ZAMA

Credit: Johannes Plenio

GUIDE TO FULLY HOMOMORPHIC ENCRYPTION OVER THE [DISCRETIZED] TORUS

Marc Joye

Version 2.6
October 14, 2021

Abstract

First posed as a challenge in 1978 by Rivest *et al.* [RAD78], fully homomorphic encryption—the ability to evaluate any function over encrypted data—was only solved in 2009 in a breakthrough result by Gentry [Gen09, Gen10]. After a decade of intense research, practical solutions have emerged and are being pushed for standardization.

This guide is intended to practitioners. It explains the innerworkings of TFHE [CGGI20], a torus-based fully homomorphic encryption scheme. More exactly, it describes its implementation on a discretized version of the torus. It also explains in detail the technique of the programmable bootstrapping.

Contents

1 Definitions

1.1 Torus and Torus Polynomials

The letter 'T' in TFHE [CGGI20] refers to the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. Basically, $\mathbb{T}$ is the set $[0, 1)$ of real numbers modulo 1.

Any two elements of $\mathbb{T}$ can be added modulo 1: $(\mathbb{T}, +)$ forms an abelian group. But it is important to observe that $\mathbb{T}$ is *not* a ring as the internal product $\times$ of torus elements is not defined.

> **!** Torus $\mathbb{T}$ is not a ring. If $\mathbb{T}$ were a ring, one would have $(a+b)\times c = a\times c+b\times c$ and $a\times(b+c) = a\times b+a\times c$, where $+$ and $\times$ are defined over the torus (i.e., where $+$ and $\times$ respectively stand for the addition and the multiplication over the real numbers modulo 1).

Example 1. Take for example $a = \frac{2}{5}$, $b = \frac{4}{5}$ and $c = \frac{1}{3}$. Over $\mathbb{T}$, we get $(a+b)\times c = \frac{1}{5}\times\frac{1}{3} = \frac{1}{15}$ and $a\times c+b\times c = \frac{2}{15}+\frac{4}{15} = \frac{6}{15}$, a contradiction.

The problem stems from the fact that 0 and 1 are equivalent as elements of $\mathbb{T}$. But viewing $0, 1 \in \mathbb{T}$, for any $t \in \mathbb{T}$, we have $0\times t = 0$ while $1\times t = t$.

External product The *external* product $\cdot$ between integers and torus elements is however well defined. Let $k \in \mathbb{Z}$ and $t \in \mathbb{T}$. If $k \geq 0$, we define

$$k\cdot t = t+\cdots+t \quad (k \text{ times}) \ .$$

If $k < 0$, we define $k\cdot t = (-k)\cdot(-t)$. Hence, for $0, 1 \in \mathbb{Z}$ and $t \in \mathbb{T}$, we have $0\cdot t = 0 \in \mathbb{T}$ and $1\cdot t = t \in \mathbb{T}$. Mathematically, $\mathbb{T}$ is endowed with a $\mathbb{Z}$*-module* structure. For any $k, l \in \mathbb{Z}$ and $a, b \in \mathbb{T}$, we have $(k+l)\cdot a = k\cdot a+l\cdot a$ and $k\cdot(a+b) = k\cdot a+k\cdot b$. Further, the external product is homogeneous: for any $k, l \in \mathbb{Z}$ and $t \in \mathbb{T}$, we have $k\cdot(l\cdot t) = (kl)\cdot t$.

Example 2. Take $k = 2$, $l = 3$, $a = \frac{2}{5}$ and $b = \frac{4}{5}$. We get $(k+l)\cdot a = 5\cdot\frac{2}{5} = 0$ and $k\cdot a+l\cdot a = \frac{4}{5}+\frac{1}{5} = 0$, as expected. We also get $k\cdot(a+b) = 2\cdot\frac{1}{5} = \frac{2}{5}$ and $k\cdot a+k\cdot b = \frac{4}{5}+\frac{3}{5} = \frac{2}{5}$. Finally, taking $t = a = \frac{2}{5}$, we get $k\cdot(l\cdot t) = 2\cdot\frac{1}{5} = \frac{2}{5}$ and $(kl)\cdot t = 6\cdot\frac{2}{5} = \frac{2}{5}$, as expected.

Torus polynomials We can as well define polynomials over the torus. Let $\Phi(X)$ denote the M-th cyclotomic polynomial (i.e., the unique irreducible polynomial with integer coefficients that divides

$X^M - 1$ but not $X^k - 1$ for any $k < M$) and let N denote its degree. For performance reasons, M is chosen as a power of 2, in which case we have $N = M/2$ and $\Phi(X) = X^N + 1$. Considering the polynomial rings $\mathbb{R}_N[X] := \mathbb{R}[X]/(X^N + 1)$ and $\mathbb{Z}_N[X] := \mathbb{Z}[X]/(X^N + 1)$, this defines the $\mathbb{Z}_N[X]$-module

$$\mathbb{T}_N[X] := \mathbb{R}_N[X]/\mathbb{Z}_N[X] = \mathbb{T}[X]/(X^N + 1) \ .$$

Elements of $\mathbb{T}_N[X]$ can therefore be seen as polynomials modulo $X^N + 1$ with coefficients in $\mathbb{T}$. Being a $\mathbb{Z}_N[X]$-module, elements in $\mathbb{T}_N[X]$ can be added together and *externally* multiplied by polynomials of $\mathbb{Z}_N[X]$.

Example 3. If $M = 4$ (and so $N = 2$) then $\Phi(X) = X^2 + 1$ and, in turn, $\mathbb{T}_2[X] = \mathbb{T}[X]/(X^2 + 1) = \{\mathcal{p}(X) = p_1X + p_0 \mid p_0, p_1 \in \mathbb{T}\}$. Take for example $\mathcal{p}(X) = \frac{2}{5}X + \frac{1}{3}$, $\mathcal{q}(X) = \frac{4}{5}X + \frac{1}{2}$, and $\mathcal{r}(X) = 2X + 7$. Then $(\mathcal{p} + \mathcal{q})(X) = \frac{1}{5}X + \frac{5}{6}$ and $(\mathcal{r} \cdot \mathcal{p})(X) = \frac{4}{5}X^2 + \frac{7}{15}X + \frac{1}{3} = -\frac{4}{5} + \frac{7}{15}X + \frac{1}{3} = \frac{7}{15}X + \frac{8}{15}$. Recall that polynomials are defined modulo $X^2 + 1$ (and thus $X^2 \equiv -1$).

1.2 Discretized Torus

Let B be an integer ≥ 2. Any torus element $t \in \mathbb{T}$ can be written as an infinite sequence of radix-B digits $(t_1, t_2, \ldots)_B$ with $t_j \in \{0, \ldots, B-1\}$ corresponding to the expansion $t = \sum_{j=1}^{\infty} t_j \cdot B^{-j}$. In practice, torus elements are not represented with an infinite number of digits. Elements are expanded up to some finite precision. With a fixed-point approach, a torus element t is written as

$$t = \sum_{j=1}^{w} t_j \cdot B^{-j} \quad \text{with } t_j \in \{0, \ldots, B-1\}$$

for some $w \geq 1$. This representation limits the torus to the subset $B^{-w}\mathbb{Z}/\mathbb{Z} \subset \mathbb{T}$ with representatives in $\{0, \frac{1}{B^w}, \frac{2}{B^w}, \ldots, \frac{B^w-1}{B^w}\}$.

Example 4. Suppose $B = 10$. We have $\sqrt{2} \bmod 1 = 0.4142\ldots = 4 \cdot 10^{-1} + 1 \cdot 10^{-2} + 4 \cdot 10^{-3} + 2 \cdot 10^{-4} + \cdots$. With $w = 3$ digits, $\sqrt{2} \bmod 1 \approx \frac{414}{10^3}$ is approximated by the torus element $4 \cdot 10^{-1} + 1 \cdot 10^{-2} + 4 \cdot 10^{-3}$.

Remark 1. In radix 2, letting $w = \Omega$, we have $t = \sum_{j=1}^{\Omega} t_j \cdot 2^{-j}$. Parameter Ω is called the *bit precision*. Furthermore, the leading bit (i.e., t_1) is sometimes called the sign bit. Indeed, elements of $\mathbb{T}$ are real numbers modulo 1. They can be viewed as unsigned real numbers in the range $[0, 1)$ or as signed real numbers in the range

$[-\frac{1}{2}, \frac{1}{2}) = [-\frac{1}{2}, 0) \cup [0, \frac{1}{2})$. Hence, if the leading bit is set, the corresponding torus element can be interpreted as a negative number; i.e., as a number in $[-\frac{1}{2}, 0)$.

Modern architectures typically have a bit precision of 32 or 64 bits; i.e., $\Omega = 32$ or 64. On such architectures, torus elements are restricted to elements of the form $\sum_{i=1}^{\Omega} t_i \cdot 2^{-i} \pmod 1$ with $t_i \in \{0, 1\}$. Essentially, the effect of working with a finite precision boils down to replacing $\mathbb{T}$ with the submodule

$$\mathbb{T}_q := q^{-1}\mathbb{Z}/\mathbb{Z} \subset \mathbb{T} \quad \text{where } q = 2^{\Omega} \enspace .$$

The representatives of $\mathbb{T}_q$ are the set of fractions $\{\frac{i}{q} \bmod 1 \mid i \in \mathbb{Z}\} = \{\frac{i}{q} \mid i \in \mathbb{Z}/q\mathbb{Z}\} = \{0, \frac{1}{q}, \dots, \frac{q-1}{q}\}$. Note that the discretization modulo q of the torus is indicated by the subscript q in $\mathbb{T}_q$. The submodule $\mathbb{T}_q \subset \mathbb{T}$ forms what is called a *discretized torus*.

> **!** For practical reasons, torus elements are not implemented with fractions, but rather as elements modulo q by identifying $\mathbb{T}_q = \frac{1}{q}\mathbb{Z}/\mathbb{Z}$ with $\mathbb{Z}/q\mathbb{Z}$. In more details, given two torus elements $t = \frac{a}{q}, u = \frac{b}{q} \in \mathbb{T}_q$, if $v := t + u = \frac{c}{q} \in \mathbb{T}_q$ then $c \equiv a + b \pmod q$. Likewise, for a torus element $t = \frac{a}{q} \in \mathbb{T}_q$ and a scalar $k \in \mathbb{Z}$, if $w := k \cdot t = \frac{d}{q} \in \mathbb{T}_q$ then $d \equiv k\,a \pmod q$. Computations over $\mathbb{T}_q$ can therefore be carried out entirely with arithmetic modulo q, taking only the numerator into account.

Likewise, on the discretized torus $\mathbb{T}_q$, we similarly define

$$\mathbb{T}_{N,q}[X] := \mathbb{T}_q[X]/(X^N + 1) \enspace .$$

We also define $\mathbb{Z}_{N,q}[X] := \mathbb{Z}_q[X]/(X^N+1)$ with $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$. Viewing $\frac{1}{q}$ as an element in $\mathbb{T}_{N,q}[X]$, any polynomial $\mathscr{p} \in \mathbb{T}_{N,q}[X]$ can be written as $\mathscr{p} = \overline{\mathscr{p}} \cdot \frac{1}{q}$ for some polynomial $\overline{\mathscr{p}} \in \mathbb{Z}_{N,q}[X]$. Addition and external multiplication in $\mathbb{T}_{N,q}[X]$ are respectively denoted with '+' and '·'.

1.3 Notation

It is useful to introduce some notation. If S is a set, $a \xleftarrow{\$} S$ indicates that a is sampled *uniformly* at random in S. If $\mathcal{D}$ is a probability distribution, $a \leftarrow \mathcal{D}$ indicates that a is sampled according to $\mathcal{D}$. For a real number x, $\lfloor x \rfloor$ denotes the largest integer $\leq x$, $\lceil x \rceil$ denotes the smallest integer $\geq x$, and $\lfloor x \rceil$ denotes the nearest integer to x.

Vectors are viewed as row matrices and are denoted with bold letters. Elements in $\mathbb{Z}$ or $\mathbb{T}$ (resp. in $\mathbb{Z}_q$ or $\mathbb{T}_q$) are denoted with roman letters while polynomials are denoted with calligraphic letters. $\mathbb{B}$ is the integer subset $\{0, 1\}$ and, for N a power of 2, $\mathbb{B}_N[X]$ is the subset of polynomials in $\mathbb{Z}_N[X]$ with coefficients in $\mathbb{B}$.

! Vectors are viewed as row matrices.
Further notations used throughout this document are listed in Appendix C.

Example 5. The vector $\boldsymbol{v} = (3, 4) \in \mathbb{Z}^2$ is regarded as the row matrix $\begin{pmatrix} 3 & 4 \end{pmatrix} \in \mathbb{Z}^{1\times 2}$, and if $\boldsymbol{A} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ then $\boldsymbol{vA} = \begin{pmatrix} 3 & 10 \end{pmatrix} = (3, 10)$.

1.4 Complexity Assumptions

In 2005, Regev [Reg05, Reg09] introduced the *learning with errors problem* (LWE). Generalizations and extensions to ring structures were subsequently proposed in [SSTX09, LPR10]. As originally stated in [CGGI20], the security of TFHE relies on the hardness of torus-based problems [BLP+13, CS15]: the LWE assumption and the GLWE assumption [BGV14, LS15] over the torus.

We consider below similar definitions, but over the *discretized* torus.

Definition 1 (LWE problem over the discretized torus)**.** Let $q, n \in \mathbb{N}$ and let $\boldsymbol{s} = (s_1, \ldots, s_n) \xleftarrow{\$} \mathbb{B}^n$. Let also $\hat{\chi}$ be an error distribution over $q^{-1}\mathbb{Z}$. The *learning with errors (LWE) over the discretized torus problem* is to distinguish the following distributions:

$$\mathcal{D}_0 = \left\{(\boldsymbol{a}, r) \mid \boldsymbol{a} \xleftarrow{\$} \mathbb{T}_q^n, r \xleftarrow{\$} \mathbb{T}_q\right\}$$

and

$$\mathcal{D}_1 = \left\{(\boldsymbol{a}, r) \mid \boldsymbol{a} = (a_1, \ldots, a_n) \xleftarrow{\$} \mathbb{T}_q^n, r = \textstyle\sum_{j=1}^n s_j \cdot a_j + e, e \leftarrow \hat{\chi}\right\} \ .$$

Definition 2 (GLWE problem over the discretized torus)**.** Let $N, q, k \in \mathbb{N}$ with N a power of 2 and let $\boldsymbol{\mathfrak{s}} = (\mathfrak{s}_1, \ldots, \mathfrak{s}_k) \xleftarrow{\$} \mathbb{B}_N[X]^k$. Let also $\hat{\chi}$ be an error distribution over $q^{-1}\mathbb{Z}_N[X]$; namely, over polynomials of $q^{-1}\mathbb{Z}_N[X]$ with coefficients drawn according to $\hat{\chi}$. The *general learning with errors (GLWE) over the discretized torus problem* is to distinguish the following distributions:

$$\mathcal{D}_0 = \left\{(\boldsymbol{\mathcal{a}}, \mathcal{r}) \mid \boldsymbol{\mathcal{a}} \xleftarrow{\$} \mathbb{T}_{N,q}[X]^k, \mathcal{r} \xleftarrow{\$} \mathbb{T}_{N,q}[X]\right\}$$

and

$$\mathcal{D}_1 = \Big\{(\boldsymbol{a}, b) \mid \boldsymbol{a} = (a_1, \ldots, a_k) \xleftarrow{\$} \mathbb{T}_{N,q}[X]^k, \\ b = \textstyle\sum_{j=1}^{k} s_j \cdot a_j + e, e \leftarrow \hat{\chi}\Big\} \ .$$

The *decisional LWE assumption* (resp. the *decisional GLWE assumption*) asserts that solving the LWE problem (resp. GLWE problem) is infeasible for some security parameter λ, where $q := q(\lambda)$, $n := n(\lambda)$, and $\hat{\chi} := \hat{\chi}(\lambda)$ (resp. $N := N(\lambda)$, $q := q(\lambda)$, $k = k(\lambda)$, and $\hat{\chi} := \hat{\chi}(\lambda)$).

