A New CRT-based Fully Homomorphic Encryption

Anil Kumar Pradhan

Vaultree Ltd.

Abstract. We have proposed a novel FHE scheme that uniquely encodes the plaintext with noise in a way that slows down the increasing noise from overflowing and corrupting the plaintext. This allows users to perform computations on encrypted data smoothly. The scheme is constructed using the Chinese Remainder Theorem (CRT), supporting a predefined number of modular operations on encrypted plaintext without the need for bootstrapping.
Although FHE recently became popular after Gentry's work [10] [11] and various developments have occurred in the last decade, the idea of "Fully Homomorphic Encryption (FHE)" scheme was first introduced in the 1970s by Rivest [13]. The Chinese Remainder Theorem is one of the most suitable tools for developing a FHE Scheme because it forms a ring homomorphism $Z_{p_1} \times Z_{p_2} \times \ldots \times Z_{p_k} \cong Z_{p_1 p_2 \ldots p_k}$. Various attempts have been made to develop a FHE using CRT, but most of them were unsuccessful, mainly due to the chosen plaintext attack (CPA) [5].
The proposed scheme overcomes the chosen plaintext attack. The scheme also adds random errors to the message during encryption. However, these errors are added in such a way that, when homomorphic operations are performed over encrypted data, the increasing values of errors never overwrite the values of the messages, as happens in LWE-based homomorphic schemes. Therefore, one can perform a predefined number of homomorphic operations (both addition and multiplication) without worrying about the increasing values of errors.

Keywords: Fully Homomorphic Encryption (FHE), Chinese Reminder Theorem, Chosen Plaintext Attack

1 Introduction

Fully Homomorphic Encryption (FHE) is a transformative cryptographic paradigm that enables computations on encrypted data without requiring decryption. Introduced by in 1970's by Rivest [13] and first secure scheme was proposed Craig Gentry in 2009 [10] [11], and later improved in [3] [2] [4] [7] [14] [9], FHE has opened new avenues for secure data processing, allowing operations to be performed on ciphertexts while maintaining the confidentiality of the underlying plaintexts. This capability is particularly valuable in scenarios where data privacy is paramount, such as in cloud computing, medical data processing, and financial transactions.

FHE schemes can execute a wide range of operations on encrypted data, including both arithmetic (addition and multiplication) and logical operations, making them Turing complete. This means that any computable function can, in theory, be evaluated on encrypted data without revealing the data itself. The potential applications of FHE are vast, encompassing secure voting systems, privacy-preserving data analysis, and encrypted search functionalities, among others.

One of the critical challenges addressed by FHE is the need to maintain data privacy while enabling the functionality of modern data analytics and machine learning. Traditional encryption schemes secure data at rest and in transit but require decryption for processing, exposing sensitive information to potential breaches. FHE, on the other hand, keeps data encrypted throughout the processing life-cycle, thus significantly enhancing security and privacy.

Despite its promising features, the practical deployment of FHE has been historically hindered by performance issues, particularly the high computational overhead associated with homomorphic operations. Early FHE schemes required bootstrapping [8] [6] [12]—a process to refresh ciphertexts to manage noise growth during computations—which was computationally expensive and impractical for real-world applications [1].

The concept of utilizing the Chinese Remainder Theorem (CRT) for constructing a fully homomorphic encryption (FHE) scheme was first introduced as a "privacy homomorphism" in 1978 by Rivest, Adleman, and Dertouzos [13]. The fundamental idea was to define an encryption function that allows computations on encrypted data without needing to decrypt it. The approach begins with enabling basic binary operations, such as addition and multiplication, over encrypted data. Since any function can be approximated by a polynomial, supporting addition and multiplication on encrypted data implies the potential to compute any function on the data.

The privacy homomorphism designed by Rivest, Adleman, and Dertouzos operates as follows:

- *Key Generation*: The user selects two large prime numbers, p and q, and computes the public parameter $n = pq$.
- *Plaintext and Ciphertext Spaces*: The plaintext space is defined as $\mathbb{Z}_n$, and the ciphertext space is defined as $\mathbb{Z}_p \times \mathbb{Z}_q$.

