

FULLY HOMOMORPHIC ENCRYPTION BASED ON NON-CYCLOTOMIC RINGS

TECHNICAL FIELD

5 The presently disclosed subject matter relates to data security, and in particular to methods of encryption and decryption.

BACKGROUND

Problems of performing fully homomorphic encryption and decryption have been recognized in the conventional art and various techniques have been developed to provide
10 solutions.

SUMMARY

According to one aspect of the presently disclosed subject matter there is a processing circuitry (PC)-based method of fully homomorphic encryption, the method comprising:

15 encrypting an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

20

b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

a. the third non-cyclotomic ring, and

b. a second ideal of the third non-cyclotomic ring,

25

wherein the second ideal is not the first ideal.

In addition to the above features, the method according to this aspect of the presently disclosed subject matter can comprise one or more of features (i) to (xxii) listed below, in any desired combination or permutation which is technically possible:

5

- (i) the irreducible non-cyclotomic polynomial is one of:

$$f(x) = x^n + x - b, \text{ or}$$

$$f(x) = x^n - x + b;$$

10

and the first ideal is:

$$x - b$$

wherein b is an integer.

- (ii) the method further comprising, prior to the encrypting:

15

encoding a given plaintext, of a given plaintext space, to the element of the first non-cyclotomic ring, the encoding being based on:

EncodedElement =

20

$$\left(\sum_{i=0}^{\text{Logb}(m)} m_i x^i \right) \bmod \text{FirstIdeal}$$

wherein EncodedElement denotes the element of the first non-cyclotomic ring, FirstIdeal denotes the first ideal, and

25

wherein m denotes the given plaintext, and wherein m_i is based on:

$$m =$$

$$\sum_{i=0}^{\text{Logb}(m)} m_i b^i,$$

and wherein $f(b)$ is equivalent to a size of the given plaintext space.

(iii) the third non-cyclotomic ring is an order of a field, the field being an extension of a fourth non-cyclotomic ring that is derivative of the irreducible non-cyclotomic polynomial.

(iv) the third non-cyclotomic ring is a final extension ring of a series of extension rings, the series of extension rings successively extending the fourth non-cyclotomic ring, the series of extension rings being of a given series length,

wherein each successive extension ring is based on an equation:

$$\text{CurrentNonCyclotomicExtensionRing} = \text{PredecessorNonCyclotomicRing}[t]/\text{MinimalPolynomial}$$

wherein $\text{PredecessorNonCyclotomicRing}$ denotes a respective immediately preceding extension ring of the series, t is a respective additional algebraic element, and

wherein MinimalPolynomial denotes a respective minimal polynomial, the respective minimal polynomial being based on an equation:

$$\text{MinimalPolynomial} =$$

$$\left(\prod_{j=0}^n (t - a_j) + (c(t) * b) + (d(t) * \text{FirstIdeal}) \right)$$

wherein FirstIdeal denotes the first ideal, and wherein each a_j is a unique element of a ring that is a quotient of $\text{PredecessorNonCyclotomicRing}$ and the first ideal, n is a respective given integer greater than 1,

and wherein $c(t)$ and $d(t)$ are polynomials in
 PredecessorNonCyclotomicRing[t], and the minimal polynomial is
 irreducible in PredecessorNonCyclotomicRing.

- 5 (v) the given series length is 1, and wherein the final extension ring of the
 series of final extensions rings is based on:

CurrentNonCyclotomicExtensionRing =
 FourthNonCyclotomicRing[t]/MinimalPolynomial

10 where FourthNonCyclotomicRing denotes the fourth non-cyclotomic ring.

- (vi) the minimal polynomial defining the final extension ring is one of:
- a. $t_n^2 + t_n + x$, or
 - b. $t_n^2 - t_n + x$, or
 - 15 c. $t_n^2 + t_n + x$, or
 - d. $t_n^2 - t_n + x$,

wherein t_n denotes the additional algebraic element of the final extension ring
 of the series of extension rings.

- (vii) the minimal polynomial defining the final extension ring is one of:
- 20 a. $t_n^2 + t_n + bt_{n-1}$, or
 - b. $t_n^2 - t_n + bt_{n-1}$, or
 - c. $t_n^2 + t_n - bt_{n-1}$, or
 - d. $t_n^2 - t_n - bt_{n-1}$,

25 wherein t_n denotes the additional algebraic element of the final extension ring of
 the series of extension rings, and

wherein t_{n-1} denotes the additional algebraic element of the extension ring, of the
 series of extension rings, immediately preceding the final extension ring.

(viii) the method further comprising, prior to the encrypting, encoding a given number of plaintexts to an element of the first non-cyclotomic ring, wherein the encoding comprises:

5

- a) generating, for each plaintext of the given number of plaintexts, a respective per-plaintext ring element of the first non-cyclotomic ring, the generating being based on:

PerPlaintextRingElement =

$$(\sum_{i=0}^{\text{Logb}(m)} m_i x^i) \bmod \text{FirstIdeal}$$

wherein PerPlaintextRingElement denotes the element of the first non-cyclotomic ring, and wherein m denotes the given plaintext, and wherein m_i is based on:

15

m =

$$\sum_{i=0}^{\text{Logb}(m)} m_i b^i,$$

and wherein f(b) is equivalent to a size of the given plaintext space; and

- b) calculating an encoded element based on the formula:

EncodedElement_{c0} =

$$\left(\sum_{i=0}^{n-1} \text{PerPlaintextRingElement}_i \cdot \lambda_i \right) \bmod \text{FirstIdeal}$$

wherein EncodedElement_{c0} denotes the element of the first non-cyclotomic ring, n denotes the given number of plaintexts, PerPlaintextRingElement_i denotes a respective per-plaintext ring element, and λ_i denotes a respective Lagrange coefficient.

25

(ix) the given number of plaintexts is:

$$\prod_{\{i=0\}}^{SeriesLength} (d_i)$$

wherein SeriesLength is the given series length of the series of extension rings,
and d_i is a respective degree of the respective minimal polynomial of the
respective extension ring.

(x) the encrypting the element of a first non-cyclotomic ring comprises:

calculating a linear combination, over the second non-cyclotomic ring of, at least:

- i. the element of the first non-cyclotomic ring,
- ii. a sum of, at least:

- a. a product of, at least, an encryption key and a randomizer, and
- b. a noise value,

the calculating thereby resulting in an element of the second non-cyclotomic ring.

(xi) the encryption key is an element of the second non-cyclotomic ring, the
noise value is sampled from the first ideal, and the randomizer is sampled
from a distribution over a ring with a bounded expected norm.

(xii) the encrypting is based on:

EncryptedElement_{c0} =

(PlaintextElement + ((Randomizer * EncryptionKey) + NoiseValue) mod
(ThirdNonCyclotomicRing / SecondIdeal))

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring,
SecondIdeal denotes the second ideal, PlaintextElement denotes the element of
the first non-cyclotomic ring, EncryptionKey denotes the encryption key,
NoiseValue denotes the noise value, and Randomizer denotes the randomizer.

(xiii) the encrypting is based on:

EncryptedElement_{c0} =

$$(\text{PlaintextElement} + ((\text{Randomizer} * ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3}))$$

5 mod (ThirdNonCyclotomicRing /SecondIdeal)

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer

10 denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

(xiv) the encrypting is based on:

15

EncryptedElement_{c0} =

$$((\text{PlaintextElement} * \frac{\text{SecondIdeal}}{\text{FirstIdeal}}) +$$

$$(\text{Randomizer} * ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue})) + \text{Randomizer3})$$

20 mod (ThirdNonCyclotomicRing /SecondIdeal)

wherein the first ideal and second ideal are principal ideals,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, Randomizer denotes the randomizer, and

25

wherein ConstantValue is a constant value that is an element of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

(xv) the encrypting is based on:

5

EncryptedElement_{c0} =

$$\begin{aligned} & ((\text{PlaintextElement} * \text{SecondIdeal}) + \\ & ((\text{Randomizer} * (\text{ConstantValue} * \text{EncryptionKey})) + \text{NoiseValue}) \\ & + \text{Randomizer3}) \end{aligned}$$

10

$$\text{mod } (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))$$

wherein the first ideal is not a principal ideal, and the second ideal is a principal ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

15

wherein ConstantValue is a constant value that is an element of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

20

(xvi) the encrypting is based on:

EncryptedElement_{c0} =

(PlaintextElement +

5 (Randomizer * (ConstantValue * EncryptionKey) + NoiseValue) +
Randomizer3)

mod (ThirdNonCyclotomicRing / (SecondIdeal * FirstIdeal))

wherein the element of the first non-cyclotomic ring is in the second ideal,
the first ideal is a principal ideal, and the second ideal is not a principal
10 ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring,
FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal,
PlaintextElement denotes the element of the first non-cyclotomic ring,
EncryptionKey denotes the encryption key, NoiseValue denotes the noise
15 value, and Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of the second
non-cyclotomic ring, and wherein Randomizer3 is sampled from a
distribution over a ring with a bounded expected norm.

20 (xvii) the encryption key is an element of a module derivative of the second non-
cyclotomic ring, the noise value is sampled from the first ideal, and the
randomizer is sampled from a distribution over a module with a bounded
expected norm.

(xviii) the encrypting is based on:

$$\begin{aligned} \text{EncryptedElement}_{c0} = \\ & (\text{PlaintextElement} + ((\text{Randomizer} * \text{EncryptionKey}) + \text{NoiseValue}) \bmod \\ 5 & (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer.

10 (xix) the encrypting is based on:

$$\begin{aligned} \text{EncryptedElement}_{c0} = \\ & (\text{PlaintextElement} + ((\text{Randomizer} * ((\text{ConstantValue} * \\ & \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3})) \\ & \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

15 wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

20 wherein ConstantValue is a constant value that is an element of a module that is derivative the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

(xx) the encrypting is based on:

EncryptedElement_{c0} =

$$\begin{aligned}
 & ((\text{PlaintextElement} * \frac{\text{SecondIdeal}}{\text{FirstIdeal}}) + \\
 & (\text{Randomizer} * ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue})) \\
 & + \text{Randomizer3}) \\
 & \text{mod } (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})
 \end{aligned}$$

wherein the first ideal and second ideal are principal ideals,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of a module that is derivative of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

(xxi) the encrypting is based on:

20

EncryptedElement_{c0} =

$$\begin{aligned}
 & ((\text{PlaintextElement} * \text{SecondIdeal}) + \\
 & ((\text{Randomizer} * (\text{ConstantValue} * \text{EncryptionKey})) + \text{NoiseValue}) \\
 & + \text{Randomizer3}) \\
 & \text{mod } (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))
 \end{aligned}$$

25

wherein the first ideal is not a principal ideal, and the second ideal is a principal ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of a module that is derivative of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

(xxii) the encrypting is based on:

EncryptedElement_{c0} =

$$\begin{aligned} & (\text{PlaintextElement} + \\ & (\text{Randomizer} * (\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \\ & \text{Randomizer3}) \\ & \text{mod } (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal})) \end{aligned}$$

wherein the element of the first non-cyclotomic ring is in the second ideal, the first ideal is a principal ideal, and the second ideal is not a principal ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of a module that is derivative of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

- 5 According to another aspect of the presently disclosed subject matter there is provided a system of fully homomorphic encryption, the system comprising a processing circuitry (PC) configured to:

encrypt an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

- 10 wherein the first non-cyclotomic ring is a quotient of:

- a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and
- b. a first non-trivial ideal of the third non-cyclotomic ring, and

- 15 wherein the second non-cyclotomic ring is a quotient of:

- a. the third non-cyclotomic ring, and
- b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

20

This aspect of the disclosed subject matter can further optionally comprise one or more of features (i) to (xxii) listed above with respect to the method, mutatis mutandis, in any desired combination or permutation which is technically possible.

- 25 According to another aspect of the presently disclosed subject matter there is provided a computer program product comprising a computer readable non-transitory storage medium containing program instructions, which program instructions when read by

a processor, cause the processing circuitry to perform a method of fully homomorphic encryption, the method comprising:

encrypting an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

5 wherein the first non-cyclotomic ring is a quotient of:

a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

b. a first non-trivial ideal of the third non-cyclotomic ring, and

10 wherein the second non-cyclotomic ring is a quotient of:

a. the third non-cyclotomic ring, and

b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

15 This aspect of the disclosed subject matter can further optionally comprise one or more of features (i) to (xxii) listed above with respect to the method, mutatis mutandis, in any desired combination or permutation which is technically possible.

|According to another aspect of the presently disclosed subject matter there is a
20 processing circuitry (PC)-based method of decrypting fully homomorphically encrypted data, the method comprising:

decrypting an element of a second non-cyclotomic ring to an element of a first non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

25 a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

b. a first non-trivial ideal of the third non-cyclotomic ring, and
wherein the second non-cyclotomic ring is a quotient of:

- a. the third non-cyclotomic ring, and
- b. a second ideal of the third non-cyclotomic ring,

5

wherein the second ideal is not the first ideal.

In addition to the above features, the method according to this aspect of the presently disclosed subject matter can comprise one or more of features (i) to (xvi) listed below, in any desired combination or permutation which is technically possible:

10

- (i) the irreducible non-cyclotomic polynomial is one of:
 $f(x) = x^n + x - b$, or

$$f(x) = x^n - x + b;$$

15

and the first ideal is:

$$x-b$$

wherein b is an integer.

(ii) the method further comprising, subsequent to the decrypting:

decoding the element of the first non-cyclotomic ring to a plaintext of a given plaintext space.

5

(iii) the decoding is based on calculating:

$f_{\text{element}}(b)$

wherein $f_{\text{element}}()$ is a polynomial function corresponding to the element of the first non-cyclotomic ring.

10

(iv) the first non-cyclotomic ring is a quotient ring based on a final ring of a series of successive rings extending a fourth non-cyclotomic ring, and wherein the decoding comprises:

15

a) determining one or more roots of a minimal polynomial associated with the first non-cyclotomic ring;

b) for each determined root:

20

calculating a result of substituting, in the element of the first non-cyclotomic ring, a respective extending algebraic element with the respective root,

thereby generating one or more elements of a quotient ring derivative of a preceding ring of the series;

25

c) responsive to the preceding ring being an extension ring of the fourth non-cyclotomic ring:

for each generated element: repeating a) - b); and

d) responsive to the preceding ring being the fourth non-cyclotomic ring:

calculating, for each generated element $f_{\text{element}}()$;

$f_{\text{element}}(b)$

thereby giving rise to one or more plaintexts.

- 5 (v) the decrypting the element of a first non-cyclotomic ring comprises subtracting, from the element of the second non-cyclotomic ring:

a value based on a product of a decryption key and a randomizing element,

- 10 the decryption key and the randomizing element being elements of the second non-cyclotomic ring,

thereby resulting in an element of the first non-cyclotomic ring.

- (vi) the decrypting is based on:

- 15
$$\text{DecryptedElement} = \text{EncryptedElement}_{c0} - (\text{DecryptionKey} * \text{EncryptedElement}_{c1}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, EncryptedElement_{c0} denotes the element of the second non-cyclotomic ring, EncryptedElement_{c1} denotes the randomizing element, DecryptedElement denotes the element of the first non-cyclotomic ring, and DecryptionKey denotes the decryption key.

20

- (vii) the decryption key is a symmetric key.
- (viii) the decryption key is a private key.
- (ix) the randomizing element is derivative of at least one randomizer.
- 25 (x) the randomizing element is further derivative of at least one constant value.

- (xi) the decrypting the element of a first non-cyclotomic ring comprises subtracting, from the element of the second non-cyclotomic ring:

5 a value based on a product of a decryption key and a randomizing element,

the decryption key and the randomizing element being elements of a module derivative of the second non-cyclotomic ring,

- (xii) the decrypting is based on:

$$10 \quad \text{DecryptedElement} = \text{EncryptedElement}_{c0} - (\text{DecryptionKey} * \text{EncryptedElement}_{c1}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, EncryptedElement_{c0} denotes the element of the second non-cyclotomic ring, EncryptedElement_{c1} denotes the randomizing element, DecryptedElement denotes the element of the first non-cyclotomic ring, 15 and DecryptionKey denotes the decryption key.

- (xiii) the decryption key is a symmetric key.
 (xiv) the decryption key is a private key.
 (xv) The randomizing element is derivative of at least one randomizer.
 (xvi) the randomizing element is further derivative of at least one constant 20 value.

According to another aspect of the presently disclosed subject matter there is provided a system of decrypting fully homomorphically encrypted data, the system comprising a processing circuitry (PC) configured to:

25 decrypt an element of a second non-cyclotomic ring to an element of a first non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

- a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

a. the third non-cyclotomic ring, and

5 b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

10 This aspect of the disclosed subject matter can further optionally comprise one or more of features (i) to (xvi) listed above with respect to the method, mutatis mutandis, in any desired combination or permutation which is technically possible.

According to another aspect of the presently disclosed subject matter there is provided a computer program product comprising a computer readable non-transitory
15 storage medium containing program instructions, which program instructions when read by a processor, cause the processing circuitry to perform a method of fully homomorphic encryption, the method comprising:

encrypting an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

20 wherein the first non-cyclotomic ring is a quotient of:

a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

b. a first non-trivial ideal of the third non-cyclotomic ring, and

25 wherein the second non-cyclotomic ring is a quotient of:

a. the third non-cyclotomic ring, and

b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

This aspect of the disclosed subject matter can further optionally comprise one or more of features (i) to (xvi) listed above with respect to the method, mutatis mutandis, in any desired combination or permutation which is technically possible.

5

BRIEF DESCRIPTION OF THE DRAWINGS

In order to understand the invention and to see how it can be carried out in practice, embodiments will be described, by way of non-limiting examples, with reference to the accompanying drawings, in which:

10 **Fig. 1** illustrates an example deployment of fully-homomorphic encryption/decryption, in accordance with some embodiments of the presently disclosed subject matter;

Fig. 2A illustrates a logical block diagram of an example computer system enabled for performance of fully homomorphic encryption (FHE), in accordance with
15 some embodiments of the presently disclosed subject matter;

Fig. 2B illustrates a logical block diagram of an example deployment of a computer system enabled for decryption of data encrypted using FHE, in accordance with some embodiments of the presently disclosed subject matter;

20 **Fig. 3** illustrates a flow diagram of an example method of ring-based fully-homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter;

Fig. 4 illustrates a flow diagram of an example method of module-based fully-homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter; and

25 **Fig. 5** illustrates a flow diagram of an example method of decrypting data that was encrypted using ring-based or module-based fully-homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those skilled in the art that the presently disclosed subject matter may be practiced without these specific details. In other instances, well-known methods, procedures, components and circuits have not been described in detail so as not to obscure the presently disclosed subject matter.

Unless specifically stated otherwise, as apparent from the following discussions, it is appreciated that throughout the specification discussions utilizing terms such as "processing", "computing", "comparing", "encrypting", "decrypting", "determining", "calculating", "receiving", "providing", "obtaining", "emulating" or the like, refer to the action(s) and/or process(es) of a computer that manipulate and/or transform data into other data, said data represented as physical, such as electronic, quantities and/or said data representing the physical objects. The term "computer" should be expansively construed to cover any kind of hardware-based electronic device with data processing capabilities including, by way of non-limiting example, the processor, mitigation unit, and inspection unit therein disclosed in the present application.

The terms "non-transitory memory" and "non-transitory storage medium" used herein should be expansively construed to cover any volatile or non-volatile computer memory suitable to the presently disclosed subject matter.

The operations in accordance with the teachings herein may be performed by a computer specially constructed for the desired purposes or by a general-purpose computer specially configured for the desired purpose by a computer program stored in a non-transitory computer-readable storage medium.

Embodiments of the presently disclosed subject matter are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the presently disclosed subject matter as described herein.

Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data. When mathematical operations are performed on a homomorphically encrypted ciphertext, the result is a ciphertext that, when decrypted, matches the result of identical operations performed on the original plaintext.

5 Certain homomorphic encryption schemes allow limited types of operations (e.g. addition or multiplication only). Fully Homomorphic Encryption (FHE) supports both addition and multiplication operations on ciphertexts, making it theoretically possible to perform any computation on encrypted data.

10 In distributed computing, homomorphic encryption can be valuable because it allows data to be processed by untrusted third parties without exposing sensitive information. For example, in cloud computing environments, users can offload computations to a cloud server without revealing the underlying data. This is useful in privacy-preserving applications such as secure voting systems, confidential machine learning, and secure data outsourcing. By ensuring that the data remains encrypted
15 throughout the process, homomorphic encryption enhances security and privacy while enabling distributed systems to perform necessary computations.

 Some embodiments of the presently disclosed subject matter are directed to methods of fully-homomorphic encryption and decryption based on non-cyclotomic rings. Some advantages of methods based on non-cyclotomic rings are presented in the
20 additional disclosure section below.

Fig. 1 illustrates an example deployment of fully-homomorphic encryption/decryption, in accordance with some embodiments of the presently disclosed subject matter.

25 Encryption system **105** can be a computer system adapted to perform encryption in a manner which enables FHE decryption. An example encryption system **105** is described below, with reference to **Fig. 2A**.

 Communication network **110** can be any kind of suitable network for computer communications (Ethernet, Wi-Fi, 3G wireless network, 4G wireless network, 5G wireless network, Infiniband, etc.). In embodiments, communication network **110**

includes a local area network (LAN), a wide area network (WAN), the Internet, and/or one or more Intranets. Communication network **110** can be operably attached to encryption system **105**, encrypted calculation system **115**, and decryption system **120**.

Encryption calculation system **115** can be a computer system adapted to perform
5 computation on FHE-encrypted data (i.e. without decrypting it first).

Decryption system **120** can be a computer system adapted to perform FHE decryption. An example encryption system **105** is described below, with reference to **Fig. 2B**.

In some examples, encryption system **105** can encrypt data using homomorphic
10 encryption in combination with e.g. a public key that is associated with a private key held by decryption system **120**. Encryption system **105** can e.g. transmit the encrypted data to encrypted calculations system **115**, which performs mathematical operations based on the encrypted data. Encrypted calculations system **115** can transmit the results of the mathematical operations to decryption system **120**, which can then decrypt the encrypted
15 results of the mathematical operations e.g. using its private key.

Fig. 2A illustrates a logical block diagram of an example computer system enabled for performance of fully homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter.

Encryption system (processing circuitry) **200A** can include processor **205A** and
20 memory **210A**.

Processor **205A** can be a suitable hardware-based electronic device with data processing capabilities, such as, for example, a general purpose processor, digital signal processor (DSP), a specialized Application Specific Integrated Circuit (ASIC), one or more cores in a multicore processor, etc. Processor **205A** can also consist, for example,
25 of multiple processors, multiple ASICs, virtual processors, combinations thereof etc.

Memory **210A** can be, for example, a suitable kind of volatile and/or non-volatile storage, and can include, for example, a single physical memory component or a plurality

of physical memory components. Memory **210A** can also include virtual memory. Memory **210A** can be configured to, for example, store various data used in computation.

Encryption system (processing circuitry) **200A** can be configured to execute several functional modules in accordance with computer-readable instructions implemented on a non-transitory computer-readable storage medium. Such functional modules are referred to hereinafter as comprised in the processing circuitry. These modules can include, for example, communications unit **215A**, optional ring-based encryption unit **220A**, optional module-based encryption unit **225A**, and optional encoding unit **230A**. The functional modules may include hardware modules, software modules, firmware modules, or combinations thereof. In some embodiments, the operations described with reference to one or more of the communications unit **215A**, optional ring-based encryption unit **220A**, optional module-based encryption unit **225A**, and optional encoding unit **230A** may be combined into fewer units. In some embodiments, the operations described with reference to one or more of the communications unit **215A**, optional ring-based encryption unit **220A**, optional module-based encryption unit **225A**, and optional encoding unit **230A** may be split up and handled by separate units.

Communications unit **215A** can be any suitable component usable for communicating via communication network **110** (ethernet controller, wi-fi controller etc.). Communications unit **215A** can facilitate e.g. reception of data for encryption and/or transmission of encrypted data.

Optional ring-based encryption unit **220A** can encrypt data using a ring-based fully homomorphic encryption method as detailed herein.

Some embodiments of the presently disclosed subject matter utilize a plaintext ring which is a quotient ring of a non-cyclotomic ring (termed **R**) and an Ideal (termed **I**). Such a plaintext ring (herein termed “**P**”) can be described formally as:

$$P = R/I$$

where **R** is a ring such that:

$$R = \mathbb{Z}[x]/f()$$

Where:

- $\mathbb{Z}[x]$ denotes the set of all polynomials with integer coefficients (and for which the operations of addition and multiplication are defined as polynomial addition and multiplication),
- $f()$ is an irreducible non-cyclotomic polynomial and
- the ideal I is a non-trivial ideal of the ring R (i.e. a proper subset of R satisfying additive closure and absorption, as known in the art).

Some embodiments of the presently disclosed subject matter utilize a ciphertext ring which is a quotient ring of the non-cyclotomic ring R and a second ideal herein termed Q (which is distinct from I). Such a ciphertext ring (herein termed “ C ”) can be described formally as:

$$C = R/Q$$

In some embodiments, the plaintext ring and ciphertext ring are based on the following definitions:

- The irreducible non-cyclotomic polynomial from which the non-cyclotomic ring R is derived is:

$$f(x) = x^n + x - b$$

for some degree n ,

and the Ideal of the non-cyclotomic ring is:

$$x-b$$

where b is an integer.

In some other embodiments, the plaintext ring and ciphertext ring are based on the following definitions:

- The irreducible non-cyclotomic polynomial from which the non-cyclotomic ring R is derived is:

$$f(x) = x^n + x + b$$

for some degree n ,

5 and the Ideal of the non-cyclotomic ring is:

$$x-b$$

where b is an integer.

In some other embodiments, the plaintext ring and ciphertext ring are based on other specifications of $f()$ and I .

10 Some embodiments of the presently disclosed subject matter are directed to methods of encrypting an element of a non-cyclotomic plaintext ring to an element of a non-cyclotomic ciphertext ring.

These methods can employ a e.g. symmetric encryption key or a public key/private key scheme. These methods can also employ one or more randomizers and/or
15 constant values. These methods can also employ noise values.

To facilitate subsequent homomorphic mathematical operations as well as decryption/reconstruction of plaintexts, some embodiments utilize a pair of ciphertext ring elements which can be transmitted together.

In some examples herein, this pair of elements making up a ciphertext are
20 described as two elements in parentheses, e.g.

$$(c_0, c_1)$$

where c_0 identifies a ciphertext ring element that is derivative of an encryption operation utilizing e.g. a plaintext element, an encryption key and other parameters, and where c_1 is a ciphertext ring element that is derivative of e.g. randomizers and/or noise
25 values and/or constant values and/or other parameters.

In some examples herein, the pair of elements making up a ciphertext are described using the terms $\text{EncryptedElement}_{c_0}$ and $\text{EncryptedElement}_{c_1}$ to refer to the respective elements. In some cases herein, $\text{EncryptedElement}_{c_1}$ is referred to as a “randomizing element”.

5 Ring-based encryption unit **220A** can, for example, utilize methods such as the methods described below with reference to **Fig. 3**.

Optional module-based encryption unit **225A** can encrypt data using a module-based fully homomorphic encryption method.

10 A module over a ring R is a generalization of a vector space, where the scalars come from a ring rather than a field.

Some embodiments of the presently disclosed subject matter are directed to encryption/decryption methods which utilize modules as randomizing elements.

In such embodiments, the second element (i.e. c_1) of the (c_0, c_1) ciphertext (i.e. $\text{EncryptedElement}_{c_1}$) is a module.

15 Module-based encryption can have desirable security properties, as detailed below.

Module-based encryption unit **225A** can, for example, utilize methods such as the methods described below with reference to **Fig. 4**.

20 Optional encoding unit **230A** can encode data (e.g. integers, binary data) to elements of a plaintext ring to facilitate encryption. Optional encoding unit **230A** can utilize encoding methods as described below with reference to **Fig. 3**.

It is noted that encryption system (processing circuitry) **200A** can additionally implement some or all of the functions of encryption system (processing circuitry) **200B**.

25 **Fig. 2B** illustrates a logical block diagram of an example deployment of a computer system enabled for decryption of data encrypted using FHE, in accordance with some embodiments of the presently disclosed subject matter.

Decryption system (processing circuitry) **200B** can include processor **205B** and memory **210B**.

Processor **205B** can be a suitable hardware-based electronic device with data processing capabilities, such as, for example, a general purpose processor, digital signal processor (DSP), a specialized Application Specific Integrated Circuit (ASIC), one or
5 more cores in a multicore processor, etc. Processor **205B** can also consist, for example, of multiple processors, multiple ASICs, virtual processors, combinations thereof etc.

Memory **210B** can be, for example, a suitable kind of volatile and/or non-volatile storage, and can include, for example, a single physical memory component or a plurality
10 of physical memory components. Memory **210B** can also include virtual memory. Memory **210B** can be configured to, for example, store various data used in computation.

Decryption system (processing circuitry) **200B** can be configured to execute several functional modules in accordance with computer-readable instructions implemented on a non-transitory computer-readable storage medium. Such functional
15 modules are referred to hereinafter as comprised in the processing circuitry. These modules can include, for example, communications unit **215B**, decryption unit **220B**, and optional decoding unit **225B**.

Decryption unit **220B** can receive ciphertexts (e.g. ring-based or module-based) and can decrypt them to e.g. plaintext ring elements unit **220B** can do this, for example,
20 utilizing methods such as the methods described below with reference to **Fig. 5**.

Decoding unit **225B** can then decode decrypted plaintext ring elements unit to e.g. integer or binary data. Decoding unit **225B** can utilize, for example methods such as those described below with reference to **Fig. 5**.

Attention is directed to **Fig. 3**, which illustrates a flow diagram of an example
25 method of ring-based fully-homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter.

Optionally: in some examples, it is desirable to perform fully homomorphic encryption (and possibly subsequent encrypted mathematical operations) upon plaintexts consisting of integers, binary data etc.

To accomplish this, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can encode (305) such a plaintext. More specifically: encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can map the plaintext to an element of a plaintext ring (e.g. a plaintext ring in accordance with the plaintext ring definition provided above), so as to facilitate performing fully homomorphic encryption on the plaintext ring element.

In some examples, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) encodes a single plaintext to a single element of the plaintext ring. In some other examples, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) encodes multiple plaintexts to a single plaintext ring element.

In some examples, a plaintext consists of a series of binary digits of a plaintext space of given size. For example: a binary plaintext can be of length 8 (i.e. the plaintext size is 256). In such embodiments, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can encode the plaintext to a polynomial element of the plaintext ring (the ring being defined by a polynomial $f()$) based on the formula:

$$\text{EncodedElement} = \sum_{i=0}^{\text{Logb}(m)-1} m_i x^i$$

wherein m denotes the given plaintext, and wherein m_i is based on:

$$m = \sum_{i=0}^{\text{Logb}(m)-1} m_i b^i,$$

and where $f(b)$ is equivalent to the size of the plaintext space, and $\text{Log}_b()$ denotes the logarithm in base b .

Accordingly, when the value of b is 2 (i.e. the first Ideal is $x-2$):

$$\begin{aligned} \text{EncodedElement} = \\ 5 \quad \sum_{i=0}^{\text{Log2}(\text{SizeOfPlaintextSpace})-1} m_i x^i \end{aligned}$$

where m denotes the given plaintext, and wherein m_i is based on:

$$\begin{aligned} m = \\ 10 \quad \sum_{i=0}^{\text{Log2}(\text{SizeOfPlaintextSpace})-1} m_i 2^i \end{aligned}$$

It is noted that in the formula above (i.e. with $b = 2$), the indexes of the polynomial terms are identical to the binary digits of the plaintext.