> ! Interestingly, identifying $\mathbb{T}_q$ with $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ (resp. $\mathbb{T}_{N,q}[X]$ with $\mathbb{Z}_{N,q}[X]$), it turns out that the decisional LWE (resp. GLWE) assumption over the *discretized* torus is equivalent to the standard decisional LWE (resp. GLWE) assumption.

Cryptographic parameters Table 1 lists typical cryptographic parameters to be used for secure instances for the LWE and GLWE assumptions. The error distribution $\hat{\chi}$ is induced by the normal distribution $\mathcal{N}(0, \sigma^2)$, centered in 0 and with variance σ^2 (σ represents the standard deviation).

Table 1: Typical parameter sets for LWE and GLWE

LWE	$n = 630$	$\mathcal{N}(0, \sigma^2)$ with $\sigma = 2^{-15}$
GLWE	$(N, k) = (1024, 1)$	$\mathcal{N}(0, \sigma^2)$ with $\sigma = 2^{-25}$

We recommend the reader to check the `lwe-estimator` script[1] to find concrete parameters for a given security level [APS15].

For an equivalent security level, a smaller value for parameter n (resp. for (N, k)) should be compensated with a larger value for σ (i.e., less concentrated noise).

[1] https://bitbucket.org/malb/lwe-estimator/

2 TLWE Encryption

2.1 Description

Intuition The LWE assumption over the discretized torus essentially says that a torus element $r \in \mathbb{T}_q$ constructed as $r = \sum_{j=1}^{n} s_j \cdot a_j + e$ cannot be distinguished from a *random* torus element $r \in \mathbb{T}_q$, even if the torus vector $(a_1, \dots, a_n) \in \mathbb{T}_q^n$ is known. Torus element $r = \sum_{j=1}^{n} s_j \cdot a_j + e$ can therefore be used as a kind of one-time pad to conceal a "plaintext message" $\mu \in \mathbb{T}_q$ so as to form a ciphertext $\boldsymbol{c} = (a_1, \dots, a_n, r + \mu) \in \mathbb{T}_q^{n+1}$, where $\boldsymbol{s} = (s_1, \dots, s_n) \in \mathbb{B}^n$ plays the role of the private encryption key. The reason why secret key $\boldsymbol{s}$ is chosen as a vector of bits is to have an efficient implementation for the bootstrapping; see Section 5.

Only part of the torus is used to input plaintext messages. The plaintext space is chosen as a proper additive subgroup $\mathcal{P} \subset \mathbb{T}_q$; specifically,

$$\mathcal{P} = \left\{0, \tfrac{1}{p}, \dots, \tfrac{p-1}{p}\right\}$$

for some integer p dividing q, $p \geq 2$. This allows for unique decryption, provided that the noise present in the ciphertext is not too large. In particular, with the above choice for $\mathcal{P}$, if $\boldsymbol{c} = (a_1, \dots, a_n, b)$ with $b = \sum_{j=1}^{n} s_j \cdot a_j + \mu + e$ is an encryption of a plaintext $\mu \in \mathcal{P}$, plaintext μ can be recovered in two steps as:

- compute $\mu^* = b - \sum_{j=1}^{n} s_j \cdot a_j \quad (\text{in } \mathbb{T}_q)$;
- return the closest plaintext in $\mathcal{P}$.

TLWE encryption scheme Given the discretized torus $\mathbb{T}_q$, the plaintext space is set as an additive subgroup of $\mathbb{T}_q$; i.e., $\mathcal{P} := p^{-1}\mathbb{Z}/\mathbb{Z} = \mathbb{T}_p \subset \mathbb{T}_q$ for some p dividing q. The discretized distribution $\hat{\chi}$ over $q^{-1}\mathbb{Z}$ is induced by an error distribution χ over $\mathbb{R}$: a noise error $e \leftarrow \hat{\chi}$ is defined as $e = \frac{\bar{e}}{q}$ with $\bar{e} = \mathrm{round}(q e_0) \in \mathbb{Z}$ for some $e_0 \leftarrow \chi$. The *mask* $(a_1, \dots, a_n) \in \mathbb{T}_q^n$ of a ciphertext is formed by drawing $\bar{a}_j \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and letting $a_j = \frac{\bar{a}_j}{q}$, for $1 \leq j \leq n$; the corresponding *body* b is given by $b = \sum_{j=1}^{n} s_j \cdot a_j + \mu + e$ where $e \leftarrow \hat{\chi}$. The TLWE encryption of $\mu \in \mathcal{P}$ is the vector $(a_1, \dots, a_n, b)$.

A private-key encryption scheme is symmetric: the same key is used for both encryption and decryption. Public-key variants are presented in Appendix A.

Formally, we get the following *private-key* encryption scheme.

TLWE Encryption

KeyGen(1^λ) On input security parameter λ, define a positive integer n, select positive integers p and q such that $p \mid q$, and define a discretized error distribution $\hat{\chi}$ over $q^{-1}\mathbb{Z}$ induced by a normal distribution $\chi = \mathcal{N}(0, \sigma^2)$ over $\mathbb{R}$. Sample uniformly at random a vector $\boldsymbol{s} = (s_1, \dots, s_n) \xleftarrow{\$} \mathbb{B}^n$. The plaintext space is $\mathcal{P} = \mathbb{T}_p \subset \mathbb{T}_q$. The public parameters are $pp = \{n, \sigma, p, q\}$ and the private key is $sk = \boldsymbol{s}$.

Encrypt$_{sk}(\mu)$ The encryption of $\mu \in \mathcal{P}$ is given by

$$\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) = (a_1, \dots, a_n, b) \in \mathbb{T}_q^{n+1}$$

with

$$\begin{cases} \mu^* = \mu + e \\ b = \sum_{j=1}^{n} s_j \cdot a_j + \mu^* \end{cases}$$

for a random vector $(a_1, \dots, a_n) \xleftarrow{\$} \mathbb{T}_q^n$ and a “small” noise $e \leftarrow \hat{\chi}$.

Decrypt$_{sk}(\boldsymbol{c})$ To decrypt $\boldsymbol{c} = (a_1, \dots, a_n, b)$, use private key $\boldsymbol{s} = (s_1, \dots, s_n)$, compute (in $\mathbb{T}_q$)

$$\mu^* = b - \sum_{j=1}^{n} s_j \cdot a_j$$

and return

$$\mu = \frac{\lfloor p\,\mu^* \rceil \bmod p}{p},$$

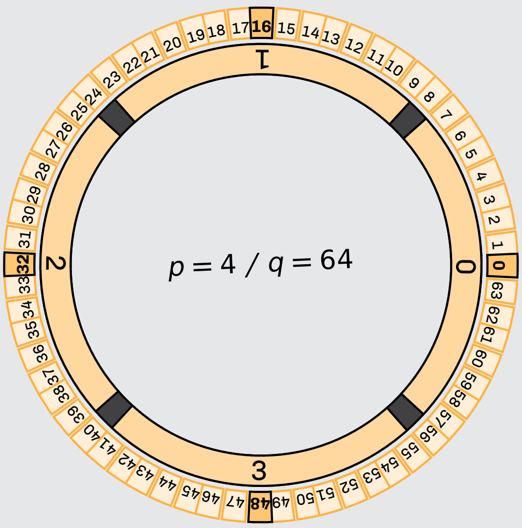
that is, the closest plaintext $\mu \in \mathcal{P}$, as the decryption of $\boldsymbol{c}$.

To ease the notation, for an integer k and a torus element $t \in \mathbb{T}_q \subset \mathbb{T}$, $\lfloor k\,t \rceil$ denotes the nearest integer to the product of k by t viewed as a *real number*. Rigorously, one should write $\lfloor k\,\mathrm{lift}(t) \rceil$ where function lift lifts elements of $\mathbb{T}$ to $\mathbb{R}$ (i.e., views elements of $\mathbb{T}$ as elements in $\mathbb{R}$).

It is easily verified that decryption succeeds in recovering plaintext μ if the noise error e satisfies $|e| < \frac{1}{2p}$.

Proof. For plaintext $\mu \in \mathcal{P} = \{0, \frac{1}{p}, \dots, \frac{p-1}{p}\}$, we let $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) = (a_1, \dots, a_n, b)$ where $(a_1, \dots, a_n) \xleftarrow{\$} \mathbb{T}_q^n$ and $b = \sum_{j=1}^{n} s_j \cdot a_j + \mu + e$ with $e \leftarrow \hat{\chi}$. Since $\mu \in \mathcal{P}$, there exists a unique integer $m \in [0, p)$ such that $\mu = \frac{m}{p}$. An application of $\mathrm{Decrypt}_{sk}(\boldsymbol{c})$ outputs $\frac{\lfloor p\mu^* \rceil \bmod p}{p}$ with $\mu^* := (\mu + e) \in \mathbb{T}_q \subset \mathbb{T}$. We have $\lfloor p\mu^* \rceil = \lfloor p((\mu + e) \bmod 1) \rceil = \lfloor p(\mu + e + \delta) \rceil = \lfloor p(\mu + e) \rceil + \delta p$ for some $\delta \in \mathbb{Z}$. We also have $\lfloor p(\mu + e) \rceil = \left\lfloor p(\frac{m}{p} + e) \right\rceil = \lfloor m + pe \rceil = m + \lfloor pe \rceil = m$ if we assume that $|e| < 1/(2p)$. In this case, it thus follows that $\lfloor p\mu^* \rceil \bmod p = \lfloor p(\mu + e) \rceil \bmod p = m$ and so $\frac{\lfloor p\mu^* \rceil \bmod p}{p} = \frac{m}{p} = \mu$. ☐

Example 6. Suppose $p = 4$ and $q = 64$ $(= 2^6)$. The plaintext space is $\mathcal{P} = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}\}$.



The outer wheel depicts the discretized torus $\mathbb{T}_q = \{0, \frac{1}{64}, \dots, \frac{63}{64}\}$. It can be observed that if the noise error e satisfies $|e| < \frac{1}{2p} = \frac{1}{8}$, that is, $e \in \{-\frac{7}{64}, \dots, \frac{7}{64}\}$, then any noisy value $\mu^* := \mu + e$ corresponds unequivocally to a plaintext $\mu \in \mathcal{P} = \{0, \frac{16}{64}, \frac{32}{64}, \frac{48}{64}\}$. The closest plaintext to $\mu^* \in \{\frac{57}{64}, \dots, \frac{63}{64}, \frac{0}{64}, \dots, \frac{7}{64}\}$ is $\mu = 0$ (note that $\frac{57}{64}$ and $-\frac{7}{64}$ are equivalent as elements of $\mathbb{T}_q$); the closest plaintext to $\mu^* \in \{\frac{9}{64}, \dots, \frac{23}{64}\}$ is $\mu = \frac{16}{64} = \frac{1}{4}$;the closest plaintext to $\mu^* \in \{\frac{25}{64}, \dots, \frac{39}{64}\}$ is $\mu = \frac{32}{64} = \frac{1}{2}$; and the closest plaintext to $\mu^* \in \{\frac{41}{64}, \dots, \frac{55}{64}\}$ is $\mu = \frac{48}{64} = \frac{3}{4}$.

2.2 Encoding/Decoding

The encryption algorithm takes (discretized) torus elements—or, more exactly, elements in $\mathcal{P}$—on input. Encoding and decoding aim at supporting further input formats.

Let $\mathcal{M}$ be an arbitrary finite message space of cardinality $\#\mathcal{M} = p$ with $p = 2^{\nu}$. The plaintext space is $\mathcal{P} = \mathbb{T}_p \subset \mathbb{T}_q$ with $q = 2^{\Omega}$. The encoding function, $\mathrm{Encode}\colon \mathcal{M} \to \mathcal{P}$, maps a message $m \in \mathcal{M}$ to an element $\mu \in \mathcal{P}$; the encoding is applied before encryption. The decoding function, $\mathrm{Decode}\colon \mathcal{P} \to \mathcal{M}$, is applied after decryption.

We discuss below the cases of message spaces consisting of bits, of integers modulo p (with p dividing q), and of fixed-precision torus elements.

Bits The message space is $\mathcal{M} = \{0, 1\}$.

For a bit $b \in \{0, 1\}$, we define $\mathrm{Encode}(b) = b/2$. Hence, bit 0 is encoded as torus element $0 = \frac{0}{q} \in \mathbb{T}_q$ and bit 1 as torus element $\frac{1}{2} = \frac{q/2}{q} \in \mathbb{T}_q$. The reverse operation is defined as $\mathrm{Decode}(\mu) = \lfloor 2\,\mu \rceil \bmod 2$, and thus if $\mu \in \{0, \frac{1}{2}\}$ then $\mathrm{Decode}(\mu) \in \{0, 1\}$.

Integers modulo *p* This generalizes the previous case (bits can be seen as integers modulo $p = 2$). We have $\mathcal{M} = \{i \bmod p \mid i \in \mathbb{Z}\} = \mathbb{Z}/p\mathbb{Z}$.

Let $\Delta = q/p \in \mathbb{Z}$. The encoding and decoding are then respectively given by

$$\mathrm{Encode}(i) = \frac{i \bmod p}{p} \qquad \left(= \tfrac{(i \bmod p)\,\Delta}{q}\right)$$

and

$$\mathrm{Decode}(\mu) = \lfloor p\,\mu \rceil \bmod p \ .$$

Fixed-precision torus elements Let $p \geq 2$ with $p \mid q$. This case is similar to the case of integers modulo p and considers torus elements of the form $t = \frac{i}{p}$ with $i \in \mathbb{Z}/p\mathbb{Z}$. These elements form a subset of fixed-precision torus elements. For $x \in \mathbb{T}_p = p^{-1}\mathbb{Z}/\mathbb{Z}$ and $\mu \in \mathbb{T}_q$, we define

$$\mathrm{Encode}(x) = x$$

and

$$\mathrm{Decode}(\mu) = \frac{\lfloor p\,\mu \rceil \bmod p}{p} \ .$$

Remark 2. The second encoding obviously applies to unsigned integers smaller than p; i.e., to integers in $\{0, \ldots, p-1\}$. It may also apply to signed integers. In the latter case, the "mod p" returns the signed representative in $\left\{-\frac{p}{2}, \ldots, \frac{p}{2} - 1\right\}$.

Example 7. Suppose $p = 4$ and $q = 64$. If $\mu = \frac{48}{64}$ then $\mathrm{Decode}(\mu) = \lfloor p\,\mu \rceil \bmod p \equiv 3 \equiv -1 \pmod 4$, which represents the unsigned integer 3 or the signed integer -1.

Likewise, the third encoding applies to unsigned (fixed-precision) numbers in $\mathbb{T}_p \cap [0, 1)$, or to signed (fixed-precision) numbers in $\mathbb{T}_p \cap [-\frac{1}{2}, \frac{1}{2})$.

2.3 Implementation Notes

Batching ciphertexts When a set of m plaintexts (torus elements) need to be encrypted, randomness can be re-used if they are all encrypted under *different* keys. Specifically, for $\mu_1, \ldots, \mu_m \in \mathcal{P}$, we set $\boldsymbol{C} = (a_1, \ldots, a_n, b_1, \ldots, b_m) \in \mathbb{T}_q^{n+m}$ as their encryption with $b_i = \sum_{j=1}^{n} s_{i,j} \cdot a_j + \mu_i + e_i$ for $1 \le i \le m$, where $(a_1, \ldots, a_n) \xleftarrow{\$} \mathbb{T}_q^n$, $\boldsymbol{s_i} = (s_{i,1}, \ldots, s_{i,n}) \xleftarrow{\$} \mathbb{B}^n$ and noise error e_i.

The security of this variant follows from [BBS03]. Since the randomness is given explicitly in a TLWE ciphertext (namely, the a_j's), it is readily verified that the "reproducibility" criterion [BBKS07, Definition 9.3] is satisfied.

Ciphertext compression TLWE ciphertexts are torus vectors with $n + 1$ components. With the parameter set of Table 1, if we suppose that torus elements are represented with 64 bits, a TLWE ciphertext typically requires $631 \times 64 = 40384$ bits (or about 5 kilobytes) for its representation.