Encryption:

$$m \in \mathbb{Z}_n \longrightarrow (m \mod p, m \mod q) = (c_1, c_2)$$

Decryption:

The plaintext m is recovered from (c_1, c_2) using the Chinese Remainder Theorem.

However, it was later demonstrated that this privacy homomorphism is vulnerable to known plaintext attacks [5]. The attack exploits the fact that if $\text{Enc}(m) = (c_1, c_2)$, then:

$$c_1 = m \mod p \Rightarrow m = pk + c_1 \Rightarrow p|(m - c_1)$$

Since $p|n$, we have:

$$p|\gcd(m - c_1, n)$$

Similarly, $q|\gcd(m - c_2, n)$. Given that $n = pq$, the greatest common divisor $\gcd(m - c_i, n)$ for $i = 1, 2$ can be either p or q. Consequently, an attacker can recover the secret keys from a known set of plaintext and ciphertext values.

This vulnerability highlighted the need for more secure methods of constructing fully homomorphic encryption schemes, leading to the exploration and development of more robust approaches in the following decades.

Our Contribution: The main idea of the scheme is to enable efficient computation on encrypted data by uniquely encoding the plaintext with noise in a way that prevents noise from increasing to the point of overflowing and corrupting the plaintext. This innovative approach allows users to perform computations on encrypted data smoothly without the common problem of noise accumulation degrading the data integrity. The techniques used in scheme ensure that the noise introduced during encryption does not interfere with the accuracy and reliability of the computations performed on the encrypted data.

This paper aims to provide a comprehensive overview of the proposed Fully Homomorphic Encryption, discussing its foundational principles, security analysis, speed, performance, and practical applications. We will also explore the implementation and benchmarking of the proposed scheme.

We reduce the security of the proposed scheme to lattice-based cryptography, meaning it is as secure as any other lattice-based cryptographic scheme. Specifically, we demonstrate that if a method exists to break the proposed scheme, the same method can be used to break the known hard problem, LWE and RLWE problem. Therefore, our proposed FHE scheme is as secure as any cryptographic scheme based on the LWE problem, which is recognized as being quantum-safe.

The rest of the section is organized as follows: we begin with an introduction to the preliminaries, followed by the definition of essential notations and parameters. Next, we describe the proposed fully homomorphic encryption (FHE) algorithm in detail. Following this, we provide a proof of the algorithm's correctness and conduct a thorough security analysis. Additionally, we present a performance analysis and benchmarking results.

2 Preliminary

In this section, we provide an overview of the basic concepts that underpin our Fully Homomorphic Encryption (FHE) scheme, focusing on the Chinese Remainder Theorem (CRT). These concepts are fundamental to understanding the security and functionality of our encryption scheme.

2.1 Chinese Remainder Theorem (CRT)

The Chinese Remainder Theorem is a key mathematical tool used in number theory and cryptography. It provides a way to solve systems of simultaneous congruences with pairwise co-prime modulo.

Let $n_1, n_2, \ldots, n_k$ be pairwise coprime integers (i.e., $\gcd(n_i, n_j) = 1$ for $i \neq j$). For any given integers $a_1, a_2, \ldots, a_k$, there exists an integer x that simultaneously satisfies the system of congruences:

$$\begin{aligned} x &\equiv a_1 \mod n_1 \\ x &\equiv a_2 \mod n_2 \\ &\vdots \\ x &\equiv a_k \mod n_k \end{aligned}$$

Moreover, the solution x is unique modulo $N = n_1 n_2 \cdots n_k$.

The CRT is useful in cryptographic applications because it allows computations to be performed independently in smaller, modular arithmetic spaces and then recombined to form the final result. This can lead to efficiencies in both computation and storage.