In some embodiments, multiple plaintexts can be encoded in a single plaintext ring element for encryption.

15 In such embodiments, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can utilize a special case of the non-cyclotomic ring R from which the plaintext ring and ciphertext ring are derived. Specifically, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can utilize a non-cyclotomic ring R that is an order of a field, the field in turn being an extension of a fourth non-cyclotomic
20 ring that is derivative of the irreducible non-cyclotomic polynomial.

In some such embodiments, this ring can be a final ring of a recursive series of extension rings of the fourth non-cyclotomic ring for which:

the irreducible non-cyclotomic polynomial $f(x)$ is either $x^n + x - b$ or $x^n - x + b$,
and

the first ideal (where the plaintext ring is derived by taking the quotient of the fourth non-cyclotomic ring R and the first ideal) is $x-b$ where b is some integer.

- 5 For example, the first extension ring of the fourth non-cyclotomic ring can be based on:

$\text{FourthNonCyclotomicRing}[t]/\text{MinimalPolynomial}$

where t is an algebraic element being added to the fourth non-cyclotomic ring, and where the minimal polynomial can be based on:

10
$$\left(\prod_{j=0}^n (t - a_j) + (c(t) * b) + (d(t) * \text{FirstIdeal}) \right)$$

- and wherein $c(t)$ and $d(t)$ are polynomials in $\text{FourthNonCyclotomicRing}[t]$, and the minimal polynomial is irreducible in the fourth non-cyclotomic ring. In some embodiments, n is an integer greater than one that defines the degree of the minimal polynomial (and determines the number of plaintext slots that will be available). Each a_j can be a unique element of a ring that is a quotient of the fourth non-cyclotomic ring and the first ideal.
- 15

Similarly, subsequent extension rings can be based on:

$\text{PredecessorNonCyclotomicRing}[t_i]/\text{MinimalPolynomial}$

- wherein $\text{PredecessorNonCyclotomicRing}$ denotes the immediately preceding extension ring of the series, and t_i is a respective additional algebraic element being added in this extension.
- 20

In this case, the minimal polynomial of the extension can again be based on:

$$\prod_{j=0}^n (t - a_j) + (c(t) * b) + (d(t) * \text{FirstIdeal})$$

and wherein $c(t)$ and $d(t)$ are polynomials in $\text{PredecessorNonCyclotomicRing}[t]$, and the minimal polynomial is irreducible in the fourth non-cyclotomic ring. n is an integer greater than one that defines the degree of the minimal polynomial (and determines the number of plaintext slots that will be available). Each a_j can be a unique element of a ring that is a quotient of $\text{PredecessorNonCyclotomicRing}$ and the first ideal. In some embodiments, n is an integer greater than one that defines the degree of this particular minimal polynomial.

The series of recursive extensions of the fourth non-cyclotomic ring can have a particular series length (e.g. 5).

The number of plaintext slots available in an element of the final extension ring of the series of extension rings can then be:

$$\prod_{\{i=0\}}^{\text{SeriesLength}} (d_i)$$

wherein SeriesLength is the given series length of the series of extension rings, and d_i is a respective degree of the respective minimal polynomial of the respective extension ring.

In some embodiments, the minimal polynomial defining the final extension ring of the series of extension rings is one of:

- a) $t_n^2 + t_n + x$, or
- b) $t_n^2 - t_n + x$, or
- c) $t_n^2 + t_n + x$, or
- d) $t_n^2 - t_n + x$,

wherein t_n denotes the additional algebraic element of the final extension ring of the series of extension rings.

In some embodiments, the minimal polynomial defining the final extension ring is one of:

- a) $t_n^2 + t_n + bt_{n-1}$, or
- b) $t_n^2 - t_n + bt_{n-1}$, or
- c) $t_n^2 + t_n - bt_{n-1}$, or
- d) $t_n^2 - t_n - bt_{n-1}$,

- 5 wherein t_n denotes the additional algebraic element of the final extension ring of the series of extension rings, and wherein t_{n-1} denotes the additional algebraic element of the extension ring, of the series of extension rings, immediately preceding the final extension ring.

- 10 Encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can encode a given number of plaintexts to a single ring element of the plaintext ring that is the quotient of the final extension ring and the first ideal.

- 15 In some examples, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) encodes a number of plaintexts that is equivalent to the number of plaintext slots available in each ring element (as given by the formula above). In some examples, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) encodes a number of plaintexts that is less than the number of plaintext slots available, and utilizes other values (e.g. 0) in the unused slots.

Encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can encode the plurality of plaintexts based on the following steps:

- 20 a) generating, for each plaintext of the given number of plaintexts, a respective per-plaintext ring element of the plaintext ring, the generating being based on:

PerPlaintextRingElement =

$$\left(\sum_{i=0}^{\text{Logb}(m)} m_i x^i \right) \bmod \text{FirstIdeal}$$

- 25 wherein PerPlaintextRingElement denotes the element of the plaintext ring, and wherein m denotes the given plaintext, and wherein m_i is based on:

$m =$

$$\sum_{i=0}^{\text{Logb}(m)} m_i b^i,$$

and wherein $f(b)$ is equivalent to a size of the given plaintext space; and

b) calculating an encoded element based on the formula:

EncodedElement_{c0} =

$$\left(\sum_{i=0}^{n-1} \text{PerPlaintextRingElement}_i \cdot \lambda_i \right) \bmod \text{FirstIdeal}$$

wherein EncodedElement_{c0} denotes the element of the plaintext ring, n denotes the given number of plaintexts, PerPlaintextRingElement_i denotes a respective per-plaintext ring element, and λ_i denotes a respective Lagrange coefficient.

It is noted that this encoding method can be utilized to encode plaintexts to rings other than the rings defined herein. Similarly, it is noted that this encoding method can be utilized in the context of other applications (e.g. other encryption applications, non-encryption applications).

Encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can obtain (**310**) an encryption key. In some examples, the encryption key is received or derived from a secure key exchange mechanism. In some examples, the encryption key is an element of the ciphertext ring. In some examples, the encryption key is an element of an algebraic structure (e.g. a module) that is derivative of the ciphertext ring.

Encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can optionally obtain one or more randomizer. A randomizer can be a random value (with suitable distribution characteristics as described herein), that is used in the encryption to enhance security. In some examples, randomizers are obtained from a random number generator implemented hardware and/or software. It is noted that some encryption schemes do not use a randomizer.

Encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can optionally obtain one or more noise values. Noise values can be a random value (with suitable distribution characteristics as described herein), that are used in homomorphic encryption to enhance security. In some examples, the noise values is
 5 obtained from a random number generator implemented hardware and/or software. It is noted that some encryption schemes do not use noise values.

Encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can next encrypt (**315**) an element of the plaintext ring to an element of the ciphertext ring.

10 In some embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) performs the encryption by calculating a linear combination, over the ciphertext ring of, at least:

- the element of the plaintext ring to be encrypted, and
- 15 - a sum of, at least:
 - (i) a product of, at least, an encryption key and a randomizer, and
 - (ii) a noise value

where the encryption key is an element of the ciphertext ring, the noise value is sampled
 20 from the first ideal, and the randomizer is sampled from a distribution over a ring with a bounded expected norm.

In some such embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a symmetric encryption key, e.g. in accordance with the formula:

$$\begin{aligned}
 25 \quad \text{EncryptedElement}_{c0} = & \\
 & (\text{PlaintextElement} + (\text{Randomizer} * \text{EncryptionKey} + \\
 & \text{NoiseValue})) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})
 \end{aligned}$$

wherein ThirdNonCyclotomicRing denotes the plaintext ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the plaintext ring, EncryptionKey is the encryption key (i.e. an element of the ciphertext ring), NoiseValue is a noise value is sampled from the first ideal, and Randomizer is
 5 sampled from a distribution over a ring with a bounded expected norm.

In this case the associated randomizing element (e.g. for use in decryption) can be defined in accordance with the formula:

$$\text{EncryptedElement}_{c1} = (\text{Randomizer}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

10 It is noted that in this case EncryptedElement_{c0} is a linear combination of the element of the plaintext ring to be encrypted, and a sum of: the product of the encryption key and a randomizer, and the noise value.

In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private
 15 encryption key pair, e.g. in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c0} = & (\text{PlaintextElement} + ((\text{Randomizer} * ((\text{ConstantValue} * \\ & \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3})) \\ & \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

20 where ConstantValue is a constant that is an element of the ciphertext ring, and Randomizer3 is sampled from distributions over respective rings with bounded expected norms.

In this case the associated randomizing element can be defined in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c1} = & ((\text{Randomizer} * \text{ConstantValue}) + \text{Randomizer2}) \\ & \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

where Randomizer2 is sampled from a distribution over a ring with a bounded expected norm.

It is noted that here also EncryptedElement_{c0} is a linear combination of, at least, the element of the plaintext ring to be encrypted, and a sum of: the product of the encryption key and a randomizer, and the noise value.

In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private encryption key pair, where the first ideal and second ideal are principal ideals, in accordance the following formula:

$$\begin{aligned} \text{EncryptedElement}_{c0} = & \\ & ((\text{PlaintextElement} * \frac{\text{SecondIdeal}}{\text{FirstIdeal}} + ((\text{Randomizer1} * \\ & ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \\ & \text{Randomizer3})) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

In this case the associated randomizing element can be defined in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c1} = & ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2}) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private encryption key pair, where the first ideal is not a principal ideal, and the second ideal is a principal ideal, in accordance the following formula:

$$\begin{aligned} \text{EncryptedElement}_{c0} = & \\ & ((\text{PlaintextElement} * \text{SecondIdeal}) + ((\text{Randomizer1} * \\ & ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \\ & \text{Randomizer3})) \end{aligned}$$

$$\text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))$$

In this case the associated randomizing element can be defined in accordance with the formula:

$$\text{EncryptedElement}_{c1} = ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2})$$

$$\text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))$$

In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private encryption key pair, where the element of the plaintext ring is in the second ideal, the first ideal is a principal ideal, and the second ideal is not a principal ideal, in accordance the following formula:

$$\text{EncryptedElement}_{c0} =$$

$$(\text{PlaintextElement} + ((\text{Randomizer1} * ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3}))$$

$$\text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))$$

In this case the associated randomizing element can be defined in accordance with the formula:

$$\text{EncryptedElement}_{c1} = ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2})$$

$$\text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))$$

$$)$$

Encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can optionally store (**320**) data derivative of the randomizer(s) and/or constant value, for use in decryption. For example: encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can compute $\text{EncryptedElement}_{c1}$ according to formulas given above.

Attention is directed to **Fig. 4**, which illustrates a flow diagram of an example method of module-based fully-homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter.

As in the case of ring-based encryption, encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can encode (**405**) a plaintext to the plaintext ring. Encryption system (processing circuitry) **200A** (e.g. encoding unit **230A**) can optionally utilize encoding methods as described with reference to **Fig. 3** above.

Encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can obtain (**410**) an encryption key. In some examples, the encryption key is received or derived from a secure key exchange mechanism. In some examples, the encryption key is an element of a module that is derivative of the ciphertext module.

Encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can optionally obtain one or more randomizer. A randomizer can be a random value (with suitable distribution characteristics as described herein), that is used in the encryption to enhance security. In some examples, randomizers are obtained from a random number generator implemented hardware and/or software. It is noted that some encryption schemes do not use a randomizer.

Encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can optionally obtain one or more noise values. Noise values can be a random value (with suitable distribution characteristics as described herein), that are used in homomorphic encryption to enhance security. In some examples, the noise values is obtained from a random number generator implemented hardware and/or software. It is noted that some encryption schemes do not use noise values.

Encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can then encrypt (**415**) an element of the plaintext ring to an element of the ciphertext ring, utilizing an encryption key that is an element of a module that is derivative of the ciphertext ring, as known in the art.

In some embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) performs the encryption by calculating a linear combination, over the ciphertext ring of, at least:

- the element of the plaintext ring to be encrypted, and
 - a sum of, at least:
 - (i) a product of, at least, an encryption key and a randomizer, and
 - (ii) a noise value
- where the encryption key is an element of a module derived from the ciphertext ring, the noise value is sampled from the first ideal, and the randomizer is sampled from a distribution over a module with a bounded expected norm.

In some such embodiments, encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can perform encryption using a symmetric encryption key, e.g. in accordance with the formula:

$$\text{EncryptedElement}_{c0} = (\text{PlaintextElement} + (\text{Randomizer} * \text{EncryptionKey} + \text{NoiseValue})) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

wherein **ThirdNonCyclotomicRing** denotes the plaintext ring, **SecondIdeal** denotes the second ideal, **PlaintextElement** denotes the element of the plaintext ring, **EncryptionKey** is the encryption key, **NoiseValue** is a noise value is sampled from the first ideal, and **Randomizer** is the randomizer.

In this case the associated randomizing element (e.g. for use in decryption) can be defined in accordance with the formula:

$$\text{EncryptedElement}_{c1} = (\text{Randomizer}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

It is noted that in this case EncryptedElement_{c0} is a linear combination of the element of the plaintext ring to be encrypted, and a sum of: the product of the encryption key and a randomizer, and the noise value.

In some other embodiments, encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can perform encryption using a public/private encryption key pair, e.g. in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c0} = & \\ & (\text{PlaintextElement} + ((\text{Randomizer1} * ((\text{ConstantValue} * \\ & \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3})) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

where ConstantValue is a constant that is an element of a module derivative of the ciphertext ring, Randomizer1 and Randomizer2 are sampled from distributions over respective modules with bounded expected norms, and Randomizer3 is sampled from a distribution over a ring with a bounded expected norm

In this case the associated randomizing element can be defined in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c1} = & ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2}) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

It is noted that here also EncryptedElement_{c0} is a linear combination of, at least, the element of the plaintext ring to be encrypted, and a sum of: the product of the encryption key and a randomizer, and the noise value.

In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private encryption key pair, where the first ideal and second ideal are principal ideals, in accordance the following formula:

$$\text{EncryptedElement}_{c0} =$$

$$\begin{aligned} & \left(\left(\text{PlaintextElement} * \frac{\text{SecondIdeal}}{\text{FirstIdeal}} + ((\text{Randomizer1} * \right. \right. \\ & \left. \left. ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \right. \right. \\ & \left. \left. \text{Randomizer3}) \right) \right) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

5 In this case the associated randomizing element can be defined in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c1} &= ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2}) \\ &\text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

10 In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private encryption key pair, where the first ideal is not a principal ideal, and the second ideal is a principal ideal, in accordance the following formula:

$$\begin{aligned} \text{EncryptedElement}_{c0} &= \\ & \left(\left(\text{PlaintextElement} * \text{SecondIdeal} \right) + ((\text{Randomizer1} * \right. \\ 15 & \left. ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue}) + \right. \\ & \left. \text{Randomizer3}) \right) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal})) \end{aligned}$$

In this case the associated randomizing element can be defined in accordance with the formula:

$$\begin{aligned} 20 \quad \text{EncryptedElement}_{c1} &= ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2}) \\ &\text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal})) \\ & \end{aligned}$$

25 In some other embodiments, encryption system (processing circuitry) **200A** (e.g. ring-based encryption unit **220A**) can perform encryption using a public/private encryption key pair, where the element of the plaintext ring is in the second ideal, the first

ideal is a principal ideal, and the second ideal is not a principal ideal, in accordance the following formula:

$$\begin{aligned} \text{EncryptedElement}_{c0} = & \\ & (\text{PlaintextElement} + ((\text{Randomizer1} * ((\text{ConstantValue} * \\ 5 \quad & \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3})) \\ & \text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal})) \end{aligned}$$

In this case the associated randomizing element can be defined in accordance with the formula:

$$\begin{aligned} \text{EncryptedElement}_{c1} = & ((\text{Randomizer1} * \text{ConstantValue}) + \text{Randomizer2}) \\ 10 \quad & \text{mod} (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal})) \\ &) \end{aligned}$$

Encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can optionally store (**420**) data derivative of the randomizer(s) and/or constant value, for use in decryption. For example: encryption system (processing circuitry) **200A** (e.g. module-based encryption unit **225A**) can compute $\text{EncryptedElement}_{c1}$ according to formulas given above.

Attention is directed to **Fig. 5**, which illustrates a flow diagram of an example method of decrypting data that was encrypted using ring-based or module-based fully-homomorphic encryption, in accordance with some embodiments of the presently disclosed subject matter.

Decryption system (processing circuitry) **200B** (e.g. decryption unit **220B**) can obtain (**505**) a decryption key corresponding to the encryption key used to encrypt the ciphertext ring element to be decrypted. A decryption key can be e.g. a symmetric key, or a private key of a public/private key pair. In some examples, the decryption key can be received or derived as part of a key distribution protocol. The encryption key can be, for example a ciphertext ring element, or an element of an algebraic structure (e.g. a module or other algebraic structure) that is derivative of the ciphertext ring.

Decryption system (processing circuitry) **200B** (e.g. decryption unit **220B**) can obtain data derivative of any randomizers/constants used in encryption. In some examples, decryption system (processing circuitry) **200B** (e.g. decryption unit **220B**) can simply obtain the randomizers and/or constants. In other examples, decryption system
 5 (processing circuitry) **200B** (e.g. decryption unit **220B**) can obtain a randomizing element that is derivative of the randomizers and/or constants (as described above with reference to encryption methods). Decryption system (processing circuitry) **200B** (e.g. decryption unit **220B**) can utilize the randomizing elements in decryption, as described below.

10 Decryption system (processing circuitry) **200B** (e.g. decryption unit **220B**) can decrypt (**510**) an element of the ciphertext ring to an element of the plaintext ring.

In some embodiments, decryption system (processing circuitry) **200B** (e.g. encryption unit **220B**) performs the decryption by a method comprising: subtracting, from the element of the second non-cyclotomic ring, a value based on a product of a
 15 decryption key and a randomizing element.

In some such embodiments, the decryption key and the randomizing element are elements of the ciphertext ring. In some other embodiments, the decryption key and the randomizing element are elements of the module derived from the ciphertext ring, as known in the art.

20 More formally: considering a case where an encryption procedure generated the encrypted element ($\text{EncryptedElement}_{c0}$, $\text{EncryptedElement}_{c1}$), the decrypting can be performed based on the following formula:

$$\text{DecryptedElement} = \text{EncryptedElement}_{c0} - (\text{DecryptionKey} * \text{EncryptedElement}_{c1}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

25 where $\text{ThirdNonCyclotomicRing}$ denotes the non-cyclotomic ring R , SecondIdeal denotes the second ideal of R (termed Q), DecryptedElement denotes the resulting element of the plaintext ring, and DecryptionKey denotes the decryption key.

In some embodiments, the decryption key is symmetric (i.e. is the same as the encryption key). In some other embodiments, the decryption key is a private key i.e. a non-public key that corresponds to a public key that was utilized in the encryption operation that generated the element of the ciphertext ring.

- 5 In some embodiments, the randomizing element $\text{EncryptedElement}_{c1}$ is identical with a randomizer that was multiplied by the encryption key in the encryption operation.

In some embodiments, the randomizing element $\text{EncryptedElement}_{c1}$ is derivative of a randomizer that was multiplied by the encryption key in the encryption operation, and is further derivative of – for example - one or more additional randomizers and/or
10 constant values and/or other parameters.

Decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can decode (**515**) the plaintext ring element to plaintext data (e.g. binary data).

In some embodiments, when parameter b is 2, decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can map an element of the plaintext ring to a
15 plaintext by calculating:

$$f_{\text{element}}(2)$$

where $f_{\text{element}}()$ is the plaintext ring element (i.e. a polynomial) resulting from the decoding. It is noted that this decoding method is the inverse of the encoding method which maps plaintext bits to coefficients of polynomial terms (as described above).

- 20 As described in detail above, a single plaintext ring element can, in some examples, be an encoding of multiple plaintexts.

Accordingly, decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can perform decoding on the plaintext ring element, thereby generating multiple plaintexts.

- 25 In some examples, the plaintext ring element is a quotient ring of: a final ring of a series of extension rings, and a non-trivial ideal of the final ring.

It is noted that, as described in detail above each extension ring can be defined as:

PredecessorNonCyclotomicRing[t]/MinimalPolynomial

where t is an algebraic element being added to the predecessor ring (i.e. to extend it), and where the minimal polynomial can be:

$$\prod_{j=0}^n (t - a_j) + (c(t) * b) + (d(t) * FirstIdeal)$$

5 To decode an element of a quotient ring derivative of an extended ring thus defined, decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can generate, from the ring element, one or more ring elements of a quotient ring derivative of the immediately preceding extension ring in the series.

To generate the one or more ring elements of the quotient ring based on the
10 immediately preceding ring, decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can:

a) determine the roots of the minimal polynomial associated with the ring of the encoded ring element. These can be identical with the a_j values in the definition of the minimal polynomial.

15

b) for each determined root a_j :

calculate a result of substituting, in the encoded ring element, the respective extending algebraic element (i.e. t) with the constant value a_j ,

thereby generating one or more elements of a quotient ring based on the preceding ring of
20 the series (i.e. generating a quotient ring element for each determined a_j).

If the preceding ring is also an extension ring, then decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can then repeat steps a) -b) on this next ring.

If the preceding ring is a non-cyclotomic ring based on either $x^n + x - b$ or $x^n - x + b$, with the ideal being $x-b$ (for some integer b) - i.e. not an extension of such a ring, then

25 decryption system (processing circuitry) **200B** (e.g. decoding unit **225B**) can calculate, for each generated element $f_{\text{element}}()$;

$f_{\text{element}}(b)$

thereby resulting in one or more plaintexts.

It is noted that this decoding method can be utilized to decode from rings other than the rings defined herein. It is similarly noted that this decoding method can be
5 utilized in the context of other applications (e.g. other decryption applications, non-decryption applications).

It is noted that sections appearing hereinbelow describe additional embodiments, and are part of the detailed description, and that any headings included therein are for ease of reference only.

We present a fully homomorphic encryption scheme which natively supports arithmetic and logical operations over “machine words”,

namely plaintexts of the form $\mathbb{Z}_{2^n}$ (e.g. $n = 64$). Our scheme builds on the well-known BGV framework, but deviates in the selection of number field and in the encoding of messages. This allows us to support large message spaces efficiently without resorting to batching. In deed, each ciphertext in our scheme efficiently encodes a single message from the message space.

Arithmetic operations (modulo 2^n) are supported natively similarly to BGV-style FHE schemes, and we present an efficient bootstrapping procedure for our scheme. Our bootstrapping algorithm has the feature that along the way it decomposes our machine word into bits, so that during bootstrapping it is possible to perform logical operations (essentially addressing each bit in the message independently). This means that during a single bootstrapping cycle we can perform logical operations on n bits. For example, a “greater than” operation (if $x \geq y$ output 1, otherwise output 0), only requires a single subtraction and a single bootstrapping cycle.

Along the way we present a number of new tools and techniques, such as a generalization of the BGV modulus switching to a setting where the plaintext and ciphertext moduli are ideals (and not numbers).

1 Introduction

Fully homomorphic encryption (FHE) [RAD⁺78, Gen09a] allows us to compute on encrypted data without decrypting it first and without any knowledge of the secret key. This makes it a prominent privacy enhancing technology with numerous potential applications [GH19, CJP21]. Whereas there is ample motivation for using FHE in the real world, its utilization in practice is hindered by the overhead it incurs in computation and in storage. More explicitly, the resources, say in running time, required to run a computation over encrypted data may be orders of magnitude longer than running the same computation in the clear. Narrowing this gap to a tolerable level has been a major research direction of the FHE community for over a decade, and indeed some success has recently been reported, notably the recent announcement of the integration of Swift FHE into Apple devices as a part of iOS 18 [BTR24]. However the overhead remains prohibitive (or at least quite restrictive) for many desirable applications.

There are two main paradigms in the literature for practically-oriented FHE candidates. Both rely on structure related to the Learning with Errors (LWE) [Reg05] and Ring Learning with Errors (RLWE) [LPR13a], as put forth in [BV11b, BV11a, GSW13]. The first is a more direct extension [BV11b, BV11a] and

is utilized in the BGV, B/FV, CKKS schemes [BGV12, Bra12, FV12, CKKS17] and their successors, and is of a more *arithmetic* nature. We therefore refer to these as *arithmetic schemes*. Software libraries that take this approach include [HS20, SEA11, CKKS16]. The second follows the paradigm of [GSW13, DM14, CGGI20] and is more native for *boolean* operations, and we therefore refer to them as *boolean schemes*. Software libraries that take this approach include [Zam21, tfh17]. Let us discuss these approaches in slightly more detail below.

Arithmetic Schemes. These schemes natively support arithmetic operations over some *plaintext modulus* p . In BGV and B/FV the operations are carried out exactly over a discrete ring, whereas in CKKS, they are real valued and noisy, but they are all natively arithmetic. When implemented naively, this approach leads to exorbitant information overhead. Namely to very large ciphertexts that encrypt a small amount of data. This is partly mitigated by the use of *batching*: the ability to encrypt multiple plaintexts in parallel “slots” in the same ciphertext, and apply operations on them in parallel. Batching reduces the *amortized* overhead, both in terms of storage and in terms of computation, since an operation on the ciphertext corresponds to operations on many plaintexts.

Importantly, batching uses algebraic properties of the RLWE problem, and in particular requires a specific relation between p and the number field over which the scheme is defined. In particular this means that the strongest form of batching requires p which is an odd prime with some specific structure. There are techniques in the literature that allow to trade off the “amount of batching” with the form of the modulus [GHS12, HS21]. These techniques use field extensions to produce plaintext spaces of the form $\text{GF}(p^d)$. Thus it is possible to batch finite-field elements of size, e.g., 2^{64} , but it is not possible to properly batch plaintexts from the ring $\mathbb{Z}_{2^{64}}$. We are not aware of solutions that allow batching for message spaces such as those targeted in this work, namely $\mathbb{Z}_p$ for a large p , specifically for p being a power of 2.

Either way, batching requires both aggregation of a large amount of data per ciphertext *and* the ability to split this large amount of data into small amounts (to fit in each slot) in such a way that per-slot homomorphic operations remain useful. This is simply not the case when we wish to work with large integers, which is how many modern computer systems operate.

As in all known approaches for FHE, if one wishes to compute functionalities with a-priori unbounded depth, then a *bootstrapping* operation needs to be executed periodically. Bootstrapping [Gen09b] reduces the “noise component” that exists in all known FHE candidates. The noise grows with every homomorphic operation, and must be kept below a certain threshold to maintain the correctness of the scheme. Bootstrapping, therefore, is the process that allows to reduce the noise level so that additional homomorphic operations can be carried out. Bootstrapping essentially requires to apply the decryption algorithm homomorphically over the ciphertext. This requires applying operations that are not “natively” arithmetic, such as rounding or truncation of least significant bits. Therefore, whereas bootstrapping is heavy on resources in all known FHE

instantiations, arithmetic schemes generally struggle with bootstrapping more than boolean ones. This is likewise the case for any non-arithmetic operation, in particular boolean operations like “greater than” are challenging to implement.

Boolean Schemes. In these schemes, each ciphertext essentially encrypts a single bit. It is possible to pack multiple ciphertexts into a more compact representation, and various optimizations have been introduced in [DM14, CGGI20], but homomorphic operations are still performed at the bit level. This allows to perform logical operations natively. A particularly successful approach is taken by the TFHE scheme [CGGI20]. This proposed a way to perform the bootstrapping operation very efficiently, and therefore elect to perform bootstrapping after (or rather, as a part of) every operation. This framework makes it harder to work with data types that are naturally composed of multiple bits, such as number modulo p . In boolean schemes, one needs to represent large numbers as a sequence of bits and apply the appropriate boolean circuits to perform operations like addition and multiplication. This is possible, and indeed works naturally even for moduli of the form 2^n , but requires a large number of bootstrapping operations even for the simplest of operations (e.g. adding two n bit numbers as integers modulo n). We note that boolean schemes are very convenient for logical operations, e.g. “if” statements. For example, boolean comparison between two numbers, i.e. the predicate $a \geq b$, can be computed as easily as the difference $a - b$. Note that this is not the case for arithmetic schemes.

The Challenge: Arithmetic-Logic Unit for Machine Words. Our goal when using FHE is to be able to take existing code and covert it into running homomorphically with as little change as possible. Indeed, our algorithms and data types should remain oblivious to our choice of privacy preserving technique. We may therefore put forth the task of constructing a homomorphic framework that natively supports the “standard” operations of computer programs. In particular, this includes arithmetics over integers modulo 2^n , since integer data types in most architectures are of this form. At the same time we wish to be able to execute conditional statements, and apply logical operations on the bits of our numbers.¹ Indeed, the “slogan” for our goal is to implement an “Arithmetic Logic Unit” (ALU) inspired by such components that exist in CPUs. Such a unit would support arithmetics modulo 2^n , as well as logical operations at the bit level.

We note that both aforementioned approaches are universal (a.k.a Turing complete) and therefore allow in principle to implement an ALU. However, when concrete efficiency is concerned, if the native representation is only arithmetic or only boolean, then a change of representation will be required for some operations, which is usually prohibitive in terms of resources. This work therefore focuses on the following question.

Is it possible to design an FHE scheme that natively supports arithmetics modulo $\mathbb{Z}_{2^n}$ and supports boolean operations as well?

¹ We are not considering float-point operations at this point.

1.1 Our Contribution – FHE for Arithmetic and Logical Operations

We present a number of novel ideas that allow us to break from the existing paradigm and present a fully homomorphic encryption scheme with new capabilities. We first show how to construct a BGV-style scheme with native support for arithmetics over $\mathbb{Z}_{2^n}$ while not incurring the penalty that is usually associated with such plaintexts in “legacy” BGV. We then present a bootstrapping algorithm for our scheme which not only performs the “standard” bootstrapping features of reducing the noise in the ciphertext, but also, along the way, provides us access to the n individual bits in the binary representation of our $\mathbb{Z}_{2^n}$ message. This allows us, in the course of bootstrapping, to perform logical operations natively. Thus achieving both arithmetic and boolean logic without additional overhead.

For the first part, we construct an encryption scheme whose native plaintext space is $\mathbb{Z}_{2^n}$ (where n is a parameter). We show that arithmetic homomorphic operations (addition, multiplication) over this plaintext space can be performed essentially the same as in BGV-style FHE. However, the parameters of the scheme and noise scaling are comparable to BGV with plaintext $\{0, 1\}$. This is a crucial difference since prior to this work, a plaintext space of $\mathbb{Z}_{2^n}$ meant an additional factor of 2^{nd} in the noise, when evaluating a depth d arithmetic circuit. In contrast, in our scheme the growth would be roughly $n^{O(d)}$, which as explained above is comparable to binary plaintext. This has a cascading effect on all parameters of the scheme, since mild noise growth allows to reduce the so-called “noise ratio” of the underlying algebraic LWE problem, which in turn upgrades the security level of the scheme, which then allows to reduce the size of other parameters and maintain the same security level as previous schemes. As a result, we can instantiate our scheme with very large plaintext spaces that were previously prohibitive, with only minimal loss in performance. We note that we did not explore the possibility of batching in our scheme, so we only consider the setting of a single message per ciphertext. However, it may be possible to incorporate batching into our framework as well.