Instead of representing a ciphertext $\boldsymbol{c}$ as $\boldsymbol{c} = (a_1, \ldots, a_n, b)$, a much more compact way is to define $\boldsymbol{c}$ as $\boldsymbol{c} = (\theta, b)$ where $\theta \xleftarrow{\$} \{0, 1\}^\lambda$ is a random λ-bit string for security parameter λ. The value of θ is used as a seed to a cryptographically secure pseudo-random number generator (PRNG) to derive the random vector $(a_1, \ldots, a_n)$:

$$(a_1, \ldots, a_n) \leftarrow \mathrm{PRNG}(\theta) \enspace .$$

With the above parameter set (which corresponds to a desired bit-security of 128 bits), the same ciphertext only needs $128 + 64 = 192$ bits for its representation.

Key storage The same trick applies to private key $\boldsymbol{s}$. Instead of plainly storing $\boldsymbol{s}$ as a n-bit string, we can store it as a λ-bit random seed that is used to generate $\boldsymbol{s}$ through a cryptographic pseudo-random number generator.

3 TGLWE Encryption

3.1 Description

TLWE encryption readily extends to torus polynomials in $\mathbb{T}_{N,q}[X]$. Operations on the torus $\mathbb{T}_q$ are simply replaced with operations on polynomials modulo X^N+1 (and modulo q). Given two polynomials $\mathfrak{a}, \mathfrak{b} \in \mathbb{T}_{N,q}[X]$, $\mathfrak{a}+\mathfrak{b}$ refers to the addition of $\mathfrak{a}$ and $\mathfrak{b}$ modulo (X^N+1, q) and, for $\mathfrak{a} \in \mathbb{Z}_{N,q}[X]$ and $\mathfrak{b} \in \mathbb{T}_{N,q}[X]$, $\mathfrak{a} \cdot \mathfrak{b}$ refers to the external product of $\mathfrak{a}$ and $\mathfrak{b}$ modulo (X^N+1, q)—remember that the internal product is not defined.

The plaintext space is the subset of polynomials

$$\mathcal{P}_N[X] := \mathcal{P}[X]/(X^N+1) = \mathbb{T}_{N,p}[X] \subset \mathbb{T}_{N,q}[X]$$

with $\mathcal{P} = \mathbb{T}_p = p^{-1}\mathbb{Z}/\mathbb{Z}$ for some p dividing q. Note that this latter condition imposes that $\mathcal{P}_N[X]$ forms an additive subgroup of $\mathbb{T}_{N,q}[X]$.

This leads to the TGLWE private-key encryption scheme.

TGLWE Encryption

KeyGen(1^λ) On input security parameter λ, define a pair of integers (N, k) with N a power of 2 and $k \geq 1$. Select positive integers p and q such that $p \mid q$. Define also a discretized error distribution $\hat{\chi}$ over $q^{-1}\mathbb{Z}_N[X]$ induced by a normal distribution $\chi = \mathcal{N}(0, \sigma^2)$ over $\mathbb{R}_N[X]$. Sample uniformly at random a vector $\mathfrak{s} = (\mathfrak{s}_1, \ldots, \mathfrak{s}_k) \xleftarrow{\$} \mathbb{B}_N[X]^k$. The plaintext space is $\mathcal{P}_N[X] = \mathbb{T}_{N,p}[X] \subset \mathbb{T}_{N,q}[X]$. The public parameters are $pp = \{k, N, \sigma, p, q\}$ and the private key is $sk = \mathfrak{s}$.

Encrypt$_{sk}(\mu)$ The encryption of $\mu \in \mathcal{P}_N[X]$ is given by

$$\mathfrak{c} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu) = (\mathfrak{a}_1, \ldots, \mathfrak{a}_k, \mathfrak{b}) \in \mathbb{T}_{N,q}[X]^{k+1}$$

with

$$\begin{cases} \mu^* = \mu + \mathfrak{e} \\ \mathfrak{b} = \sum_{j=1}^{k} \mathfrak{s}_j \cdot \mathfrak{a}_j + \mu^* \end{cases}$$

for a random vector $(\mathfrak{a}_1, \ldots, \mathfrak{a}_k) \xleftarrow{\$} \mathbb{T}_{N,q}[X]^k$ and a "small" noise $\mathfrak{e} \leftarrow \hat{\chi}$.

Decrypt$_{sk}(\boldsymbol{\mathscr{c}})$ To decrypt $\boldsymbol{\mathscr{c}} = (\mathscr{a}_1, \ldots, \mathscr{a}_k, \mathscr{b})$, use private key $\boldsymbol{\mathscr{s}} = (\mathscr{s}_1, \ldots, \mathscr{s}_k)$, compute (in $\mathbb{T}_{N,q}[X]$)

$$\mu^* = \mathscr{b} - \sum_{j=1}^{k} \mathscr{s}_j \cdot \mathscr{a}_j$$

and return the closest plaintext $\mu \in \mathcal{P}_N[X]$ as the decryption of $\boldsymbol{\mathscr{c}}$.

Remark 3. Since $\mathbb{T}_{N,q}[X] = \mathbb{T}_q$ when $N = 1$, it turns out that the TLWE encryption (Section 2.1) can be seen as a special instantiation of the TGLWE encryption with parameters $(k, N) = (n, 1)$.

At this point, the reader may wonder why there are two versions for the encryption: one over $\mathbb{T}_q$ and one over $\mathbb{T}_{N,q}[X]$. For the encryption of a single torus element $\mu \in \mathcal{P}$, TLWE should be preferred to TGLWE because the resulting ciphertext is shorter. For the encryption of multiple torus elements, TGLWE can be a better option; see next section. But the main reason of having two different schemes is for the implementation of the programmable bootstrapping where both TLWE and TGLWE are needed; see Section 5.

3.2 Encoding/Decoding

The TGLWE encryption scheme supports the encryption of an arbitrary polynomial $\mu \in \mathcal{P}_N[X]$. In many applications, μ is restricted to a polynomial of degree 0 and can therefore be seen as an element in $\mathcal{P}$. In this case, the encoding and decoding functions presented in Section 2.2 equally apply.

When up to N torus elements $\mu_0, \ldots, \mu_{N-1} \in \mathcal{P}$ need to be encrypted, they can each be represented as a coefficient of polynomial $\mu(X) = \mu_0 + \mu_1 X + \cdots + \mu_{N-1} X^{N-1} \in \mathcal{P}_N[X]$. Such an optimization is known as *coefficient packing*.

3.3 Implementation Notes

The (external) product of two polynomials is a demanding operation. The special form of cyclotomic polynomial $\Phi(X) = X^N + 1$ makes however computations slightly easier.

Example 8. Let $N = 4$ and thus $\Phi(X) = X^4 + 1$. Let also $q = 8$. Suppose we want to externally multiply $\mathscr{p} \in \mathbb{Z}_{N,q}[X]$ and $\mathscr{q} \in \mathbb{T}_{N,q}[X]$ with $\mathscr{p}(X) = 2X^3 + 5X + 3$

and $\mathcal{q}(X) = \frac{1}{4}X^3 + \frac{1}{8}$. Then the product $\mathcal{r} := \mathcal{p} \cdot \mathcal{q} \in \mathbb{T}_{N,q}[X]$ verifies

$$\begin{aligned}
\mathcal{p}(X) \cdot \mathcal{q}(X) &\equiv (2X^3 + 5X + 3) \cdot (\tfrac{1}{4}X^3 + \tfrac{1}{8}) \\
&\equiv \tfrac{1}{2}X^6 + \tfrac{1}{4}X^3 + \tfrac{5}{4}X^4 + \tfrac{5}{8}X + \tfrac{3}{4}X^3 + \tfrac{3}{8} \\
&\equiv \tfrac{1}{2}X^6 + \tfrac{1}{4}X^4 + X^3 + \tfrac{5}{8}X + \tfrac{3}{8} \\
&\equiv (X^4 + 1) \cdot (\tfrac{1}{2}X^2 + \tfrac{1}{4}) + X^3 + \tfrac{5}{8}X + \tfrac{3}{8} - \tfrac{1}{2}X^2 - \tfrac{1}{4} \\
&\equiv X^3 + \tfrac{1}{2}X^2 + \tfrac{5}{8}X + \tfrac{1}{8} \pmod{(X^4 + 1, 8)} \ .
\end{aligned}$$

Hence, $\mathcal{r}(X) = X^3 + \frac{1}{2}X^2 + \frac{5}{8}X + \frac{1}{8} \in \mathbb{T}_{N,q}[X]$.

In the general case, for $\Phi(X) = X^N + 1$, let $\mathcal{p} \in \mathbb{Z}_{N,q}[X]$ and $\mathcal{q} \in \mathbb{T}_{N,q}[X]$ given by $\mathcal{p}(X) = p_0 + p_1X + \cdots + p_{N-1}X^{N-1}$ and $\mathcal{q}(X) = q_0 + q_1X + \cdots + q_{N-1}X^{N-1}$. Using the relation $X^{N+i} \equiv -X^i \pmod{X^N + 1}$, their product satisfies

$$\begin{aligned}
\mathcal{p}(X) \cdot \mathcal{q}(X) = (p_0 + p_1X + \cdots + p_{N-1}X^{N-1}) \cdot& \\
(q_0 + q_1X + \cdots + q_{N-1}X^{N-1})& \\
= p_0 \cdot q_0 - p_1 \cdot q_{N-1} - \cdots - p_{N-1} \cdot q_1& \\
+ (p_0 \cdot q_1 + p_1 \cdot q_0 - \cdots - p_{N-1} \cdot q_2)X& \\
+ \dots& \\
+ (p_0 \cdot q_{N-1} + p_1 \cdot q_{N-2} + \cdots + p_{N-1} \cdot q_0)X^{N-1}& \ .
\end{aligned}$$

This requires N^2 external torus products for evaluating $p_i \cdot q_j$ with $0 \le i, j \le N-1$. For large values of N, an alternative way is to rely on the fast Fourier transform (FFT) [vzGG13, Chapter 8].

When $\mathcal{p}(X)$ is the monomial X^j for some $j \in \{0, \dots, N-1\}$, the previous product formula simplifies into

$$X^j \cdot \mathcal{q}(X) = \begin{cases} q_0 + q_1X + q_2X^2 + \cdots + q_{N-2}X^{N-2} + q_{N-1}X^{N-1} & j = 0 \\ -q_{N-1} + q_0X + q_1X^2 + \cdots + q_{N-3}X^{N-2} + q_{N-2}X^{N-1} & j = 1 \\ \vdots & \vdots \\ -q_1 - q_2X - q_3X^2 - \cdots - q_{N-1}X^{N-2} + q_0X^{N-1} & j = N-1 \end{cases}$$

or, more concisely,

$$X^j \cdot \mathcal{q}(X) = \sum_{i=0}^{j-1} -q_{i+N-j}X^i + \sum_{i=j}^{N-1} q_{i-j}X^i$$

and $X^{N+j} \cdot \mathcal{q}(X) = -X^j \cdot \mathcal{q}(X)$. This relation is known as the *negacyclic property*.

Example 9. To better exhibit the negacyclic property, we represent polynomials by their vectors of coefficients. Take $N = 4$ and consider the polynomial $q(X) = q_0 + q_1 X + q_2 X^2 + q_3 X^3$. Then

$$
\begin{aligned}
q(X) &= [q_0, q_1, q_2, q_3] & X^4 q(X) &= [-q_0, -q_1, -q_2, -q_3]\\
X \cdot q(X) &= [-q_3, q_0, q_1, q_2] & X^5 \cdot q(X) &= [q_3, -q_0, -q_1, -q_2]\\
X^2 \cdot q(X) &= [-q_2, -q_3, q_0, q_1] & X^6 \cdot q(X) &= [q_2, q_3, -q_0, -q_1]\\
X^3 \cdot q(X) &= [-q_1, -q_2, -q_3, q_0] & X^7 \cdot q(X) &= [q_1, q_2, q_3, -q_0]
\end{aligned}
$$

$X^8 \cdot q(X) = [q_0, q_1, q_2, q_3] = q(X)$, and so on. At each multiplication by X, it turns out that the polynomial coefficients are circularly shifted one position to the right and the entering coefficient is negated.

4 Working over Encrypted Data

Clearly, TLWE encryption and TGLWE encryption are additively homomorphic.The approach of Gentry–Sahai–Waters [GSW13] using matrix product is employed to turn these encryption schemes into *somewhat homomorphic encryption schemes*—that is, schemes supporting a limited number of multiplications.

4.1 TLWE Ciphertexts

4.1.1 Addition of ciphertexts

Let $\boldsymbol{c_1} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu_1)$ and $\boldsymbol{c_2} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu_2)$ (in $\mathbb{T}_q^{n+1}$) be respective TLWE encryptions of μ_1 and μ_2 (in $\mathcal{P}$):

$$\boldsymbol{c_1} = (a_1, \dots, a_n, b) \quad \text{and} \quad \boldsymbol{c_2} = (a'_1, \dots, a'_n, b')$$

with $(a_1, \dots, a_n) \xleftarrow{\$} \mathbb{T}_q^n$ and $b = \sum_{j=1}^n s_j \cdot a_j + \mu_1 + e_1$, $(a'_1, \dots, a'_n) \xleftarrow{\$} \mathbb{T}_q^n$ and $b' = \sum_{j=1}^n s_j \cdot a'_j + \mu_2 + e_2$, and e_1, e_2 “small”. Then $\boldsymbol{c_3} := \boldsymbol{c_1} + \boldsymbol{c_2}$ (in $\mathbb{T}_q^{n+1}$) is a valid encryption of $\mu_3 := \mu_1 + \mu_2$ (in $\mathcal{P}$); i.e.,

$$\boldsymbol{c_3} = (a''_1, \dots, a''_n, b'') \quad \text{with} \quad \begin{cases} a''_j = a_j + a'_j & (1 \le j \le n) \\ b'' = b + b' \end{cases}$$

provided that the additive noise $e_3 := e_1 + e_2$ keeps “small”.

> **!** Addition of ciphertexts explains why $\mathcal{P}$ was chosen as an additive *subgroup* of $\mathbb{T}_q$ in the definition of TLWE encryption. Doing so implies that if $\mu_1, \mu_2 \in \mathcal{P}$ then so does $\mu_3 = \mu_1 + \mu_2$.

4.1.2 Multiplication by a known constant

Multiplying by a constant can be obtained as a series of additions. As a result, given the TLWE ciphertext $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu)$ with $\mu \in \mathcal{P}$, the TLWE encryption of $K \cdot \mu$ for some known (small) integer $K \neq 0$ can be obtained as

$$K \cdot \boldsymbol{c} = \underbrace{\boldsymbol{c} + \cdots + \boldsymbol{c}}_{K \text{ times}}$$

if $K > 0$, and $K \cdot \boldsymbol{c} = (-K) \cdot (-\boldsymbol{c})$ if $K < 0$. This boils down to multiplying every vector component of $\boldsymbol{c}$ by K; namely, if $\boldsymbol{c} = (a_1, \ldots, a_n, b) \in \mathbb{T}_q^{n+1}$ then

$$K \cdot \boldsymbol{c} = (K \cdot a_1, \ldots, K \cdot a_n, K \cdot b) \enspace .$$

Again, $K \cdot \boldsymbol{c}$ (in $\mathbb{T}_q^{n+1}$) is a valid encryption of $K \cdot \mu$ (in $\mathcal{P}$), provided that the resulting noise keeps "small".

4.1.3 Multiplication of ciphertexts

The main challenge in working over encrypted data resides in multiplying ciphertexts. In order to make the Gentry–Sahai–Waters' approach work, ciphertexts in TLWE encryption need to be expressed as matrices.

Gadget matrix *Flattening* is a method that modifies vectors without affecting dot products [BGV14, Bra12]. As will become apparent, this technique helps controlling the noise.

