi+4},\quad\text{and}\end{aligned}\tag{44}$$

$$\mathsf{Var}\big(\mathsf{Err}(\mathbf{h})\big)\lessapprox 2^{-2\nu}\cdot\underbrace{{}^1\!/_6\cdot 2^{2(\varphi_0+\gamma_0+2\nu)-2\chi}}_{=d_N^{(\mathbf{H})}}\cdot 4^\nu={}^1\!/_6\cdot 2^{4\nu+2\varphi_0+2\gamma_0-2\chi},\tag{45}$$

and the cyclic results follow.

For the negacyclic convolution, we feed DFT with a folded and twisted input vector; cf. (31). It enters DFT with error bounded as

$$\|\mathsf{Err}(\mathbf{f}'')\|_\infty\lessapprox(1\cdot 0+2^{\varphi_0+1/2}\cdot 2^{-\chi-1})\cdot\sqrt{2}=2^{\varphi_0-\chi}.\tag{46}$$

Regarding variance, it shows that the term with $\mathsf{Var}\big(\mathsf{Err}(\mathbf{f}'')\big)$ will be neglected. Next, we precompute

$$\begin{aligned}c_H^{(\mathbf{f}'')}&=2(\sqrt{2}-1)\cdot\|\mathsf{Err}(\mathbf{f}'')\|_\infty+(2-\sqrt{2})\cdot 2^{\varphi_0+1/2-\chi+1}\lessapprox\\ &\lessapprox 6(\sqrt{2}-1)\cdot 2^{\varphi_0-\chi},\quad\text{and}\end{aligned}\tag{47}$$

$$d_N^{(\mathbf{f}'')}={}^1\!/_6\,2^{2(\varphi_0+1/2)-2\chi},\tag{48}$$

and apply into

$$\begin{aligned}
\|\mathsf{Err}(\bar{\mathbf{F}} \odot \bar{\mathbf{G}})\|_\infty &\lessapprox \Big(\underbrace{c_H^{(\mathbf{f}'')} \cdot \big(\sqrt{2}+1\big)^{\nu-1}}_{\gtrapprox \|\mathsf{Err}(\bar{\mathbf{F}})\|_\infty} \cdot \underbrace{2^{\gamma_0+1/2+\nu-1}}_{\geq \|\bar{\mathbf{G}}\|_\infty} + \\
&+ c_H^{(\mathbf{g}'')} \cdot \big(\sqrt{2}+1\big)^{\nu-1} \cdot 2^{\varphi_0+1/2+\nu-1}\Big) \cdot \sqrt{2} = \\
&= 3\big(\sqrt{2}+1\big)^{\nu-2} \cdot 2^{\nu+\varphi_0+\gamma_0-\chi+2} =: E_{\bar{\mathbf{H}}}, \quad \text{and} \qquad (49) \\
\mathsf{Var}\big(\mathsf{Err}(\bar{\mathbf{F}} \odot \bar{\mathbf{G}})\big) &\lessapprox \Big(\underbrace{d_N^{(\mathbf{f}'')} \cdot 4^{\nu-1}}_{\gtrapprox \mathsf{Var}\left(\mathsf{Err}(\bar{\mathbf{F}})\right)} \cdot \underbrace{2^{2\gamma_0+1+2\nu-2}}_{\geq \|\bar{\mathbf{G}}\|_\infty^2} + \\
&+ d_N^{(\mathbf{g}'')} \cdot 4^{\nu-1} \cdot 2^{2\varphi_0+1+2\nu-2}\Big) \cdot 2 = \\
&= 1/3 \cdot 2^{4\nu+2\varphi_0+2\gamma_0-2\chi-1} =: V_{\bar{\mathbf{H}}}. \qquad (50)
\end{aligned}$$

Next, we apply these estimates as the initial error and variance bound into (22) and (23), respectively, together with multiplication by $2/N = 2^{-\nu+1}$. We have

$$\begin{aligned}
\|\mathsf{Err}(\mathbf{h}'')\|_\infty &\lessapprox 2^{-\nu+1} \cdot \underbrace{2(\sqrt{2}-1) \cdot E_{\bar{\mathbf{H}}}}_{\approx c_H^{(\bar{\mathbf{H}})}} \cdot \big(\sqrt{2}+1\big)^{\nu-1} \approx \\
&\approx 3\big(\sqrt{2}+1\big)^{2\nu-4} \cdot 2^{\varphi_0+\gamma_0-\chi+4}, \quad \text{and} \qquad (51) \\
\mathsf{Var}\big(\mathsf{Err}(\mathbf{h}'')\big) &\lessapprox 2^{-2\nu+2} \cdot \underbrace{1/6 \cdot 2^{(2\varphi_0+2\gamma_0+2+4\nu-4)-2\chi}}_{= d_N^{(\bar{\mathbf{H}})}} \cdot 4^{\nu-1} = \\
&= 1/3 \cdot 2^{4\nu+2\varphi_0+2\gamma_0-2\chi-3}, \qquad (52)
\end{aligned}$$

while in (52), it has shown that the term with $V_{\bar{\mathbf{H}}}$ was not the leading term, hence it was neglected. By (31) it remains to untwist and unfold, we have

$$\begin{aligned}
\|\mathsf{Err}(\mathbf{h}')\|_\infty &\lessapprox \big(1 \cdot \underbrace{3(\sqrt{2}+1)^{2\nu-4} \cdot 2^{\varphi_0+\gamma_0-\chi+4}}_{\gtrapprox \|\mathsf{Err}(\mathbf{h}'')\|_\infty} + \underbrace{2^{2\nu+\varphi_0+\gamma_0-1}}_{\geq \|\mathbf{h}''\|_\infty} \cdot 2^{-\chi-1}\big) \cdot \sqrt{2} \approx \\
&\approx 3\sqrt{2} \cdot (\sqrt{2}+1)^{2\nu-4} \cdot 2^{\varphi_0+\gamma_0-\chi+4}, \quad \text{and} \qquad (53) \\
\mathsf{Var}\big(\mathsf{Err}(\mathbf{h}')\big) &\lessapprox (1^2 \cdot \underbrace{1/3 \cdot 2^{4\nu+2\varphi_0+2\gamma_0-2\chi-3}}_{\gtrapprox \mathsf{Var}\left(\mathsf{Err}(\mathbf{h}'')\right)} + \underbrace{2^{4\nu+2\varphi_0+2\gamma_0-2}}_{\geq \|\mathbf{h}''\|_\infty^2} \cdot 1/12 \cdot 2^{-2\chi}) \cdot 2 = \\
&= 2^{4\nu+2\varphi_0+2\gamma_0-2\chi-3}. \qquad (54)
\end{aligned}$$

Since the unfolding operation does not change the error, the negacyclic results follow. □