This is achieved by examining the properties of the BGV scheme in the algebraic setting (i.e. when it is based on structures stemming from the RLWE assumption and similar ones). We notice that, under the hood, BGV encryption can seamlessly be made to support any message space that is defined by an *ideal* in the ring where the scheme is defined (the *ring of integers* of some *number field*, or a subring thereof). This property was already noticed in the context of the NTRU scheme by Hoffstein and Silverman [HS99], and was used in the context of FHE in a number of works, e.g. [CLPX18, BCIV20]. However, these works fell short of achieving the goal of naturally supporting $\mathbb{Z}_{2^n}$ since they insisted on sticking with the use of *cyclotomic number fields* which has indeed been prevailing in the FHE literature. We deviate from this paradigm and show that working with non-cyclotomic number fields opens the door for great versatility in the design of the message space. We view this as a major contribution of this work. Indeed, this modification requires us to use algebraic variants of LWE that are defined over such number fields. Whereas this is not as widely used in

the literature, we notice that to the best of our knowledge there is no reason to speculate that cyclotomic number fields would lead to more secure constructions (in fact, some argue that the opposite is more likely). We provide a detailed discussion on the security of algebraic LWE in our context.

As an additional contribution, we discuss the possibility of creating “Double CRT” encoding of the ciphertexts in our scheme. Double CRT [HS20] is a way to encode ciphertexts that relies on the decomposition of the ciphertext modulus into prime ideals of a cyclotomic ring. This implies a representation of a large ciphertext as an array of fairly-small numbers, where addition and multiplication in the ring translates into pointwise addition and multiplication of the elements of the array. Whereas on the face of it we may not use Double CRT in our scheme, since we do not know how to decompose ordinary integers into prime ideals in our ring, we show that it is possible to take the opposite approach and construct the ciphertext modulus as a product of “simple” ideals. This would indeed allow for decomposition, but would imply that the ciphertexts in our scheme can no longer be represented as vectors of polynomials, where each coefficient is in $\mathbb{Z}_q$ for some integer q , but instead are cosets of some ideal in our ring. We show that nevertheless it is possible to apply homomorphic evaluation in this case.

In the second part we wish to devise a bootstrapping algorithm for our scheme. The bootstrapping process is computationally labor-intensive since it requires evaluating a pretty complex function. Furthermore, the noise accumulation *during bootstrapping* directly reduces the homomorphic capacity of the scheme after bootstrapping. Therefore, there is a lot of effort in the literature in order to reduce the complexity of bootstrapping in various FHE schemes [GHS12, HS21, DM14, CGGI20, KDE⁺24]. In particular, to come up with “decryption circuits” that require the least amount of resources and have the least noise accumulation.

In our scheme, we manage to introduce a decryption functionality which is both relatively mild in terms of resources, and allows us to extract the individual bits of the message in the course of bootstrapping. This means that during bootstrapping we can perform bit-level operations on the encrypted message, which we leverage to obtain logical/boolean evaluation capacity for our scheme.

Our starting point is the method of [KDE⁺24] who proposed to bootstrap BGV-style ciphertext by reducing the task to that of bootstrapping non-algebraic ciphertexts (i.e. ones that are not defined over a ring). In fact, this is quite straightforward in BGV-style encryption, since one can decompose an algebraic ciphertext into a collection of non-algebraic ciphertexts simply by thinking about ring elements as polynomials, and considering one coefficient at a time (see technical overview below for additional details). They then use the [CGGI20, CJP21] bootstrapping approach which has been designed for non-algebraic ciphertexts, and use it essentially as a building block.

We cannot follow this blueprint as is, since our algebraic ciphertexts do not decompose well in terms of coefficients. That is, our use of a general ideal to define the ciphertext space means that noise in our ciphertext is not added per-coefficient as in previous FHE schemes. We therefore design a novel approach

for recursive decomposition of our algebraic ciphertext into a collection of non-algebraic ciphertexts. Essentially this works by noticing that we can extract a non-algebraic ciphertext encrypting the least significant bit. We then bootstrap this ciphertext and show how to use the bootstrapped ciphertext in order to produce a non-algebraic encryption of the next bit of the message. At the end of the process, we have bootstrapped versions of non-algebraic encryptions of all bits of the message. This allows us to perform many logical operations “for free”: e.g. apply any permutation or shift on the bits, remove some of the bits or XOR them with each other. We can also apply more sophisticated operations such as bitwise AND using a bit more work.

We refer the reader to the technical overview below for a more detailed explanation on how our scheme works.

Finally, we consider concrete parameters for our scheme, with a plaintext space of $n = 64$ bits. We implemented our scheme and report implementation-specific details.

1.2 Other Related Works

This work addresses FHE in the circuit model. This means that we consider computation that is represented in a combinatorial form as a circuit with boolean or arithmetic gates. Until recently, FHE was only known to be applicable in this model. Recently, Lin, Mook and Wichs [LMW23] introduced the first FHE scheme that operates in the RAM model (with suitable preprocessing). Whether our techniques have implications on RAM-FHE remains a subject for future inquiry.

1.3 Paper Organization

Section 2 contains a technical overview of our scheme. Section 3 contains preliminaries and definitions. The basic scheme and basic arithmetic homomorphic evaluation are presented in Sections 4, 5. Our bootstrapping algorithm and the derived homomorphic logical operations are described in Section 6. In Section 7 we provide a detailed discussion on the security of algebraic-LWE in our ring. Implementation details are provided in Section 8. Our analysis for using algebraic modulus for the sake of double CRT is provided in Appendix A.

2 Technical Overview

We start with a common blueprint for LWE-based encryption. The ciphertext is a (column) vector $\mathbf{c}$, say in $\mathbb{Z}_q^n$, and the secret key $\mathbf{s}$ is a (row) vector of the same dimension, say in $\mathbb{Z}^n$. The ciphertext is generated so that

$$\mathbf{s} \cdot \mathbf{c} = \mu + p\varepsilon \pmod{q} . \quad (1)$$

We refer to q is the “ciphertext modulus” and p is the “plaintext modulus”, and we require that p, q are coprime. The encrypted message μ can be interpreted

as an element in $\mathbb{Z}_p$ and the variable ε is the “noise”. So long as $|\varepsilon| \ll q$, it is possible to recover $\mathbf{m}$ from the ciphertext.

In algebraic LWE variants, $\mathbb{Z}$ is replaced with some other ring. Particularly the “ring of integers” of a number field, or a full-rank subring thereof. A very common choice for such a ring is $\mathcal{R} = \mathbb{Z}[x]/\langle f(x) \rangle$, where f is an irreducible monic polynomial of degree n .² A prevalent choice for f is a *cyclotomic* polynomial, in particular of the form $f(x) = x^n + 1$ for n being a power of 2. The reader may keep this example in mind until we explain how we deviate from it in this work.

Given such a ring, we may analogously consider $\mathbf{c} \in \mathcal{R}_q$ (where $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$), $\mathbf{s} \in \mathcal{R}$ so that

$$\mathbf{s} \cdot \mathbf{c} = \mu + p\varepsilon \pmod{q\mathcal{R}}, \quad (2)$$

where $\varepsilon \in \mathcal{R}$ and the messages are now drawn from $\mathcal{R}_p$.³ Since elements in $\mathcal{R}_q$ can be represented as degree $(n-1)$ polynomials, the term $\mu + p\varepsilon \pmod{q}$ can be considered one coefficient at a time. In each such coefficient there is a message part that comes from $\mathbb{Z}_p$ and a noise part that comes from the respective coefficient of ε . We conclude that if all coefficients of ε are small compared to q , denote this $|\varepsilon| \ll q$, then $\mu \in \mathcal{R}_p$ can be fully recovered.⁴

As shown in [BV11b, BV11a, BGV12], the above scheme can be made homomorphic and support arithmetic operations over the plaintext ring $\mathcal{R}_p$. Namely, addition and multiplication. Each such operation incurs a penalty in the size of ε , most significantly during multiplication. Indeed, even in the most noise-efficient multiplication methods, a depth d multiplication incurs a penalty of roughly p^d in the noise (in addition to other penalties that are not directly dependent on p). Furthermore, arithmetics in $\mathcal{R}_p$ is often not what is actually required. Indeed, common data types take a form of $\mathbb{Z}_p$ and not of polynomials with $\mathbb{Z}_p$ coefficients that are multiplied modulo $f(x)$. One way to achieve homomorphism with respect to $\mathbb{Z}_p$ is to encode messages $\mathbf{m} \in \mathbb{Z}_p$ as polynomials $\mu \in \mathcal{R}_p$ by just setting the free coefficient of μ to be $\mathbf{m}$, and keep all other coefficients at 0 (or in more abstract term use the fact that $\mathbb{Z}_p$ is a subring of $\mathcal{R}_p$). This indeed does the trick but has two main drawbacks. First, the information rate achieved is quite poor. An element in $\mathcal{R}_p$ can encode n elements from $\mathbb{Z}_p$ in terms of information content, but the homomorphic requirement forces us to lose a factor of n in

² This ring coincides with the ring of integers of the number field defined by $f(x)$ in some useful special cases, but not always.

³ In fact, algebraic variants of LWE (such as RLWE) were originally defined slightly differently, using the *dual ring*. This definition allowed to prove worst-case hardness for the problem. However, it is possible to convert into the form presented here while paying a “penalty” in the size of the noise. See, e.g., [RSW13], and also Section 7 in this work.

⁴ For the informed reader, we note that we work here with the so-called “coefficient embedding”. Whereas in some cases it is useful to work with the “canonical embedding” we find that for the analysis of our scheme the coefficient embedding provides more convenience and flexibility.

utilizing this capacity. Second, if p is large, then the noise growth is completely prohibitive.

Prior works, starting with [SS19], proposed a way to *amortize* the information utility. They show that by properly selecting f and p , it is possible to set up the $\mathcal{R}$ so that $\mathcal{R}_p \cong \mathbb{Z}_p^n$, as rings, with pointwise addition and multiplication. This means that it is possible to pack n elements from $\mathbb{Z}_p$ and apply operations on them in parallel (see also [GHS12] and other followup works). However, this solution is limited to specific values of p , and to a setting where it is indeed possible to “collect” n messages into one ciphertext. This method had been extended to other types of plaintext spaces, specifically field or ring extensions, see [HS21]. We are not aware of solutions that allow batching for message spaces such as those targeted in this work, namely $\mathbb{Z}_p$ for a large p , and in particular $p = 2^\ell$ for some ℓ . However, using the existing methods, even if batching can be achieved, the noise blowup problem remains.

Using Ideal Plaintext Modulus. We notice, as others before [HS00, GC14, CLPX18], that p needs not be in $\mathbb{Z}$ in order for the scheme to work. Indeed, we may consider an ideal $\mathfrak{p}$ in the ring $\mathcal{R}$, and consider ciphertexts for which

$$\mathbf{s} \cdot \mathbf{c} = \mu + e \pmod{q\mathcal{R}}, \quad (3)$$

where we are guaranteed that $e \in \mathfrak{p}$. For our purposes we consider $\mathfrak{p} = \langle x - 2 \rangle$, so we can think of $e = (x - 2)\varepsilon$, where ε is as before. In this case, the plaintext μ is in the ring $\mathcal{R}_{\mathfrak{p}} = \mathcal{R}/\mathfrak{p}$. Importantly, in many cases there is a ring homomorphism $\mathcal{R}_{\mathfrak{p}} \cong \mathbb{Z}_p$ for $p = |f(2)|$ (this holds for any $f(x)$ that is of interest in this work). However, in a cyclotomic ring, such p can never be a power of 2. We therefore propose to use *non-cyclotomic* polynomials. In particular we consider polynomials of the form $f(x) = x^n - x + 2$.

We note while the FHE literature mostly uses cyclotomics, this is done for reasons of convenience and structure (e.g. having automorphisms that can be used for some functionalities). There is no evidence that working over cyclotomics makes algebraic LWE more secure. In fact, some claim that to the contrary, that the additional structure of cyclotomic rings makes them more risky in terms of security, and designers should opt for other polynomials [BCLvV16]. We refer the reader to Section 7 for a more detailed discussion.

We can now explain our scheme. Our message space is $\mathbb{Z}_{2^n}$ for some parameter n , and our ring is $\mathcal{R} = \mathbb{Z}[x]/\langle f(x) \rangle$, $f(x) = x^n - x + 2$. We require an encoding method to map an elements $\mathbf{m} \in \mathbb{Z}_{2^n}$ into $\mu \in \mathcal{R}_{\mathfrak{p}}$ and vice versa. It is important that $|\mu|$ is as small as possible, for reasons we explain below. This is in fact not very difficult as one can verify that one can map any $\mathbf{m}$ into μ with $\{0, 1\}$ coefficients by simply considering the binary representation of the number $\mathbf{m} \pmod{2^n}$. In the other direction, for any $\mu = \mu(x)$, we may consider $\mu(2) \pmod{2^n}$. This mapping indeed implement the aforementioned ring homomorphism. In addition it has the useful property that the bits of $\mathbf{m}$ are exactly the coefficients of μ . This is a consequence of using the non-cyclotomic polynomial and will be invaluable to us down the line.

Therefore, when we wish to encrypt a message m , we first encode it into a polynomial μ and then use the above scheme. The arithmetic homomorphic properties such as addition and multiplication work very similarly to previous schemes, but now, instead of p^d penalty for depth d multiplication as in [BGV12] and related schemes, the penalty scales roughly with $(\log p)^d$, which is a significant improvement. We will not go into the details here, but the reason, essentially, is that the noise growth factor depends on the maximal ℓ_1 norm of plaintexts in the space. Since we can encode m into μ with $\{0, 1\}$ coefficient, this factor is roughly $n = \log p$.

We remark that in the description so far, n plays a double role. Both as a dimension for the algebraic LWE problem and as a parameter that determines the plaintext space. If desired we can decouple these two roles by working with the so-called “module” version of the algebraic LWE problem. In this case $c \in \mathcal{R}_q^r$ for some “rank” r (and likewise for s). This means that the dimension for the LWE problem is $r \cdot n$, but the plaintext space is determined by n . This subtlety is not going to be of particular importance for this high level discussion.

Arithmetic Homomorphic Operations. Homomorphic addition and multiplication are performed similarly to the BGV scheme [BGV12, HS20]. Whereas many of the subroutines carry over, there is one that requires rethinking and updating. This is the so-called “modulus switching” procedure. The goal of this operation is essentially to “shrink down” the ciphertext modulus q into a new smaller q' . In this process, the relative noise level $|\varepsilon|/q$ remains roughly unchanged: $|\varepsilon|/q \approx |\varepsilon'|/q'$, which means that the absolute noise level goes down (since $q' < q$). This subroutine is quite central to BGV-style schemes and it needs to be carried out after every multiplication operation, and it also plays a role in the bootstrapping procedure (to be discussed below). Furthermore, [HS20] requires modulus switching in order to optimize the performance of another subroutine, called “key switching”, which can in principle be carried out without modulus switching but with worse performance in terms of noise growth.

In this context, we first consider the straightforward extension of “legacy” modulus switching to the setting where $\mathfrak{p}$ is a general ideal. This is a relatively straightforward extension but it requires that $q = q' \pmod{\mathfrak{p}}$. Whereas prior to our work this was not considered a problem since essentially by definition, the plaintext space was such that $p \ll q, q'$, in our case this could be rather prohibitive. In fact, our paradigm even supports a setting where $q \ll p$. In prior works, [GC14] considered the straightforward extension as defined above, without noticing its drawbacks for very large p , whereas [CLPX18] did not face this challenge since they worked with the so-called B/FV approach, and do not perform bootstrapping, so at least asymptotically one can do without modulus switching, an approach that is not suitable for our needs.

We propose two methods to mitigate this problem. The first is to notice that even if $\alpha = qq'^{-1} \pmod{\mathfrak{p}} \neq 1$, so long as this value α is invertible modulo $\mathfrak{p}$, it is possible to “correct” the modulus switching procedure at the cost of an additional homomorphic scalar multiplication, which is relatively mild in terms of noise cost. The second is to do away with the ciphertext modulus q being an

integer, and taking it as an element of the ring as well. This means that for any q it is possible to find q' with the proper “volume” (the volume in this context is the index of the ideal in the ring) so that modulus switching has the desired properties. We notice that working with such algebraic ciphertext modulus may have additional advantages. For example, setting $q = \prod_i q_i$, where each q_i is “small” will allow to perform modulus switching as desired, and also provide a method for double-CRT encoding of the ciphertext, as explained above. We elaborate on this in Section A. Finally, going back to the first method, we notice that the scaling by α may be performed only “conceptually” without actually performing the multiplication. We can just “carry” the α factor throughout the computation and cancel it only at decryption (or at bootstrapping).

Bootstrapping. We now describe the bootstrapping procedure for our scheme. As explained above, we follow [KDE⁺24] and rely on the TFHE bootstrapping procedure [CGG20] as a building block. Their goal was to bootstrap schemes with syntax similar to Eq. (2). They do this by decomposing the ciphertext into n non-algebraic ciphertexts, each of the form of Eq. (1). To see why this is the case, we recall that Eq. (2) is an equality between two ring elements, which can be seen as an equality between two polynomials of degree $n-1$. If we consider the i -th coefficient of this polynomial, then on the right hand side we get $\mu_i + p\varepsilon_i$, and on the left hand side we get a bilinear operation on the coefficients of c and s . This can be interpreted as a non-algebraic ciphertext that is decryptable using the coefficient vector of s . The [KDE⁺24] approach is to bootstrap each such non-algebraic ciphertext using the methods of [?], and then putting the outcomes back together.

This approach doesn’t carry over to our setting. We embrace the basic idea of splitting an algebraic ciphertext into multiple non-algebraic ones, but we require a very different mechanism for this purpose, one that will ultimately also allow us to perform boolean operations in the course of bootstrapping. To see why this is the case, let us analyze the right hand side of Eq. (3) similarly to what we did with Eq. (2). The i -th coefficient of $\mu + e$ equals $\mu_i + e_i$, but now recall that $e = (x-2)\varepsilon \pmod{f(x)}$, so e_i can no longer be written as $p\varepsilon_i$ as before. Let us write down the coefficients of e for $n=4$ to get a sense of what is going on (note that ε_{n-1} wraps around because of the modulation in $f(x)$).

$$\begin{aligned} e_0 &= -2\varepsilon_0 - 2\varepsilon_3 \\ e_1 &= \varepsilon_0 - 2\varepsilon_1 - \varepsilon_3 \\ e_2 &= \varepsilon_1 - 2\varepsilon_2 \\ e_3 &= \varepsilon_2 - 2\varepsilon_3 \end{aligned}$$

Note that other than e_0 , in all other components there is no guarantee that the noise belongs to some ideal. Indeed the structure of the noise in our setting is not axis-parallel in the coefficient embedding. Nevertheless, we notice that at least for e_0 we do have a guarantee that it can be written as $e_0 = 2e'_0$. We may therefore extract the free coefficient from equation Eq. (3), and use it to bootstrap and obtain a non-algebraic low-noise encryption of μ_0 .

It may seem that we have reached a dead end, since the other e_i are not as accommodating. To get us out of this barrier, let us consider a ciphertext for which we are guaranteed that $\mu_0 = 0$. Namely, it is possible to write $\mu = x\mu'$. We also notice that in our ring, since we work modulo $f(x) = x^n - x + 2$, it holds that $(x - 2) = x^n$. It follows that we can write the right-hand side of Eq. (3), for such a ciphertext, as $x\mu' + x^n\varepsilon$. Using a standard technique (essentially multiplying the ciphertext by $x^{-1} \pmod{q}$) it is possible to remove the common factor and convert this to one whose right hand side is $\mu' + x^{n-1}\varepsilon$. Letting $e' = x^{n-1}\varepsilon$, we can see that $e'_0 = -2\varepsilon_1$, which is again even. In fact, for every power of x it holds that the free component of $x^i\varepsilon$ is even. This is not an accident and follows directly from our choice of ring polynomial $f(x)$. Our approach, therefore, is to first extract μ_0 , and then subtract it from the original ciphertext to obtain a ciphertext as described above. Then extract μ_1 from this ciphertext and so on. However, implementing this approach is not without difficulties.

So far, we did not get into the question of what the output of the bootstrapping actually looks like. We only mentioned that it is a “non-algebraic low-noise encryption”. The right-hand side of this encryption would normally be of the form $\lceil q/2 \rceil \mu_0 + \hat{\varepsilon}_0 \pmod{q}$ for some small integer value $\hat{\varepsilon}_0 \in \mathbb{Z}$. How can we use it to cancel out μ_0 from an algebraic ciphertext? To do this, we utilize the versatility of the bootstrapping procedure of [?], which allows to produce non-algebraic ciphertexts with-right hand side $g(\mu_0) + \hat{\varepsilon}_0 \pmod{q}$, for any function g with $\mathbb{Z}_q$ values (in fact, it is even somewhat more versatile, but this suffices for us). We use it to recover all powers of two multiples of μ_0 . That is $2^j \mu_0 + \hat{\varepsilon}_{0,j} \pmod{q}$. This allows us, via subset sum, to recover an encryption for any integer multiple of μ_0 . Indeed, this is only for non-algebraic ciphertexts, but it can be extended to algebraic ciphertexts as well. We therefore recover an algebraic ciphertext of the form $(x - 2)^{-1} \mu_0 + \hat{\varepsilon}'_0$, which can then be scaled to obtain a ciphertext of the form $\mu_0 + (x - 2)\hat{\varepsilon}'_0$. This ciphertext can then be subtracted from the original algebraic encryption of μ , and allow us to continue to μ_1 and then further down the line.

To summarize, in each step of the bootstrapping, we recover an algebraic ciphertext of the form $x^i \mu_i + (x - 2)\hat{\varepsilon}'_i$. We can eventually add all of these together to obtain an encryption of μ with small noise, which concludes the bootstrapping functionality.

Logical Operations. As explained above, and demonstrated in our description of bootstrapping, in the course of bootstrapping we recover individual encryptions of all bits μ_i , scaled by any algebraic or non-algebraic integer. This allows us to very easily perform bitwise operations on the ciphertext. Bit shifts and permutations are essentially trivial since we can recover $x^i \mu_j + (x - 2)\hat{\varepsilon}'_i$ for any i, j , and add them together to obtain the new ciphertext. Linear GF(2) operations over the bits of the message can likewise be computed.

More elaborate operations are also easily possible. For example the logical “ $a \geq b$ ” predicate which outputs 1 if $a \geq b$ and 0 otherwise can be implemented homomorphically by computing (arithmetically) the difference $a - b \pmod{2^n}$

and then extract the most significant bit of the difference which exactly implements the above functionality.

We may even consider further operations such as bitwise AND between ciphertexts, which can also be performed once we extract all bits. One way to implement this is to recover non-algebraic ciphertexts of the form $\mu_i + 4\varepsilon_i$. Adding two ciphertexts of this form, and then extracting the second-least-significant bit, recovers the AND of the two numbers. One may think of more sophisticated algorithms that can be carried out in this way.

3 Preliminaries

3.1 Notation

The security parameter is denoted by κ . We use bold letters for vectors and matrices.

An expression that includes a multiplication of a matrix and a ciphertext, or a vector of ciphertexts as in $D \cdot (c_0, \dots, c_{n-1})^t$ is done as in the regular way, however when a matrix entry by a vector entry, the operation is a homomorphic scalar-by-ciphertext multiplication one.

3.2 Algebraic Number Theory

We consider a monic irreducible polynomial $f(x)$ of degree n and integer coefficients, the number field $K = \mathbb{Q}[x]/\langle f(x) \rangle$, and the ring $\mathcal{R} = \mathbb{Z}[x]/\langle f(x) \rangle$. Note that this ring is not necessarily the ring of integers of K , but it is nonetheless an order of the field (a full-rank subring of the ring of integers). For any ideal $\mathfrak{p}$ in $\mathcal{R}$ we denote $\mathcal{R}_{\mathfrak{p}} = \mathcal{R}/\mathfrak{p}$. For brevity, when $\mathfrak{q} = \langle q \rangle$ is generated by an integer $q \in \mathbb{Z}$, we denote $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. If $\mathfrak{p} = \langle x - a \rangle$ then there exists a ring-isomorphism $\mathcal{R}_{\mathfrak{p}} \cong \mathbb{Z}_p$ where $p = \mathfrak{N}(\mathfrak{p}) = |f(a)|$ (the notation $\mathfrak{N}(\cdot)$ refers to the *absolute norm* of the ideal which is defined as the index of the ideal in the ring: $\mathfrak{N}(\mathfrak{p}) = (\mathcal{R} : \mathfrak{p})$). In this work we will consider ideals for which the above holds. We are particularly interested in the setting where $f(x) = x^n - x + 2$ for some n , and $\mathfrak{p} = \langle x - 2 \rangle$. This results in $p = |f(2)| = 2^n$.

We consider two popular methods for *embedding* elements in K into Euclidean or complex space.

- **The Coefficient Embedding.** For a number field K as above, it is always possible to express any element of K uniquely as a polynomial of degree less than n and rational coefficients. This induces a map $K \rightarrow \mathbb{Q}^n$ by mapping $t \in K$ whose polynomial representation is $t = \sum_{i=0}^{n-1} t_i x^i$ to $[t] = (t_0, \dots, t_{n-1})^t \in \mathbb{Q}^n$.
- **The Canonical Embedding.** A number field K has exactly n ring embeddings (injective ring homomorphisms) into $\mathbb{C}$, which are induced by taking x to the (complex) roots of f . Denote them by $\sigma_i : K \rightarrow \mathbb{C}$. The canonical embedding of t is the vector $[[t]] = (\sigma_1(t), \dots, \sigma_n(t))^t \in \mathbb{C}^n$. It is also possible to embed this vector into $\mathbb{R}^n$ (essentially since f is a real-valued polynomial

so its complex roots come in conjugate pairs). However, we will not require this real embedding here. In the canonical embedding, field addition and multiplication are done component-wise.

These embeddings induce geometric norms on elements of K by using ℓ_p norms over the embedding vectors. In this work we work mostly with the coefficient embedding (even though in some cases the canonical embedding would allow for a tighter analysis). We therefore let $\ell_p(\cdot)$ denote the ℓ_p norm of a field element in the coefficient embedding.

When working with the coefficient embedding, it is useful to consider the *expansion factor* of the field:

$$\gamma_p = \gamma_p(K) = \max \left\{ \frac{\ell_p(a \cdot b)}{\ell_p(a)\ell_p(b)} : a, b \in K \right\} \quad (4)$$

By default, when p is not specified, we consider the infinity norm $p = \infty$.

For $w \in \mathcal{R}$ we define

$$exf_w = \max \left\{ \frac{\ell_\infty(a \cdot b)}{\ell_\infty(a)\ell_\infty(b)} : a, b \in K \right\}$$

Our secret key has small ℓ_1 norm, therefore we also define:

$$\hat{\gamma} = \max \left\{ \frac{\ell_\infty(a \cdot b)}{\ell_1(a)\ell_\infty(b)} : a, b \in K \right\} .$$

Note that $\hat{\gamma} \geq \gamma/n$. The following proposition bounds the expansion factor for our polynomial.

Proposition 3.1. *Let $f(x) = x^n - x + 2$, then for all $e_1, e_2 \in \mathcal{R}$ it holds that the expansion factor γ is upper bounded by $3n$, and moreover $\hat{\gamma}$ is upper bounded by 3.*

Proof. For $e \in \mathcal{R}$ we have:

$$\begin{aligned} (e_1 \cdot e_2)^{(i)} &= \sum_{j=0}^i e_1^{(j)} e_2^{(i-j)} - 2 \sum_{j=i+1}^n e_1^{(j)} e_2^{(n+i-j)} + \sum_{j=i}^n e_1^{(j)} e_2^{(n-1+i-j)} \\ &\leq 3 \cdot \ell_1(e_1) \cdot \ell_\infty(e_2) \\ &\leq 3n \cdot \ell_\infty(e_1) \cdot \ell_\infty(e_2) . \end{aligned}$$

For any field element $t \in K$, we can consider the “multiply by t ” operator $\text{mult}_t : K \rightarrow K$, where $\text{mult}_t(w) = t \cdot w$. This is a linear operator (in any embedding) and can be described by a matrix acting on the embedding of w . In the canonical embedding, this matrix is diagonal and less useful for our purposes. In the coefficient embedding we define $\mathcal{M}_t \in \mathbb{Q}^{n \times n}$ as the matrix such that for all w it holds that $[w'] = \mathcal{M}_t[w]$, where $w' = tw$.

3.3 Homomorphic Encryption

We now define homomorphic encryption and its desired properties. Throughout this section (and this work) we use κ to indicate the security parameter.

A homomorphic (public-key) encryption scheme

$$\text{HE} = (\text{HE.Keygen}, \text{HE.Enc}, \text{HE.Dec}, \text{HE.Eval})$$

is a quadruple of PPT algorithms as follows:

- **Key generation:** The algorithm $(\text{pk}, \text{evk}, \text{sk}) \leftarrow \text{HE.Keygen}(1^\kappa, \text{aux})$ takes a unary representation of the security parameter and auxiliary input that in our case encodes the plaintext space and the homomorphic capacity and outputs a public encryption key pk , a public evaluation key evk , and a secret decryption key sk .
- **Encryption:** The algorithm $c \leftarrow \text{HE.Enc}_{\text{pk}}(\mu)$ takes the public key pk and a message $\mu \in \mathbb{Z}/p\mathbb{Z}$ and outputs a ciphertext c .
- **Decryption:** The algorithm $\mu^* \leftarrow \text{HE.Dec}_{\text{sk}}(c)$ takes the secret key sk and a ciphertext c and outputs a message $\mu^* \in \mathbb{Z}/p\mathbb{Z}$.
- **Homomorphic evaluation:** The algorithm $c_f \leftarrow \text{HE.Eval}_{\text{evk}}(f, c_1, \dots, c_\ell)$ takes the evaluation key evk , a function $f : (\mathbb{Z}/p\mathbb{Z})^\ell \rightarrow \mathbb{Z}/p\mathbb{Z}$, and a set of ℓ ciphertexts $c_1, \dots, c_\ell$, and outputs a ciphertext c_f .

Definition 3.1. A homomorphic encryption scheme is said to correctly evaluate a circuit family F if for all $f \in F$ and for all $m_1, \dots, m_\ell \in \mathbb{Z}/p\mathbb{Z}$, the following holds: If sk, pk are correctly generated by HE.Keygen with security parameter κ , and if $c_i = \text{HE.Enc}_{\text{pk}}(m_i)$ for all i , and $c_f = \text{HE.Eval}_{\text{pk}}(f, c_1, \dots, c_\ell)$, then

$$\Pr[\text{HE.Dec}_{\text{sk}}(c_f) \neq f(m_1, \dots, m_\ell)] = \text{negl}(\kappa),$$

where the probability is taken over all randomness in the error and key distributions.

Furthermore, the scheme is said to compactly evaluate the family if, in addition, the running time of the decryption circuit depends only on κ and not on its input.

The only security notion we consider in this chapter is semantic security, namely security with respect to passive adversaries. We use its widely known formulation as IND-CPA security, defined as follows.

Definition 3.2 (IND-CPA Security).