We present the "gadget decomposition" technique over the discretized torus $\mathbb{T}_q = q^{-1}\mathbb{Z}/\mathbb{Z}$ for a general integer q (i.e., not necessarily a power of 2). For a radix B and some integer $\ell \geq 1$ such that $B^\ell \mid q$, we consider the so-called *gadget matrix* $\boldsymbol{G} \in \mathbb{T}_q^{(n+1)\times(n+1)\ell}$ given by

$$\boldsymbol{G}^\intercal = \boldsymbol{I}_{n+1} \otimes \boldsymbol{g}^\intercal = \mathrm{diag}(\underbrace{\boldsymbol{g}^\intercal, \ldots, \boldsymbol{g}^\intercal}_{n+1}) = \begin{pmatrix} 1/B & & & \\ \vdots & & & \\ 1/B^\ell & & & \\ & 1/B & & \\ & \vdots & & \\ & 1/B^\ell & & \\ & & \ddots & \\ & & & 1/B \\ & & & \vdots \\ & & & 1/B^\ell \end{pmatrix}$$

with $\boldsymbol{g} = (1/B, \ldots, 1/B^\ell) \in \mathbb{T}_q^{\,\ell}$, so that for an input vector $\boldsymbol{u} \in \mathbb{Z}^{(n+1)\ell}$ the product $\boldsymbol{u} \cdot \boldsymbol{G}^\intercal$ yields a vector in $\mathbb{T}_q^{n+1}$. We also consider the associated inverse transformation $G^{-1}\colon \mathbb{T}_q^{n+1} \to \mathbb{Z}^{(n+1)\ell}$ such that for any vector $\boldsymbol{v} \in \mathbb{T}_q^{n+1}$, we have

$$G^{-1}(\boldsymbol{v}) \cdot \boldsymbol{G}^\intercal \approx \boldsymbol{v} \quad \text{and} \quad G^{-1}(\boldsymbol{v}) \text{ is ``small''} \enspace .$$

This inverse transformation replaces each entry of a vector by its signed radix-B expansion. Explicitly, if $\boldsymbol{v} = (v_1, \dots, v_{n+1}) \in \mathbb{T}_q^{n+1}$ with $v_i \in [-\frac{1}{2}, \frac{1}{2})$, we set $\overline{v_i} = \lfloor B^\ell v_i \rceil$ and write

$$\overline{v_i} \equiv \sum_{j=1}^{\ell} u_{i,j}\, B^{\ell-j} \pmod{B^\ell} \quad \text{where } u_{i,j} \in [-\lfloor B/2 \rfloor, \lceil B/2 \rceil) \enspace .$$

We define $g^{-1}(v_i) := \bigl(u_{i,1}, \dots, u_{i,\ell}\bigr) \in \mathbb{Z}^\ell$. Then

$$\begin{aligned} G^{-1}(\boldsymbol{v}) &:= \bigl(g^{-1}(v_1), g^{-1}(v_2), \dots, g^{-1}(v_{n+1})\bigr) \\ &= \bigl(u_{1,1}, \dots, u_{1,\ell}, \dots, u_{2,1}, \dots, u_{2,\ell}, \dots, \\ &\qquad\qquad u_{n+1,1}, \dots, u_{n+1,\ell}\bigr) \in \mathbb{Z}^{(n+1)\ell} \enspace . \end{aligned}$$

Note that when $B^\ell = q$, all the components $v_i \in [-\frac{1}{2}, \frac{1}{2})$ of $\boldsymbol{v}$ satisfy $\overline{v_i} = B^\ell v_i$. It then follows that, over $\mathbb{T}_q$, $G^{-1}(\boldsymbol{v}) \cdot \boldsymbol{G}^\intercal = \boldsymbol{v}$ holds exactly.

Example 10. Take $n = 1$, $\ell = 2$, $B = 4$, and $q = 64$ (and so $\mathbb{T}_q = \frac{1}{64}\mathbb{Z}/\mathbb{Z}$). Hence,

$$\boldsymbol{G}^\intercal = \begin{pmatrix} 1/4 & 0 \\ 1/16 & 0 \\ 0 & 1/4 \\ 0 & 1/16 \end{pmatrix} \in \mathbb{T}_q^{4\times 2} \enspace .$$

Suppose that $\boldsymbol{v} = (\frac{41}{64}, \frac{26}{64}) \equiv (-\frac{23}{64}, \frac{26}{64}) \pmod 1$. We get $\overline{v_1} = \lfloor 4^2\,(-\frac{23}{64}) \rceil = -6$ and $\overline{v_2} = \lfloor 4^2\,\frac{26}{64} \rceil = 7$. We have $-6 = -1 \cdot 4^1 - 2$ and $7 = 1 \cdot 4^2 - 2 \cdot 4^1 - 1 \equiv -2 \cdot 4^1 - 1 \pmod{4^2}$, and so $G^{-1}(\boldsymbol{v}) = (-1, -2, -2, -1)$. We can verify that $G^{-1}(\boldsymbol{v}) \cdot \boldsymbol{G}^\intercal = (-\frac{24}{64}, -\frac{36}{64}) \equiv (\frac{40}{64}, \frac{28}{64}) \approx \boldsymbol{v}$.

Now with the same parameters but with $\ell = 3$ (and thus $B^\ell = q$), we have

$$\boldsymbol{G}^\intercal = \begin{pmatrix} 1/4 & 0 \\ 1/16 & 0 \\ 1/64 & 0 \\ 0 & 1/4 \\ 0 & 1/16 \\ 0 & 1/64 \end{pmatrix} \in \mathbb{T}_q^{6\times 2} \enspace .$$

We have $\overline{v_1} = -23$ and $\overline{v_2} = 26$. We obtain $G^{-1}(\boldsymbol{v}) = (-1, -2, 1, -2, -1, -2)$ and $G^{-1}(\boldsymbol{v}) \cdot \boldsymbol{G}^\intercal = (-\frac{23}{64}, -\frac{38}{64}) \equiv (\frac{41}{64}, \frac{26}{64}) = \boldsymbol{v}$.

Remark 4. The inverse transformation G^{-1} naturally extends to matrices. For a matrix $\boldsymbol{M} \in \mathbb{T}_q^{m\times(n+1)}$, $G^{-1}(\boldsymbol{M}) \in \mathbb{Z}^{m\times(n+1)\ell}$ is defined as the $m \times (n+1)\ell$ matrix whose row $\#i$ is $G^{-1}(\boldsymbol{m_i})$ where $\boldsymbol{m_i}$ is row $\#i$ of $\boldsymbol{M}$. It satisfies $G^{-1}(\boldsymbol{M}) \cdot \boldsymbol{G} \approx \boldsymbol{M}$.

TGSW encryption The gadget matrix gives rise to a torus-based variant of the Gentry–Sahai–Waters (GSW) encryption scheme.

Let an integer $p \mid q$ where $q = 2^{\Omega}$. The gadget decomposition over $\mathbb{T}_q$ supposes integers B and ℓ such that $B^{\ell} \mid q$. Actually, since all its elements are 0 or of the form B^{-j} with $1 \le j \le \ell$, the gadget matrix $\boldsymbol{G}$ is actually defined over $B^{-\ell}\mathbb{Z}/\mathbb{Z} \subseteq \mathbb{T}_q$. We assume that $p = B^{\ell}$. In this case, $\boldsymbol{G}$ is defined over $\mathbb{T}_p = p^{-1}\mathbb{Z}/\mathbb{Z}$.

The private key is $\boldsymbol{s} = (s_1, \dots, s_n) \in \mathbb{B}^n$ and the plaintext space is $\overline{\mathcal{P}} := \mathbb{Z}/p\mathbb{Z}$. The *TGSW encryption* of $m \in \overline{\mathcal{P}}$ under key $\boldsymbol{s}$ is defined as

$$\mathrm{TGSW}_{\boldsymbol{s}}(m) = \boldsymbol{Z} + m \cdot \boldsymbol{G}^{\intercal} \quad (\in \mathbb{T}_q^{(n+1)\ell\times(n+1)})$$

where

$$\boldsymbol{Z} \leftarrow \begin{pmatrix} \mathrm{TLWE}_{\boldsymbol{s}}(0) \\ \mathrm{TLWE}_{\boldsymbol{s}}(0) \\ \vdots \\ \mathrm{TLWE}_{\boldsymbol{s}}(0) \end{pmatrix} \left.\vphantom{\begin{pmatrix} a \\ a \\ \vdots \\ a \end{pmatrix}}\right\} (n+1)\ell \text{ rows} \enspace .$$

> **!** The last row of $\mathrm{TGSW}_{\boldsymbol{s}}(m) \in \mathbb{T}_q^{(n+1)\ell\times(n+1)}$ contains $\mathrm{TLWE}_{\boldsymbol{s}}(0) + m \cdot (0, \dots, 0, \frac{1}{B^{\ell}}) \in \mathbb{T}_q^{n+1}$, that is, a TLWE encryption of $\mu := \frac{m}{B^{\ell}} \in \mathcal{P}$ where $\mathcal{P} = \mathbb{T}_p$.

Being defined over the ring $\overline{\mathcal{P}} = \mathbb{Z}/p\mathbb{Z}$, TGSW plaintexts can be multiplied. For $m_1, m_2 \in \overline{\mathcal{P}}$, given their respective ciphertexts $\boldsymbol{C_1} \leftarrow \mathrm{TGSW}_{\boldsymbol{s}}(m_1)$ and $\boldsymbol{C_2} \leftarrow \mathrm{TGSW}_{\boldsymbol{s}}(m_2)$, we let $\boldsymbol{C_3} = \boldsymbol{C_1} \boxtimes \boldsymbol{C_2} := G^{-1}(\boldsymbol{C_2}) \cdot \boldsymbol{C_1}$. This is known as the [internal] product of ciphertexts [GSW13, AP14, DM15]. It can be verified that $\boldsymbol{C_3} = \boldsymbol{C_1} \boxtimes \boldsymbol{C_2}$ is a TGSW of $m_3 = m_1 \times m_2 \pmod{p}$, up to rounding error and multiplicative noise.

Proof. From the definition, we have $\boldsymbol{C_3} = \boldsymbol{C_1} \boxtimes \boldsymbol{C_2} = G^{-1}(\boldsymbol{C_2}) \cdot \boldsymbol{C_1} = G^{-1}(\boldsymbol{C_2}) \cdot (\boldsymbol{Z_1} + m_1 \cdot \boldsymbol{G}^{\intercal}) = G^{-1}(\boldsymbol{C_2}) \cdot \boldsymbol{Z_1} + (G^{-1}(\boldsymbol{C_2})\, m_1) \cdot \boldsymbol{G}^{\intercal}$, letting $\boldsymbol{C_1} = \boldsymbol{Z_1} + m_1 \cdot \boldsymbol{G}^{\intercal}$ where $\boldsymbol{Z_1} \leftarrow \mathrm{TGSW}_{\boldsymbol{s}}(0)$.

Let $\boldsymbol{\epsilon_2} := G^{-1}(\boldsymbol{C_2}) \cdot \boldsymbol{G}^{\intercal} - \boldsymbol{C_2}$ denote the rounding error matrix. We so get $\boldsymbol{C_3} = G^{-1}(\boldsymbol{C_2}) \cdot \boldsymbol{Z_1} + m_1 \cdot (\boldsymbol{C_2} + \boldsymbol{\epsilon_2}) = G^{-1}(\boldsymbol{C_2}) \cdot \boldsymbol{Z_1} + m_1 \cdot \boldsymbol{Z_2} + (m_1 m_2) \cdot \boldsymbol{G}^{\intercal} + m_1 \cdot \boldsymbol{\epsilon_2}$, letting $\boldsymbol{C_2} = \boldsymbol{Z_2} + m_2 \cdot \boldsymbol{G}^{\intercal}$ where $\boldsymbol{Z_2} \leftarrow \mathrm{TGSW}_{\boldsymbol{s}}(0)$. Assuming the error resulting from the rounding (i.e, $m_1 \cdot \boldsymbol{\epsilon_2}$) keeps “small” and that the multiplicative noise keeps “small”, we can write $\boldsymbol{C_3} = \boldsymbol{Z_3} + (m_1 m_2) \cdot \boldsymbol{G}^{\intercal}$ for some $\boldsymbol{Z_3} \leftarrow \mathrm{TGSW}_{\boldsymbol{s}}(0)$. □

> **!** If $\boldsymbol{Z} \in \mathbb{T}_q^{(n+1)\times(n+1)}$ is a matrix whose rows are TLWE encryptions of 0 then, for any (small) matrix $\boldsymbol{A} \in \mathbb{Z}^{m\times(n+1)}$, $\boldsymbol{Z'} = \boldsymbol{A} \cdot \boldsymbol{Z} \in \mathbb{T}_q^{m\times(n+1)}$ is a matrix whose rows are TLWE encryptions of 0 (up to the noise).

Example 11. To see it, suppose $m = n = 2$. Letting

$$\boldsymbol{A} = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \end{pmatrix} \quad\text{and}\quad \boldsymbol{Z} = \begin{pmatrix} a_{1,1} & a_{1,2} & b_1 \\ a_{2,1} & a_{2,2} & b_2 \\ a_{3,1} & a_{3,2} & b_3 \end{pmatrix} \text{ with } b_i = \sum_{j=1}^{2} s_j \cdot a_{i,j} + e_i$$

we get $\boldsymbol{Z'} = \boldsymbol{A} \cdot \boldsymbol{Z} := \begin{pmatrix} a'_{1,1} & a'_{1,2} & b'_1 \\ a'_{2,1} & a'_{2,2} & b'_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{3} \alpha_{1,i} \cdot a_{i,1} & \sum_{i=1}^{3} \alpha_{1,i} \cdot a_{i,2} & \sum_{i=1}^{3} \alpha_{1,i} \cdot b_i \\ \sum_{i=1}^{3} \alpha_{2,i} \cdot a_{i,1} & \sum_{i=1}^{3} \alpha_{2,i} \cdot a_{i,2} & \sum_{i=1}^{3} \alpha_{2,i} \cdot b_i \end{pmatrix}$.
Remark that $b'_1 = \sum_{i=1}^{3} \alpha_{1,i} \cdot b_i = \sum_{i=1}^{3} \alpha_{1,i} \cdot (\sum_{j=1}^{2} s_j \cdot a_{i,j} + e_i) = \sum_{i=1}^{3} \sum_{j=1}^{2} \alpha_{1,i} \cdot (s_j \cdot a_{i,j}) + \sum_{i=1}^{3} \alpha_{1,i} \cdot e_i = \sum_{j=1}^{2} s_j \cdot (\sum_{i=1}^{3} \alpha_{1,i} \cdot a_{i,j}) + \sum_{i=1}^{3} \alpha_{1,i} \cdot e_i = \sum_{j=1}^{2} s_j \cdot a'_{1,j} + e'_1$ with $e'_1 := \sum_{i=1}^{3} \alpha_{1,i} \cdot e_i$; and similarly $b'_2 = \sum_{j=1}^{2} s_j \cdot a'_{2,j} + e'_2$ with $e'_2 := \sum_{i=1}^{3} \alpha_{2,i} \cdot e_i$.

Inspecting the proof shows that the resulting error term present in $\boldsymbol{Z_3}$ comprises three components: (i) one coming from the noise present in $\boldsymbol{Z_1}$, which is amplified by $G^{-1}(\boldsymbol{C_2})$; (ii) one coming from the noise present in $\boldsymbol{Z_2}$, which is amplified by m_1; and (iii) one coming from the rounding error $\boldsymbol{\epsilon_2}$, which is also amplified by m_1. The multiplicative noise can grow quickly. The use of the gadget matrix leads however to a favorable situation since by construction $\|G^{-1}(\boldsymbol{C_2})\|_\infty \le B/2$. Furthermore, the two other components can be contained if plaintext m_1 keeps small (for example, if m_1 is restricted to elements in $\{0, 1\}$).

External product of ciphertexts TLWE ciphertexts are [much] shorter than TGSW ciphertexts and should therefore be preferred. The best we can do for TLWE is to consider the external product of plaintexts: $m_1 \cdot \mu_2$ for some integer $m_1 \in \overline{\mathcal{P}}$ and a plaintext $\mu_2 \in \mathcal{P} \subset \mathbb{T}_q$. Corresponding to $m_1 \cdot \mu_2$ is the *external* product of ciphertexts. The $\boxdot$ operation enables the external multiplication of ciphertexts. It is given by

$$\boxdot\colon \text{TGSW} \times \text{TLWE} \to \text{TLWE}, \quad (\boldsymbol{C_1}, \boldsymbol{c_2}) \mapsto \boldsymbol{C_1} \boxdot \boldsymbol{c_2} = G^{-1}(\boldsymbol{c_2}) \cdot \boldsymbol{C_1}$$

where $\boldsymbol{C_1} \leftarrow \text{TGSW}_{\boldsymbol{s}}(m_1)$ with $m_1 \in \overline{\mathcal{P}}$ and where $\boldsymbol{c_2} \leftarrow \text{TLWE}_{\boldsymbol{s}}(\mu_2)$ with $\mu_2 \in \mathcal{P}$. In more detail, we have:

$$\boldsymbol{C_1} = \boldsymbol{Z_1} + m_1 \cdot \boldsymbol{G}^\intercal \in \mathbb{T}_q^{(n+1)\ell \times (n+1)} \quad\text{and}\quad \boldsymbol{c_2} \in \mathbb{T}_q^{n+1}$$

where

$$\boldsymbol{Z_1} = \begin{pmatrix} a_{1,1} & \dots & a_{1,n} & b_1 \\ a_{2,1} & \dots & a_{2,n} & b_2 \\ \vdots & & \vdots & \vdots \\ a_{(n+1)\ell,1} & \dots & a_{(n+1)\ell,n} & b_{(n+1)\ell} \end{pmatrix}$$

with

$$\begin{cases} (a_{i,1}, \dots, a_{i,n}) \xleftarrow{\$} \mathbb{T}_q^n \\ b_i = \sum_{j=1}^{n} s_j \cdot a_{i,j} + (e_1)_i \end{cases}$$

and

$$\boldsymbol{c_2} = (a'_1, \ldots, a'_n, b') \quad \text{with} \begin{cases} (a'_1, \ldots, a'_n) \xleftarrow{\$} \mathbb{T}_q^n \\ b' = \sum_{j=1}^n s_j \cdot a_j + \mu_2 + e_2 \end{cases},$$

and where $(e_1)_i$ for $1 \le i \le (n+1)\ell$ and e_2 are "small". Then

$$\begin{aligned}
\boldsymbol{c_3} &:= \boldsymbol{C_1} \boxdot \boldsymbol{c_2} = G^{-1}(\boldsymbol{c_2}) \cdot \boldsymbol{C_1} = G^{-1}(\boldsymbol{c_2}) \cdot (\boldsymbol{Z_1} + m_1 \cdot \boldsymbol{G}^\intercal) \\
&= \underbrace{G^{-1}(\boldsymbol{c_2}) \cdot \boldsymbol{Z_1}}_{=\mathrm{TLWE}_{\boldsymbol{s}}(0)} + m_1 \cdot (\underbrace{G^{-1}(\boldsymbol{c_2}) \cdot \boldsymbol{G}^\intercal}_{\approx \boldsymbol{c_2}}) \\
&= \mathrm{TLWE}_{\boldsymbol{s}}(0) + m_1 \cdot \boldsymbol{c_2} \\
&= \mathrm{TLWE}_{\boldsymbol{s}}(0) + m_1 \cdot \mathrm{TLWE}_{\boldsymbol{s}}(\mu_2) = \mathrm{TLWE}_{\boldsymbol{s}}(m_1 \cdot \mu_2)
\end{aligned}$$

is a valid TLWE encryption of $\mu_3 := m_1 \cdot \mu_2$ (in $\mathcal{P}$), provided that

1. the rounding error $\|G^{-1}(\boldsymbol{c_2}) \cdot \boldsymbol{G}^\intercal - \boldsymbol{c_2}\|_\infty$ keeps "small";
2. the multiplicative noise $e_3 := G^{-1}(\boldsymbol{c_2}) \cdot \boldsymbol{e_1}^\intercal + m_1 \cdot e_2$ keeps "small", where $\boldsymbol{e_1} = ((e_1)_1, \ldots, (e_1)_{(n+1)\ell})$.

4.2 TGLWE Ciphertexts

Again, the operations and underlying techniques developed for TLWE and TGSW extend to polynomials. Torus elements are replaced with torus polynomials. Addition and external multiplication are performed modulo $X^N + 1$. The same trick using a gadget matrix (over $\mathbb{T}_{N,q}[X]$) is used to control the noise growth.

4.2.1 Addition of ciphertexts

Let $\mu_1, \mu_2 \in \mathcal{P}_N[X]$. Let also the ciphertexts $\boldsymbol{\mathscr{c}_1} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu_1) = (\mathscr{a}_1, \ldots, \mathscr{a}_k, \mathscr{b}) \in \mathbb{T}_{N,q}[X]^{k+1}$ and $\boldsymbol{\mathscr{c}_2} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu_2) = (\mathscr{a}'_1, \ldots, \mathscr{a}'_k, \mathscr{b}') \in \mathbb{T}_{N,q}[X]^{k+1}$. If $\mathscr{e}_1$ and $\mathscr{e}_2$ are the respective noise present in $\boldsymbol{\mathscr{c}_1}$ and $\boldsymbol{\mathscr{c}_2}$ then $\boldsymbol{\mathscr{c}_3} := \boldsymbol{\mathscr{c}_1} + \boldsymbol{\mathscr{c}_2} = (\mathscr{a}_1 + \mathscr{a}'_1, \ldots, \mathscr{a}_k + \mathscr{a}'_k, \mathscr{b} + \mathscr{b}') \in \mathbb{T}_{N,q}[X]^{k+1}$ is a valid TGLWE encryption of $\mu_3 := \mu_1 + \mu_2$ (in $\mathcal{P}_N[X]$), provided that the additive noise $\mathscr{e}_3 := \mathscr{e}_1 + \mathscr{e}_2$ keeps "small".

4.2.2 Multiplication by a known polynomial

Let $\mu \in \mathcal{P}_N[X]$ and let $K \in \mathbb{Z} \subset \mathbb{Z}_N[X]$ (i.e., viewed as a degree 0 polynomial in $\mathbb{Z}_N[X]$). Given the ciphertext $\boldsymbol{\mathscr{c}} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu)$,

$$\boldsymbol{\mathscr{c}}' := K \cdot \boldsymbol{\mathscr{c}}$$

is a valid ciphertext of $\mu' = K \cdot \mu$ (in $\mathcal{P}_N[X]$), provided that the resulting noise keeps "small". More generally, for a (small) polynomial $\mathscr{k} \in \mathbb{Z}_N[X]$, $\boldsymbol{c}' = \mathscr{k} \cdot \boldsymbol{c}$ is a valid ciphertext of $\mu' = \mathscr{k} \cdot \mu$ (in $\mathcal{P}_N[X]$), provided that the resulting noise keeps "small".

4.2.3 Multiplication of ciphertexts

Gadget matrix The "gadget vector" $\boldsymbol{g} = (1/B, \ldots, 1/B^\ell) \in \mathbb{T}_q^\ell$ that we used for TLWE/TGSW encryption can be seen as an element in $\mathbb{T}_{N,q}[X]^\ell$. It therefore applies to the polynomial setting too.

Adapting the dimension, we define the *gadget matrix* $\boldsymbol{G}$ *over* $\mathbb{T}_{N,q}[X]$, $\boldsymbol{G} \in \mathbb{T}_{N,q}[X]^{(k+1)\times(k+1)\ell}$, as

$$\boldsymbol{G}^\intercal = \boldsymbol{I}_{k+1} \otimes \boldsymbol{g}^\intercal = \begin{pmatrix} 1/B & & & \\ \vdots & & & \\ 1/B^\ell & & & \\ & 1/B & & \\ & \vdots & & \\ & 1/B^\ell & & \\ & & \ddots & \\ & & & 1/B \\ & & & \vdots \\ & & & 1/B^\ell \end{pmatrix} .$$

The associated inverse transformation $G^{-1}(\cdot)$ flattens a vector of $(k+1)$ polynomials of $\mathbb{T}_{N,q}[X]$ into a vector of $(k+1)\ell$ polynomials of $\mathbb{Z}_N[X]$ with small coefficients (i.e., in the range $[-\lfloor B/2 \rfloor, \lceil B/2 \rceil)$). The definition of $G^{-1}(\cdot)$ is similar to the one of Section 4.1.3 where vectors in $\mathbb{T}_q^{n+1}$ are replaced by vectors in $\mathbb{T}_{N,q}[X]^{k+1}$. Also, for any polynomial vector $\boldsymbol{p} \in \mathbb{T}_{N,q}[X]^{k+1}$, it holds that $G^{-1}(\boldsymbol{p}) \cdot \boldsymbol{G}^\intercal \approx \boldsymbol{p}$ and $G^{-1}(\boldsymbol{p})$ is "small".

Example 12. Take $k = 1$, $N = 2$, $\ell = 3$, $B = 4$, and $q = 256$. Hence,

$$\boldsymbol{G}^\intercal = \begin{pmatrix} 1/4 & 0 \\ 1/16 & 0 \\ 1/64 & 0 \\ 0 & 1/4 \\ 0 & 1/16 \\ 0 & 1/64 \end{pmatrix} .$$

If $\boldsymbol{p} = (\frac{41}{256} + \frac{26}{256}X, \frac{231}{256} + \frac{35}{256}X) \equiv (\frac{41}{256} + \frac{26}{256}X, -\frac{25}{256} + \frac{35}{256}X) \pmod{(X^2+1, 1)}$ then

$$\begin{aligned} \overline{p_1} &= \lfloor 4^3 \tfrac{41}{256} \rceil + \lfloor 4^3 \tfrac{26}{256} \rceil X = 10 + 7X \\ &= (1 \cdot 4^2 - 1 \cdot 4^1 - 2) + (1 \cdot 4^2 - 2 \cdot 4^1 - 1)X \\ &= (1 + X) \cdot 4^2 + (-1 - 2X) \cdot 4^1 + (-2 - X) \end{aligned}$$

and

$$\overline{p_2} = \lfloor 4^3\,(-\tfrac{25}{64})\rceil + \lfloor 4^3\,\tfrac{35}{64}\rceil X = -6 + 9X$$
$$= (0\cdot 4^2 - 1\cdot 4^1 - 2) + (1\cdot 4^2 - 2\cdot 4^1 + 1)X$$
$$= X\cdot 4^2 + (-1-2X)\cdot 4^1 + (-2+X)$$

and so $G^{-1}(\boldsymbol{p}) = (1+X, -1-2X, -2-X, X, -1-2X, -2+X)$.

TGGSW ciphertexts Again, it is worth noting that a TGLWE ciphertext can be seen as $\mathrm{TGLWE}_{\mathfrak{s}}(\mu) \equiv \mathrm{TGLWE}_{\mathfrak{s}}(0) + (0, \dots, 0, 1)\cdot \mu$.

Let $p = B^\ell$ and such that $p \mid q$. Let also $\mathfrak{s} = (\mathfrak{s}_1, \dots, \mathfrak{s}_k) \in \mathbb{B}_N[X]^k$. The *TGGSW encryption* of $m \in \overline{\mathcal{P}}_N[X]$ under private key $\mathfrak{s}$ is defined as

$$\mathrm{TGGSW}_{\mathfrak{s}}(m) = \mathfrak{Z} + m\cdot \boldsymbol{G}^\intercal \quad (\in \mathbb{T}_{N,q}[X]^{(k+1)\ell\times(k+1)})$$

where

$$\mathfrak{Z} \leftarrow \left.\begin{pmatrix} \mathrm{TGLWE}_{\mathfrak{s}}(0) \\ \mathrm{TGLWE}_{\mathfrak{s}}(0) \\ \vdots \\ \mathrm{TGLWE}_{\mathfrak{s}}(0) \end{pmatrix}\right\} (k+1)\ell \text{ rows} \;.$$

External product of ciphertexts Let $m_1 \in \overline{\mathcal{P}}_N[X]$ and $\mu_2 \in \mathcal{P}_N[X]$ and their ciphertexts $\mathfrak{C}_1 \leftarrow \mathrm{TGGSW}_{\mathfrak{s}}(m_1)$ $(\in \mathbb{T}_{N,q}[X]^{(k+1)\ell\times(k+1)})$ and $\boldsymbol{c_2} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu_2)$ $(\in \mathbb{T}_{N,q}[X]^{k+1})$. The *external product* $\boxdot$ of a TGGSW ciphertext by a TGLWE ciphertext is defined as

$$\boxdot\colon\ \mathrm{TGGSW}\times\mathrm{TGLWE} \to \mathrm{TGLWE},$$
$$(\mathfrak{C}_1, \boldsymbol{c_2}) \mapsto \mathfrak{C}_1 \boxdot \boldsymbol{c_2} = G^{-1}(\boldsymbol{c_2})\cdot \mathfrak{C}_1 \;.$$

The resulting ciphertext $\boldsymbol{c_3} := \mathfrak{C}_1 \boxdot \boldsymbol{c_2}$ $(\in \mathbb{T}_{N,q}[X]^{k+1})$ is a valid encryption of $\mu_3 := m_1\cdot\mu_2$ $(\in \mathcal{P}_N[X])$, provided that the rounding error resulting from $G^{-1}(\cdot)$ and the multiplicative noise keep “small”.

CMUX The main application of the external product in TFHE is the “controlled” multiplexer, or CMUX in short. Given two TGLWE ciphertexts $\boldsymbol{c_0} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu_0)$ and $\boldsymbol{c_1} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu_1)$, the CMux operator acts as a selector to choose between $\boldsymbol{c_0}$ and $\boldsymbol{c_1}$ according to a TGGSW encryption $\mathfrak{C}_\mathbf{b} \leftarrow \mathrm{TGGSW}_{\mathfrak{s}}(\mathrm{b})$ of a control bit $\mathrm{b} \in \{0, 1\}$.

This can be computed through an external product as

$$\begin{aligned}\mathrm{CMux}(\mathscr{C}_{\mathbf{b}}, \boldsymbol{c}_{\mathbf{0}}, \boldsymbol{c}_{\mathbf{1}}) &\leftarrow \mathscr{C}_{\mathbf{b}} \boxdot (\boldsymbol{c}_{\mathbf{1}} - \boldsymbol{c}_{\mathbf{0}}) + \boldsymbol{c}_{\mathbf{0}} \\ &\leftarrow \mathrm{TGGSW}_{\mathfrak{s}}(\mathrm{b}) \boxdot \mathrm{TGLWE}_{\mathfrak{s}}(\mu_1 - \mu_0) \\ &\qquad\qquad\qquad\qquad\qquad + \mathrm{TGLWE}_{\mathfrak{s}}(\mu_0) \\ &\leftarrow \mathrm{TGLWE}_{\mathfrak{s}}\big(\mathrm{b}(\mu_1 - \mu_0) + \mu_0\big) \\ &\leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu_{\mathrm{b}}) \ .\end{aligned}$$

The output is a TGLWE encryption of μ_{b}.

4.3 Implementation Notes

The encoding for integers modulo p (including bits when $p = 2$) presented in Section 2.2 respects the addition. In more details, for any $i_1, i_2 \in \mathbb{Z}/p\mathbb{Z}$, letting $i_3 = i_1 + i_2 \bmod p$, we have $\mathrm{Encode}(i_3) = \mathrm{Encode}(i_1) + \mathrm{Encode}(i_2)$ (in $\mathbb{T}_p$). The encoding also respects the external product: for any $i \in \mathbb{Z}/p\mathbb{Z}$ and any integer k, letting $i_k = k \cdot i \bmod p$, we have $\mathrm{Encode}(i_k) = k \cdot \mathrm{Encode}(i)$ (in $\mathbb{T}_p$). In other words, the encoding is homomorphic and so complies with the homomorphic structure of the encryption.

The same holds true for the encoding for fixed-precision torus elements presented in Section 2.2.

5 Programmable Bootstrapping

As aforementioned, both TLWE and TGLWE encryptions are needed for implementing certain operations. We will see in this section that their combination is central to refreshing noisy TLWE ciphertexts. Such an operation is referred to as *bootstrapping*. Furthermore, this operation can be programmed to evaluate at the same time a selected function.

5.1 Gentry's Recryption

For a (symmetric) fully homomorphic encryption algorithm Encrypt, given the encryption of x under private key sk, the homomorphic evaluation of a univariate function f yields the encryption of $f(x)$. This is illustrated in the next figure.