Let $(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$ be a public key encryption scheme. We define an experiment $\text{Expr}_b^{\text{cpa}}[\mathcal{A}]$ parameterized by a bit $b \in \{0, 1\}$ and an efficient adversary $\mathcal{A}$:

$$\text{Expr}_b^{\text{cpa}}[\mathcal{A}](1^\kappa) : \begin{cases} (sk, pk, ek) \leftarrow \text{KeyGen}(1^\kappa) \\ (x_0, x_1) \leftarrow \mathcal{A}(1^\kappa, pk, ek) \\ ct \leftarrow \text{Enc}_{pk}(x_b) \\ b' \leftarrow \mathcal{A}(ct) \\ \text{Return } b' \end{cases}$$

We say that the scheme is IND-CPA secure if for every PPT adversary A , it holds that

$$\text{Adv}^{\text{cpa}}[A](\kappa) := |\Pr\{\text{Expr}_0^{\text{cpa}}[A](1^\kappa) = 1\} - \Pr\{\text{Expr}_1^{\text{cpa}}[A](1^\kappa) = 1\}| = \text{negl}(\kappa).$$

3.4 Gadget Decomposition

We present a ring generalization of the widely used “gadget decomposition” technique [MP12]. For a ring $\mathcal{R}$, a *gadget vector* $\mathbf{g} \in \mathcal{R}^l$, is a row vector of ring elements, equipped with a (not necessarily linear) *decomposition operator* $\mathbf{g}^{-1} : \mathcal{R} \rightarrow \mathcal{R}^l$. We require that for all $x \in \mathcal{R}$, it holds that $\mathbf{g} \cdot \mathbf{g}^{-1}(x) = x$. Namely the gadget is an “inverse” of the decomposition operator. Importantly, the gadget is defined so as to ensure that for all $x \in \mathcal{R}$ it holds that $\mathbf{g}^{-1}(x)$ is “small”, with respect to some measure. Thus the decomposition operator allows to trade-off size for dimension, while allowing linear reconstruction.

The *gadget matrix* $\mathbf{G}_{\mathbf{g}}$ corresponding to a gadget vector $\mathbf{g}$ is defined as $\mathbf{G}_{\mathbf{g}} = \mathbf{I}_k \otimes \mathbf{g} = \text{diag}(\mathbf{g}, \dots, \mathbf{g}) \in \mathcal{R}^{k \times kl}$ (where the diag operator organizes its operands along the block-diagonal of a matrix, with 0 in the off-diagonal), where $\mathbf{I}_k$ is the identity matrix of size k . The decomposition operation, $\mathbf{G}_{\mathbf{g}}^{-1} : \mathcal{R}^{k \times k'} \rightarrow \mathcal{R}^{kl \times k'}$, applies $\mathbf{g}^{-1}$ coordinate-wise, expanding each entry into an l -dimensional column vectors.

More generally we may consider a gadget matrix that is generated by a set of different gadget vectors: $\mathbf{G}_{\mathbf{g}_1, \dots, \mathbf{g}_k}$. This is defined as $\mathbf{G}_{\mathbf{g}_1, \dots, \mathbf{g}_k} = \text{diag}(\mathbf{g}_1, \dots, \mathbf{g}_k) \in \mathcal{R}^{k \times (\sum l_i)}$. The decomposition operation, denoted as $\mathbf{G}_{\mathbf{g}_1, \dots, \mathbf{g}_k}^{-1} : \mathcal{R}^{k \times k'} \rightarrow \mathcal{R}^{(\sum l_i) \times k'}$ is defined analogously.

Next we define a specific class of gadgets that are commonly used in the paper.

Definition 3.3 (The Powers Gadget). Let $\mathcal{R}$ be either a polynomial ring $\mathbb{Z}[x]/\langle f(x) \rangle$ or $\mathbb{Z}$ itself. For $x \in \mathcal{R}_q$ we define $\text{Decomp}_d(x)$ be the decomposition of x into the following base d representation, if $x = \sum_{j=0}^{\lceil \log_d(q) \rceil} (b_j d^j)$ for $b_j \in \mathcal{R}/d$ with coefficients smaller than $\lfloor d/2 \rfloor$, $\text{Decomp}_d(x) = (b_0, \dots, b_{\lceil \log_d(q) \rceil}) \in \mathcal{R}/d$. The $\text{Decomp}_d(x)$ operator is the decomposition operator $\mathbf{g}_{\text{Powers}_d}^{-1}$ corresponding to the Powers_d gadget vector: $\text{Powers}_d^q = (1, d, \dots, d^{\lceil \log_d(q) \rceil}) \in \mathcal{R}_q^{\lceil \log_d(q) \rceil}$. We Notice that $\ell_\infty(\mathbf{g}_{\text{Powers}_d}^{-1}(x)) \leq \lfloor d/2 \rfloor$.

3.5 Learning with Errors and Related Problems

The Learning with Errors (LWE) problem was introduced by Regev [Reg05] and became one of the most widely used and influential building blocks in cryptography. We use the following definition.

Definition 3.4 (Learning with Errors). Let $n, q \in \mathbb{N}$, χ_s a distribution over $\mathbb{Z}^n$, χ_e a distribution over $\mathbb{Z}$.

For a row vector $\mathbf{s} \in \mathbb{Z}^n$, consider the distribution $\mathbf{A}_{\mathbf{s}}$ over $\mathbb{Z}_q^{n+1}$ defined as $\{(\mathbf{a}, \mathbf{sa} + e \pmod{q})\}$, where $\mathbf{a}$ is uniform in $\mathbb{Z}_q^n$, e is sampled from χ_e .

Then the (decisional) Learning with Errors (LWE) problem with respect to parameters $\text{lweparams} = (n, q, \chi_s, \chi_e)$ is to distinguish A_s from the uniform distribution over $\mathbb{Z}_q^{n+1}$, given an a-priori unbounded number of samples, where s is drawn from χ_s .

We refer to χ_s as the secret distribution and to χ_e as the noise distribution.

The Module LWE problem (MLWE) was introduced by the name “Generalized LWE” in [BGV12] as a generalization of the Ring LWE problem (RLWE) [LPR10, LPR13a]. Here we use a variant that analogous to the Polynomial LWE (PLWE) problem [RSW18], and defined as follows.⁵

Definition 3.5 (Module Polynomial LWE). Let $f(x)$ be an irreducible monic polynomial of degree n , let $K = \mathbb{Q}[x]/\langle f(x) \rangle$ be a number field, define the ring $\mathcal{R} = \mathbb{Z}[x]/\langle f(x) \rangle$ and let $\mathfrak{q}$ be an ideal in this ring, denoting $\mathcal{R}_{\mathfrak{q}} = \mathcal{R}/\mathfrak{q}$. Let $r \in \mathbb{N}$, χ_s a distribution over $\mathcal{R}^r$ and χ_e a distribution over $\mathcal{R}$.

For a row vector $s \in \mathcal{R}^r$, consider the distribution A_s over $\mathcal{R}_{\mathfrak{q}}^{r+1}$ defined as $\{(a, sa + e \pmod{\mathfrak{q}})\}$, where a is uniform in $\mathcal{R}_{\mathfrak{q}}^r$, e is sampled from χ_e .

Then the (decisional) Module Polynomial LWE problem (MPLWE) with respect to parameters $\text{mplweparams} = (f, n, r, \mathfrak{q}, \chi_s, \chi_e)$ is to distinguish A_s from the uniform distribution over $\mathcal{R}_{\mathfrak{q}}^{r+1}$, given an a-priori unbounded number of samples, where s is drawn from χ_s .

Remark 3.1. The special case of MPLWE where $r = 1$ is known as Polynomial LWE (PLWE) [?].

Remark 3.2. Letting $\chi_e = t \cdot \chi_{\varepsilon}$ for a ring element t which is coprime to q , it holds that MPLWE with noise sampled from χ_e is equivalent to the setting where the noise is sampled from χ_{ε} .

Remark 3.3. In this work, we use the term MPLWE to refer to the module polynomial LWE problem. We note that in prior work [RSS17], the name MP-LWE is used for the “middle-product” LWE problem. The latter is not directly related to our work.

Starting with [Reg05] there are numerous hardness results relating the hardness of LWE to the worst-case hardness of lattice problems, for ensembles of lweparams that range asymptotically with the security parameter. Similarly, there are worst-case hardness results for RLWE [?] and MLWE [?]. There are also known methods for relating PLWE security to that of RLWE [RSW18]. These methods readily apply also to relate MPLWE security to MLWE. See Section 7 for a thorough discussion about the security of our assumptions.

⁵ The difference between our variant in MLWE is the same as the difference between PLWE and RLWE. Whereas in RLWE, MLWE the secret, noise and arithmetics are done modulo the dual ring $\mathcal{R}^{\vee}$, in PLWE and MPLWE everything is defined over $\mathcal{R}$. Indeed one has to be careful when analyzing the security of such variants which we address in the appropriate section.

4 Our Scheme

In this section, we introduce the encryption scheme, with the homomorphic properties being discussed in the subsequent section. As global parameters for our scheme, we consider the parameters $f(x) = x^n - x + 2$, $K = \mathbb{Q}[x]/\langle f(x) \rangle$, $\mathcal{R} = \mathbb{Z}[x]/\langle f(x) \rangle$, $\mathfrak{p} = \langle x - 2 \rangle$, $p = 2^n$, as defined in Section 3.2. We recall the ring isomorphism $\mathbb{Z}_p \cong \mathcal{R}_{\mathfrak{p}}$.

The ciphertext space is defined over $\mathcal{R}_q$ as defined in 3.2. Unless stated otherwise, additions and multiplications in the following sections are over $\mathcal{R}_q$. Given $a \in \mathcal{R}$, we denote by $[a]_{\mathcal{R}_q} \in \mathcal{R}$ the unique ring element with coefficients in the range $[-q/2, q/2)$ such that $[a]_{\mathcal{R}_q} = a \bmod \langle q \rangle$.

In Section 4.1 we describe an encoding - decoding procedure for elements in $\mathbb{Z}_p$ into elements in $\mathcal{R}_{\mathfrak{p}}$. We proceed by introducing our encryption scheme in Section 4.2 and analyze correctness and security in Section 4.3. Homomorphic properties are discussed in subsequent sections.

4.1 Messages vs. Plaintexts

Our scheme encrypts plaintexts μ which are elements in $\mathcal{R}_{\mathfrak{p}}$, i.e. as cosets of the ideal $\mathfrak{p}$. We use these plaintexts to encode messages $\mathbf{m} \in \mathbb{Z}_p$, where we recall that $p = (\mathcal{R} : \mathfrak{p})$.

We therefore require an efficient implementation of the ring isomorphism $\mathbb{Z}_p \cong \mathcal{R}_{\mathfrak{p}}$, in both directions, so that messages can be properly encoded and decoded. In terms of terminology, we differentiate between “*messages*” which are elements in $\mathbb{Z}_p$, and “*plaintexts*” which are elements in $\mathcal{R}_{\mathfrak{p}}$. Our encoding and decoding translate messages into plaintexts and vice versa.

We note that both $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ and $\mathcal{R}_{\mathfrak{p}} = \mathcal{R}/\mathfrak{p}$ are quotient rings, so we need to consider specific representatives that are produced by the encoding and decoding. We consider two procedures `Encode`, `Decode` that run in $\text{polylog}(p)$ time, take as input elements from $\mathbb{Z}, \mathcal{R}$ respectively, and output elements from $\mathcal{R}, \mathbb{Z}$, and implement the ring isomorphism. We further require that `Encode` produces “short” elements. Our measure of length in this context is infinity norm in the coefficient embedding. We let $B_{\text{enc}} = \max_{\mathbf{m} \in \mathbb{Z}} \|\text{Encode}(\mathbf{m})\|_{\infty}$.

Notably, in our scheme with $\mathfrak{p} = \langle x - 2 \rangle$, it is possible to achieve $B_{\text{enc}} = 1$ since any coset of $\mathcal{R}_{\mathfrak{p}}$ has a representative with $\{0, 1\}$ coefficients obtained by considering the binary representation of $\mathbf{m} \bmod p$. Namely, assume without loss of generality that $\mathbf{m} \in [0, p - 1)$ and that $\mathbf{m} = \sum_i \mu_i 2^i$, where $\mu_i \in \{0, 1\}$. Then $\text{Encode}(\mathbf{m}) = \mu(x) = \sum_i \mu_i x^i$ is a valid encoding procedure whose output only has $\{0, 1\}$ coefficients. For `Decode`, we notice that given $\mu = \sum_i \mu_i x^i$, we can output $\sum_i \mu_i 2^i \bmod p$.

We let $\mathcal{R}_{\{0,1\}}$ denote the set of plaintexts of our scheme. That is, $\mathcal{R}_{\{0,1\}}$ is the set of elements in $\mathcal{R}$ that can be represented as degree $(n - 1)$ polynomials with $\{0, 1\}$ coefficients.

We extend the `Encode`($\cdot$) function to act also on elements from $\mathcal{R}$ by taking their small-coefficient representative modulo $\mathfrak{p}$. More formally, for $\mu \in \mathcal{R}$, we define $\text{Encode}(\mu) = \text{Encode}(\text{Decode}(\mu)) \in \mathcal{R}_{\{0,1\}}$.

4.2 The Encryption Scheme

In this section we present our encryption scheme called *Ring Embedding FHE* (REFHE), based on the MPLWE hardness assumption (see Definition 3.5). We embed messages $\mathbf{m} \in \mathbb{Z}_p$ as ring elements $\mu = \text{Encode}(\mathbf{m}) \in \mathcal{R}_{\{0,1\}}$, which results in a compact ciphertext.

Conventions ε is sampled from χ_ε in $\mathcal{R}$. e is in $\mathfrak{p}$. We denote by B_{χ_ε} the bound on the ℓ_∞ of χ_ε and by B_{χ_e} the bound on the ℓ_∞ of $(x-2) \cdot \chi_e$.

Algorithm 4.1: The Encryption Scheme

1. **Setup:** $\text{pp} \leftarrow \text{REFHE.Setup}(1^n, 1^\kappa)$ gets n the degree of the polynomial, κ the security parameter and Returns $\chi_s, \chi_\varepsilon, q, r$ as pp , such that $\gcd(q, 2) = 1$, the MPLWE problem with $\text{mplweparams}(f, n, r, q, \chi_s, \chi_\varepsilon)$ is κ -hard, and $q > q_0$ as defined in lemma 4.2.
2. **Key Generation:** $(\text{pk}, \text{sk}, \text{evk}) \leftarrow \text{REFHE.Keygen}(1^\kappa, \text{pp})$. Sample row vectors $\mathbf{s}_1 \leftarrow \chi_s$, $\varepsilon \leftarrow \chi_\varepsilon^r$, and a uniformly random matrix $\mathbf{A} \in \mathcal{R}_q^{r \times r}$. Returns the secret key $\mathbf{s} = (1, -\mathbf{s}_1)$, and public key $\text{pk} = (\mathbf{b}, \mathbf{A})$ where $\mathbf{b} = \mathbf{s}_1 \mathbf{A} + (x-2)\varepsilon$. evk consists of the public parameters needed for homomorphic evaluations - relinearization key and bootstrap key, which we will describe through the paper.
3. **Encryption:** $\mathbf{c} \in \mathcal{R}_q^{r+1} \leftarrow \text{REFHE.Enc}_{\text{pk}}(\mu)$. For $\mu \in \mathcal{R}_{\{0,1\}}$, sample $\mathbf{r} \leftarrow \chi_s$, $\varepsilon_0 \leftarrow \chi_\varepsilon$, $\varepsilon_1 \leftarrow \chi_\varepsilon^r$. Then $c_0 = \mathbf{b} \cdot \mathbf{r} + (x-2)\varepsilon_0 + \mu$, $\mathbf{c}_1 = \mathbf{A}\mathbf{r} + (x-2)\varepsilon_1$. Return the column vector $\mathbf{c} = (c_0, \mathbf{c}_1)$.
4. **Decryption:** $\mu \leftarrow \text{REFHE.Dec}_{\text{sk}}(\mathbf{c})$ returns $[\mathbf{s} \cdot \mathbf{c}]_{\mathcal{R}_q} \bmod \mathfrak{p} = [c_0 - \mathbf{s}_1 \cdot \mathbf{c}_1]_{\mathcal{R}_q} \bmod \mathfrak{p}$.

When the scheme is used as an homomorphic encryption scheme, the setup and key generation phases also depend on the homomorphic capacity, meaning in which circuit evaluation the scheme supports. In this case for the key generation also generates ‘Key Switch matrices’ as seen in Algorithm 5.3. The decryption REFHE.Dec might be with respect to another modulus, as the modulus changes during the circuit evaluation due to the Key Switching and Modulus switching Algorithms 5.3, 5.5.

4.3 Correctness and Security

To keep track of the noise, we have the following definition:

Definition 4.1. Let $\mathbf{c} \in \mathcal{R}_q^{r+1}$ be a ciphertext, let $\mathbf{s} \in \mathcal{R}^{r+1}$, and let $\mu \in \mathcal{R}_p$. We define

$$\eta_{\mathbf{s}}(\mathbf{c}, \mu) = \min \{ \ell_\infty(\mu + e) : e \in \mathfrak{p}, \mu + e = \mathbf{s} \cdot \mathbf{c} \pmod{\mathfrak{q}} \}.$$

We omit the subscript when $\mathbf{s}$ is clear from the context. We also omit μ when clear from context

Lemma 4.1. *If $\eta_s(\mathbf{c}, \mu) \leq \lfloor (q-1)/2 \rfloor$, then $\text{REFHE.Dec}_s(\mathbf{c}) = \mu$.*

Proof. By the assumption there is $e \in \mathfrak{p}$ such that $\mathbf{s} \cdot \mathbf{c} = \mu + e \pmod q$. Since $\ell_\infty(\mu + e) \leq \lfloor (q-1)/2 \rfloor$ we also have:

$$[\mathbf{s} \cdot \mathbf{c}]_{\mathcal{R}_q} = \mu + e$$

Now $(\mu + e) \pmod{\mathfrak{p}} = \mu$ as we wanted.

In the following sections we will use Lemma 4.1 when analyzing correctness of homomorphic evaluation. Indeed, given a function $f : (\mathbb{Z}/p)^N \rightarrow \mathbb{Z}/p$ and input ciphertexts $\mathbf{c}_1 = \text{REFHE.Enc}(\mu_1), \dots, \mathbf{c}_N = \text{REFHE.Enc}(\mu_N)$, let $\mathbf{c}$ be the output ciphertext of the homomorphic evaluation. Then by Lemma 4.1 it suffices to bound $\eta(\mathbf{c}, \mu)$ for $\mu = \text{Encode} \circ f \circ \text{Decode}(\mu_1, \dots, \mu_N)$.

First, we analyse correctness of REFHE as an encryption scheme, without applying additional homomorphic operations:

Lemma 4.2. *Let $\mathbf{c} = \text{REFHE.Enc}_{\text{pk}}(\mu)$. Denote by $q_0/2 := (1 + 2 \cdot \hat{\gamma} \cdot \ell_1(\mathbf{s})) \cdot B_{\chi_e} + 1$. Then $\eta_s(\mathbf{c}, \mu) \leq q_0/2$.*

Proof. Notice that

$$\begin{aligned} \mathbf{s} \cdot \mathbf{c} &= \mathbf{b} \cdot \mathbf{r} + e_0 + \mu - \mathbf{s}_1 \cdot (\mathbf{A}\mathbf{r} + \mathbf{e}_1) \\ &= (\mathbf{s}_1 \mathbf{A} + \mathbf{e})\mathbf{r} + e_0 + \mu - \mathbf{s}_1 \mathbf{A}\mathbf{r} + \mathbf{s}_1 \cdot \mathbf{e}_1 \\ &= \mu + \mathbf{r} \cdot \mathbf{e} + e_0 + \mathbf{s}_1 \cdot \mathbf{e}_1 \end{aligned}$$

Where $\mathbf{s}_1, \mathbf{r} \leftarrow \chi_s$, $\mathbf{e}, \mathbf{e}_1 \in \chi_e^r$, $e_0 \in \chi_e$, and $\ell_\infty(\mu) = 1$. It follows that $\mathbf{c} \cdot \mathbf{s} = \mu + e_3$ when $e_3 \leq (1 + 2 \cdot \hat{\gamma} \cdot \ell_1(\mathbf{s})) \cdot B_{\chi_e} + 1$

Lemma 4.3. *The public-key encryption scheme REFHE is IND-CPA secure based on the MPLWE assumption (Definition 3.5) with parameters $\text{mplweparams} = (f, n, r, \mathfrak{q}, \chi_s, \chi_e)$.*

Proof (Proof Sketch.). The proof of this lemma follows the standard argument for proving security for encryption schemes based on LWE-style assumptions. See, e.g., [Reg05, LPR10]. The proof follows by a hybrid argument where we consider an adversary that is given the public key pk and a ciphertext $\mathbf{c}$ which are properly generated by the scheme and encrypt some value μ . We recall Remark 3.2 with $t = (x-2)$.

We first replace the $\mathbf{b}$ vector in the public key to be sampled uniformly rather than being computed using MPLWE. By the MPLWE assumption with secret $\mathbf{s}_1$, this hybrid is computationally indistinguishable from the original experiment.

Then we replace $\mathbf{c}$ by a uniform vector. This is computationally indistinguishable from the previous hybrid again relying on MPLWE with secret $\mathbf{r}$.

Finally, $\mathbf{c}$ is completely uniform and independent of μ , which implies the security of the scheme.

In Section 7, we relate the hardness of MPLWE to that of more commonly used assumptions in lattice-based cryptography, such as RLWE, and discuss the proper choice of parameters.

5 Homomorphic Arithmetic Operations

In this section we present our algorithms for the homomorphic evaluation of arithmetic operations: addition, multiplication by a scalar and multiplication of ciphertexts. For the sake of the multiplication operations, we also introduce our versions of the key switching and modulus switching techniques.

Homomorphic evaluation of logical operations will be implemented as a part of our bootstrapping process, see Section ?? for details.

During this section there are addition and multiplication between elements that naturally live in different spaces. Some of them are in $\mathcal{R}_q$ (the ciphertexts), some in $\mathfrak{p}$ (the errors), some in $\mathcal{R}/\mathfrak{p}$ (the messages), and some in $\mathcal{R}$ (the secret keys). Although they live in different spaces, the operations and equalities in this section are in $\mathcal{R}_q$, unless stated otherwise.

Also always $e_j \in \mathfrak{p}$, μ_j is the message.

5.1 Addition

REFHE.Add takes two ciphertexts encrypted under the same secret key $\mathbf{s}$. The addition is performed by adding the two ciphertexts (as vectors of ring elements).

Algorithm 5.1: REFHE.Add (Homomorphic Addition)

$\mathbf{c}_3 \leftarrow \text{REFHE.Add}(\mathbf{c}_1, \mathbf{c}_2)$: Output $\mathbf{c}_3 \leftarrow \mathbf{c}_1 + \mathbf{c}_2$

Lemma 5.1. For $\mathbf{c}_3 = \text{REFHE.Add}(\mathbf{c}_1, \mathbf{c}_2)$ we have $\eta(\mathbf{c}_3, \mu_1 + \mu_2) \leq \eta(\mathbf{c}_1, \mu_1) + \eta(\mathbf{c}_2, \mu_2)$.

Proof. Let $\mathbf{s} \cdot \mathbf{c}_1 = \mu_1 + e_1$, $e_1 \in \mathfrak{p}$ such that $\ell_\infty(\mu_1 + e_1) = \eta_{\mathbf{s}}(\mu_1 + e_1)$, $\mathbf{s} \cdot \mathbf{c}_2 = \mu_2 + e_2$, $e_2 \in \mathfrak{p}$ such that $\ell_\infty(\mu_2 + e_2) = \eta_{\mathbf{s}}(\mu_2 + e_2)$. Then $\mathbf{s} \cdot (\mathbf{c}_1 + \mathbf{c}_2) = \mu_1 + e_1 + \mu_2 + e_2 = (\mu_1 + \mu_2) + (e_1 + e_2)$.

5.2 Scalar Multiplication

Same as with the message, we have to encode the scalar as a polynomial that equal to the scalar mod $\mathfrak{p}$. Then multiply each coordinate of the ciphertext by it. Notice that since the norm of the encoded scalar is small, when multiplying by it the noise grows roughly by the expansions factor, which is logarithmic in the size of the plaintext space. In BGV this factor is linear in the size of the plaintext space, so for large plaintext spaces we get a significant improvement in the noise growth.

Algorithm 5.2: Multiplication by a scalar

$\mathbf{c}' \leftarrow \text{REFHE.ScalarMult}(\alpha, \mathbf{c})$: Outputs $\mathbf{c}' = \text{REFHE.Encode}(\alpha) \cdot \mathbf{c}$.

Lemma 5.2. *Let $\mathbf{c}$ be a ciphertext, then for $\mathbf{c}' = \text{REFHE.ScalarMult}(\mathbf{c}, \alpha)$ we have $\eta_{\mathbf{s}}(\mathbf{c}', \alpha \cdot \mu) \leq \gamma \cdot B_{\text{enc}} \cdot \eta_{\mathbf{s}}(\mathbf{c}, \mu)$.*

Proof. Let $\mathbf{s} \cdot \mathbf{c}' = \mu + e$ when $\ell_{\infty}(\mu + e) = \eta_{\mathbf{s}}(\mu + e)$. Observe that

$$\mathbf{s} \cdot \mathbf{c}' = \text{REFHE.Encode}(\alpha) \cdot \mu + \text{REFHE.Encode}(\alpha) \cdot e$$

Since $\ell_{\infty}(\text{REFHE.Encode}(\alpha)) \leq B_{\text{enc}}$, we have

$$\ell_{\infty}(\text{REFHE.Encode}(\alpha) \cdot \mu + \text{REFHE.Encode}(\alpha) \cdot e) \leq \gamma \cdot B_{\text{enc}} \cdot \ell_{\infty}(\mu + e)$$

Also $\text{REFHE.Encode}(\alpha) \cdot \mu + \text{REFHE.Encode}(\alpha) \cdot e \pmod{\mathbf{p}} = \alpha \cdot \text{REFHE.Decode}(\mu) \pmod{\mathbf{p}}$ as we wanted

5.3 Key switching

5.3.1 HELib inspired optimized keyswitch The following Key switching technique is based on the one presented in [HS20], and is done in order to reduce the relative noise, in comparison to the standard approach in [BGV12]. This is indeed a generalization of the BGV keyswitching, which can be obtained as a special case if $q = q'$.

Algorithm 5.3: Keyswitch

1. **Key Generation:** $\text{SwitchKeyGen}_{\mathbf{G}}(q, q', \mathbf{s}_1 \in \mathcal{R}_{q'}^{r_1}, \mathbf{s}_2 \in \mathcal{R}_{q'}^{r_2})$ for $q|q'$ outputs a matrix $\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2} \mathbf{G}$ over $\mathcal{R}_{q'}$, satisfying:

$$\mathbf{s}_2 \cdot \tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2} = T \cdot \mathbf{s}_1 \cdot \mathbf{G} + \mathbf{e}^t \pmod{q'}$$

2. **Key Switch:** $\text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1, w)$ outputs $\mathbf{c}_2 = \tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) \in (\mathcal{R}_{q'})^{r_2}$ for $T = q'/q$, and $\mathbf{c}'_1 = w \cdot \mathbf{c}_1 \pmod{q}$.

w will be chosen as an element in $\mathcal{R}_{\{0,1\}}$. In order for everything to be defined $T, \mathbf{p}$ need to be co-prime. In practice, we always take $q'|q$ odd integers, so this condition is satisfied.

Lemma 5.3. *Let $\mathbf{s}_1, \mathbf{s}_2, q, q'$ be as in $\text{SwitchKeyGen}(q, q', \mathbf{s}_1, \mathbf{s}_2)$. Let $\mathbf{c}_1 \in \mathcal{R}_q^{r_1}$, $\mathbf{c}'_1 = w \cdot \mathbf{c}_1 \pmod{q}$ and $\mathbf{c}_2 \leftarrow \text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1, w)$. Then,*

$$\mathbf{s}_2 \mathbf{c}_2 = \mathbf{e} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) + T \cdot \mathbf{s}_1 \cdot \mathbf{c}'_1 \pmod{q'}$$

Proof. Let $\mathbf{A} = \tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}$. We have:

$$\begin{aligned} \mathbf{s}_2 \cdot \mathbf{c}_2 &= \mathbf{s}_2 \cdot \mathbf{A} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) \\ &= (T \cdot \mathbf{s}_1 \cdot \mathbf{G} + \mathbf{e}) \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) \\ &= \mathbf{e} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) + T \cdot (\mathbf{s}_1 \cdot \mathbf{c}'_1 + qE) \\ &= \mathbf{e} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) + T \cdot (\mathbf{s}_1 \cdot \mathbf{c}'_1 + qE) \end{aligned}$$

When the equations are mod q' .

$$\mathbf{s}_2 \cdot \mathbf{c}_2 = \mathbf{e} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) + T \cdot \mathbf{s}_1 \cdot \mathbf{c}'_1 \pmod{q'}$$

Corollary 5.1. *Let $\mathbf{c}_1 \in \mathcal{R}_q^{r_1}$ and $\mathbf{c}_2 \leftarrow \text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1)$. Then for the powers of d gadget $\mathbf{G}_{\text{Powers}_d}$ (defined in Definition 3.3) we have that $\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}$ is in $\mathcal{R}_{q'}^{r_1 \cdot \lceil \log_d(q) \rceil \times r_2}$, and:*

$$\eta_{\mathbf{s}_2}(\mathbf{c}_2, \mu w \cdot T \pmod{\mathbf{p}}) \leq q'/q \cdot \gamma_w \cdot \eta_{\mathbf{s}_1}(\mathbf{c}_1, \mu) + r_1 \cdot \lfloor d/2 \rfloor \cdot \lceil \log_d(q) \rceil (B_{\chi_e}(q')) \cdot \gamma$$

Proof. Notice that

$$\mathbf{e} \cdot \mathbf{G}^{-1}(\mathbf{c}'_1) + T \cdot \mathbf{s}_1 \cdot \mathbf{c}'_1 \pmod{\mathbf{p}} = (Tw \cdot \mu + Tw \cdot e + e') \pmod{\mathbf{p}} = Tw\mu$$

Lemma 5.4 (Security). *For every known $\mathbf{s}_1 \in \mathcal{R}_q^{r_1}$ and a random $\mathbf{s}_2 \leftarrow \chi_s$, the $\text{SwitchKeyGen}(\mathbf{s}_1, \mathbf{s}_2)$ distribution is computationally indistinguishable from uniform.*

5.3.2 Relinearization Typically, homomorphic multiplication outputs a ciphertext encrypted under $\mathbf{s} \otimes \mathbf{s}$. In order to keep the dimension fixed, it is common to wrap this procedure with a *relinearization* procedure that switches the key from $\mathbf{s} \otimes \mathbf{s}$ back to $\mathbf{s}$. The relinearization key is $\text{SwitchKeyGen}(\mathbf{s}, \mathbf{s} \otimes \mathbf{s})$, which is considered as part of the public key. We can then invoke REFHE.SwitchKey from $\mathbf{s} \otimes \mathbf{s}$ to $\mathbf{s}$ after every multiplication.

5.4 Modulus Switching

We denote $y = \lceil a \rceil_{c+\mathbf{p}}$ the 1-rounding algorithm with respect to ℓ_∞ for the lattice $\mathbf{p}$, meaning for $a \in \mathbb{Q}[x]/f(x)$, $c \in \mathcal{R}$. finding a $y \in \mathcal{R}$ such that $(y - c) \in \mathbf{p}$, $\ell_\infty(y - a) \leq 1$.

Algorithm 5.4: Rounding

Given an input a , output $\lceil a \rceil_{c+\mathbf{p}} = \lfloor a \rfloor + \text{Encode}(c - \lfloor a \rfloor)$ where $\lfloor \cdot \rfloor$ is performed separately to each coefficient in the coefficient embedding.