Figure 1: Homomorphic evaluation

$$\mathsf{Encrypt}_{sk}(x) \longrightarrow \boxed{\;f(\cdot)\;} \longrightarrow \mathsf{Encrypt}_{sk}(f(x))$$

Gentry's key idea to reduce the noise present in a ciphertext is to homomorphically evaluate the decryption of the ciphertext using a homomorphic encryption of its own decryption key [Gen10]. The encryption of the decryption key (matching the encryption key used to produce the ciphertext) forms what is called the *bootstrapping key*.

Specifically, let $c \leftarrow \mathfrak{E}\mathsf{ncrypt}_{sk_1}(m)$ denote a noisy ciphertext encrypting a plaintext m and let $\mathsf{bsk} \leftarrow \mathsf{Encrypt}_{sk_2}(sk_1)$ denote the bootstrapping key. Assume that function f in the above figure is the decryption function dedicated to ciphertext c, viewed as the univariate function $\mathfrak{D}\mathsf{ecrypt}(\cdot, c)$. Then, letting $x = sk_1$, the homomorphic evaluation of f yields

$$\begin{aligned} \mathsf{Encrypt}_{sk_2}(f(x)) &= \mathsf{Encrypt}_{sk_2}(\mathfrak{D}\mathsf{ecrypt}(sk_1, c)) \\ &= \mathsf{Encrypt}_{sk_2}(m) \enspace . \end{aligned}$$

The procedure is detailed in Fig. 2.

Figure 2: Recryption

$$\underset{\text{[bootstrapping key]}}{\mathsf{Encrypt}_{sk_2}(sk_1)} \longrightarrow \underset{\text{with } c \leftarrow \mathfrak{E}\mathsf{ncrypt}_{sk_1}(m)}{\boxed{\;\mathfrak{D}\mathsf{ecrypt}(\cdot, c)\;}} \longrightarrow \mathsf{Encrypt}_{sk_2}(m)$$

Starting with the noisy ciphertext $c \leftarrow \mathfrak{E}\mathsf{ncrypt}_{sk_1}(m)$, the recryption process ends up with a new ciphertext $\mathsf{Encrypt}_{sk_2}(m)$, encrypting the same plaintext m. Note that the encryption keys are different. The encryption algorithms Encrypt and $\mathfrak{E}$ncrypt may be distinct or not. In the latter case, the resulting ciphertext can be reverted back into a ciphertext under the initial key sk_1 thanks to a standard key-switching technique.

5.2 Bootstrapping

General description Let $\boldsymbol{s} = (s_1, \dots, s_n) \in \mathbb{B}^n$. Consider a TLWE encryption of $\mu \in \mathcal{P}$: we have $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) = (a_1, \dots, a_n, b) \in$

$\mathbb{T}_q^{n+1}$ where $a_j \xleftarrow{\$} \mathbb{T}_q$ and $b = \sum_{j=1}^{n} s_j \cdot a_j + \mu^*$ with $\mu^* = \mu + e$ for some "small" noise error e. The goal of the bootstrapping procedure is to produce a TLWE ciphertext of the same plaintext but with a reduced amount of noise $e' < e$. So far, the only known way to bootstrap a ciphertext is Gentry's recryption technique. In the case of TFHE, using the previous notations, its application involves two steps:

1. obtaining the noisy plaintext μ^* as $\mu^* = b - \sum_{j=1}^{n} s_j \cdot a_j \in \mathbb{T}_q$;
2. recovering the plaintext μ by rounding μ^* to the closest plaintext as $\mu = \frac{\lfloor p\,\mu^* \rceil \bmod p}{p} \in \mathcal{P}$.

These two steps have to be performed over encrypted data. The first step being linear is easy given an encryption of the s_j's. The second step (i.e., the rounding) is more problematic. This is where polynomials come to the rescue.

Rounding with polynomials Consider polynomial $\mathscr{v}(X) = v_0 + v_1 X + \cdots + v_{N-1} X^{N-1} \in \mathbb{T}_{N,p}[X] = \mathbb{T}_p[X]/(X^N + 1)$. The formula of the external multiplication in $\mathbb{T}_{N,p}[X]$ by a monomial (cf. Section 3.3) teaches that

$$X^{-j} \cdot \mathscr{v}(X) = X^{2N-j} \cdot \mathscr{v}(X) = \begin{cases} v_j + \ldots & \text{for } 0 \le j < N \\ -v_j + \ldots & \text{for } N \le j < 2N \end{cases} .$$

In other words, when $0 \le j < N$, the constant term of polynomial $X^{-j} \cdot \mathscr{v}(X)$ is v_j. As we will see, this simple observation provides a way to round a torus element $\mu^* \in \mathbb{T}_q$ as an element of $\mu \in \mathbb{T}_p$, where $p \mid q$.

Since $\mu^* \in \mathbb{T}_q$, we can write $\mu^* = \bar{\mu}^*/q$ where $\bar{\mu}^* := \lfloor q\,\mu^* \rceil \bmod q$ with $0 \le \bar{\mu}^* < q$. If we suppose for a moment that $N \ge q$, we have $0 \le \bar{\mu}^* < N$. It also means that polynomial $\mathscr{v}$ has more coefficients than the number of possible values for $\bar{\mu}^*$. We can therefore assign a chosen value for v_j, for any $0 \le j < q$, and an application of $X^{-j} \cdot \mathscr{v}(X)$ will yield $v_j + \ldots$ In particular, if we select $v_j := \frac{\lfloor (p\,j)/q \rceil \bmod p}{p} \in \mathbb{T}_p$ plugging $j = \bar{\mu}^*$ in the relation $X^{-j} \cdot \mathscr{v}(X) = v_j + \ldots$ yields

$$\begin{aligned} X^{-\bar{\mu}^*} \cdot \mathscr{v}(X) &= \frac{\lfloor (p\,\bar{\mu}^*)/q \rceil \bmod p}{p} + \ldots \\ &= \frac{\lfloor p\,\mu^* \rceil \bmod p}{p} + \ldots \\ &= \mu + \ldots \end{aligned}$$

namely, a polynomial whose constant term is the rounded value $\mu \in \mathbb{T}_p$.

Example 13. As an illustration, suppose we wish to round 5-bit precision torus elements μ^* to 2-bit precision torus elements μ, for $0 \le \mu^* \le 25/32$; rounding by convention downwards in the case of a tie. This setting corresponds to $q = 32$ and $p = 4$ (that is, $\mathbb{T}_q = \frac{1}{32}\mathbb{Z}/\mathbb{Z}$ and $\mathbb{T}_p = \frac{1}{4}\mathbb{Z}/\mathbb{Z}$).

μ^*		μ
$\frac{0}{32}$	$\to$	$\frac{0}{4}$
$\vdots$		$\vdots$
$\frac{4}{32}$	$\to$	$\frac{0}{4}$
$\frac{5}{32}$	$\to$	$\frac{1}{4}$
$\vdots$		$\vdots$
$\frac{12}{32}$	$\to$	$\frac{1}{4}$
$\frac{13}{32}$	$\to$	$\frac{2}{4}$
$\vdots$		$\vdots$
$\frac{20}{32}$	$\to$	$\frac{2}{4}$
$\frac{21}{32}$	$\to$	$\frac{3}{4}$
$\vdots$		$\vdots$
$\frac{25}{32}$	$\to$	$\frac{3}{4}$

Since there are 26 possible values for μ^*, we set $N = 32$ (i.e., as the smallest power of 2 that is ≥ 26). We set polynomial $\mathcal{v}$ as

$$\begin{aligned}\mathcal{v}(X) = {} & \tfrac{0}{4} + \tfrac{0}{4}X + \tfrac{0}{4}X^2 + \tfrac{0}{4}X^3 + \tfrac{0}{4}X^4 \\ & + \tfrac{1}{4}X^5 + \tfrac{1}{4}X^6 + \tfrac{1}{4}X^7 + \tfrac{1}{4}X^8 + \tfrac{1}{4}X^9 \\ & + \tfrac{1}{4}X^{10} + \tfrac{1}{4}X^{11} + \tfrac{1}{4}X^{12} + \tfrac{2}{4}X^{13} + \tfrac{2}{4}X^{14} \\ & + \tfrac{2}{4}X^{15} + + \tfrac{2}{4}X^{16} + \tfrac{2}{4}X^{17} + \tfrac{2}{4}X^{18} + \tfrac{2}{4}X^{19} \\ & + \tfrac{2}{4}X^{20} + \tfrac{3}{4}X^{21} + \tfrac{3}{4}X^{22} + \tfrac{3}{4}X^{23} + \tfrac{3}{4}X^{24} + \tfrac{3}{4}X^{25} \ .\end{aligned}$$

It can be checked that any 5-bit precision element $\mu^* \in \left[0, \frac{25}{32}\right] \subset \mathbb{T}_q$ verifies

$$X^{-\lfloor 32\mu^* \rceil} \cdot \mathcal{v}(X) = \mu + \dots$$

where $\mu \in \mathbb{T}_p$ denotes the matching rounded value.

5.2.1 Blind rotation

As above, let $\bar{\mu}^* = \lfloor q\,\mu^* \rceil \bmod q$. Let also $\bar{a}_j = \lfloor q\,a_j \rceil \bmod q$ and $\bar{b} = \lfloor q\,b \rceil \bmod q$. In order to bootstrap, one way to look at the decryption (without the rounding) is to see that

$$-\bar{\mu}^* = -\bar{b} + \textstyle\sum_{j=1}^n s_j\,\bar{a}_j \pmod q \ .$$

This value can then be put at the exponent of X to get the monomial $X^{-\bar{\mu}^*}$, which leads to plaintext μ from the evaluation of $X^{-\bar{\mu}^*} \cdot \mathcal{v}(X)$. There are a couple of complications in implementing this idea as it supposes $q < N$, which is not verified in practical settings. Typical cryptographic parameters mandate $N \in \{2^{10}, 2^{11}, 2^{12}\}$ and $q \in \{2^{32}, 2^{64}\}$.

First, the relation $X^{-\bar{\mu}^*} \cdot \mathcal{v}(X)$ being defined modulo $X^N + 1$, this means that, as a multiplicative element of $\mathbb{Z}_N[X]$, X is of order $2N$ (i.e., $X^{2N} = 1$) and thus exponent $-\bar{\mu}^*$ in $X^{-\bar{\mu}^*} \cdot \mathcal{v}(X)$ is defined modulo $2N$. The value of $\bar{\mu}^*$ needs therefore to be rescaled modulo $2N$. As a consequence, instead of starting with the relation $-\bar{\mu}^* = -\bar{b} + \sum_{j=1}^n s_j\,\bar{a}_j \pmod q$, we rely on the approximation

$$-\tilde{\mu}^* = -\tilde{b} + \textstyle\sum_{j=1}^n s_j\,\tilde{a}_j \pmod{2N} \,,$$

where $\tilde{b} = \lfloor 2Nb \rceil \bmod 2N$ and $\tilde{a}_j = \lfloor 2Na_j \rceil \bmod 2N$. This approximation may generate a small additional error that adds to the noise.

> **!** The additional error introduced by the discretization modulo $2N$ is called *drift*. Its impact on the result can be dealt with by a careful choice of the parameters.

Second, because polynomial $\mathscr{v}$ lies in $\mathbb{T}_{N,p}[X]$ and thus has N coefficients, at most N values for $\tilde{\mu}^*$ can be encoded. This is addressed by ensuring that the most significant bit of $\tilde{\mu}^*$ is set to 0. In this case, $\tilde{\mu}^*$ can take at most N possible values.

From the above considerations, the so-called *test polynomial* $\mathscr{v}$ is formed as

$$\mathscr{v} := \mathscr{v}(X) = \sum_{j=0}^{N-1} v_j X^j \quad \text{with } v_j = \frac{\lfloor \frac{pj}{2N} \rceil \bmod p}{p} \in \mathcal{P}$$

and the relation

$$X^{-\tilde{b}+\sum_{j=1}^{n} s_j \tilde{a}_j} \cdot \mathscr{v}(X) = X^{-\tilde{\mu}^*} \cdot \mathscr{v}(X) = \mu + \dots$$

holds, provided that the drift is contained and that $0 \le (\tilde{\mu}^* \bmod 2N) < N$. For conciseness, we let $\mathscr{q}_j := X^{-\tilde{b}+\sum_{i=1}^{j} s_i \tilde{a}_i} \cdot \mathscr{v}$. The external product being homogeneous, it follows that

$$\begin{aligned}\mathscr{q}_j &= \left(X^{-\tilde{b}+\sum_{i=1}^{j-1} s_i \tilde{a}_i} X^{s_j \tilde{a}_j}\right) \cdot \mathscr{v} = X^{s_j \tilde{a}_j} \cdot \left(X^{-\tilde{b}+\sum_{i=1}^{j-1} s_i \tilde{a}_i} \cdot \mathscr{v}\right) = X^{s_j \tilde{a}_j} \cdot \mathscr{q}_{j-1} \\ &= \begin{cases} \mathscr{q}_{j-1} & \text{if } s_j = 0 \\ X^{\tilde{a}_j} \cdot \mathscr{q}_{j-1} & \text{if } s_j = 1 \end{cases} .\end{aligned}$$

This provides an iterative method to get $\mathscr{q}_n = X^{-\tilde{b}+\sum_{i=1}^{n} s_i \tilde{a}_i} \cdot \mathscr{v}$, starting at $\mathscr{q}_0 = X^{-\tilde{b}} \cdot \mathscr{v}$ and then iterating on j from 1 to n.

Gentry's recryption does the same but over encrypted data. As the rounding method involves polynomials, we rely on TGLWE encryption. Let $\mathscr{s}' \in \mathbb{B}_N[X]^{k+1}$. We assume that we are given the *bootstrapping keys* $\mathsf{bsk}[j] \leftarrow \mathrm{TGGSW}_{\mathscr{s}'}(s_j) \in \mathbb{T}_{N,q}[X]^{(k+1)\ell \times (k+1)}$, for $1 \le j \le n$. We have:

in the clear	over encrypted data
$\mathscr{q}_0 \leftarrow X^{-\tilde{b}} \cdot \mathscr{v}$	$\boldsymbol{\mathscr{c}}'_0 \leftarrow X^{-\tilde{b}} \cdot \mathrm{TGLWE}_{\mathscr{s}'}(\mathscr{v})$
for $j = 1$ **to** n **do**	**for** $j = 1$ **to** n **do**
$\quad \mathscr{q}_j \leftarrow \begin{cases} \mathscr{q}_{j-1} & \text{if } s_j = 0 \\ X^{\tilde{a}_j} \cdot \mathscr{q}_{j-1} & \text{if } s_j = 1 \end{cases}$	$\quad \boldsymbol{\mathscr{c}}'_j \leftarrow \mathrm{CMux}(\mathsf{bsk}[j], \boldsymbol{\mathscr{c}}'_{j-1}, X^{\tilde{a}_j} \cdot \boldsymbol{\mathscr{c}}'_{j-1})$
end for	**end for**
return $\mathscr{q}_n$	**return** $\boldsymbol{\mathscr{c}}'_n$

Clearly, the output ciphertext $\mathfrak{c}' := \mathfrak{c}'_n$ is a TGLWE encryption of $\mathfrak{q}_n = X^{-\tilde{b}+\sum_{j=1}^{n} s_j \tilde{a}_j} \cdot \mathfrak{v}$; i.e., $\mathfrak{c}'_n \leftarrow \mathrm{TGLWE}_{\mathfrak{s}'}\big(X^{-\tilde{b}+\sum_{j=1}^{n} s_j \tilde{a}_j} \cdot \mathfrak{v}\big) = \mathrm{TGLWE}_{\mathfrak{s}'}\big(X^{-\tilde{\mu}^*} \cdot \mathfrak{v}\big)$. Finally, we remark that $(0, \dots, 0, \mathfrak{v}) \in \mathbb{T}_{N,q}[X]^{k+1}$ is a valid TGLWE encryption for $\mathfrak{v}$; we can thus take $\mathfrak{c}'_0 \leftarrow X^{-\tilde{b}} \cdot (0, \dots, 0, \mathfrak{v})$.

Summing up, given a TLWE ciphertext $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) \in \mathbb{T}_q^{n+1}$ under the key $\boldsymbol{s} = (s_1, \dots, s_n) \in \mathbb{B}^n$ and the matching bootstrapping-key vector $\mathbf{bsk} = (\mathrm{bsk}[1], \dots, \mathrm{bsk}[n])$ with $\mathrm{bsk}[j] \leftarrow \mathrm{TGGSW}_{\mathfrak{s}'}(s_j)$ and $\mathfrak{s}' = (\mathfrak{s}'_1, \dots, \mathfrak{s}'_k) \in \mathbb{B}_N[X]^k$, we get a TGLWE ciphertext $\mathfrak{c}' \leftarrow \mathrm{TGLWE}_{\mathfrak{s}'}(X^{-\tilde{\mu}^*} \cdot \mathfrak{v}) = \mathrm{TGLWE}_{\mathfrak{s}'}(\mu + \dots) \in \mathbb{T}_{N,q}[X]^{k+1}$ under the key $\mathfrak{s}'$ for the predefined polynomial $\mathfrak{v}(X) = \sum_{j=0}^{N-1} \frac{\lfloor pj/(2N) \rceil \bmod p}{p} X^j \in \mathcal{P}_N[X]$, in two steps as:

1. define $\mathfrak{c} := (0, \dots, 0, \mathfrak{v})$ and $\tilde{\boldsymbol{c}} := (\tilde{a}_1, \dots, \tilde{a}_n, \tilde{b}) \leftarrow \lfloor \boldsymbol{c}\, 2N \rceil \bmod 2N$;
2. do
$$\begin{cases} \mathfrak{c}'_0 \leftarrow X^{-\tilde{b}} \cdot \mathfrak{c} \\ \mathfrak{c}'_j \leftarrow \mathrm{CMux}(\mathrm{bsk}[j], \mathfrak{c}'_{j-1}, X^{\tilde{a}_j} \cdot \mathfrak{c}'_{j-1}) \quad \text{for } 1 \le j \le n \end{cases}$$
and set $\mathfrak{c}' := \mathfrak{c}'_n$.

We write $\mathfrak{c}' \leftarrow \mathrm{BlindRotate}_{\mathbf{bsk}}(\mathfrak{c}, \tilde{\boldsymbol{c}})$ where $\mathbf{bsk} = (\mathrm{bsk}[1], \dots, \mathrm{bsk}[n])$.

! Algorithms in pseudo-code are provided in Appendix B.

5.2.2 Sample extraction

The previous conversion step turns the TLWE encryption of a plaintext $\mu \in \mathcal{P}$ into a TGLWE encryption of a polynomial plaintext $\upmu(X) := X^{-\tilde{\mu}^*} \cdot \mathfrak{v} \in \mathcal{P}_N[X]$ whose constant term is μ. The constant-term component is then extracted to give rise to a refreshed TLWE encryption of μ, but under a different key. This is referred to as *sample extraction*. We note that, although it is applied to the constant term, the technique readily adapts to extract other components of $\upmu$.

In more detail, on input a TLWE ciphertext $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) \in \mathbb{T}_q^{n+1}$, the previous step yields at the end of the blind rotation a TGLWE ciphertext $\mathfrak{c}' \leftarrow \mathrm{TGLWE}_{\mathfrak{s}'}(X^{-\tilde{\mu}^*} \cdot \mathfrak{v}) = \mathrm{TGLWE}_{\mathfrak{s}'}(\mu + \dots) \in \mathbb{T}_{N,q}[X]^{k+1}$.

We let $\mathfrak{s}' = (\mathfrak{s}'_1, \dots, \mathfrak{s}'_k) \in \mathbb{B}_N[X]^k$ and $\mathfrak{c}' = (\mathfrak{a}'_1, \dots, \mathfrak{a}'_k, \mathfrak{b}') \in \mathbb{T}_N[X]^{k+1}$ where, for $1 \le j \le k$, $\mathfrak{s}'_j := \mathfrak{s}'_j(X) = (s'_j)_0 + (s'_j)_1 X + \dots + (s'_j)_{N-1} X^{N-1}$ and $\mathfrak{a}'_j := \mathfrak{a}'_j(X) = (a'_j)_0 + (a'_j)_1 X + \dots + (a'_j)_{N-1} X^{N-1}$.

We also let $\upmu = X^{-\tilde{\mu}^*} \cdot \mathcal{v} = \mu + \cdots$. By definition of a TLWE ciphertext, there exists $\mathcal{e} := \mathcal{e}(X) = e_0 + e_1 X + \cdots + e_{N-1}X^{N-1}$ such that $\mathcal{b}' = \sum_{j=1}^{k} \mathcal{s}'_j \cdot \mathcal{a}'_j + \upmu + \mathcal{e}$.

Expanding polynomial $\mathcal{b}'$, we get

$$\begin{aligned}\mathcal{b}' &:= \mathcal{b}'(X) = b'_0 + b'_1 X + \cdots + b'_{N-1}X^{N-1}\\ &= \sum_{j=1}^{k}\big((s'_j)_0 + \cdots + (s'_j)_{N-1}X^{N-1}\big)\cdot\big((a'_j)_0 + \cdots + (a'_j)_{N-1}X^{N-1}\big)\\ &\qquad + \upmu + \mathcal{e}\enspace.\end{aligned}$$

Now, if we take a close look at the constant term $b'_0 \in \mathbb{T}_q$ of polynomial $\mathcal{b}'$, we see that it satisfies

$$\begin{aligned}b'_0 &= \sum_{j=1}^{k}\big[(s'_j)_0\cdot(a'_j)_0 - (s'_j)_1\cdot(a'_j)_{N-1} - \cdots - (s'_j)_{N-1}\cdot(a'_j)_1\big] + \mu + e_0\\ &= \big((s'_1)_0, (s'_1)_1, \dots, (s'_1)_{N-1}, \dots, \dots, (s'_k)_0, (s'_k)_1, \dots, (s'_k)_{N-1}\big)\cdot\\ &\quad \big((a'_1)_0, -(a'_1)_{N-1}, \dots, -(a'_1)_1, \dots, \dots, (a'_k)_0, -(a'_k)_{N-1}, \dots,\\ &\qquad -(a'_k)_1\big) + \mu + e_0\enspace.\end{aligned}$$

As a result, defining $\boldsymbol{s'} := \big((s'_1)_0, (s'_1)_1, \dots, (s'_k)_{N-1}\big) \in \mathbb{B}^{kN}$ and $\dot{\boldsymbol{a}}' := \big((a'_1)_0, -(a'_1)_{N-1}, \dots, -(a'_k)_1\big) \in \mathbb{T}_q{}^{kN}$, the vector $\boldsymbol{c'} := (\dot{\boldsymbol{a}}', b'_0) \in \mathbb{T}_q{}^{kN+1}$ can be viewed as a TLWE encryption of μ under key $\boldsymbol{s'}$.

We write $\boldsymbol{s'} \leftarrow \mathsf{Recode}(\mathcal{s}')$ and $\boldsymbol{c'} \leftarrow \mathsf{SampleExtract}(\mathcal{c}')$.

5.2.3 Key switching

The loop is almost closed. With the above procedure, ciphertexts $\boldsymbol{c}$ and $\boldsymbol{c'} \leftarrow \mathsf{SampleExtract}(\mathsf{BlindRotate}_{\mathbf{bsk}}(\mathcal{c}, \tilde{\boldsymbol{c}}))$ both encrypt plaintext μ but they feature a different set of parameters: $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) \in \mathbb{T}_q{}^{n+1}$ and $\boldsymbol{c'} \leftarrow \mathrm{TLWE}_{\boldsymbol{s'}}(\mu) \in \mathbb{T}_q{}^{kN+1}$. The *key switching algorithm* converts a ciphertext under a key into a ciphertext under another key. Its implementation requires *key-switching keys*, i.e., TLWE encryptions of the key bits of $\boldsymbol{s'}$ with respect to the original key $\boldsymbol{s}$. The procedure may seem conceptually very similar to the bootstrapping, but there is a fundamental difference between the two techniques: bootstrapping reduces the noise (and is computationally demanding) whereas the key switching makes the noise increase (but is cheaper to evaluate).

Assume we are given the key-switching keys

$$\mathsf{ksk}[i,j] \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(s'_i \cdot B^{-j}) \qquad (1 \le i \le kN \text{ and } 1 \le j \le \ell)$$

for some parameters B and ℓ defining a gadget decomposition (see Section 4.1.3). On input ciphertext $\boldsymbol{c'} \leftarrow \mathrm{TLWE}_{\boldsymbol{s'}}(\mu) = (a'_1, \ldots, a'_{kN}, b') \in \mathbb{T}_q^{\,kN+1}$ under the key $\boldsymbol{s'} = (s'_1, \ldots, s'_{kN}) \in \mathbb{B}^{kN}$, the ciphertext

$$\boldsymbol{c''} := (0, \ldots, 0, b') - \sum_{i=1}^{kN} \sum_{j=1}^{\ell} (\overline{a'_i})_j \cdot \mathrm{ksk}[i,j]$$

where

$$\big((\overline{a'_i})_1, \ldots, (\overline{a'_i})_\ell\big) = g^{-1}(a'_i) \quad \text{with } (\overline{a'_i})_j \in [-\lfloor B/2 \rfloor, \lceil B/2 \rceil)$$

is a TLWE encryption of μ under the key $\boldsymbol{s} \in \mathbb{B}^n$, provided that the resulting noise error remains contained.

We write $\boldsymbol{c''} \leftarrow \mathrm{KeySwitch}_{\mathbf{ksk}}(\boldsymbol{c'})$ with $\mathbf{ksk} = (\mathrm{ksk}[i,j])_{\substack{1 \le i \le kN \\ 1 \le j \le \ell}}$.

Proof. The gadget decomposition leads to $g^{-1}(a'_i) \cdot \boldsymbol{g}^\intercal = \sum_{j=1}^{\ell} (\overline{a'_i})_j \cdot B^{-j} = a'_i + \epsilon_i$ where ϵ_i denotes the rounding error. Hence, $\sum_{j=1}^{\ell} (\overline{a'_i})_j \cdot \mathrm{ksk}[i,j] = \sum_{j=1}^{\ell} (\overline{a'_i})_j \cdot \mathrm{TLWE}_{\boldsymbol{s}}(s'_i \cdot B^{-j}) = \mathrm{TLWE}_{\boldsymbol{s}}(s'_i \cdot (a'_i + \epsilon_i))$. Moreover, $(0, \ldots, 0, b')$ is a valid TLWE encryption for b'. Letting e' the noise present in $\boldsymbol{c'}$, we therefore see that $\boldsymbol{c''} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}\big(b' - \sum_{i=1}^{kN} s'_i \cdot (a'_i + \epsilon_i)\big) = \mathrm{TLWE}_{\boldsymbol{s}}\big(\mu + e' + \sum_{i=1}^{kN} s'_i \epsilon_i\big)$, which decrypts to μ if the error $e'' := e' + \sum_{i=1}^{kN} s'_i \epsilon_i$ keeps small. □

5.2.4 Putting it all together

To sum up, the bootstrapping of a TLWE ciphertext $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) \in \mathbb{T}_q^{\,n+1}$ with $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbb{B}^n$ proceeds as a series of 3 steps.

1. $\boldsymbol{\mathfrak{c}'} \leftarrow \mathrm{BlindRotate}_{\mathbf{bsk}}(\boldsymbol{\mathfrak{c}}, \tilde{\boldsymbol{c}}) \quad (\in \mathbb{T}_{N,q}[X]^{k+1})$, where
   - $\boldsymbol{\mathfrak{c}} = (0, \ldots, 0, \mathfrak{v}) \in \mathbb{T}_{N,q}[X]^{k+1}$ with $\mathfrak{v} := \mathfrak{v}(X) = \sum_{j=0}^{N-1} \frac{\lfloor pj/(2N) \rceil \bmod p}{p} X^j \in \mathcal{P}_N[X] \subsetneq \mathbb{T}_{N,q}[X]$
   - $\tilde{\boldsymbol{c}} = \lceil \boldsymbol{c}\, 2N \rfloor \in (\mathbb{Z}/2N\mathbb{Z})^{n+1}$;
   - $\mathbf{bsk} = (\mathrm{bsk}[j])_{1 \le j \le n}$ with $\begin{cases} \mathrm{bsk}[j] \leftarrow \mathrm{TGGSW}_{\boldsymbol{\mathfrak{s}'}}(s_j) \in \mathbb{T}_{N,q}[X]^{(k+1)\ell \times (k+1)} \\ \boldsymbol{\mathfrak{s}'} = (\mathfrak{s}'_1, \ldots, \mathfrak{s}'_k) \in \mathbb{B}_N[X]^k \end{cases}$;
2. $\boldsymbol{c'} \leftarrow \mathrm{SampleExtract}(\boldsymbol{\mathfrak{c}'}) \quad (\in \mathbb{T}_q^{\,kN+1})$;
3. $\boldsymbol{c''} \leftarrow \mathrm{KeySwitch}_{\mathbf{ksk}}(\boldsymbol{c'}) \quad (\in \mathbb{T}_q^{\,n+1})$, where

- $\mathbf{ksk} = (\mathsf{ksk}[i,j])_{\substack{1\le i\le kN\\1\le j\le \ell}}$
  with $\begin{cases}\mathsf{ksk}[i,j] \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(s'_i \cdot B^{-j}) \in \mathbb{T}_q{}^{n+1}\\ \boldsymbol{s'} = (s'_1, \dots, s'_{kN}) \leftarrow \mathrm{Recode}(\boldsymbol{\mathfrak{s}'}) \in \mathbb{B}^{kN}\end{cases}$.

5.3 Programmable Bootstrapping

The (regular) bootstrapping essentially relies on the observation that $X^{-j} \cdot \mathfrak{v}(X) = v_j + \dots$, for any $0 \le j < N$. In the above section, test polynomial $\mathfrak{v} \in \mathbb{T}_N[X]$ was defined as $\mathfrak{v}(X) = \sum_{j=0}^{N-1} \frac{\lfloor pj/(2N)\rceil \bmod p}{p} X^j$.

Now, given a function $f\colon \mathbb{T}_p \to \mathbb{T}_p$, if we instead define test polynomial $\mathfrak{v}$ as

$$\mathfrak{v}(X) = \sum_{j=0}^{N-1} f\left(\frac{\lfloor pj/(2N)\rceil \bmod p}{p}\right) X^j ,$$

we remark that the resulting polynomial $X^{-\tilde{\mu}^*} \cdot \mathfrak{v}(X)$ has for constant term $f\left(\frac{\lfloor p\tilde{\mu}^*/(2N)\rceil \bmod p}{p}\right) = f(\mu)$, assuming the absence of drift impact and $0 \le (\tilde{\mu}^* \bmod 2N) < N$. Under these conditions, on input a (noisy) TLWE ciphertext $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu)$, the above procedure (cf. Section 5.2.4) outputs a TLWE ciphertext $\boldsymbol{c'} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(f(\mu))$ featuring a small amount of noise. Observe that the regular bootstrapping corresponds to the identity function for f.

We note that the range restriction on $\tilde{\mu}^*$ can be suppressed when function f is negacyclic (i.e., if $f(\mu + \frac{1}{2}) = -f(\mu)$, $\forall \mu \in \mathbb{T}_p$). The "sign" function over the torus is an example of negacyclic function.

References

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014. doi:10.1007/978-3-662-44371-2_17.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. doi:10.1515/jmc-2015-0016.

[BBKS07] Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. Multi-recipient encryption schemes: How to save on bandwidth and computation without sacrificing security.

IEEE Transactions on Information Theory, 53(11):3927–3943, 2007. doi:10.1109/TIT.2007.907471.

[BBS03] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemes. In Y. Desmedt, editor, *Public Key Cryptography (PKC 2003)*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2003. doi:10.1007/3-540-36288-6_7.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3):13:1–13:36, 2014. doi:10.1145/2633600.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584. ACM Press, 2013. doi:10.1145/2488608.2488680.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012. doi:10.1007/978-3-642-32009-5_50.

[CGGI20] 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. doi:10.1007/s00145-019-09319-x.