Lemma 5.5. *Algorithm 5.4 is a 1-rounding algorithm to the lattice $\mathbf{p}$ with respect to ℓ_∞ . Meaning $\ell_\infty(\lceil a \rceil_{c+\mathbf{p}}) \leq 1$, $\lceil a \rceil_{c+\mathbf{p}} = c \pmod{\mathbf{p}}$*

Proof. Let $a' = \lfloor a \rfloor$, $b = c(2) - a'(2) \pmod{(2^n)}$, $c' = \lceil a \rceil_{c+\mathbf{p}}$. We need to prove that $c' - c \in \mathbf{p}$ and that $\ell_\infty(a - c') \leq 1$. For the second claim we have

$$|a^{(i)} - c'^{(i)}| = |b_i - \{a^{(i)}\}| \leq 1$$

And regarding the first claim, we have:

$$a'(2) + \sum b_i 2^i = c(2) \pmod{2^n}$$

Now using the fact that x divides $x - x^{n-1} = 2$ we get:

$$\begin{aligned} a'(2) + \sum b_i 2^i &= c(2) \pmod{x^n} \\ a'(2) + \sum b_i 2^i &= c(2) \pmod{x-2} \\ c'(x) &= a'(x) + \sum b_i x^i = c(x) \pmod{x-2} \end{aligned}$$

As desired.

Algorithm 5.5: Modulus Switching

For $\mathbf{c} \in \mathcal{R}_q^{r+1}$, $\mathbf{c}' \leftarrow \text{ModSwitch}_{q,q',w}(\mathbf{c})$, outputs $\mathbf{c}' \in \mathcal{R}_{q'}^{r+1}$: $\mathbf{c}'[i] = \lceil \frac{wq'}{q} \cdot \mathbf{c}[i] \rceil_{q^{-1}q'w\mathbf{c}[i]+\mathbf{p}}$, for $0 < i \leq r+1$, for $w \in \mathcal{R}_{\{0,1\}}$

Notice that since the ciphertext contains multiple elements from $\mathcal{R}$, the rounding is done on each element separately.

We start with a lemma on the error rate of modulus switching for (rational) integer moduli. While this is the most efficient modulus switching we have for our scheme, it provides a good sense of the underlying concepts.

Lemma 5.6. *Assume that $q, q' \in \mathbb{Z}$ and let $\mathbf{c}$ be a ciphertext. Then for $\mathbf{c}' = \text{ModSwitch}_{q,q',w}(\mathbf{c})$,*

$$\eta(\mathbf{c}', \mu \cdot \frac{wq'}{q}) \leq \gamma_w \cdot \frac{q'}{q} \eta(\mathbf{c}, \mu) + \hat{\gamma} \ell_1(\mathbf{s})$$

Proof. We have,

$$\mathbf{s} \cdot \mathbf{c} = \mu + \epsilon + qE$$

and

$$\frac{q'w}{q} \mathbf{s} \cdot \mathbf{c} = \frac{q'w}{q} \mu + \frac{q'w}{q} \epsilon + q'wE.$$

We have

$$\mathbf{s} \cdot \mathbf{c}' = \frac{q'w}{q} \mu + \frac{q'w}{q} \epsilon + q'wE + \mathbf{s} \cdot (\mathbf{c}' - \frac{q'w}{q} \mathbf{c}).$$

Denote $v = \frac{q'w}{q} \mu + \frac{q'w}{q} \epsilon + \mathbf{s} \cdot (\mathbf{c}' - \frac{q'w}{q} \mathbf{c})$. then:

$$\begin{aligned} \ell_\infty(v) &\leq \frac{q'}{q} \gamma_w \cdot \eta(\mathbf{c}, \mu) + \ell_\infty(\mathbf{s} \cdot (\mathbf{c}' - \frac{q'w}{q} \mathbf{c})) \\ &\leq \frac{q'}{q} \gamma_w \cdot \eta(\mathbf{c}, \mu) + \hat{\gamma} \cdot \ell_1(\mathbf{s}) \end{aligned}$$

we also have

$$\mathbf{s} \cdot \mathbf{c}' - q'wE = v.$$

We have that $\mathbf{s} \cdot \mathbf{c}' \bmod \mathfrak{p} = \frac{q'w}{q} \mathbf{s} \cdot \mathbf{c} \bmod \mathfrak{p}$. Therefore,

$$(\mathbf{s} \cdot \mathbf{c}' - q'wE) \bmod \mathfrak{p} = \frac{q'w}{q} (\mathbf{s} \cdot \mathbf{c} - qE) \bmod \mathfrak{p},$$

which gives us

$$v \bmod \mathfrak{p} = \frac{q'w}{q} \mu$$

So to conclude, $\mathbf{s} \cdot \mathbf{c}' \bmod q' = v$, for v s.t. $\ell_\infty(v) \leq \frac{q'w}{q} \cdot \eta(\mathbf{c}, \mu) + \hat{\gamma} \cdot \ell_1(\mathbf{s})$, and $v \bmod \mathfrak{p} = \frac{q'w}{q} \mu$. This completes the proof.

Remark 5.1. If $q = q' \bmod \mathfrak{p}$ and multiplication by w is trivial and we can see from the proof that $\eta(\mathbf{c}') \leq \frac{q'}{q} \eta(\mathbf{c}) + \hat{\gamma} \ell_1(s)$ which removes the γ_w from the noise bound. As mentioned at the beginning of this section, it may not be ideal to choose integers q and q' that are congruent modulo $\mathfrak{p}$, as this could lead to very large ciphertext moduli values. Specifically, this condition implies $q \equiv q' \pmod{p}$, resulting in $q \geq p$. This would require a much larger ciphertext modulus than what is typically used in our implementations. For that we will have to choose $w \neq 1$ and suffer from a multiplicative expansion factor in the resulting error. In order to overcome that we introduce the ideal modulus switching, which will result in working modulo $\mathfrak{q}$ which is not a rational number.

5.5 Multiplication

Algorithm 5.6: Tensor multiplication

$\mathbf{c}_3 \leftarrow \text{REFHE.TensorMult}(\mathbf{c}_1, \mathbf{c}_2)$ gets $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_q^{r+1}$ and returns $\mathbf{c}_3 = \mathbf{c}_1 \otimes \mathbf{c}_2 \in \mathcal{R}_q^{(r+1)^2}$

Given ciphertexts encrypting μ_1, μ_2 under $\mathbf{s}$, the algorithm returns ciphertext that encrypts $\mu_1 \cdot \mu_2 \bmod \mathfrak{p}$ under $\mathbf{s} \otimes \mathbf{s}$. This is formalized via the following Lemma:

Lemma 5.7. *Let $\mathbf{c}_1, \mathbf{c}_2$ be ciphertexts that encrypt μ_1, μ_2 under $\mathbf{s}_1, \mathbf{s}_2$ respectively. Then*

$$\eta_{\mathbf{s} \otimes \mathbf{s}}(\text{REFHE.TensorMult}(\mathbf{c}_1, \mathbf{c}_2), \mu_1 \cdot \mu_2) \leq \gamma \cdot \eta_{\mathbf{s}}(\mathbf{c}_1, \mu_1) \cdot \eta_{\mathbf{s}}(\mathbf{c}_2, \mu_2)$$

Proof. Let $\mathbf{s} \cdot \mathbf{c}_1 = e_1$, $\mathbf{s} \cdot \mathbf{c}_2 = e_2$, where $e_1 = \mu_1 + e'_1$, $e_2 = \mu_2 + e'_2$, $e'_1, e'_2 \in \mathfrak{p}$ and $\eta_{\mathbf{s}}(\mathbf{c}_1, \mu_1) = \ell_\infty(e_1)$, $\eta_{\mathbf{s}}(\mathbf{c}_2, \mu_2) = \ell_\infty(e_2)$. Observe that

$$(\mathbf{s} \otimes \mathbf{s}) \cdot (\mathbf{c}_1 \otimes \mathbf{c}_2) = (\mathbf{s} \cdot \mathbf{c}_1) \cdot (\mathbf{s} \cdot \mathbf{c}_2) = e_1 \cdot e_2$$

and $e_1 \cdot e_2 \bmod \mathfrak{p} = (\mu_1 + e'_1) \cdot (\mu_2 + e'_2) \bmod \mathfrak{p} = \mu_1 \cdot \mu_2$. The lemma follows.

The dimension of $\text{REFHE.TensorMult}(\mathbf{c}_1, \mathbf{c}_2)$ is squared compared to the input ciphertexts. Therefore, when multiplying we perform $\mathbf{c}_4 = \text{REFHE.SwitchKey}(\tau_{\mathbf{s} \otimes \mathbf{s} \rightarrow \mathbf{s}}, \mathbf{c}_3)$ and then $\mathbf{c}_5 = \text{ModSwitch}_{q,q',w}(\mathbf{c}_4)$ for q, q' determined in the scheme parameters, in order to reduce the noise.

Algorithm 5.7: Multiplication

$\mathbf{c}_3 \leftarrow \text{REFHE.Mult}(\mathbf{c}_1, \mathbf{c}_2)$ gets $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_q^{r+1}$, sets
 $\mathbf{c}_4 = \text{REFHE.SwitchKey}(\tau_{\mathbf{s} \otimes \mathbf{s} \rightarrow \mathbf{s}}, \text{REFHE.TensorMult}(\mathbf{c}_1, \mathbf{c}_2), w)$
 and returns
 $\mathbf{c}_3 = \text{ModSwitch}_{q,q',w'}(\mathbf{c}_4)$

Notice that w, w' can be chosen in $\mathcal{R}_{\{0,1\}}$ such that $\mathbf{c}_3$ encrypts the multiplication of the messages $\mathbf{c}_1, \mathbf{c}_2$ encrypt (given that the noise is small), and not a scalar multiplication of it.

Moduli ladder As in [BGV12], the setup phase of the scheme in algorithm 4.1 produces also a “ladder” of the cipher text moduli that will be used during the evaluation of the scheme. Notice that after each multiplication we perform key switch and then modulus switch, so the evaluation key should consist of a Public keyswitch matrix for each multiplication before the bootstrapping, and the relevant modulus.

5.6 Optimizations

Optimization for keyswitch and modulus switch

Notice that there is a multiplication by a scalar in both algorithms: multiplying $\mathbf{c}_1$ by w in Algorithm ??, and multiplying $\mathbf{c}$ by w in Algorithm ??. It turns out that we can ignore these scalar multiplications and account for them only in the final step. Specifically, by skipping these scalar multiplications, i.e., setting $w = w' = 1$, and given bounds on $\eta_{\mathbf{s}}(\mathbf{c}_1, \mu_1)$ and $\eta_{\mathbf{s}}(\mathbf{c}_2, \mu_2)$, Algorithm 5.7 provides a bound on $\eta_{\mathbf{s}}(\mathbf{c}_3, \mu_1 \cdot \mu_2 \cdot \alpha)$, where α is determined solely by the evaluated circuit, and the chosen primes. This means that the multiplication algorithm effectively becomes a composition of multiplication and a multiplication by a known constant scalar α .

We now claim that every circuit can be evaluated using this modified multiplication instead of the regular one. Indeed, we take the same circuit—with the modified multiplication in place of the standard one, and associate with each cell in the circuit a scalar $\alpha_l \in \mathbb{Z}_p$. Let circuit 1 represent the circuit using the “new” multiplication and circuit 2 the one using standard multiplication. Before computation, we can determine the scalar α_l for each cell l , which is the ratio of the value of the cell in circuit 1 to the value in circuit 2 (and we need to prove that this ratio is independent of the inputs to the circuit).

Notice that when performing a homomorphic evaluation, we generally work with layered circuits, where each cell corresponds to a layer that reflects the

depth of multiplications required to reach that cell. Homomorphic operations have inputs only from the same layer, as the inputs must have the same key and modulus, which are determined by the depth of multiplications. Returning to our claim, we wish to show that all the cells in the same layer have the same scalar multiplier. Indeed, we need to explain that for multiplication, addition, and scalar multiplication. For multiplication it is clear: $\alpha_l \cdot m_1 \times \alpha_l m_2 \times \alpha_{l_3} = \alpha_{l+1} \times m_1 \times m_2$ where α_{l+1} depends only on α_l, α . For addition: $\alpha_l \cdot (m_1 + m_2) = \alpha_l \cdot m_1 + \alpha_l \cdot m_2$. And for multiplication by scalar $\alpha' \cdot (\alpha_l \cdot m) = \alpha_l \cdot (\alpha' \cdot m)$.

Notice that without multiplying by scalar we have

$$\eta(\text{ModSwitch}'(\mathbf{c}), \mu) \leq \frac{q'}{q} \eta(\mathbf{c}, \mu) + \hat{\gamma} \ell_1(\mathbf{s})$$

$$\eta_{s_2}(\text{SwitchKey}'(\mathbf{c}, \tau_{s_1 \rightarrow s_2}), \mu) \leq q'/q \cdot \eta_{s_1}(\mathbf{c}, \mu) + r_1 \cdot \lfloor d/2 \rfloor \cdot \lceil \log_d(q) \rceil B_{\chi_e}(q') \cdot \gamma$$

There are several ways to take advantage of this, depending on the specific usecase:

- The first and simplest way is to always use the modulus switching and key switching without the multiplication by scalar, and only in the last modswitch before bootstrapping multiply by a scalar w , that instead of being chosen as equal to $q'/q \bmod \mathfrak{p}$ it will be chosen such that the associated scalar with the new cell is 1.
- Another way is that the decryption algorithm will include also division by the relevant scalar $\bmod \mathfrak{p}$. The issue with this approach is that it can't be used during binary operations during bootstrapping.
- We can also try to affect this factor during encryption, middle steps, or choosing of moduli ladder, in order for the final multiplier to have an inverse with small l_1 norm.

Tradeoff between n and r Security of the scheme depends on $n \cdot r$. While working with a ciphertext $\mathbb{Z}/2^{n_1}$, f can be every $n \geq n_1$. In practice when $n_1 = 64$ we work with $n > 64$ in order to reduce the size of the Key Switching matrix in Algorithm 5.3.

6 Bootstrapping and Boolean Operations

We now present our bootstrapping algorithm. We start by introducing a generic notation that will be used throughout this section. In the course of the bootstrapping we switch between a number of forms of algebraic and non-algebraic LWE. All of these schemes have a very similar syntax, namely a linear decryption over some ring results in an encoding of a plaintext with some noise. To capture this, we denote by $\text{LWE}_{q,s}^{a,b}(\mu; \varepsilon)$ the set of vectors of elements in $\mathbb{Z}_q$ such that for a secret key $\mathbf{s}$ it holds that

$$\mathbf{s} \cdot \mathbf{c} = b\mu + a\varepsilon \pmod{q}, \quad a, b, \varepsilon \in \mathbb{Z}_q.$$

Intuitively $\text{LWE}_{q,s}^{a,b}(\mu; \varepsilon)$ can be thought of as a set of ciphertexts and related objects (i.e. modulus-switching parameters) generated by a scheme which is based on the LWE assumption. We therefore refer to such objects as “LWE ciphertexts”.

Similarly, we denote by $\text{MPLWE}_{q,s}^{a,b}(\mu; \varepsilon)$ the set of vectors of elements in some polynomial ring $\mathcal{R}$ such that for a secret key $\mathbf{s}$ it holds that

$$\mathbf{s} \cdot \mathbf{c} = b\mu + a\varepsilon \pmod{q}, \quad a, b \in \mathcal{R}_q.$$

$\text{MPLWE}_{q,s}^{a,b}(\mu; \varepsilon)$ can be thought of as the analogous notion to $\text{MPLWE}_{q,s}^{a,b}(\mu; \varepsilon)$ but for schemes based on the MPLWE assumption. We therefore refer to such objects as “MPLWE ciphertexts”.

We then denote $\text{BGV}_{q,s}(\mu; \varepsilon)$, $\text{BFV}_{q,s}(\mu; \varepsilon)$, $\text{REFHE}_{q,s}(\mu; \varepsilon)$ to mean $\text{LWE}_{q,s}^{2,1}(\mu; \varepsilon)$, $\text{LWE}_{q,s}^{1,\frac{q}{2}}(\mu; \varepsilon)$, $\text{MPLWE}_{q,s}^{(x-2),1}(\mu; \varepsilon)$ respectively, which in the same manner can intuitively be thought of as BGV, BFV, REFHE ciphertexts. We occasionally omit one or more of the parameters $\mathbf{s}, a, b$ when they are clear from the context. Also, we use $q, Q \in \mathbb{Z}$ to denote ciphertext space moduli. Q is used to denote the modulus of a newly encrypted REFHE ciphertext, as opposed to q which denotes a modulus of a ciphertext that is the result of homomorphic operations and has potentially consumed ciphertext levels reaching to a level for which the modulus is q .

The rest of this section is organized as follows. In Section 6.1 we cover the subroutines in more depth, and in Section 6.2 we present our bootstrap algorithm in detail. The description above is provided mainly for intuition. In Section we show how logical operations can be performed homomorphically during bootstrap.

6.1 Subroutines of the Bootstrapping Algorithm

6.1.1 Programmable Bootstrapping for B/FV This is the procedure from [KDE⁺24] that allows to use the programmable bootstrapping of the TFHE scheme in order to bootstrap schemes that are based on the BGV and B/FV paradigms.

- Setup algorithm $\text{PB.Setup}(1^n, 1^q, \mathbf{s}, 1^{n'}, q')$ that takes as input the security parameter, as well as the following parameters. LWE dimension and modulus parameters (n, q) , both given in unary representation, and a corresponding binary LWE secret key $\mathbf{s} \in \{0, 1\}^n$. Additional LWE dimension and modulus parameters (n', q') , where n' is given in unary representation, as well as a vector $\tilde{\mathbf{c}}$ of Ring-LWE zero-encryptions with noise bound B . The algorithm outputs the programmable bootstrapping public parameters pbpp of bit length $\text{poly}(n, q, n', \log q')$.
- Programmable Bootstrapping Algorithm $\text{PB.Bootstrap}(\text{pbpp}, \mathbf{c}, f)$ which takes as input the programmable bootstrapping parameters pbpp an LWE ciphertext $\mathbf{c}$ with dimension n and modulus q , a function $f : \mathbb{Z}_q \rightarrow \mathbb{Z}_{q'}$ represented

by its truth table. It outputs $\mathbf{c}'$, an LWE ciphertext with dimension n' and modulus q' .

We overload the notation of PB.Bootstrap with respect to the last operand.

Let F be a matrix of functions: $F \in (\mathbb{Z}_q \rightarrow \mathbb{Z}_{q'})^{m_1 \times m_2}$, then $\text{PB.Bootstrap}(\text{pbpp}, \mathbf{c}, F)$ is a shorthand for the procedure that applies $\text{PB.Bootstrap}(\text{pbpp}, \mathbf{c}, F_{i,j})$ for every entry of F and outputs a matrix $m_1 \times m_2$ of LWE ciphertexts corresponding to the outputs of the executions.

Theorem 6.1 (Programmable Bootstrapping [KDE⁺24]). *There exist polynomial time algorithms with syntax as above, with the following properties:*

- **Correctness.** *Letting $\text{pbpp} = \text{PB.Setup}(1^n, 1^q, \mathbf{s}, 1^{n'}, q', \tilde{\mathbf{c}})$ and then computing $\mathbf{c}' = \text{PB.Bootstrap}(\text{pbpp}, \mathbf{c}, f)$, it holds that*

$$\mathbf{s}' \cdot \mathbf{c}' \pmod{q'} = f(\mathbf{s} \cdot \mathbf{c}) + e, \quad (5)$$

where $|e| \leq B_{PB} = B_{\chi_e} \cdot \text{poly}(n, \log q, n', \log q')$, for some (fixed) polynomial.

- **Security.** *If the elements of $\tilde{\mathbf{c}}$ are computationally indistinguishable from uniform, then pbpp can be simulated without any knowledge of $\mathbf{s}$.*

6.1.2 Sample Extract We next describe how to extract a LWE sample encrypting the free coefficient (recall that we face the constraint of being able to only use the free coefficient in our bootstrapping mechanism, the reason for which is explained in Section 2) of the input REFHE ciphertext as a BGV ciphertext. For a field element $t \in K$ we consider the operator $J : K \rightarrow \mathbb{Q}$ to be the operator that outputs the free coefficient $u \in \mathbb{Q}$, of the multiplication of t by a field element $w \in K$. Note that if we consider such a multiplication of elements in the ring $\mathcal{R}$ the result is in $\mathbb{Z}$ and hence, viewed as a linear operator acting on the coefficient embedding of $w \in \mathcal{R}$, its output is a coefficient vector in $\mathbb{Z}^n$. Extending J to operate on vectors of such ring elements $\mathcal{R}^r$ we take the output to be a concatenation of the resulting coefficient vectors, one for each of the ring elements, and denote it formally as $J : \mathcal{R}^r \rightarrow \mathbb{Z}^{nr}$.

The extraction of the first coefficient of a REFHE ciphertext $\mathbf{c} = (c_0, \mathbf{c}_1)$ encrypted under the secret key $\mathbf{s} = (1, -\mathbf{s}_1)$ is simply $([c_0]_0, J(\mathbf{c}_1))$, which will output an BGV ciphertext encrypting the same message under the extracted secret key, namely the coefficient embedding of $\mathbf{s}_1$.

After extraction we switch the output BGV ciphertext to a BFV ciphertext. This is done for two reasons: the first is that TFHE requires a modulus switching to one that is a power of 2 which is not possible for BGV since BGV requires q and p to be coprime, and the second is that during programmable bootstrapping we evaluate a function that takes as input a ciphertext in which the noise term e is not multiplied by a coefficient as in BFV.

Starting with a BGV ciphertext $\mathbf{c}$ s.t. $\mathbf{s} \cdot \mathbf{c} = \mu + 2\varepsilon \pmod{q}$, we switch it to a BFV ciphertext $\mathbf{c}'$ by multiplying by $2^{-1} \pmod{q} = (q+1)/2$, so we have:

$$\mathbf{c}' = \frac{q+1}{2} \mathbf{c}$$

and get:

$$\begin{aligned} \mathbf{s} \cdot \mathbf{c}' &= \frac{q+1}{2} \cdot \mu + \frac{q+1}{2} \cdot \varepsilon \pmod{q} \\ &= \frac{q+1}{2} \cdot \mu + (q+1) \cdot \varepsilon \pmod{q} \end{aligned}$$

Since we are performing modular arithmetic modulus q , we can omit the term $q \cdot \varepsilon$ and obtain:

$$\mathbf{s} \cdot \mathbf{c}' = \frac{q+1}{2} \cdot \mu + \varepsilon \pmod{q}$$

which is indeed a BFV ciphertext.

Algorithm 6.1: Sample Extract

Input: $\mathbf{c} = (c_0, \mathbf{c}_1) \in \mathcal{R}_q^{r+1}$ a REFHE ciphertext encrypted under the secret key $\mathbf{s} = (1, -\mathbf{s}_1)$
Output: $c_{\text{BFV}} \in \mathbb{Z}^{nr+1}$ a BFV ciphertext encrypting the first coefficient of $\mathbf{c}$ under the secret key $(1, -[\mathbf{s}_1])$ (recall that $[\mathbf{s}_1]$ is the coefficient vector of $\mathbf{s}_1$).
 $c' \leftarrow ([c_0]_0, J(\mathbf{c}_1))$
return $c_{\text{BFV}} \leftarrow \frac{q+1}{2} \cdot c'$

6.1.3 Repacking with Key Switching *[revisited version]*

The aim of the Repacking procedure is to construct a MPLWE ciphertext from the n LWE encryptions of its message polynomial. This means converting each of them into a MPLWE ciphertext encrypting the same message multiplied by the corresponding power of the formal variable x^i , and sum them up in the end. We next show how to accomplish this using key-switching. We start with a vector of the n LWE ciphertexts: $(\mathbf{c}_0, \dots, \mathbf{c}_{n-1})$, each such that:

$$\mathbf{s} \cdot \mathbf{c}_i = [\alpha(x)]_i + \varepsilon_i \pmod{Q}$$

however, each $\mathbf{c}_i$ is a potentially large vector, so we binary-decompose it $\mathbf{c}_i = \sum_{j,l} c_i[j]_l 2^l$ and obtain:

$$\left(\sum_{l,j} 2^l \cdot s_j \right) \cdot \left(\sum_i c_i[j]_l \right) = [\alpha(x)]_i + \varepsilon_i \pmod{Q, f}$$

At this stage, since the terms s_j which are elements of the private key and hence are obviously not given, we substitute the term $2^l s_j$ with the key-switching keys (which are created at Section 6.2.1). Recall that the key-switching keys are

MPLWE encryptions of powers of 2 multiplied by the secret key elements over the scheme's ring under a different key denoted here as $\hat{\mathbf{s}}$, and so are of the form:

$$a_{j,l} + \mathbf{b}_{j,l}\hat{\mathbf{s}} = 2^l s_j + \hat{\varepsilon}(x) \pmod{Q, f}$$

we substitute for $2^l s_j$ in the last equation and denote the resulting MPLWE (in the scheme's ring $R = \mathbb{Z}[x]/\langle f(x) \rangle$) as $(d_i[0], \mathbf{d}_i[1]) \in \mathcal{R}_Q^{r+1}$

$$\begin{aligned} (d_i[0], \mathbf{d}_i[1]) &= \sum_{j,l} c_i[j]_l \cdot (a_{j,l}, \mathbf{b}_{j,l}) \\ d_i[0] + \mathbf{d}_i[1]\hat{\mathbf{s}} &= \sum_{j,l} c_i[j]_l (a_{j,l} + \mathbf{b}_{j,l} \cdot \hat{\mathbf{s}}) = \sum_{j,l} c_i[j]_l [2^l \cdot s_j + \hat{\varepsilon}_{j,l}(x)] \\ &= [\alpha(x)]_i + \varepsilon_i + \sum_{j,l} c_i[j]_l \cdot \hat{\varepsilon}_{j,l}(x) \end{aligned}$$

we now denote $d_0 = \sum_i d_i[0]x^i$, $\mathbf{d}_1 = \sum_i \mathbf{d}_i[1]x^i$ and obtain:

$$d_0 + \mathbf{d}_1\hat{\mathbf{s}} = \sum_i (d_i[0] + \mathbf{d}_i[1]\hat{\mathbf{s}})x^i = \alpha(x) + \varepsilon(x) + \sum_{i,j,l} c_i[j]_l \cdot \hat{\varepsilon}_{j,l}(x)x^i \in \mathcal{R}_Q$$

Algorithm 6.2: Repackaging

Input: $\mathbf{ksk} \in \mathcal{R}_Q^{r \times n \lceil \log_2 Q \rceil}$ key-switching keys as REFHE ciphertexts
 $\mathbf{c}$ a vector of n LWE ciphertexts
Output: a MPLWE under the ring $\mathcal{R}_Q^{r+1}$ to be constructed in which each of the coefficients correspond to one of the n LWE ciphertexts recieved as input
for $i = 0, \dots, n-1$ **do**
 $\tilde{c}_j \leftarrow \text{Decomp}(c_i[j], 2)$
 $d_i = \sum_{j,l} \tilde{c}_{j,l} \cdot \mathbf{ksk}_{i,l}$
return $\sum_{i=0}^n d_i x^i$

6.2 Bootstrapping Procedure and Analysis

6.2.1 Bootstrapping Setup The procedure REFHE.BS.Setup defines public parameters bspp , which include: a function D the function we evaluate during bootstrapping, gadget decomposition elements (as defined in Section 3.4); a decomposition base B used for the decomposition, gadget vector: $\mathbf{g} = \mathbf{g}_{B,Q} = (1, B^1, \dots, B^{\lceil \log_B Q \rceil})$, gadget matrix $\mathbf{G} = \mathbf{G}_{\mathbf{g}} = \mathbf{I}_k \otimes \mathbf{g}$ and a gadget decomposition operation $\mathbf{G}^{-1} = \mathbf{G}_{\mathbf{g}}^{-1}$, key-switching keys; for a secret key $\mathbf{s}$ generated by REFHE.Keygen as in Algorithm 4.1 in Section 4.2 a vector of key-switching keys consists of encryptions of the different powers of 2 multiplied by the bits of

the secret key $\mathbf{s}$, namely a key-switching key $\mathbf{ksk}_{j,l}$ is in a REFHE encryption of $2^l \cdot \mathbf{s}_j$ where $0 \leq l < \lfloor \log_2 Q \rfloor$, TFHE bootstrapping parameters; $\mathbf{tfbspp}$ output by the algorithm TF.Bootstrap as in Section 6.1.1

6.2.2 Bootstrapping Algorithm

Definition 6.1. For $q, Q \in \mathbb{N}$ we define $D : \mathbb{Z}_q \rightarrow \mathbb{Z}_Q$ to be the function that extracts the most significant bit of its input. Specifically $D(z)$ represents z as $(q/2)\mu + \varepsilon$ where ε has the smallest possible absolute value and $\mu \in \{0, 1\}$, and outputs μ .

Algorithm 6.3: Bootstrapping

Input: $c_{\text{IN}} = \text{REFHE}_q(\mu; \varepsilon)$ REFHE ciphertext to be bootstrapped,
 $\mathbf{bspp}$ bootstrapping parameters obtained at scheme setup,
Parameters: D bootstrapping function, $\mathbf{g}$ gadget vector, n security parameter, q ciphertext modulus, Q ciphertext modulus of freshly bootstrapped ciphertexts
Output: A REFHE ciphertext with dimension n and modulus Q

- 1: $c_{\ll} \leftarrow c' \leftarrow c_{\text{IN}}$
- 2: **for** $i = 0, \dots, n-1$ **do**
- 3: $c_{\text{lsb}} \leftarrow \text{REFHE.SampleExtract}(c_{\ll}) \quad \triangleright c_{\text{lsb}} = \text{BFV}_q(\mu_0; \varepsilon)$
- 4: $\tilde{\mathbf{c}} \leftarrow \text{TF.Bootstrap}(\mathbf{tfbspp}, c_{\text{lsb}}, \mathbf{g} \cdot D) \quad \triangleright D$ per Definition 6.1
- 5: $\tilde{\mathbf{c}}_i \leftarrow \tilde{\mathbf{c}} \cdot \mathbf{G}_{B,Q}^{-1}(\lfloor \frac{x^i}{x-2} \rfloor)$
- 6: $c_{\text{MPLWE}}^i \leftarrow \text{REFHE.Repack}(n, Q, \mathbf{pbpp.ksk}, \tilde{\mathbf{c}}_i)$
- 7: $c_{\text{REFHE}}^i \leftarrow (x-2) \cdot c_{\text{MPLWE}}^i \quad \triangleright c_{\text{REFHE}}^i = \text{REFHE}_Q(\mu_i x^i; \varepsilon''')$
- 8: $c' \leftarrow c' - \text{ModSwitch}(c_{\text{REFHE}}^i, q)$
- 9: $c_{\ll} \leftarrow c' \cdot x^{-i} \pmod{q}$
- 10: **return** $\sum_{i=0}^{n-1} c_{\text{REFHE}}^i$

6.2.3 Correctness and Security Analysis For the purpose of correctness analysis we consider an execution of the bootstrapping algorithm (Algorithm 6.3) on inputs $\mathbf{bspp} = \text{REFHE.BS.Setup}(1^n, 1^q, \mathbf{s}, 1^{n'}, Q, \mathbf{c})$, $c_{\text{IN}} \in \text{REFHE}_q(\mu; \varepsilon)$, $|\varepsilon| \leq ?$.