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 513–536. Springer, 2015. doi:10.1007/978-3-662-46800-5_20.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015. doi:10.1007/978-3-662-46800-5_24.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, 2009. doi:10.1145/1536414.1536440.

[Gen10] Craig Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010. doi:10.1145/1666420.1666444.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013. doi:10.1007/978-3-642-40041-4_5.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. doi:10.1007/978-3-642-13190-5_1.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. doi:10.1007/s10623-014-9938-4.

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Detouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 165–179. Academic Press, 1978. Available at https://people.csail.mit.edu/rivest/pubs.html#RAD78.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, 2005. doi:10.1145/1060590.1060603.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. doi:10.1145/1568318.1568324.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Y. Ishai, editor, *Theory of Cryptography (TCC 2011)*, volume 6597 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2011. doi:10.1007/978-3-642-19571-6_14.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, 2009. doi:10.1007/978-3-642-10366-7_36.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013. doi:10.1017/CBO9781139856065.

A From Private Key to Public Key

As described in Sections 2 and 3, TLWE and TGLWE are private-key encryption schemes. This is not a restriction because, as demonstrated in [Rot11], any additively homomorphic private-key encryption scheme can be converted into a public-key encryption scheme. In this appendix, we expand on how to extend TLWE and TGLWE to the public-key setting.

Let $\mu \in \mathcal{P}$. We noticed in Section 4.1.3 that the encryption of μ using the private-key TLWE encryption scheme (Section 2.1) can be put under the form

$$\mathrm{TLWE}_{\boldsymbol{s}}(\mu) \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(0) + (0, \dots, 0, \mu) \enspace .$$

Only the first part—i.e., $\mathrm{TLWE}_{\boldsymbol{s}}(0)$—involves the private key $\boldsymbol{s}$.

Now consider m private-key TLWE encryptions of '0'. TLWE encryption being additively homomorphic, any linear combination of these encryptions of '0' is also a private-key TLWE encryption of '0' (provided that the resulting noise keeps "small"). This leads to a [public-key] version of TLWE encryption. The public key is $pk = \boldsymbol{Z}$, a $m \times (n+1)$ matrix whose rows are private-key TLWE encryptions of 0. The [public-key] encryption of $\mu \in \mathcal{P}$ is then obtained by adding together a random subset of the encryptions of 0 present in the public key $\boldsymbol{Z}$ and adding to it $(0, \dots, 0, \mu)$. Specifically, the *public-key* encryption of μ is given by $\mathrm{TLWE}_{pk}(\mu) = \boldsymbol{r} \cdot \boldsymbol{Z} + (0, \dots, 0, \mu)$ where $\boldsymbol{r} \xleftarrow{\$} \mathbb{B}^m$.

More formally:

Public-key TLWE Encryption

KeyGen(1^λ) On input security parameter λ, define two positive integers m, n, select positive integers p and q such $p \mid q$, and define a discretized error distribution $\hat{\chi}$ over $q^{-1}\mathbb{Z}$ induced by a normal distribution $\chi = \mathcal{N}(0, \sigma^2)$ over $\mathbb{R}$. Sample uniformly at random a vector $\boldsymbol{s} = (s_1, \dots, s_n) \xleftarrow{\$}$

$\mathbb{B}^n$. The plaintext space is $\mathcal{P} = \mathbb{T}_p \subset \mathbb{T}_q$ where $\mathbb{T}_q = q^{-1}\mathbb{Z}/\mathbb{Z}$. Using $\boldsymbol{s}$, randomly generate m [private-key] TLWE encryptions of 0 (see Section 2.1), and form the corresponding matrix

$$\boldsymbol{Z} \leftarrow \begin{pmatrix} \mathrm{TLWE}_{\boldsymbol{s}}(0) \\ \vdots \\ \mathrm{TLWE}_{\boldsymbol{s}}(0) \end{pmatrix} \quad (\in \mathbb{T}_q{}^{m\times(n+1)}) \ .$$

The public parameters are $pp = \{m, n, \sigma, p, q\}$, the public key is $pk = \boldsymbol{Z}$, and the private key is $sk = \boldsymbol{s}$.

$\mathrm{Encrypt}_{pk}(\mu)$ The [public-key] encryption of $\mu \in \mathcal{P}$ is given by

$$\boldsymbol{c} = \boldsymbol{r} \cdot \boldsymbol{Z} + (0, \dots, 0, \mu) \in \mathbb{T}_q^{n+1}$$

for a random vector $\boldsymbol{r} \xleftarrow{\$} \mathbb{B}^m$.

$\mathrm{Decrypt}_{sk}(\boldsymbol{c})$ To decrypt $\boldsymbol{c} = (a_1, \dots, a_n, b)$, using secret decryption key $\boldsymbol{s} = (s_1, \dots, s_n)$, compute (in $\mathbb{T}_q$)

$$\mu^* = b - \textstyle\sum_{j=1}^{n} s_j \cdot a_j$$

and return closest plaintext $\mu \in \mathcal{P}$ as the decryption of $\boldsymbol{c}$.

The public-key variant of private-key TGLWE encryption (see Section 3.1) is obtained analogously. We present it below for completeness. The key observations are that (i) for $\mu \in \mathcal{P}_N[X]$ we have $\mathrm{TGLWE}_{\mathfrak{s}}(\mu) \equiv \mathrm{TGLWE}_{\mathfrak{s}}(0) + (0, \dots, 0, \mu)$—see Section 4.2.3, and (ii) private-key TGLWE encryption is additively homomorphic.

Public-key TGLWE Encryption

$\mathrm{KeyGen}(1^\lambda)$ On input security parameter λ, define integers N, k, m with N a power of 2 and $m, k \geq 1$. Select positive integers p and q such $p \mid q$. Define also a discretized error distribution $\hat{\chi}$ over $q^{-1}\mathbb{Z}_N[X]$ induced by a normal distribution $\chi = \mathcal{N}(0, \sigma^2)$ over $\mathbb{R}_N[X]$. Sample uniformly at random a vector $\mathfrak{s} = (\mathfrak{s}_1, \dots, \mathfrak{s}_k) \xleftarrow{\$} \mathbb{B}_N[X]^k$. Using $\mathfrak{s}$, randomly generate m [private-key] TGLWE encryptions of 0 (see Section 3.1), and form the corresponding matrix

$$\mathfrak{Z} \leftarrow \begin{pmatrix} \mathrm{TGLWE}_{\mathfrak{s}}(0) \\ \vdots \\ \mathrm{TGLWE}_{\mathfrak{s}}(0) \end{pmatrix} \quad (\in \mathbb{T}_{N,q}[X]^{m\times(k+1)}) \ .$$

The plaintext space is $\mathcal{P}_N[X] \subset \mathbb{T}_{N,q}[X]$ where $\mathbb{T}_q = q^{-1}\mathbb{Z}/\mathbb{Z}$. The public parameters are $pp = \{m, k, N, \sigma, p, q\}$, the public key is $pk = \mathfrak{Z}$, and the private key is $sk = \mathfrak{s}$.

Encrypt$_{pk}(\mu)$ The [public-key] encryption of $\mu \in \mathcal{P}_N[X]$ is given by

$$\boldsymbol{c} = \boldsymbol{r} \cdot \mathfrak{Z} + (0, \ldots, 0, \mu) \in \mathbb{T}_{N,q}[X]^{k+1}$$

for a random vector $\boldsymbol{r} \xleftarrow{\$} \mathbb{B}_N[X]^m$.

Decrypt$_{sk}(\boldsymbol{c})$ To decrypt $\boldsymbol{c} = (a_1, \ldots, a_n, b)$, using secret decryption key $\mathfrak{s} = (s_1, \ldots, s_n)$, compute (in $\mathbb{T}_{N,q}[X]$)

$$\mu^* = b - \sum_{j=1}^{k} s_j \cdot a_j$$

and return the closest plaintext $\mu \in \mathcal{P}_N[X]$ as the decryption of $\boldsymbol{c}$.

B Pseudo-Code

CMux

Input: 1) $\boldsymbol{c_0}, \boldsymbol{c_1} \in \mathbb{T}_{N,q}[X]^{k+1}$
2) $\mathfrak{K} \in \mathbb{T}_{N,q}[X]^{(k+1)\ell\times(k+1)}$ where $\mathfrak{K} \leftarrow \mathrm{TGGSW}_{\mathfrak{s}}(\mathrm{b})$ with $\mathrm{b} \in \{0, 1\}$ and $\mathfrak{s} \in \mathbb{B}_N[X]^k$

Output: $\boldsymbol{c}' \leftarrow \mathrm{CMux}(\mathfrak{K}, \boldsymbol{c_0}, \boldsymbol{c_1}) \in \mathbb{T}_{N,q}[X]^{k+1}$

$\boldsymbol{c}' \leftarrow \mathfrak{K} \boxdot (\boldsymbol{c_1} - \boldsymbol{c_0}) + \boldsymbol{c_0}$

return $\boldsymbol{c}'$

BlindRotate

Input: 1) $\mathcal{c} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu) \in \mathbb{T}_{N,q}[X]^{k+1}$
2) $\tilde{\boldsymbol{c}} = (\tilde{a}_1, \dots, \tilde{a}_n, \tilde{b}) \in (\mathbb{Z}/2N\mathbb{Z})^{n+1}$
3) $\mathbf{bsk} = (\mathrm{bsk}[1], \dots, \mathrm{bsk}[n]) \in \left(\mathbb{T}_{N,q}[X]^{(k+1)\ell \times (k+1)}\right)^n$
where $\mathrm{bsk}[j] \leftarrow \mathrm{TGGSW}_{\mathfrak{s}}(s_j)$ with $\mathfrak{s} \in \mathbb{B}_N[X]^k$
and $\boldsymbol{s} = (s_1, \dots, s_n) \in \mathbb{B}^n$

Output: $\mathcal{c}' \leftarrow \mathrm{BlindRotate}_{\mathbf{bsk}}(\mathcal{c}, \tilde{\boldsymbol{c}}) \in \mathbb{T}_{N,q}[X]^{k+1}$

$\mathcal{c}' \leftarrow X^{-\tilde{b}} \cdot \mathcal{c}$
for $j = 1$ **to** n **do**
  $\mathcal{c}' \leftarrow \mathrm{CMux}(\mathrm{bsk}[j], \mathcal{c}', X^{\tilde{a}_j} \cdot \mathcal{c}')$
end for

return $\mathcal{c}'$

SampleExtract

Input: $\mathcal{c} \leftarrow \mathrm{TGLWE}_{\mathfrak{s}}(\mu) = (\mathcal{a}_1, \dots, \mathcal{a}_k, \mathcal{b}) \in \mathbb{T}_{N,q}[X]^{k+1}$
with $\mathcal{a}_j(X) = (a_j)_0 + (a_j)_1 X + \cdots + (a_j)_{N-1} X^{N-1}$ for
$1 \le j \le k$ and $\mathcal{b}(X) = b_0 + b_1 X + \cdots + b_{N-1} X^{N-1}$,
and where $\mu(X) = \mu_0 + \cdots + \mu_{N-1} X^{N-1} \in \mathcal{P}_N[X]$

Output: $\boldsymbol{c}' \leftarrow \mathrm{SampleExtract}(\mathcal{c}) \in \mathbb{T}_q^{kN+1}$

$\boldsymbol{a}' \leftarrow \big((a_1)_0, -(a_1)_{N-1}, \dots, -(a_1)_1,$
$\quad \dots, \dots, (a_k)_0, -(a_k)_{N-1}, \dots, -(a_k)_1\big)$
$\boldsymbol{c}' \leftarrow (\boldsymbol{a}', b_0)$

return $\boldsymbol{c}'$

Recode

Input: $\mathfrak{s} = (\mathfrak{s}_1, \dots, \mathfrak{s}_k) \in \mathbb{B}_{N,q}[X]^k$ with
$\mathfrak{s}_j(X) = (s_j)_0 + (s_j)_1 X + \cdots + (s_j)_{N-1} X^{N-1}$ for $1 \le j \le k$

Output: $\boldsymbol{s}' \leftarrow \mathrm{Recode}(\mathfrak{s}) \in \mathbb{B}^{kN}$

$\boldsymbol{s}' \leftarrow \big((s_1)_0, (s_1)_1, \dots, (s_1)_{N-1},$
$\quad \dots, \dots, (s_k)_0, (s_k)_1, \dots, (s_k)_{N-1}\big)$

return $\boldsymbol{s}'$

KeySwitch

Input: 1) $\boldsymbol{c} \leftarrow \mathrm{TLWE}_{\boldsymbol{s}}(\mu) = (a_1, \dots, a_n, b) \in \mathbb{T}_q^{n+1}$
with $\boldsymbol{s} = (s_1, \dots, s_n) \in \mathbb{B}^n$

2) $\mathbf{ksk} = (\mathrm{ksk}[i,j])_{\substack{1 \le i \le n \\ 1 \le j \le \ell}}$ with $\mathrm{ksk}[i,j] \in \mathbb{T}_q^{n'+1}$
where $\mathrm{ksk}[i,j] \leftarrow \mathrm{TLWE}_{\boldsymbol{s'}}(s_i \cdot B^{-j})$ with $\boldsymbol{s'} \in \mathbb{B}^{n'}$

Output: $\boldsymbol{c'} \leftarrow \mathrm{KeySwitch}_{\mathbf{ksk}}(\boldsymbol{c}) \in \mathbb{T}_q^{n'+1}$

$\boldsymbol{c'} \leftarrow (0, \dots, 0, b)$
for $i = 1$ **to** n **do**
  $((\bar{a})_1, \dots, (\bar{a})_\ell) \leftarrow g^{-1}(a_i)$
  $\boldsymbol{d'} \leftarrow (\bar{a})_1 \cdot \mathrm{ksk}[i,1]$
  for $j = 2$ **to** ℓ **do** $\boldsymbol{d'} \leftarrow \boldsymbol{d'} + (\bar{a})_j \cdot \mathrm{ksk}[i,j]$
  $\boldsymbol{c'} \leftarrow \boldsymbol{c'} - \boldsymbol{d'}$
end for

return $\boldsymbol{c'}$

C Index to Notations

In the following notations, letters have the following significance:

N	a power of two;
q	ciphertext modulus, $q = 2^{\Omega}$ where Ω is the bit-precision for the representation;
p	plaintext modulus such that $p \mid q$.

Formal symbolism	Meaning	Section reference
$\mathbb{B}$	$\mathbb{B} = \{0, 1\}$	Section 1.3
$\mathbb{B}_N[X]$	$\mathbb{B}_N[X] = \mathbb{B}[X]/(X^N + 1)$	Section 1.3
$\mathcal{P}$	$\mathcal{P} = \mathbb{T}_p$ (plaintext space)	Section 2.1
$\overline{\mathcal{P}}$	$\overline{\mathcal{P}} = \mathbb{Z}/p\mathbb{Z}$	Section 4.1.3
$\mathcal{P}_N[X]$	$\mathcal{P}_N[X] = \mathcal{P}[X]/(X^N + 1)$	Section 3.1
$\overline{\mathcal{P}}_N[X]$	$\overline{\mathcal{P}}_N[X] = \overline{\mathcal{P}}[X]/(X^N + 1)$	Section 4.2.3
$\mathbb{R}_N[X]$	$\mathbb{R}_N[X] = \mathbb{R}[X]/(X^N + 1)$	Section 1.1
$\mathbb{T}$	$\mathbb{T} = \mathbb{R}/\mathbb{Z}$ (real torus)	Section 1.1
$\mathbb{T}_q$	$\mathbb{T}_q = \frac{1}{q}\mathbb{Z}/\mathbb{Z}$ (discretized torus)	Section 2.1
$\mathbb{T}_N[X]$	$\mathbb{T}_N[X] = \mathbb{T}[X]/(X^N + 1)$	Section 1.1
$\mathbb{T}_{N,q}[X]$	$\mathbb{T}_{N,q}[X] = \mathbb{T}_q[X]/(X^N + 1)$	Section 3.1
$\mathbb{Z}_N[X]$	$\mathbb{Z}_N[X] = \mathbb{Z}[X]/(X^N + 1)$	Section 1.1