We recall Theorem ??

Lemma 6.1 (Blind Rotation). For all i , consider $\tilde{\mathbf{c}}_i$, computed in Line 4 of Algorithm 6.3. If we denote $\tilde{\mathbf{c}}_i = (\mathbf{c}'_0, \dots, \mathbf{c}'_l)$ then it holds that

$$\mathbf{c}'_j \in \text{BFV}(B^j \cdot \mu; \varepsilon'_j), \quad 0 \leq j < l \quad (6)$$

where $|\varepsilon'_j| \leq B_{PB} \cdot \text{poly}(n, \log(q), n', \log(Q))$.

6.3 Boolean Operations

We now explain how to use our bootstrapping framework in order to perform boolean operations on the ciphertext in the course of bootstrapping. Our solution hinges on the property that the coefficients of the encoded plaintext μ are the bits of the message $\mathbf{m}$, and those are the values that are recovered in the course of our bootstrapping procedure. We start by considering boolean operations that permute ciphertext bits, and move on to logical operations.

Permuting Ciphertext Bits. In Steps 5, 6, 7 of Algorithm 6.3 we construct $c_{\text{REFHE}}^i = \text{REFHE}_Q(\mu_i x^i; \varepsilon''')$. Let us consider a slight change to Step 5 and set $\tilde{c}_i \leftarrow \tilde{c} \cdot \mathbf{G}_{B,Q}^{-1}(\lfloor \frac{1}{x-2} \rfloor)$, that is, the operand of $\mathbf{G}_{B,Q}^{-1}(\cdot)$ becomes independent of i . In this case we get $c_{\text{REFHE}}^i = \text{REFHE}_Q(\mu_i; \varepsilon''')$ instead of the above. However, we note that this is almost as good, since we can always multiply post-facto by x^j , for any j that we want, to obtain $c_{\text{REFHE}}^{i,j} = \text{REFHE}_Q(\mu_i x^j; \varepsilon''')$, at the cost of a factor $\hat{\gamma} \leq 3$ increase in the noise bound. In particular, $c_{\text{REFHE}}^{i,i}$ would be the value c_{REFHE}^i from Step 7 in the algorithm, which allows to continue the execution of the bootstrapping loop as before.

After the end of the execution of the loop, we can compute immediately any $c_{\text{REFHE}}^{i,j}$ that we want. Therefore, any shuffling of the bits may be implemented seamlessly. Let $\phi : \{0, \dots, n-1\} \rightarrow \{0, \dots, n-1, \perp\}$ be any function, then we can return, in Step 10 of Algorithm 6.3, the value $\sum_{i=0}^{n-1} c_{\text{REFHE}}^{\phi(i),i}$, where we syntactically define $c_{\text{REFHE}}^{\perp,i}$ to indicate 0. By properly choosing ϕ , it is possible to implement circular/non-circular shifts (e.g. $\phi(i) = i+1 \pmod{n}$ is a single-slot cyclic left shift), apply bit-masks, duplicate values, extract specific bits and perform similar operations. It is also possible to output more than one output ciphertext, e.g. store the lower $n/2$ bits in one ciphertext and the upper $n/2$ bits in another.

All of these operations incur a minimal penalty in terms of noise and runtime compared to ordinary bootstrapping.

Comparisons. An additional class of logical operations is producing a boolean value which corresponds to the truth value of some numerical comparison. It suffices to consider the setting where we have two input ciphertexts c_1, c_2 encrypting messages $\mathbf{m}_1, \mathbf{m}_2$, and we wish to recover c' that encrypts the value 1 if $\mathbf{m}_1 > \mathbf{m}_2$, and 0 otherwise. This is of course instrumental for branching operations, loop variables and such. This can be achieved straightforwardly using our techniques. First we can use arithmetic homomorphism to subtract $c_3 = c_2 - c_1$. Then we can bootstrap c_3 and extract a ciphertext containing only the most significant bit of the message encrypted by c_3 . Indeed, by standard boolean logic (two's complement representation), the most significant bit is 1 if and only if the output is negative, which is the case if and only if $\mathbf{m}_1 > \mathbf{m}_2$.

Multi-Bit Boolean Operations. Let us now consider operations that are performed over multiple bits. We note that using the above, equality to 0 can be tested by checking that both $\mathbf{m} > -1$ and $\mathbf{m} < 1$, which can be done using two parallel bootstrapping sessions. Equality to 0 is also the n -bit NOR operation. Similarly other logical operators over the n bits can be implemented.

We may then turn our attention to performing bitwise AND, say between two numbers. That is, we have two plaintexts μ, μ' , and we want to recover μ'' so that $\mu''_i = \mu_i \cdot \mu'_i$ for all i . This requires a bit more work, and we describe one possible method for performing this operation using a constant number of bootstrapping operations. First, we use a shuffling subroutine to split each input operand into two ciphertexts, where all even bits are set to 0, and the odd bits correspond to the odd bits of the original ciphertexts. That is, we have (from LSB to MSB): $\mu^{\text{lower}} = (\mu_0 0 \mu_1 0 \dots \mu_{n/2} 0)$, $\mu^{\text{upper}} = (\mu_{n/2+1} 0 \dots \mu_{n-1} 0)$, and likewise for μ' . We then add $c^{\text{lower}} = c^{\text{lower}} + c^{\text{lower}}$, $c^{\text{upper}} = c^{\text{upper}} + c^{\text{upper}}$, and notice that the *even* bits of the $c^{\text{lower}}, c^{\text{upper}}$ are exactly the bits of μ'' . Another bootstrapping operation will allow to reorganize the bits and obtain a c'' that encrypts μ'' as desired.

7 Security of MPLWE in Our Ring

In the section we discuss the hardness of the MPLWE problem (Definition 3.5). We do so by first explaining how it relates to other algebraic variants of the Learning With Errors problems. Later we argue about the security in our specific ring, defined by polynomials of the form $f(x) = x^n - x + 2$.

7.1 Algebraic Variants of LWE and Their Relations

Algebraic variants of LWE work over a number field K . The Ring LWE (RLWE) problem [?] is defined over the so-called ring of integers of K , whereas the Order LWE problem (OLWE) [?] is more general and can be instantiated over any full rank subring of the ring of integers (such subrings are called “orders”, and the ring of integers itself is the maximal order). Also relevant to our work is the Polynomial LWE (PLWE) problem [?] which has a somewhat simpler definition. In PLWE, all arithmetics are done in the ring itself (rather than the “dual ring” which can be thought of as a fraction over the ring), and noise is generated in the coefficient embedding (rather than the canonical embedding). We provide some formal definitions below.

Let $f(x)$ be an irreducible monic polynomial of degree n , let $K = \mathbb{Q}[x]/\langle f(x) \rangle$ be a number field, define the ring $\mathcal{R} = \mathbb{Z}[x]/\langle f(x) \rangle$, $\mathcal{O}_K$ to be its ring of integers of the number field.

Definition 7.1 (Module Order LWE). Let $\mathcal{O}$ be an order of K and $\mathfrak{q}$ be an ideal in the order. Denoting $\mathcal{O}_{\mathfrak{q}} = \mathcal{O}/\mathfrak{q}$. Let $r \in \mathbb{N}$, χ_s a distribution over $(\mathcal{O}^\vee)^r$ and χ_ε a distribution over $\mathcal{O}^\vee$.

For a row vector $\mathbf{s} \leftarrow \chi_s$, consider the distribution $A_{\mathbf{s}}$ over $\mathcal{O}_{\mathfrak{q}}^r \times \mathcal{O}_{\mathfrak{q}}^\vee$ defined as $\{(\mathbf{a}, \mathbf{s}\mathbf{a} + \varepsilon \pmod{\mathfrak{q}})\}$, where $\mathbf{a}$ is uniform in $\mathcal{O}_{\mathfrak{q}}^r$, ε is sampled from χ_ε .

Then the (decisional) Module Order LWE problem (MOLWE) with respect to parameters $\text{molweparams} = (f, n, \mathcal{O}, r, \mathfrak{q}, \chi_s, \chi_\varepsilon)$ is to distinguish $A_{\mathbf{s}}$ from the uniform distribution over $\mathcal{O}_{\mathfrak{q}}^r \times \mathcal{O}_{\mathfrak{q}}^\vee$, given an a-priori unbounded number of samples, where $\mathbf{s}$ is drawn from χ_s .

We point out a few important special cases:

1. The case $\mathcal{O} = \mathcal{O}^\vee = \mathcal{O}_K$ is called *Module-LWE (MLWE)*.
2. The case $r = 1$ is called *Order-LWE (OLWE)*.
3. The case where both $\mathcal{O} = \mathcal{O}^\vee = \mathcal{O}_K$ and $r = 1$ is called *Ring-LWE (RLWE)*.

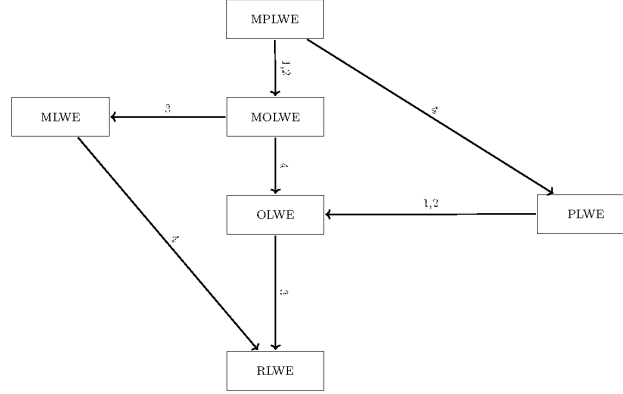


Fig. 1. Reduction relations between various relevant Algebraic LWE problems. The numbering refers to the different security issues 1, 2, 3 and 4 numbered below.

Known connections between the problems are summarized in Figure 1. The numbers on the arrows correspond to the following reductions.

1. **Coefficient vs. Canonical Sampling** [RSW18]. Let X be a Gaussian random variable with expected value μ and co-variance matrix Σ and V some linear transformation. Then VX is also a Gaussian random variable with expected value $V\mu$ and co-variance matrix $V^t\Sigma V$. In particular when moving between the coefficient and canonical embedding when uses the Vandermonde matrix V_f defined by the complex roots of f . Thus in order to control the error after the transformation one has to analyze the Singular Value Decomposition (SVD) of the matrix V_f .
2. **The Dual Ring** [RSW18]. Every order is a full rank lattice and thus one can define the dual of this lattice as $\mathcal{O}^\vee = \{\alpha \in K : \text{Tr}_{K/\mathbb{Q}}(\alpha\mathcal{O}) \subseteq \mathbb{Z}\}$. The trick is to multiply by an element in the co-different ideal [Con09], this takes an instance from the dual ring to the ring itself. When this is done the error growth by the norm of this element. As it turns out understanding the dual ring and finding an element of small norm there is rather difficult. In [RSW18] the authors show that it is always possible to find such element in non-uniform time. Another options is to note that $f'(x)$ is always a member of the different ideal (meaning that $f'(x)^{-1}$ is in the co-different ideal)[Cor 3.5 [Con09]] which in some cases may give a decent reduction.

⁶ Implicit.

3. **The Order $\mathbb{Z}[x]/\langle f(x) \rangle$ vs. The Maximal Order [RSW18].** For any order $\mathcal{O}$ of K one can define the conductor ideal $\mathcal{C}_{\mathcal{O}} = \{x \in K : x\mathcal{O}_K \subseteq \mathcal{O}\}$. Multiplying by an element of the conductor can move a RLWE instance into an OLWE instance, again increasing the error by its norm. In a similar way to the co-different ideal [RSW18] shows it is always possible to find a small element in the conductor or use $f'(x)$.
4. **Module to Ring Reductions [PP24].** The reduction here looks at a order $\mathcal{O}'$ of higher degree in a field extension of K'/K such that it is also a module over the base order $\mathcal{O}$. The reduction maintains the "total degree" (The degree of the module times the degree of the ring) and works for a wide class of extension fields.

7.2 Security in Our Ring Compared to a Cyclotomic

It is a common choice to take K to be the cyclotomic field, this is done as many structural properties are known about them. This enables faster computations [LPR13b] but also simplifies some of the security assumptions. Namely cyclotomic polynomials are monogenic, meaning that $\mathcal{O}_K = \mathbb{Z}[\zeta_m]$. Furthermore as long as the defining cyclotomic number m does not admit a lot of distinct factors the transformation between the coefficient and canonical embeddings is asymptotically tame [BC22]. Lastly for power of two cyclotomic one can see that the dual ring is a scaling of the ring.

Where it comes to the polynomial $f(x) = x^n - x + 2$ we make the following notes. We verified numerically that for all $n \leq 64$ our order $\mathcal{R}$ is indeed the ring of integers. We provide the details for $n = 32, 64$ in Section 8.1.2. For larger values of n it becomes hard to verify square-freeness. However, there is no evidence that working in the ring of integers provides additional security compared to other orders. In particular we are not familiar with attacks that exploit this. In addition it admits a good transformation between the canonical and coefficient embeddings. For $n = 2^k$ this can be argued directly via Rouché's theorem similarly to the classes of polynomials describe in [RSW18]. Otherwise a heuristic argument can be made, as the polynomial has a large gap and therefore its roots are somewhat equidistributed in terms of their angle from the origin, and its roots are between 1 to $3^{\frac{1}{n}}$ with a geometric average of $2^{\frac{1}{n}}$. We calculated numerically the largest and smallest singular values up to $n = 1000$, and we see that they both scale with $\sqrt{n}$ with a small constant, see Section 8.1.3 for details. For $n = 32$ and $n = 64$ we get $(\approx 4.6, \approx 15.2)$ and $(\approx 6.5, \approx 22.3)$ respectively.

As for working in the dual or the ring itself the case for our polynomial and cyclotomics are similar, in that we have to either multiply by the inverse of $f'(x)$ enlarging the error in the order of magnitude of the degree, or follow the reduction in [RSW18].

We also point out that some have argued that cyclotomic polynomials may be a worse choice in terms of security, compared to non-cyclotomics, since their structure may give rise to attacks. For example, in the context of the NTRU-Prime scheme, [BCLV16] proposed to work with the non-cyclotomic $x^n - x + 1$

(which is quite similar to ours). They claim that working with such polynomials can also be done quite efficiently, and is less risky in terms of security.

Lastly we want to address a line of attacks on polynomial LWE using polynomials of the form $x^n + ax + b$ [EHL14, ELOS15, CIV16]. The general concept is to start with polynomials of the form $x^n + b$ for a “large” b and show it behave similarly to $x^n + ax + b$ for a “small” a . The first idea is that if $b = q - 1$ we get that $f(1) = 0 \pmod q$ which in turns means that if there is a ring equation modulo both f and q it is possible to assign $x = 1$ and still obtain a valid equation. This collapses the PLWE instance into a one-dimensional instance with small noise. Notice that this attack hinges on the coefficient embedding of the noise $e(x)$ being small. Peikert [Pei16] surveys such “field dependent attacks” and concludes that they result from improper choice of noise parameters, that is enabled by pathological properties of the number field. Our interpretation of Peikert’s conclusion is that so long as we add the noise “properly”, i.e. in such a way that when looking at the respective dual instance, the noise is large enough, then no vulnerabilities are known. In essence this means that while we discuss the transformation between the canonical and coefficient embeddings for the security reduction, this distinction may be void for practical attacks as long as we work in the dual ring.

8 Performance and Implementation

We didn’t implement the optimized Key switch yet, and the algebraic modulus. We expect additional improvements from those.

Our parameters y are for security x

8.1 Parameter Estimations

We are going to estimate parameters generically i.e. for a plaintext of a certain size, a polynomial with certain parameters and varying security levels. In particular for f we will denote by σ_i the singular values of the Vandermonde matrix V_f defined by the roots of f . We assume that s is sampled such that it has enough entropy but not anymore then necessary for the security parameter. In particular s is chosen uniformly from $\{g \in \mathcal{R} : g_i \in \{0, 1\}, g(1) = k\}$. This provides enough entropy given that n choose k is greater then 2^κ . As for security we are going to assume that the RLWE problem is as hard as the LWE problem i.e. there are no better attacks on RLWE. Further we are going to assume that the hardness of a RLWE instance is determined by the smallest singular values of the co-variance matrix. Pictorially, the error is sampled from an ellipsoid and we assume instead the error is taken from the largest sphere contained in the ellipsoid. Lastly we are going to assume that modulus switching and re-linearization are preformed after each operation.

8.1.1 Irreducibility of our polynomials

Lemma 8.1. *The polynomial $f(x) = x^n - x + 2$ is irreducible over the integers for $n \geq 2$.*

Proof. Notice that the complex roots of f have absolute value > 1 , by the triangle inequality. assume $f(x) = h(x) \cdot g(x)$. We have that $h(0) \cdot g(0) = 2$, and since $h(0), g(0)$ are integers, one of them is of absolute value 1, with out loss of generality g . Then g must have a root of absolute value ≤ 1 , a contradiction since this root is also a root of f .

8.1.2 Monogenicity of our polynomials It is well known that if the discriminant of a polynomial f is square free, then $\mathbb{Z}[x]/f(x)$ is the ring of integers of $\mathbb{Q}(x)/f(x)$. For a polynomial of the form $x^{2^k} - x + 2$, by theorem 4 in [GD84], the discriminant is $D = 2^{k-1} \cdot 2^{k \cdot 2^k} - (2^k - 1)^{2^k - 1}$. For $k = 64$:

$$2^{63} \cdot 2^{64 \cdot 2^{64}} - 63^{63}$$

factorizes into the following primes:

1399,
315883,
1054894487,
1609025206302091,
300524395301294803,
102580173634571360137,
8668017673543442201810164494961,
1813131374962222709188812217190299

For $k = 32$:

$$2^{31} \cdot 2^{32 \cdot 2^{32}} - 31^{31}$$

factorizes into the following primes:

53,
683,
1189674929,
72879316190189456125055364884319727806855527

So both of the polynomials are monogenic.

8.1.3 Computing Singular Values We have computed the singular values of the polynomial $x^n - x + 2$ from $n = 4$ to $n = 1000$.

This gives $O(\sqrt{n})$ similar to the power of two case.

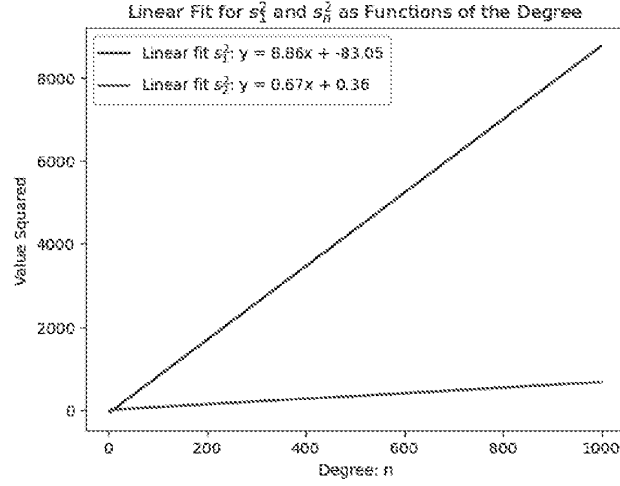


Fig. 2. Logarithmic Fit for the square of the minimal and maximal singular values as a function of the degree

8.1.4 Parameters for Security We sample our error from a Spherical Gaussian with standard deviation r then in the canonical embedding we have a distribution with a smallest singular value of $\frac{r}{\varepsilon_f}$ where ε_f is the distortion factor resulting from the linear transformation V_f . For now we are going to assume that $\varepsilon_f = s_1^2$ since this will be the singular value of the matrix $V_f V_f^T$ but I think we may be able to switch it with something like $\frac{1}{n} \sum_{i=1}^n \sigma_i^2$. This assures us that if there is no advantage to using cyclotomic field in the canonical embedding then our scheme is as secure as RLWE in the cyclotomic field with error r . That being said there is an attack called Aurora-Go which actually work in the coefficient embedding: Assume that the error is bounded in the range $[-t, t]$ then the following polynomial $P(ax - c) = x \prod_{i=1}^t x^2 - i$. This is a polynomial equation without error. I don't see how this would transfer into the canonical embedding in which errors behave considerably different. Thus we may be able to take for security $|e| = \max(\varepsilon_f, 2\sqrt{n})$ since I'm not fully convinced in this observation I'll see the parameters when assuming it and one not assuming it. So overall the parameter for security will be:

1. The computational security parameter κ taking the values 128, 160, 192
2. The statistical security parameter γ taking the values 40, 80, κ
3. The distortion of the Vandermonde matrix ε_f taking the values 1, 10, 10^2 , 10^3 , 10^4 , 10^5 .
4. A boolean value indicating the respect to the Aurora-Ge attack in the canonical embedding i.e. is the error at least $2\sqrt{n}\varepsilon_f$ for True or $\max(2\sqrt{n}, \varepsilon_f)$ for False.

5. Which gap proof do we use, with gap proportional to 2^γ or exact.

We would like to choose parameters n, q such that the scheme is secure according to the LWE estimator with errors $\frac{r}{\epsilon_f}$ that exact commands shall be

$$params = LWE.Parameters(n, q, \chi_s, \chi_e); LWE.Estimate(params)$$

Where $\chi_e = ND.DiscreteGaussian(r)$, $\chi_s = ND.Choose(n, k)$?

8.2 Implementation

This section evaluates concrete runtime of our scheme encryption and decryption procedures, homomorphic arithmetic operations: addition and multiplication, as well as modulus switching and key switching, without algebraic ciphertext modulus and bootstrapping.

Parameters

We generated parameters for our scheme, such that correctness is achieved with high probability based on the somewhat naive bounds given in sections ?? and that the lattice problems are 128-bit secure according to the lattice estimator.

Setup The tests and results presented in the next section were obtained with a naive implementation in which no specific optimizations were applied to enhance performance. These results thus, reflect the raw computational costs associated with the building blocks of our scheme without the influence of advanced heuristics, algorithms or hardware-specific optimizations. The test bed for our results is as follows. We implemented our tests in Rust run on and bench-marked single-threaded speed for all procedures.

We use the Key Switching Algorithm for $d = 2$ and $q = q'$, And while the implementation supports working over modulus, for the concrete parameters we chose $r = 1$. While talking about depth of multiplications - Each plaintext goes through the following procedure:

1. Encoding
2. Encryption
3. Modulus Switching

And then for each Multiplication:

- Tensor multiplication
- Key Switching
- Modulus Switching

In case of zero multiplications, after the third action (the modulus switch) addition is performed. The ciphertext size is measured after the modulus switch, as this is the size required for transmission during communication.

Results

Comparison with BGV We generated parameters to the original BGV [BGV12] scheme, with similar ℓ_∞ to those we obtained. We did that for plaintext spaces of sizes $2^{16}, 2^{32}, 2^{64}$. Below you can see the comparison in the ciphertext sizes, when modulus switch is performed right after encryption to reduce the ciphertext size..

A Optimizing Performance Using Ideal Ciphertext Moduli

We shortly present the scheme working with $\mathcal{R}_q$ where q is not a rational integer. We believe it holds both theoretical and practical significance. In particular, we don't need to use multiplication by w in the key switching and modulus switching algorithms, since we can switch between input and output moduli with an element which is congruent to 1 mod $\mathfrak{p}$. Indeed there are algebraic elements which are congruent to 1 mod $\mathfrak{p}$ of much smaller ℓ_∞ norm, than rational ones. Another advantage is that we can work with ciphertexts in the “Double CRT Representation” [?] which means representing the ciphertext ring as a direct product of $\mathbb{Z}/n_i$. This representation allows multiplying ciphertexts without an FFT transformation to the frequency domain. Returning to our case, notice that $\mathcal{R}/\langle ax - b \rangle \cong \mathbb{Z}/\langle b^n - ba^{n-1} + 2a^n \rangle$, So for cipher text modulus of the form $q = \prod \langle a_i x - b_i \rangle$, we have a double CRT representation.

In particular, we notice that for our scheme it is beneficial to set up the moduli-ladder by setting $\hat{q}_j = \langle 1 - k_j(x - 2) \rangle$ for rational integers $k_j \in \mathbb{Z}$, for which $\hat{q}_j$ are coprime. We then let $q_i = q_0 \cdot \prod_{j=0}^i \hat{q}_j$. Here $q_0 \in \mathbb{N}$ is the “bottom step” in the ladder, which means that as we get to decrypt or bootstrap we fall back into the rational setting.

A.1 Algebraic Number Theory facts

$N(t)$ i.e. the algebraic norm of the element $t = ax - b$ (and the ideal $\langle t \rangle$ in our ring is the resultant of $ax - b$ and $x^n - x + 2$, which is $b^n - ba^{n-1} + 2a^n$. The quotient ring satisfies $\mathcal{R}/\langle t \rangle \cong \mathbb{Z}/N(t)$.

A.2 The difference in the scheme from rational q

The scheme remains structurally identical, where operations which previously performed modulo q are now conducted modulo q . The only aspect of the original scheme, as outlined in Algorithm 4.1, that requires further definition is the decryption algorithm. However, it is important to note that our modulus ladder ends with a rational integer. Consequently, decryption is only necessary when the modulus is a rational integer, which ensures that the scheme functions correctly without further modification.

Correctness and Security. Standard security assumptions talk about RLWE type problems with respect to a rational ciphertext modulus. Although there are

assumptions with respect to algebraic ideals - as the GLWE security assumption in [PP19], we wish to talk about a simple reduction from M/P/R/O - LWE problems over an algebraic ciphertext modulus to a rational one. Working with an algebraic modulus q , we can perform modulus switch to the modulus $\alpha \cdot q$ for a rational q and α a polynomial with 0,1 coefficients. Notice that $\langle q \rangle | \langle \alpha \cdot q \rangle$ so in particular we get an MPLWE problem with the ciphertext modulus q - which is rational - where we lose roughly a factor of the expansion factor in the parameters.

We measure noise in ℓ_∞ in the coefficient embedding, the same as in definition 4.1. Note that the analysis of the noise growth after encryption, and the bounds needed for decryption stay the same.

Homomorphic Properties. Multiplication, Multiplication by scalar and Addition are the same as in integer modulus, with q replaced by q in the algorithm. The analysis and bounds on the noise $\eta_s(c, m)$ stay the same. There difference is a new gadget in the key switching and a new modulus switching.

A.3 Representing the Ciphertext

Lemma A.1. Assume $z \bmod (ax - b)$ (for $b = 2a + 1, a = k$) can be written as a polynomial of degree $n-1$ with coefficients in the range $(-b/4(1-1/2^n), b/4(1-1/2^n))$ which is $(\lceil -\frac{b}{4} \rceil, \lfloor \frac{b}{4} \rfloor)$ ($\ell_\infty(z) \leq b/4$). Then there is an efficient algorithm to find this polynomial. Also there is a unique way to write it that way.

Proof. For (res) the resultant of $ax - b$ and $f(x)$ the generating polynomial of the number field we have:

$$z = \sum c_i \left(\frac{b}{a}\right)^i \bmod (\text{res})$$

So

$$z \cdot a^{n-1} = \sum c_i b^i a^{n-1-i}$$

Notice that the RHS is in the interval $(-\text{res}/2, \text{res}/2)$. Indeed:

$$\begin{aligned} |\text{RHS}| &\leq (1 - 1/2^n) \cdot \frac{b}{4} \cdot a^{n-1} \cdot \sum \left(\frac{b}{a}\right)^i \\ &\leq (1 - 1/2^n) \cdot \frac{b}{4} \cdot a^{n-1} \cdot 2 \left(\frac{b}{a}\right)^{n-1} \\ &\leq (1 - 1/2^n) \cdot \frac{b^n}{2} \\ &\leq \frac{b^n - a^{n-1}}{2} = \frac{\text{res}}{2} \end{aligned}$$

So we look at $z' = z \cdot a^{n-1} \bmod \text{res}$ as an element in $\mathbb{Z}$ and provide an algorithm to write it there as a sum $\sum c_i b^i a^{n-1-i}$. Assuming such a representation exists, we must have $c_0 = z'/a^{n-1} \bmod b$. Similar equation holds for $c_1 \cdot b \cdot a^{n-2}$

mod (b^2) and we can continue for the latter coefficients by taking mod b^i every time. Notice it also proves that there is a unique way to write the element as a polynomial with coefficients in this range.

Lemma A.2. *Every element t in $\mathcal{R}$ is congruent modulo $ax - b$ (where $b = 2a + 1$ and $a = k$) to a polynomial z with $\ell_\infty(z) \leq \frac{b}{2} \cdot \alpha$, where $\alpha > (1 + \frac{a}{b} \cdot \zeta) \approx \frac{3}{2}$, and ζ is the real positive root of $x^n - x - 2$.*

Proof. Let $z = \sum a_i x^i$ with $a_i \in \mathbb{Z}$. Define $|z|(c)$ as the evaluation of $\sum |a_i| x^i$ at $c \in \mathbb{R}$, i.e., $|z|(c) = \sum |a_i| c^i$. Consider the polynomial z with minimal $|z|(\zeta)$ that is congruent to t modulo $ax - b$. We claim that this z cannot have coefficients with absolute value larger than $\frac{b}{2} \cdot \alpha$.

Indeed, if the i -th coefficient is larger, then we add or subtract $ax^{i+1} - bx^i$ to reduce the absolute value of the i -th coefficient (by at least $\min(b, \frac{b}{2} \cdot (2\alpha - 2))$) and claim that $|z|(\zeta)$ decreases. There are two cases to consider:

1. **Case $i < n - 1$:** In this case, the absolute value of the $(i + 1)$ -th coefficient increases by at most a . If $a \cdot \zeta < \min(b, \frac{b}{2} \cdot (2\alpha - 2))$ (which holds by the assumption on α), then $|z|(\zeta)$ decreases.
2. **Case $i = n - 1$:** The absolute value of the constant term increases by at most $2a$, and the absolute value of the x^1 coefficient increases by at most a . Therefore, $|z|(\zeta)$ decreases if

$$2a + a \cdot \zeta < \zeta^{n-1} \cdot \min(b, \frac{b}{2} \cdot (2\alpha - 2)),$$

which after dividing by ζ^{n-1} is the same inequality as in the previous case.

Working Modulo $\mathfrak{q}_j$. Recall that $\mathfrak{q}_j = q_0 \cdot \prod (a_i x - b_i)$. As long as the $a_i x - b_i$ are coprime (also to q_0), by CRT we have $\mathcal{R}/\mathfrak{q} \equiv \mathcal{R}/q_0 \prod \mathcal{R}/(a_i x - b_i)$, so calculations can be done coordinatewise. Notice that when doing the calculations modulo $(a_i x - b_i)$ we can choose $t \in \mathbb{Z}$ as a representative in the conjugacy class for each $c \in \mathcal{R}/(a_i x - b_i)$. So its like working in $\mathbb{Z}/(b_i^n - b_i a_i^{n-1} + 2a_i^n)$ in each coordinate, and then NTT can be used for multiplying elements in the quotient ring efficiently.

A.4 Key Switching

We use a variant of the Key Switching presented in algorithm 5.3, with $q, q' = \mathfrak{q}, \mathfrak{q}'$ and $T = \dots$. For that we introduce the following gadget:

CRT decomposition gadget Notice that we have the CRT isomorphism

$$f : \mathcal{R}_{\prod_j^l \mathfrak{q}_j} \rightarrow \prod_j^l \mathcal{R}_{\mathfrak{q}_j}$$

given that the $\mathfrak{q}_j^l$ are coprime. For $c \in \mathcal{R}_{\prod_j^l \mathfrak{q}_j}$ we look at the gadget vector $\mathbf{g}_{\text{crt}} = (f^{-1}(e_i))_i$. The decomposition operation $\mathbf{g}_{\text{crt}}^{-1} = f(c)$ viewed as a vector

in $\mathcal{R}^l$. Notice that this embedding is not uniquely defined. For $\mathfrak{q}_j = ax - b$ we choose the smallest representative in absolute value in $\mathbb{Z}$ of $c \bmod (ax - b)$. For $\mathfrak{q}_j = q_0 \in \mathbb{Z}$ we choose the smallest representative in ℓ_∞ in the coefficient embedding of $c \bmod q_0$.

composition with powers-of-d gadget We can compose the CRT decomposition operator with the $\mathbf{g}_{\text{Powers}_d}^{-1}$ operator. The output of $\mathbf{g}_{\text{crt}}^{-1}$ is a vector when each coordinate is in $\mathbb{Z}$ with $\ell_\infty \leq N(\mathfrak{q}_j)/2$ (N is the algebraic norm) or in $\mathcal{R}_{q_0}$ with $\ell_\infty \leq q_0/2$. The composition is applying $\mathbf{g}_{\text{Powers}_d}^{-1}$ to each coordinate of the output meaning $\mathbf{G}_{\text{Powers}_{d_0}}^{-1} \cdot \mathbf{g}_{\text{crt}}^{-1}(x)$. Notice that applying $\mathbf{g}_{\text{Powers}_d}^{-1}$ result in different vector size in each coordinate. Notice also that this operator is the decomposition operator for the gadget $\mathbf{g}_{\text{crt}} \cdot \mathbf{G}_{\text{powers of } d_0}$.

bounding the inner product To use the Keyswitch algorithm with the powers-of-d composed to CRT decomposition gadget, we need to bound $\langle \mathbf{G}^{-1}(\mathbf{c}'_1), \mathbf{e} \rangle$ as seen in lemma 5.3.

Lemma A.3. *For $\mathbf{G}^{-1}$ defined above, and $\mathbf{e}$ defined in Algorithm 5.3 we have: $\langle \mathbf{G}^{-1}(\mathbf{c}'_1), \mathbf{e} \rangle \leq \ell_\infty(\mathbf{e}) \leq \lfloor d/2 \rfloor \cdot \log_d(N(\mathfrak{q})) B_{\chi_e}(\mathfrak{q}')$*

Corollary A.1. *Let $\mathbf{c}_1 \in \mathcal{R}_{\mathfrak{q}}^{r_1}$ and $\mathbf{c}_2 = \text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1)$ defined as in algorithm 5.3. Then for the gadget we defined above we have that $\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}$ is in $\mathcal{R}_{\mathfrak{q}}^{r_1 \cdot \lceil \log_d(N(\mathfrak{q})) \rceil \times r_2}$, and if $\mathfrak{q}' = \mathfrak{q} \cdot \langle w \rangle$*

$$\eta_{\mathbf{s}_2}(\mathbf{c}_2, \mu) \leq \gamma_w \cdot \eta_{\mathbf{s}_1}(\mathbf{c}_1, \mu) + r_1 \cdot \lfloor d/2 \rfloor \cdot \lceil \log_d(N(\mathfrak{q})) \rceil B_{\chi_e}(\mathfrak{q}') \cdot \gamma$$

A.5 Modulus Switching

We now describe the modulus switching algorithm for switching from (ideal) modulus $\mathfrak{q}$ int (ideal) modulus $\mathfrak{q}'$. Formally, we let $\mathfrak{q}, \mathfrak{q}'$ be ideals in the ring $\mathcal{R}$. We considered the special case where $\mathfrak{q} = \langle q \rangle, \mathfrak{q}' = \langle \alpha \cdot q' \rangle$ for (rational) integers q, q' .

We let $A_{\mathfrak{p}}$ denote a rounding algorithm for the plaintext lattice $\mathfrak{p}$ in the canonical embedding, with distance parameter $d_{\mathfrak{p}}$. Indeed, we always select $\mathfrak{p}$ so that it has a short basis, so such $A_{\mathfrak{p}}$ should exist, however its exact properties are determined by the specific instantiation we choose.

We require the existence of a *ratio parameter* $t \in K$ (we stress that t is a fraction so usually $t \notin R$) with the following properties: $t \in \mathfrak{q}' \cdot \mathfrak{q}^{-1}$ and furthermore $t = 1 \pmod{\mathfrak{p}}$. In the integer setting, if we may take $q = q' \pmod{\mathfrak{p}}$ then $t = q'/q$ is a valid ratio parameter.

Algorithm A.1: Algebraic Modulus Switching

$\mathbf{c}' \leftarrow \text{ModSwitch}_{\mathfrak{q}, \mathfrak{q}', t}(\mathbf{c})$, for any input $\mathbf{c} \in \mathcal{R}_{\mathfrak{q}}^{r+1}$, outputs $\mathbf{c}' \in \mathcal{R}_{\mathfrak{q}'}^{r+1}$ such that $\mathbf{c}'[i] = \lceil t \cdot \mathbf{c} \rceil_{\mathfrak{c}+\mathfrak{p}}$, as in Algorithm 5.4. Notice that since the ciphertext contains multiple elements from $\mathcal{R}$, the rounding is done on each element separately.

The following lemma summarizes the performance of the algorithm.

Definition A.1. For $t \in \mathbb{Q}[x]/f(x)$ Let $\mathcal{M}_t : \mathbb{Q}[x]/f(x) \rightarrow \mathbb{Q}[x]/f(x)$ be the linear operator of multiplying by $t: a \rightarrow t \cdot a$. Define $\tau(t) = \max \left\{ \frac{\ell_\infty(A_t(a))}{\ell_\infty(a)} : a \in \mathbb{Q}[x]/f(x) \right\}$.

Notice that looking at $\mathcal{M}_t$ as a matrix in the coefficient embedding, $\tau(t)$ is equal to the maximal ℓ_1 of row of the matrix

Lemma A.4. Let $\mathbf{p}, \mathbf{q}, \mathbf{q}', t$ be as above. Let $\mathbf{c} \in \mathcal{R}_{\mathbf{q}}^{r+1}$ be a ciphertext that encrypts μ under the key $\mathbf{s}$. Let $\mathbf{c}' = \text{ModSwitch}_{\mathbf{q}, \mathbf{q}', t}(\mathbf{c})$. Then

$$\eta(\mathbf{c}', \mu) \leq \tau(t) \cdot \eta(\mathbf{c}, \mu) + \hat{\gamma} \cdot \ell_1(\mathbf{s})$$

Proof. Denote $\mathbf{s} \cdot \mathbf{c} = v + E$ where $E \in \mathbf{q}$. By definition of ModSwitch , it holds that

$$\mathbf{c}' = \lceil t \cdot \mathbf{c} \rceil_{\mathbf{c}+\mathbf{p}}^{A_{\mathbf{p}}} . \quad (7)$$

Then

$$\mathbf{s} \cdot \mathbf{c}' = t \cdot v + t \cdot E + \langle \mathbf{c}' - t \cdot \mathbf{c}, \mathbf{s} \rangle .$$

Denote $v' = t \cdot v + \langle \mathbf{c}' - t \cdot \mathbf{c}, \mathbf{s} \rangle$, $E' = tE$. Then

$$\mathbf{s} \cdot \mathbf{c}' = v' + E' . \quad (8)$$

Since $t \in \mathbf{q}'\mathbf{q}^{-1}$, it holds that $E' \in \mathbf{q}'$. Let us now analyze the norm of v' .

$$\ell_\infty(v') = \ell_\infty(t \cdot v + \langle \mathbf{c}' - t \cdot \mathbf{c}, \mathbf{s} \rangle) \quad (9)$$

$$\leq \ell_\infty(tv) + \ell_\infty(\langle \mathbf{c}' - t \cdot \mathbf{c}, \mathbf{s} \rangle) \quad (10)$$

$$\leq \tau(t)\ell_\infty(v) + \hat{\gamma} \cdot \ell_1(\mathbf{s}) . \quad (11)$$

Finally, since $t \equiv 1 \pmod{\mathbf{p}}$, then $E' = tE \equiv E \pmod{\mathbf{p}}$. Furthermore we recall that $\mathbf{c}' \equiv \mathbf{c} \pmod{\mathbf{p}}$. Therefore

$$v' = c'_0 + \mathbf{s} \cdot \mathbf{c}' - E' \quad (12)$$

$$\equiv c_0 + \mathbf{s} \cdot \mathbf{c} - E \pmod{\mathbf{p}} \quad (13)$$

$$= v . \quad (14)$$

This concludes the proof of the lemma.

Next we show a bound on $\tau(t)$.

Lemma A.5. The map $A : f \rightarrow \frac{f}{1+k(x-2)}$ is reducing the ℓ_∞ norm by at least $2/3 \cdot (k+1)$, meaning $\ell_\infty(Af) \leq \frac{3}{2(k+1)} \cdot \ell_\infty(f)$

Corollary A.2. $\tau\left(\frac{1}{1+k(x-2)}\right) \leq \frac{3}{2(k+1)}$

Proof. We upper bound the ℓ_1 norm of the rows of A as a linear operator, by $2/3 \cdot (k+1)$, which proves the lemma.

The inverse matrix of $ax+b$:

The matrix A that represents multiplying by $ax+b$ in the coefficient embedding is the following: b 's on the main diagonal, a 's is the diagonal below $((n-2)$ a 's), $A(0, n-1) = -2a$, $A(1, n-1) = a$ (indices are between 0 and $n-1$). For $n=4$ it looks like

$$A = \begin{pmatrix} b & 0 & 0 & -2a \\ a & b & 0 & a \\ 0 & a & b & 0 \\ 0 & 0 & a & b \end{pmatrix}$$

We claim that for even n , the inverse of that matrix, is $\frac{1}{b^n + a^{n-1}b + 2a^n}$ multiplied by the following matrix B :

$$\begin{aligned} B[0, 0] &= b^{n-1} + a^{n-1} \\ B[0, i] &= 2 \cdot (-b)^{i-1} a^{n-i}, & \text{for } i \geq 1 \\ B[i, i] &= b^{n-1}, & \text{for } i \geq 1 \\ B[i, j] &= (-1)^{j-i-1} \cdot (2a^{n-(j-i)} b^{j-i-1} + a^{n-1-(j-i)} b^{j-i}), & \text{for } i \geq 1 \text{ and } j \geq i+1 \\ B[i, j] &= (-1)^{i-j} \cdot b^{n-1-(i-j)} a^{i-j}, & \text{for } i \geq j \text{ works also for } i=j \end{aligned}$$

For $n=4$, the matrix B is given by:

$$B = \begin{pmatrix} b^3 + a^3 & 2a^3 & -2a^2b & 2ab^2 \\ -ab^2 & b^3 & 2a^3 + ba^2 & -(2a^2b + ab^2) \\ a^2b & -ab^2 & b^3 & 2a^3 + ba^2 \\ -a^3 & a^2b & -ab^2 & b^3 \end{pmatrix}$$

Let's see that it is indeed the inverse:

$$\begin{aligned}
(B \cdot A)[i, i] &= b \cdot B(i, i) + a \cdot B(i, i+1) \\
&= b^n + a \cdot (2a^{n-1} + a^{n-2}b), \\
&\quad \text{for } i < n-1, \\
(B \cdot A)[n-1, n-1] &= -2a \cdot (-a^{n-1}) + a \cdot a^{n-2}b + b \cdot b^n, \\
(B \cdot A)[i, n-1] &= -2a \cdot B(i, 0) + a \cdot B(i, 1) + b \cdot B(i, n-1) \\
&= -2a \cdot (-1)^i b^{n-1-i} a^i + a \cdot (-1)^{i-1} b^{n-i} a^{i-1} \\
&\quad + b \cdot (-1)^{n-1-i-1} \cdot (2a^{n-(n-1-i)} b^{n-2-i} + a^i b^{n-1-i}) = 0, \\
(B \cdot A)[i, j] &= b \cdot B(i, j) + a \cdot B(i, j+1) \\
&= b \cdot (-1)^{j-i-1} \cdot (2a^{n-(j-i)} b^{j-i-1} + a^{n-1-(j-i)} b^{j-i}) \\
&\quad + a \cdot (-1)^{j-i} \cdot (2a^{n-(j+1-i)} b^{j-i} + a^{n-1-(j+1-i)} b^{j+1-i}) = 0, \\
&\quad \text{for } i < j < n-1 \\
(B \cdot A)[i, j] &= b \cdot B(i, j) + a \cdot B(i, j+1) \\
&= b \cdot (-1)^{i-j} \cdot b^{n-1-(i-j)} a^{i-j} \\
&\quad + a \cdot (-1)^{i-j-1} \cdot b^{n-1-(i-j-1)} a^{i-j-1} = 0, \\
&\quad \text{for } j < n-1, j < i.
\end{aligned}$$

For $a = -k$, $b = 2k + 1$ we can see that by multiplying by the inverse of A the ℓ_∞ of the vector is multiplied by at most the maximum between the ℓ_1 of the rows of the matrix, which is approximately $\frac{3}{2k}$. Indeed

$$\begin{aligned}
\ell_1 &\leq \frac{b^{n-1} + a^{n-1} + 2|a|^{n-1} + 2|a|^{n-2}b + \dots + 2|a|b^{n-2}}{b^n + a^{n-1}b + 2a^n} \\
&= \frac{b^{n-1} + a^{n-1} + 2 \frac{|a|b^{n-1} - |a|^n}{b - |a|}}{b^n + a^{n-1}b + 2a^n} \\
&\leq \frac{3}{2(k+1)}
\end{aligned}$$

Where we used that $a = -k$, $b = 2k + 1$.

Corollary A.3. Let $\mathbf{q} = \langle a \cdot (1 + k(x-2)) \rangle$, $\mathbf{q}' = \langle a \rangle$ and $t = \frac{1}{1+k(x-2)}$. For $\mathbf{c}, \mathbf{c}'$ as in lemma A.4, we have $\eta(\mathbf{c}') \leq \frac{3}{2(k+1)} \cdot \eta(\mathbf{c}) + \hat{\gamma} \cdot \ell_1(\mathbf{s})$

References

- BC22. Iván Blanco-Chacón. On the rlwe/plwe equivalence for cyclotomic number fields. *Applicable Algebra in Engineering, Communication and Computing*, 33(1):53–71, 2022.

- BCIV20. Carl Bootland, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Efficiently processing complex-valued data in homomorphic encryption. *J. Math. Cryptol.*, 14(1):55–65, 2020.
- BCLvV16. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime: reducing attack surface at low cost. Cryptology ePrint Archive, Paper 2016/461, 2016.
- BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325. ACM, 2012.
- Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.
- BTR24. Fabian Boemer, Karl Tarbe, and Rehan Rishi. Announcing swift homomorphic encryption, 2024. <https://www.swift.org/blog/announcing-swift-homomorphic-encryption/>.
- BV11a. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011.
- BV11b. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.
- CGGI20. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.*, 33(1):34–91, 2020.
- CIV16. Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. Cryptology ePrint Archive, Paper 2016/239, 2016.
- CJP21. Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander A. Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning - 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8-9, 2021, Proceedings*, volume 12716 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2021.
- CKKS16. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Paper 2016/421, 2016.
- CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017*,

- Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, 2017.
- CLPX18. Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia. High-precision arithmetic in homomorphic encryption. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 116–136. Springer, 2018.
- Con09. Kieth Conrad. The different ideal. *Expository papers/Lecture notes*. Available at: <http://www.math.uconn.edu/~kconrad/blurbs/gradnumthy/different.pdf>, 2009.
- DM14. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. *Cryptology ePrint Archive*, Paper 2014/816, 2014.
- EHL14. Kirsten Eisentraeger, Sean Hallgren, and Kristin Lauter. Weak instances of PLWE. *Cryptology ePrint Archive*, Paper 2014/784, 2014.
- ELOS15. Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Provably weak instances of ring-lwe, 2015.
- FV12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, page 144, 2012.
- GC14. Matthias Geihs and Daniel Cabarcas. Efficient integer encoding for homomorphic encryption via ring isomorphisms. In Diego F. Aranha and Alfred Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014 - Third International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, September 17-19, 2014, Revised Selected Papers*, volume 8895 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2014.
- GD84. Gary R. Greenfield and Daniel Drucker. On the discriminant of a trinomial. *Linear Algebra and its Applications*, 62:105–112, 1984.
- Gen09a. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.
- Gen09b. Craig Gentry. Fully homomorphic encryption using ideal lattices. *Symposium on the Theory of Computing*, page 169–178, 2009.
- GH19. Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464. Springer, 2019.
- GHS12. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, pages 465–482, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Cryptology ePrint Archive*, Paper 2013/340, 2013.
- HS00. Jeffrey Hoffstein and Joseph H Silverman. Optimizations for ntru. In *Proc. the Conf. on Public Key Cryptography and Computational Number Theory, Warsaw*, pages 77–88, 2000.

- HS20. Shai Halevi and Victor Shoup. Design and implementation of HELib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481, 2020. <https://eprint.iacr.org/2020/1481>.
- HS21. Shai Halevi and Victor Shoup. Bootstrapping for helib. *J. Cryptol.*, 34(1):7, 2021.
- KDE⁺24. Andrey Kim, Maxim Deryabin, Jieun Eom, Rakyong Choi, Yongwoo Lee, Whan Ghang, and Donghoon Yoo. General bootstrapping approach for rlwe-based homomorphic encryption. *IEEE Trans. Computers*, 73(1):86–96, 2024.
- LMW23. Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 595–608. ACM, 2023.
- LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.
- LPR13a. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.
- LPR13b. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 35–54. Springer, 2013.
- MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.
- Pei16. Chris Peikert. How (not) to instantiate ring-LWE. Cryptology ePrint Archive, Paper 2016/351, 2016.
- PP19. Chris Peikert and Zachary Pepin. Algebraically structured LWE, revisited. In *Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part I 17*, pages 1–23. Springer, 2019.
- PP24. Chris Peikert and Zachary Pepin. Algebraically structured lwe, revisited. *J. Cryptol.*, 37(3):28, 2024.
- RAD⁺78. Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
- Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.
- RSSS17. Miruna Rosca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2017.

- RSW18. Miruna Rosca, Damien Stehlé, and Alexandre Wallet. On the ring-LWE and polynomial-LWE problems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 146–173. Springer, 2018.
- SEA11. Microsoft SEAL. Microsoft seal is an easy-to-use and powerful homomorphic encryption library., 2011. <https://github.com/Microsoft/SEAL>.
- SS10. Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.
- tfh17. tfhe. Tfh: Fast fully homomorphic encryption library over the torus, 2017. <https://github.com/tfhe/tfhe>.
- Zam21. Zama. Concrete: Tfh compiler that converts python programs into the equivalent, 2021. <https://github.com/zama-ai/concrete>.

OVERVIEW

A homomorphic encryption scheme enables mathematical operations or computations to be performed on encrypted data without requiring its decryption. Therefore, the data remains confidential while being processed, even in an untrusted environment. As a result, useful operations may be performed on the encrypted data by a third party without requiring the third party to properly secure the data, as the original data cannot be identified without the decryption key. For example, a person or organization may apply standard encryption techniques to secure sensitive data on cloud platforms, but processing or validating the encrypted data in the cloud would require its decryption, which raises privacy issues and security vulnerabilities and may entail additional costs and resources. By utilizing homomorphic encryption, private data may be shared and evaluated securely in commercial cloud environments without endangering privacy. The cloud service provider only has access to the encrypted data (ciphertext) and can perform computations on the data without decryption. The results of the encrypted processing may be provided to the owner of the private data who can then decrypt the data (into plaintext) via the decryption key. In addition to cloud services, homomorphic encryption can be used by entities in a wide variety of businesses and industries, such as financial services, healthcare, retail, information technology and artificial intelligence systems, to allow access to encrypted data while protecting client or patient privacy.

Homomorphic encryption schemes may be classified into several categories, including partially homomorphic encryption, somewhat homomorphic encryption, and fully homomorphic encryption. A partially homomorphic encryption allows only a single operation to be performed on the ciphertext, such as addition or multiplication. In contrast, fully homomorphic encryption (FHE) supports multiple operations indefinitely, such as both addition and multiplication, enabling a wider range of arbitrary computations to be performed on the ciphertext. However, FHE is characterized by certain limitations, particularly a very large computational overhead and high inefficiency due to protracted time for performing computations. Many FHE schemes are based on lattice mathematics, such as derived from the ring learning with errors (RLWE) problem, and are generally considered secure from breaches by quantum computing (i.e., post-quantum cryptography).

The present disclosure relates to a fully homomorphic encryption (FHE) scheme based on lattice problems which natively support a large plaintext space, such as a 256-bit value or higher.

SUMMARY

In accordance with one aspect of the present disclosure, there is provided a method for a fully homomorphic encryption (FHE) scheme that natively supports a large plaintext space. The method includes the step of, given a desired plaintext size “ p ”, generating an irreducible polynomial “ f ” of a chosen degree $d > \log p$, such that $f(a) = p$ for some a co-prime to p . The method further includes the step of defining a set of parameters including: a number field: $K = \mathbb{Q}[x]/f$; a corresponding ring of integers: $\mathcal{O}_K$; an order of ring: $R = \mathbb{Z}[x]/f$; a linear polynomial: $g(x) = x - a$, and an ideal I generated by g and p , such that a quotient ring R/I is isomorphic to a desired plaintext space $\mathbb{Z}_p$, and such that $\{gx^i\}$ constitute a short basis for a lattice defined by the ideal I , where a value “ q ” is selected to be large enough and co-prime to p such that the corresponding Order-LWE problem is considered hard. The method further includes the step of applying an encryption process. In a first variation, the encryption process includes: sampling a secret key “ s ” from a dual of R , sampling an error “ e ” from a Gaussian distribution defined on R with standard deviation “ σ ”, and sampling a randomizer “ α ” uniformly from R ; and determining an encryption according to: $E(m, e) = (a, m + as + ge) \bmod(R, q)$. In a second variation, the encryption process includes: sampling a secret key “ s ” from a dual of R , sampling an error “ e ” from a Gaussian distribution on a lattice defined by the ideal I , and sampling a randomizer “ α ” uniformly from R ; and determining an encryption according to: $E(m, e) = (a, m + as + e) \bmod(R, q)$. In a third variation, the encryption process includes sampling a secret key “ s ” from a dual of R , sampling an error “ e ” from a Gaussian distribution defined on a lattice translated by the message “ m ”, and sampling a randomizer “ α ” uniformly from R ; and determining an encryption according to: $E(m, e) = (a, as + e) \bmod(R, q)$.

According to an aspect of the present disclosure, a decryption of the encrypted data includes subtracting “ as ” and reducing modulo the lattice basis.

According to an aspect of the present disclosure, the method further includes the step of applying a Brakerski, Gentry, Vaikuntanathan (BGV) construction to transform the scheme into a public key FHE scheme.

In accordance with another aspect of the present disclosure, there is provided a method for a fully homomorphic encryption (FHE) scheme that natively supports a large plaintext space. The method includes the step of applying a double Chinese Remainder Theorem (CRT) encryption process. The encryption includes the steps of: receiving as input a power of two n , and a message in an extended plaintext space: $\vec{a} \in (\mathbb{Z}_p)^n$, where: $P = \prod_{i=1, \dots, k} \vec{p}_i$ for a splitting coprime sequence $\vec{p}$ of length k , wherein the splitting coprime sequence comprises a vector such that: $\vec{p}_i \equiv 1 \bmod(2n)$ and $\gcd(\vec{p}_i, \vec{p}_j) = 0$ unless $i = j$; and outputting a message: $(\vec{u}_1, \dots, \vec{u}_k)$, where $\vec{u}_i \in R_{\vec{p}_i}$; applying a pre-processing step comprising identifying roots: $\theta_{i,j}$ of a cyclotomic polynomial: $\phi_{2n}(x) = x^n + 1 \bmod(\vec{p}_i)$; applying a rational CRT conversion

step: set $\vec{a} = (\vec{a} \bmod(\vec{p}_1), \dots, \vec{a} \bmod(\vec{p}_k))$; applying a polynomial CRT conversion step: set $\hat{a}_i = (\hat{a}_1(\theta_1), \dots, \hat{a}_1(\theta_n))$, for $i = 1, \dots, k$; and providing an output of $\hat{a}$.

According to an aspect of the present disclosure, the method may further include the step of applying a double CRT decoding process. The decryption includes the steps of: receiving as input a power of two n , and a message in an extended plaintext space $\hat{a} \in \prod_{i=1, \dots, k} R_{\vec{p}_i}$, where $\vec{p}$ is a splitting coprime sequence of length k ; and outputting a message in the extended plaintext space: $\vec{a} \in (Z_p)^n$, where: $P = \prod_{i=1, \dots, k} \vec{p}_i$; applying a pre-processing step comprising identifying roots: $\theta_{i,j}$ of a cyclotomic polynomial: $\phi_{2n}(x) = x^n + 1 \bmod(\vec{p}_i)$; calculating an inverse of a Vandermonde matrix V_i defined by the roots $\theta_{i,j}$, and calculating rational CRT coefficients: $\gamma_i = \frac{P}{\vec{p}_i} \left(\vec{p}_i^{-1} \bmod \left(\frac{P}{\vec{p}_i} \right) \right)$; applying a polynomial CRT conversion step such that: $\vec{a} = (V_1^{-1}(\hat{a}_1), \dots, V_1^{-1}(\hat{a}_k))$; applying a rational CRT conversion step such that: $\vec{a}_j = \sum_{i=1, \dots, k} \gamma_i \hat{a}_{i,j}$ for $j = 1, \dots, n$; and providing an output of $\hat{a}$.

According to an aspect of the present disclosure, the method may further include the step of constructing a leveled FHE for an arbitrary plaintext space. The constructing includes the steps of: for a desired plaintext space size “ p ”, selecting a splitting coprime sequence $\vec{p}$ of length k such that: $P = \prod_{i=1, \dots, k} \vec{p}_i > p$; wherein a desired depth of the leveled FHE is denoted by h ; using k instances of a BGV scheme, constructing an encryption scheme that operates in a plaintext space $\prod_{i=1, \dots, k} R_{(\vec{p}_i)^k}$; using the CRT representation, transforming the plaintext space into an extended plaintext space $(Z_p)^n$; for every BGV ciphertext encryption, applying a zero knowledge (ZK) proof π_{range} such that the encryption is of value $0 < x < p$, to ensure that after h homomorphic operations, the encrypted value will not modulate and the homomorphic operations will be valid in Z and therefore in Z_p ; after BGV homomorphic evaluations and before published values, adding an encryption of a random multiple of p to the new ciphertext, to ensure that no information on a value of a computation over Z is revealed other than a result $\bmod(p)$, whereby for a ciphertext c , sampling: $temp \leftarrow [2^\sigma p^h]$ and providing an output: $c' = add(c, temp \cdot p)$, where “ σ ” represents a statistical security parameter, to provide a final value bounded by: $(1 + 2^\sigma)p^{h+1}$.

According to an aspect of the present disclosure, at least one of the FHE schemes is used by a client to encrypt private and/or sensitive data to be shared with a third party operative for performing computations or processing the encrypted data without decryption thereof.

According to an aspect of the present disclosure, the third party may include at least one of: a cloud service provider; a medical service provider; a financial service provider; or an information technology service provider.

According to an aspect of the present disclosure, the large plaintext space comprises an integer size of at least 256-bits.

According to an aspect of the present disclosure, the method is applied in a decentralized network.

According to an aspect of the present disclosure, the method is applied in a blockchain system.

According to an aspect of the present disclosure, the method is applied in a large scale secret information sharing system.

DETAILED DESCRIPTION

The present disclosure relates to a fully homomorphic encryption (FHE) scheme based on lattice problems which natively support a large plaintext space, such as a 256-bit value or higher. Specifically, this may be achieved based on the Order-LWE problem, which is proven to be equivalent to known lattice problems [bolboceanu2019order].

The FHE scheme is detailed hereinbelow.

First, given a desired plaintext size “ p ” (i.e., the size of the plain message before encryption), it can be shown how to generate a relatively sparse irreducible polynomial “ f ” of a chosen degree $d > \log p$, such that $f(a) = p$ for some a co-prime to p .

The following parameters are then considered: the number field $K = \mathbb{Q}[x]/f$; the corresponding ring of integers $\mathcal{O}_K$; and an order of this ring $R = \mathbb{Z}[x]/f$. A linear polynomial is defined as follows: $g(x) = x - a$, and an ideal I generated by g and p is further defined. It is observed that the quotient ring R/I is isomorphic to the desired plaintext space $\mathbb{Z}_p$, and that $\{gx^i\}$ constitute a short basis for the lattice defined by the ideal I . A value “ q ” is selected to be large enough and co-prime to p such that the corresponding Order-LWE problem is considered hard.

Using the aforementioned notations, the following three encryption schemes are provided:

- i. In a first variation, a secret key “ s ” is sampled from the dual of R ; an error “ e ” is sampled from a Gaussian distribution defined on R with standard deviation “ σ ”; and a randomizer “ α ” is sampled uniformly from R . An encryption is then calculated as follows: $E(m, e) = (a, m + as + ge) \bmod(R, q)$.
- ii. A second variation is similar to the first variation, but the error e is sampled from a Gaussian distribution on a lattice defined by the ideal I , and the encryption is of the form: $E(m, e) = (a, m + as + e) \bmod(R, q)$.
- iii. A third variation is similar to the second variation, but the Gaussian distribution is defined on a lattice translated by the message “ m ”. and the encryption is defined as follows: $E(m, e) = (a, as + e) \bmod(R, q)$.

For all these variations, decryption may be performed by subtracting “ as ” and reducing modulo the lattice basis.

A construction similar to Brakerski, Gentry, Vaikuntanathan (BGV) [cryptoeprint:2011/277] encryption may be utilized to transform the aforementioned symmetric cryptosystems to a public key FHE.

Two alternative approaches may be considered.

In a first alternative approach, the plaintext $m \in [-p, p]$ is represented in a Chinese Remainder Theorem (CRT) representation. In other words, considering the message modulo

multiple small primes p_j such that $\prod_j p_j \gg p$. Taking the product to be greater than p is necessary to avoid modulations, making computation hold essentially over the integers. However, this may introduce an inefficiency as most of the plaintext space is not used. In addition, a large multiple of p must be added before decryption, to hide the number of modular reductions mod p that take place in the homomorphic evaluations, which can expose additional information. Moreover, for security reasons should add range proofs to convince that no modular reductions $\prod_j p_j$ took place, which may further entail a large computational overhead.

In a second alternative approach, a BGV encryption scheme may be applied in its existing form with a large plaintext space. However, this approach may require a large ciphertext modulus “ q ”, which in turn requires a large number of slots “ n ”, which may become intractable and infeasible relatively quickly.

According to an embodiment based on the first alternative approach, the plaintext space of BGV-based FHE schemes may be represented utilizing both rational and polynomial CRT. This technique may be useful for optimizations and for representing larger plaintext spaces. An alternative scheme is described hereinbelow.

In a double CRT encoding process, an algorithm receives as input a power of two n , a message in the extended plaintext space: $\vec{a} \in (\mathbb{Z}_p)^n$, where: $P = \prod_{i=1, \dots, k} \vec{p}_i$ for some splitting coprime sequence $\vec{p}$ of length k . A splitting coprime sequence is a vector which satisfies that: $\vec{p}_i \equiv 1 \pmod{2n}$ and $\gcd(\vec{p}_i, \vec{p}_j) = 0$ unless $i = j$. The algorithm then outputs a message: $(\vec{u}_1, \dots, \vec{u}_k)$, where $\vec{u}_i \in R_{\vec{p}_i}$. A subsequent pre-processing stage involves finding the roots: $\theta_{i,j}$ of the cyclotomic polynomial: $\phi_{2n}(x) = x^n + 1 \pmod{\vec{p}_i}$. Next is a rational CRT conversion step: set $\vec{a} = (\vec{a} \pmod{\vec{p}_1}, \dots, \vec{a} \pmod{\vec{p}_k})$. This is followed by a polynomial CRT conversion step: set $\hat{a}_i = (\vec{a}_1(\theta_1), \dots, \vec{a}_1(\theta_n))$, for $i = 1, \dots, k$. The encoding process then provides an output of $\hat{a}$.

In a double CRT decoding process, an algorithm receives as input a power of two n , a message in the extended plaintext space $\hat{a} \in \prod_{i=1, \dots, k} R_{\vec{p}_i}$, where $\vec{p}$ is some splitting coprime sequence of length k . The algorithm then outputs a message in the extended plaintext space: $\vec{a} \in (\mathbb{Z}_p)^n$, where: $P = \prod_{i=1, \dots, k} \vec{p}_i$. A subsequent pre-processing stage involves finding the roots: $\theta_{i,j}$ of the cyclotomic polynomial: $\phi_{2n}(x) = x^n + 1 \pmod{\vec{p}_i}$. In addition, the pre-processing includes calculating the inverse of the Vandermonde matrix V_i defined by the roots $\theta_{i,j}$, and calculating the rational CRT coefficients: $\gamma_i = \frac{P}{\vec{p}_i} \left(\vec{p}_i^{-1} \pmod{\left(\frac{P}{\vec{p}_i} \right)} \right)$. Next in the decoding process is a polynomial CRT conversion step where: $\vec{a} = (V_1^{-1}(\hat{a}_1), \dots, V_1^{-1}(\hat{a}_k))$. This is followed by a rational CRT conversion step where: $\vec{a}_j = \sum_{i=1, \dots, k} \gamma_i \vec{a}_{i,j}$ for $j = 1, \dots, n$. The decoding process then provides an output of $\hat{a}$.

The above alternative scheme can be used to construct a leveled FHE for an arbitrary plaintext space as follows. For a desired plaintext space size “ p ”, a splitting coprime sequence $\vec{p}$ of length k is selected such that: $P = \prod_{i=1, \dots, k} \vec{p}_i > p$. Preferably, the elements of the

sequence are roughly the same size and P is close to p . The desired depth of the leveled FHE is denoted by h . Using k instances of the BGV scheme, an encryption scheme that operates in the plaintext space $\prod_{i=1,\dots,k} R_{(p_i)}^k$ may be constructed. Using the CRT representation, the plaintext space of the scheme may be transformed into the extended plaintext space $(Z_P)^n$. For every BGV ciphertext encryption, a zero knowledge (ZK) proof π_{range} may be applied such that the encryption is of value $0 < x < p$. This may ensure that after h homomorphic operations, the encrypted value will not modulate and thus the homomorphic operations will be valid in Z and therefore in Z_P . After BGV homomorphic evaluations and before the values are published, an encryption of a random multiple of p is added to the new cipher text. This may ensure that no information on the value of the computation over Z is revealed other than the result $\text{mod}(p)$. This may be required for circuit privacy. In particular, for a ciphertext c , one may sample: $\text{temp} \leftarrow [2^\sigma p^h]$ and output: $c' = \text{add}(c, \text{temp} \cdot p)$, where “ σ ” represents a statistical security parameter. It is noted that this may provide a final value bounded by: $(1 + 2^\sigma)p^{h+1}$. Therefore, to support depth- h multiplication, the ciphertext modulus q must be greater than $(1 + 2^\sigma)p^{h+1}$. The described scheme can be batched, as typical for BGV schemes.

The different variations and approaches presented herein may enable configurable, native plaintext space for a fully homomorphic encryption (FHE) scheme. The disclosed FHE scheme can allow for encrypting message in Z_P and performing homomorphic operations on such messages in Z_P . As opposed to the aforementioned alternative approaches (i.e., based on CRT or based on BGV), the size of the ciphertext is comparable and proportional to the size of the plaintext (and does not grow exponentially with “ p ” for example). The disclosed FHE scheme may be employed in various applications. One example relates to cloud computing services. A lightweight client (i.e., computationally or memory limited) may use third party cloud services while maintaining data privacy by sending private data encrypted to the cloud, which could homomorphically operate on the ciphertext and return an encrypted output. Only the client who holds the private key can restore the output into its original form. For example, such a client may be a smartphone application that performs real-time face recognition operations but cannot store and compute the required large complex neural network architectures. Additional exemplary applications involve sensitive data analysis, such as in the fields of medical or financial services. Such corporations or institutions may possess mass quantities of sensitive client information that can be kept encrypted under a carefully protected private key, while allowing third parties to perform requisite analyses over the encrypted sensitive data.

The disclosed fully homomorphic encryption (FHE) scheme may overcome certain disadvantages over existing FHE schemes having a limited size plaintext space. Firstly, for a sufficiently large plaintext space, the ciphertext consists of a single slot, in which the ciphertext bloat is constant (i.e., independent of security parameter). Other schemes may rely on packing technique, encrypting multiple plaintexts in a single ciphertext to obtain an amortized small ciphertext bloat. However, these extra slots cannot always be exploited, and in any case they increase the latency. Furthermore, supporting large plaintext space, such as 2^{128} , is consistent

with the operation of modern computers, processors and data registers. This may result not only in faster homomorphic computation, but also provide for a more natural, straightforward and more efficient translation between an ordinary computer program and one operating with encrypted data.

METHODS

1. A method for a fully homomorphic encryption (FHE) scheme that natively supports a large plaintext space, the method comprising the steps of:

given a desired plaintext size “ p ”, generating an irreducible polynomial “ f ” of a chosen degree $d > \log p$, such that $f(\alpha) = p$ for some α co-prime to p ;

defining a set of parameters comprising: a number field: $K = \mathbb{Q}[x]/f$; a corresponding ring of integers: $\mathcal{O}_K$; an order of ring: $R = \mathbb{Z}[x]/f$; a linear polynomial: $g(x) = x - \alpha$, and an ideal I generated by g and p , such that a quotient ring R/I is isomorphic to a desired plaintext space Z_p , and such that $\{gx^i\}$ constitute a short basis for a lattice defined by the ideal I , wherein a value “ q ” is selected to be large enough and co-prime to p such that the corresponding Order-LWE problem is considered hard; and

applying an encryption process selected from the group consisting of:

(i) sampling a secret key “ s ” from a dual of R , sampling an error “ e ” from a Gaussian distribution defined on R with standard deviation “ σ ”, and sampling a randomizer “ α ” uniformly from R ; and determining an encryption according to: $E(m, e) = (a, m + as + ge) \bmod(R, q)$;

(ii) sampling a secret key “ s ” from a dual of R , sampling an error “ e ” from a Gaussian distribution on a lattice defined by the ideal I , and sampling a randomizer “ α ” uniformly from R ; and determining an encryption according to: $E(m, e) = (a, m + as + e) \bmod(R, q)$; and

(iii) sampling a secret key “ s ” from a dual of R , sampling an error “ e ” from a Gaussian distribution defined on a lattice translated by the message “ m ”, and sampling a randomizer “ α ” uniformly from R ; and determining an encryption according to: $E(m, e) = (a, as + e) \bmod(R, q)$.

2. The method of method 1, wherein a decryption of the encrypted data comprises subtracting “ as ” and reducing modulo the lattice basis.
3. The method of method 1, further comprising the step of applying a Brakerski, Gentry, Vaikuntanathan (BGV) construction to transform the scheme into a public key FHE scheme.

4. A method for a fully homomorphic encryption (FHE) scheme that natively supports a large plaintext space, the method comprising the steps of:

applying a double Chinese Remainder Theorem (CRT) encryption process comprising the steps of:

receiving as input a power of two n , and a message in an extended plaintext space: $\vec{a} \in (\mathbb{Z}_p)^n$, where: $P = \prod_{i=1, \dots, k} \vec{p}_i$ for a splitting coprime sequence $\vec{p}$ of length k , wherein the splitting coprime sequence comprises a vector such that: $\vec{p}_i \equiv 1 \bmod(2n)$ and $\gcd(\vec{p}_i, \vec{p}_j) = 0$ unless $i = j$; and outputting a message: $(\bar{u}_1, \dots, \bar{u}_k)$, where $\bar{u}_i \in R_{\vec{p}_i}$;

applying a pre-processing step comprising identifying roots: $\theta_{i,j}$ of a cyclotomic polynomial: $\phi_{2n}(x) = x^n + 1 \bmod(\vec{p}_i)$;

applying a rational CRT conversion step: set $\vec{a} = (\vec{a} \bmod(\vec{p}_1), \dots, \vec{a} \bmod(\vec{p}_k))$;

applying a polynomial CRT conversion step: set $\hat{a}_i = (\vec{a}_1(\theta_1), \dots, \vec{a}_1(\theta_n))$, for $i = 1, \dots, k$; and

providing an output of $\hat{a}$.

5. The method of method 4, further comprising applying a double CRT decoding process comprising the steps of:

receiving as input a power of two n , and a message in an extended plaintext space $\hat{a} \in \prod_{i=1, \dots, k} R_{\vec{p}_i}$, where $\vec{p}$ is a splitting coprime sequence of length k ; and outputting a message in the extended plaintext space: $\vec{a} \in (\mathbb{Z}_p)^n$, where: $P = \prod_{i=1, \dots, k} \vec{p}_i$;

applying a pre-processing step comprising identifying roots: $\theta_{i,j}$ of a cyclotomic polynomial: $\phi_{2n}(x) = x^n + 1 \bmod(\vec{p}_i)$; calculating an inverse of a Vandermonde matrix V_i defined by the roots $\theta_{i,j}$, and calculating rational CRT coefficients: $\gamma_i = \frac{P}{\vec{p}_i} \left(\vec{p}_i^{-1} \bmod \left(\frac{P}{\vec{p}_i} \right) \right)$;

applying a polynomial CRT conversion step such that: $\vec{a} = (V_1^{-1}(\hat{a}_1), \dots, V_1^{-1}(\hat{a}_k))$;

applying a rational CRT conversion step such that: $\vec{a}_j = \sum_{i=1, \dots, k} \gamma_i \vec{a}_{i,j}$ for $j = 1, \dots, n$;

and

providing an output of $\hat{a}$.

6. The method of method 4, further comprising the step of constructing a leveled FHE for an arbitrary plaintext space, the constructing comprising the steps of:

for a desired plaintext space size “ p ”, selecting a splitting coprime sequence $\vec{p}$ of length k such that: $P = \prod_{i=1, \dots, k} p_i > p$; wherein a desired depth of the leveled FHE is denoted by h ;

using k instances of a BGV scheme, constructing an encryption scheme that operates in a plaintext space $\prod_{i=1, \dots, k} R_{(p_i)}^k$;

using the CRT representation, transforming the plaintext space into an extended plaintext space $(Z_P)^n$;

for every BGV ciphertext encryption, applying a zero knowledge (ZK) proof π_{range} such that the encryption is of value $0 < x < p$, to ensure that after h homomorphic operations, the encrypted value will not modulate and the homomorphic operations will be valid in Z and therefore in Z_P ;

after BGV homomorphic evaluations and before published values, adding an encryption of a random multiple of p to the new ciphertext, to ensure that no information on a value of a computation over Z is revealed other than a result $mod(p)$, whereby for a ciphertext c , sampling: $temp \leftarrow [2^\sigma p^h]$ and providing an output: $c' = add(c, temp \cdot p)$, where “ σ ” represents a statistical security parameter, to provide a final value bounded by: $(1 + 2^\sigma)p^{h+1}$.

7. The method of either of methods 1 or 4, wherein the FHE scheme is used by a client to encrypt private and/or sensitive data to be shared with a third party operative for performing computations or processing the encrypted data without decryption thereof.
8. The method of method 7, wherein the third party is selected from the group consisting of:
- a cloud service provider;
 - a medical service provider;
 - a financial services provider; and

an information technology service provider.

9. The method of either of methods 1 or 4, wherein the large plaintext space comprises an integer size of at least 256-bits.
10. The method as in any of the method 1 to 9, applied in a decentralized network.
11. The method as in any of the method 1 to 10, applied in a blockchain system.
12. The method as in any of the method 1 to 11, applied in a large scale secret information sharing system.

It is to be understood that the invention is not limited in its application to the details set forth in the description contained herein or illustrated in the drawings. The invention is capable of other embodiments and of being practiced and carried out in various ways. Hence, it is to be understood that the phraseology and terminology
5 employed herein are for the purpose of description and should not be regarded as limiting. As such, those skilled in the art will appreciate that the conception upon which this disclosure is based may readily be utilized as a basis for designing other structures, methods, and systems for carrying out the several purposes of the presently disclosed subject matter.

10 It will also be understood that the system according to the invention may be, at least partly, implemented on a suitably programmed computer. Likewise, the invention contemplates a computer program being readable by a computer for executing the method of the invention. The invention further contemplates a non-transitory computer-readable memory tangibly embodying a program of instructions executable by the computer for
15 executing the method of the invention.

Those skilled in the art will readily appreciate that various modifications and changes can be applied to the embodiments of the invention as hereinbefore described without departing from its scope, defined in and by the appended claims.

CLAIMS

1. A processing circuitry (PC)-based method of fully homomorphic encryption, the method comprising:

5 encrypting an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

- a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and
- 10 b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

- c. the third non-cyclotomic ring, and
- d. a second ideal of the third non-cyclotomic ring,

15 wherein the second ideal is not the first ideal.

2. The method of claim 1, wherein the irreducible non-cyclotomic polynomial is one of:

20 $f(x) = x^n + x - b$, or

$f(x) = x^n - x + b$;

and wherein the first ideal is:

$x - b$

25 wherein b is an integer.

3. The method of claim 2, the method further comprising, prior to the encrypting:

encoding a given plaintext, of a given plaintext space, to the element of the first non-cyclotomic ring, the encoding being based on:

5

EncodedElement =

$$\left(\sum_{i=0}^{\text{Logb}(m)} m_i x^i \right) \bmod \text{FirstIdeal}$$

10

wherein EncodedElement denotes the element of the first non-cyclotomic ring, FirstIdeal denotes the first ideal, and

wherein m denotes the given plaintext, and wherein m_i is based on:

m =

15

$$\sum_{i=0}^{\text{Logb}(m)} m_i b^i,$$

and wherein $f(b)$ is equivalent to a size of the given plaintext space.

4. The method of claim 2, wherein the third non-cyclotomic ring is an order of a field, the field being an extension of a fourth non-cyclotomic ring that is derivative of the irreducible non-cyclotomic polynomial.

5

5. The method of claim 4, wherein the third non-cyclotomic ring is a final extension ring of a series of extension rings, the series of extension rings successively extending the fourth non-cyclotomic ring, the series of extension rings being of a given series length,

10

wherein each successive extension ring is based on an equation:

$$\text{CurrentNonCyclotomicExtensionRing} = \text{PredecessorNonCyclotomicRing}[t]/\text{MinimalPolynomial}$$

wherein PredecessorNonCyclotomicRing denotes a respective immediately preceding extension ring of the series, t is a respective additional algebraic element, and

15

wherein MinimalPolynomial denotes a respective minimal polynomial, the respective minimal polynomial being based on an equation:

$$\text{MinimalPolynomial} =$$

$$\left(\prod_{j=0}^n (t - a_j) + (c(t) * b) + (d(t) * \text{FirstIdeal}) \right)$$

20

wherein FirstIdeal denotes the first ideal, and wherein each a_j is a unique element of a ring that is a quotient of PredecessorNonCyclotomicRing and the first ideal, n is a respective given integer greater than 1,

and wherein c(t) and d(t) are polynomials in PredecessorNonCyclotomicRing[t], and the minimal polynomial is irreducible in PredecessorNonCyclotomicRing.

25

6. The method of claim 5, wherein the given series length is 1, and wherein the final extension ring of the series of final extensions rings is based on:

CurrentNonCyclotomicExtensionRing =
 5 FourthNonCyclotomicRing[t]/MinimalPolynomial

where FourthNonCyclotomicRing denotes the fourth non-cyclotomic ring.

7. The method of claim 5, wherein the minimal polynomial defining the final
 10 extension ring is one of:

- a) $t_n^2 + t_n + x$, or
- b) $t_n^2 - t_n + x$, or
- c) $t_n^2 + t_n + x$, or
- d) $t_n^2 - t_n + x$,

- 15 wherein t_n denotes the additional algebraic element of the final extension ring of the series of extension rings.

8. The method of claim 5, wherein the minimal polynomial defining the final extension ring is one of:

- a) $t_n^2 + t_n + bt_{n-1}$, or
- 20 b) $t_n^2 - t_n + bt_{n-1}$, or
- c) $t_n^2 + t_n - bt_{n-1}$, or
- d) $t_n^2 - t_n - bt_{n-1}$,

- 25 wherein t_n denotes the additional algebraic element of the final extension ring of the series of extension rings, and

wherein t_{n-1} denotes the additional algebraic element of the extension ring, of the series of extension rings, immediately preceding the final extension ring.

9. The method of claim 5, the method further comprising, prior to the encrypting, encoding a given number of plaintexts to an element of the first non-cyclotomic ring, wherein the encoding comprises:

- 5 a) generating, for each plaintext of the given number of plaintexts, a respective per-plaintext ring element of the first non-cyclotomic ring, the generating being based on:

PerPlaintextRingElement =

$$\left(\sum_{i=0}^{\text{Logb}(m)} m_i x^i \right) \bmod \text{FirstIdeal}$$

10 wherein PerPlaintextRingElement denotes the element of the first non-cyclotomic ring, and wherein m denotes the given plaintext, and wherein m_i is based on:

m =

$$15 \quad \sum_{i=0}^{\text{Logb}(m)} m_i b^i,$$

and wherein f(b) is equivalent to a size of the given plaintext space; and

- b) calculating an encoded element based on the formula:

EncodedElement_{c0} =

$$\left(\sum_{i=0}^{n-1} \text{PerPlaintextRingElement}_i \cdot \lambda_i \right) \bmod \text{FirstIdeal}$$

20 wherein EncodedElement_{c0} denotes the element of the first non-cyclotomic ring, n denotes the given number of plaintexts, PerPlaintextRingElement_i denotes a respective per-plaintext ring element, and λ_i denotes a respective Lagrange coefficient.

10. The method of claim 9, wherein the given number of plaintexts is:

$$\prod_{\{i=0\}}^{SeriesLength} (d_i)$$

wherein SeriesLength is the given series length of the series of extension rings,
and d_i is a respective degree of the respective minimal polynomial of the
respective extension ring.

11. The method of claim 1, wherein the encrypting the element of a first non-cyclotomic ring comprises:

calculating a linear combination, over the second non-cyclotomic ring of, at least:

a. the element of the first non-cyclotomic ring,

b. a sum of, at least:

(i) a product of, at least, an encryption key and a randomizer, and

(ii) a noise value,

the calculating thereby resulting in an element of the second non-cyclotomic ring.

12. The method of claim 11, wherein the encryption key is an element of the second non-cyclotomic ring, the noise value is sampled from the first ideal, and the randomizer is sampled from a distribution over a ring with a bounded expected norm.

13. The method of claim 12, wherein the encrypting is based on:

$$\begin{aligned} \text{EncryptedElement}_{c0} = \\ (\text{PlaintextElement} + ((\text{Randomizer} * \text{EncryptionKey}) + \text{NoiseValue}) \bmod \\ (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})) \end{aligned}$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring,
SecondIdeal denotes the second ideal, PlaintextElement denotes the element of
the first non-cyclotomic ring, EncryptionKey denotes the encryption key,
NoiseValue denotes the noise value, and Randomizer denotes the randomizer.

14. The method of claim 12, wherein the encrypting is based on:

$$\begin{aligned} \text{EncryptedElement}_{c0} = & \\ & (\text{PlaintextElement} + ((\text{Randomizer} * ((\text{ConstantValue} * \\ & \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3})) \\ 5 \quad & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer

10

denotes the randomizer, and wherein ConstantValue is a constant value that is an element of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

15. The method of claim 12, wherein the encrypting is based on:

15

$$\begin{aligned} \text{EncryptedElement}_{c0} = & \\ & ((\text{PlaintextElement} * \frac{\text{SecondIdeal}}{\text{FirstIdeal}}) + \\ & (\text{Randomizer} * ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue})) \\ & + \text{Randomizer3}) \\ 20 \quad & \text{mod} (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

wherein the first ideal and second ideal are principal ideals,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, Randomizer denotes the randomizer, and

25

wherein ConstantValue is a constant value that is an element of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

16. The method of claim 12, wherein the encrypting is based on:

5

EncryptedElement_{c0} =

$$\begin{aligned} & ((\text{PlaintextElement} * \text{SecondIdeal}) + \\ & ((\text{Randomizer} * (\text{ConstantValue} * \text{EncryptionKey})) + \text{NoiseValue}) \\ & + \text{Randomizer3}) \end{aligned}$$

10

$$\text{mod } (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))$$

wherein the first ideal is not a principal ideal, and the second ideal is a principal ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

15

wherein ConstantValue is a constant value that is an element of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

20

17. The method of claim 12, wherein the encrypting is based on:

EncryptedElement_{c0} =

(PlaintextElement +

5 (Randomizer * (ConstantValue * EncryptionKey) + NoiseValue) +
Randomizer3)

mod (ThirdNonCyclotomicRing / (SecondIdeal * FirstIdeal))

wherein the element of the first non-cyclotomic ring is in the second ideal,
the first ideal is a principal ideal, and the second ideal is not a principal
10 ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring,
FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal,
PlaintextElement denotes the element of the first non-cyclotomic ring,
EncryptionKey denotes the encryption key, NoiseValue denotes the noise
15 value, and Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of the second
non-cyclotomic ring, and wherein Randomizer3 is sampled from a
distribution over a ring with a bounded expected norm.

20 18. The method of claim 11, wherein the encryption key is an element of a module
derivative of the second non-cyclotomic ring, the noise value is sampled from the
first ideal, and the randomizer is sampled from a distribution over a module with a
bounded expected norm.

19. The method of claim 18, wherein the encrypting is based on:

$$\begin{aligned} \text{EncryptedElement}_{c0} = \\ & (\text{PlaintextElement} + ((\text{Randomizer} * \text{EncryptionKey}) + \text{NoiseValue}) \bmod \\ 5 & (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer.

10 20. The method of claim 18, wherein the encrypting is based on:

$$\begin{aligned} \text{EncryptedElement}_{c0} = \\ & (\text{PlaintextElement} + ((\text{Randomizer} * ((\text{ConstantValue} * \\ & \text{EncryptionKey}) + \text{NoiseValue}) + \text{Randomizer3})) \\ & \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

15 wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

20 wherein ConstantValue is a constant value that is an element of a module that is derivative the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

21. The method of claim 18, wherein the encrypting is based on:

EncryptedElement_{c0} =

$$\begin{aligned}
 & ((\text{PlaintextElement} * \frac{\text{SecondIdeal}}{\text{FirstIdeal}}) + \\
 & (\text{Randomizer} * ((\text{ConstantValue} * \text{EncryptionKey}) + \text{NoiseValue})) \\
 & + \text{Randomizer3}) \\
 & \text{mod } (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})
 \end{aligned}$$

wherein the first ideal and second ideal are principal ideals,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of a module that is derivative of the second non-cyclotomic ring, and wherein Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

22. The method of claim 18, wherein the encrypting is based on:

20

EncryptedElement_{c0} =

$$\begin{aligned}
 & ((\text{PlaintextElement} * \text{SecondIdeal}) + \\
 & ((\text{Randomizer} * (\text{ConstantValue} * \text{EncryptionKey})) + \text{NoiseValue}) \\
 & + \text{Randomizer3}) \\
 & \text{mod } (\text{ThirdNonCyclotomicRing} / (\text{SecondIdeal} * \text{FirstIdeal}))
 \end{aligned}$$

25

wherein the first ideal is not a principal ideal, and the second ideal is a principal ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal, PlaintextElement denotes the element of the first non-cyclotomic ring, EncryptionKey denotes the encryption key, NoiseValue denotes the noise value, and Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of a module that is derivative of the second non-cyclotomic ring, and wherein

Randomizer3 is sampled from a distribution over a ring with a bounded expected norm.

23. The method of claim 18, wherein the encrypting is based on:

EncryptedElement_{c0} =

(PlaintextElement +

5 (Randomizer * (ConstantValue * EncryptionKey) + NoiseValue) +
Randomizer3)

mod (ThirdNonCyclotomicRing / (SecondIdeal * FirstIdeal))

wherein the element of the first non-cyclotomic ring is in the second ideal,
the first ideal is a principal ideal, and the second ideal is not a principal
10 ideal,

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring,
FirstIdeal denotes the first ideal, SecondIdeal denotes the second ideal,
PlaintextElement denotes the element of the first non-cyclotomic ring,
EncryptionKey denotes the encryption key, NoiseValue denotes the noise
15 value, and Randomizer denotes the randomizer, and

wherein ConstantValue is a constant value that is an element of a module
that is derivative of the second non-cyclotomic ring, and wherein
Randomizer3 is sampled from a distribution over a ring with a bounded
20 expected norm.

24. A system of fully homomorphic encryption, the system comprising a processing circuitry (PC) configured to:

encrypt an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

5 wherein the first non-cyclotomic ring is a quotient of:

a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

b. a first non-trivial ideal of the third non-cyclotomic ring, and

10 wherein the second non-cyclotomic ring is a quotient of:

a. the third non-cyclotomic ring, and

b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

15

25. A computer program product comprising a computer readable non-transitory storage medium containing program instructions, which program instructions when read by a processor, cause the processor to perform a method of fully homomorphic encryption, the method comprising:

20

encrypting an element of a first non-cyclotomic ring to an element of a second non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

25

b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

- a. the third non-cyclotomic ring, and
- b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

5

26. A processing circuitry (PC)-based method of decrypting fully homomorphically encrypted data, the method comprising:

10 decrypting an element of a second non-cyclotomic ring to an element of a first non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

- a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

15

- b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

- a. the third non-cyclotomic ring, and
- b. a second ideal of the third non-cyclotomic ring,

20

wherein the second ideal is not the first ideal.

27. The method of claim 26, wherein the irreducible non-cyclotomic polynomial is one of:

$$f(x) = x^n + x - b, \text{ or}$$

25

$$f(x) = x^n - x + b;$$

and wherein the first ideal is:

$x-b$

wherein b is an integer.

28. The method of claim 27, the method further comprising, subsequent to the decrypting:

5

decoding the element of the first non-cyclotomic ring to a plaintext of a given plaintext space.

29. The method of claim 28 where the decoding is based on calculating:

10

$f_{\text{element}}(b)$

wherein $f_{\text{element}}()$ is a polynomial function corresponding to the element of the first non-cyclotomic ring.

30. The method of claim 28 wherein the first non-cyclotomic ring is a quotient ring based on a final ring of a series of successive rings extending a fourth non-cyclotomic ring, and wherein the decoding comprises:
- 5 a) determining one or more roots of a minimal polynomial associated with the first non-cyclotomic ring;
- b) for each determined root:
- 10 calculating a result of substituting, in the element of the first non-cyclotomic ring, a respective extending algebraic element with the respective root,
- thereby generating one or more elements of a quotient ring derivative of a preceding ring of the series;
- 15 c) responsive to the preceding ring being an extension ring of the fourth non-cyclotomic ring:
- for each generated element: repeating a) - b); and
- d) responsive to the preceding ring being the fourth non-cyclotomic ring:
- 20 calculating, for each generated element $f_{\text{element}}()$;
- $f_{\text{element}}(b)$
- thereby giving rise to one or more plaintexts.
31. The method of claim 26, wherein the decrypting the element of a first non-cyclotomic ring comprises subtracting, from the element of the second non-cyclotomic ring:
- 25

a value based on a product of a decryption key and a randomizing element,

the decryption key and the randomizing element being elements of the second non-cyclotomic ring,

5 thereby resulting in an element of the first non-cyclotomic ring.

32. The method of claim 31, wherein the decrypting is based on:

$$\text{DecryptedElement} = \text{EncryptedElement}_{c0} - (\text{DecryptionKey} * \text{EncryptedElement}_{c1}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal})$$

10 wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, EncryptedElement_{c0} denotes the element of the second non-cyclotomic ring, EncryptedElement_{c1} denotes the randomizing element, DecryptedElement denotes the element of the first non-cyclotomic ring, and DecryptionKey denotes the decryption key.

15 33. The method of claim 31, wherein the decryption key is a symmetric key.

34. The method of claim 31, wherein the decryption key is a private key.

35. The method of claim 31, wherein the randomizing element is derivative of at least one randomizer.

20 36. The method of claim 35, wherein the randomizing element is further derivative of at least one constant value.

37. The method of claim 26, wherein the decrypting the element of a first non-cyclotomic ring comprises subtracting, from the element of the second non-cyclotomic ring:

25

a value based on a product of a decryption key and a randomizing element,

the decryption key and the randomizing element being elements of a module derivative of the second non-cyclotomic ring,

38. The method of claim 37, wherein the decrypting is based on:

$$\begin{aligned} & \text{DecryptedElement} = \text{EncryptedElement}_{c0} - (\text{DecryptionKey} * \\ 5 \quad & \text{EncryptedElement}_{c1}) \bmod (\text{ThirdNonCyclotomicRing} / \text{SecondIdeal}) \end{aligned}$$

wherein ThirdNonCyclotomicRing denotes the third non-cyclotomic ring, SecondIdeal denotes the second ideal, EncryptedElement_{c0} denotes the element of the second non-cyclotomic ring, EncryptedElement_{c1} denotes the randomizing element, DecryptedElement denotes the element of the first non-cyclotomic ring,
10 and DecryptionKey denotes the decryption key.

39. The method of claim 37, wherein the decryption key is a symmetric key.

40. The method of claim 37, wherein the decryption key is a private key.

41. The method of claim 37, wherein the randomizing element is derivative of at least one randomizer.

15 42. The method of claim 41, wherein the randomizing element is further derivative of at least one constant value.

43. A system of decrypting fully homomorphically encrypted data, the system comprising a processing circuitry (PC) configured to:

20

decrypt an element of a second non-cyclotomic ring to an element of a first non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

a. a third non-cyclotomic ring that is derivative of an irreducible
25 non-cyclotomic polynomial, and

b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

- a. the third non-cyclotomic ring, and
- b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.

5

44. A computer program product comprising a computer readable non-transitory storage medium containing program instructions, which program instructions when read by a processor, cause the processor to perform a method of decrypting fully homomorphically encrypted data, the method comprising:

10

decrypting an element of a second non-cyclotomic ring to an element of a first non-cyclotomic ring;

wherein the first non-cyclotomic ring is a quotient of:

15

- a. a third non-cyclotomic ring that is derivative of an irreducible non-cyclotomic polynomial, and

- b. a first non-trivial ideal of the third non-cyclotomic ring, and

wherein the second non-cyclotomic ring is a quotient of:

20

- a. the third non-cyclotomic ring, and
- b. a second ideal of the third non-cyclotomic ring,

wherein the second ideal is not the first ideal.