Definition 1 (CRT Function). *For pairwise coprime integers $p_1, \ldots, p_k$, we define the CRT function $CRT_{(p_1,\ldots,p_k)}$ for inputs $(m_1, \ldots, m_k)$ as a number in $\mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_k}$ which is congruent to m_i modulo p_i for all $i \in \{1, \ldots, k\}$, where $p = \prod_{i=1}^{k} p_i$.*

Formally, we have:

$$CRT_{(p_1,\ldots,p_k)}(m_1, \ldots, m_k) \equiv \sum_{i=1}^{k} m_i \hat{p}_i (\hat{p}_i^{-1} \mod p_i) \mod p,$$

where

$$\hat{p}_i = \frac{p}{p_i} = \prod_{\substack{j=1 \\ j \neq i}}^{k} p_j$$

2.2 NTT Representation

Polynomial Representation In the context of Fully Homomorphic Encryption (FHE) schemes like BFV, data is often represented as polynomials. A polynomial in the ring $R = \mathbb{Z}[X]/(X^n + 1)$ can be written as:

$$\mathbf{a}(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1}$$

where each a_i is a coefficient in $\mathbb{Z}$. Operations such as addition and multiplication of these polynomials are performed modulo $X^n + 1$. This representation aligns directly with the algebraic structures used in FHE schemes.

For instance, in the BFV scheme:

- The plaintext modulus is denoted by t.
- The ciphertext modulus is denoted by q.

– $R_t = \mathbb{Z}_t[X]/(X^n + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ are the polynomial rings with coefficients in $\mathbb{Z}_t$ and $\mathbb{Z}_q$, respectively.

In polynomial representation, addition and multiplication operations are straightforward but can be computationally expensive for large polynomials due to the convolution involved in polynomial multiplication.

NTT Representation The Number Theoretic Transform (NTT) is a discrete Fourier transform (DFT) performed over finite fields. It transforms a polynomial from its coefficient representation into a point-value representation, significantly speeding up polynomial multiplication.

Transform Process

Given a polynomial $\mathbf{a}(X)$ in R, the NTT of $\mathbf{a}(X)$ is:

$$\mathbf{A} = \text{NTT}(\mathbf{a})$$

This transformation maps the polynomial to an array of its values at specific points, known as roots of unity, in a finite field. The inverse NTT (INTT) transforms it back to the coefficient representation:

$$\mathbf{a} = \text{INTT}(\mathbf{A})$$

Advantages of NTT

– Efficiency in Multiplication: Multiplying two polynomials in coefficient form requires a convolution operation, which is computationally expensive ($O(n^2)$). In NTT representation, multiplication becomes pointwise and linear in complexity ($O(n)$), followed by an inverse transform, making the overall complexity $O(n \log n)$.
– Scalability: NTT-based multiplication handles large polynomials efficiently, crucial for FHE schemes that operate over high-degree polynomials to achieve security.

Steps in NTT-Based Polynomial Multiplication

– Forward Transform: Apply NTT to both polynomials, $\mathbf{a}$ and $\mathbf{b}$, obtaining $\mathbf{A}$ and $\mathbf{B}$.
– Pointwise Multiplication: Multiply the transformed polynomials pointwise, $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$.
– Inverse Transform: Apply the inverse NTT to $\mathbf{C}$ to obtain the resulting polynomial in coefficient form.

In the context of the BFV scheme [4]:

– Encryption involves transforming the plaintext polynomial to NTT form, performing the encryption operation, and transforming the resulting ciphertext polynomial back to coefficient form if necessary.

– Homomorphic multiplication benefits significantly from NTT, as it reduces the complexity of polynomial multiplication.

In summary, while polynomial representation is simpler and more intuitive, NTT representation is preferred in practical homomorphic encryption implementations due to its substantial efficiency benefits in polynomial multiplication, which are essential for the performance of schemes like BFV.

2.3 Terminologies

The notation commonly employed includes $\mathbb{Z}$ for the set of integers and $\mathbb{Z}_q$ for the ring of integers modulo q. The plaintext modulus is denoted by t, while the ciphertext modulus is denoted by q. The degree of the cyclotomic polynomial, usually a power of 2, is represented by n. The polynomial ring R is defined as $\mathbb{Z}[X]/(X^n + 1)$, and R_q is the corresponding ring with coefficients in $\mathbb{Z}_q$. Within this scheme, $\mathbf{s}$ represents the secret key polynomial, $\mathbf{e}$ denotes an error polynomial with small coefficients, and $\mathbf{a}$ and $\mathbf{b}$ are components of the public key. Ciphertexts are typically represented as a vector $\mathbf{c} = (c_0, c_1)$. The security of the scheme is grounded in the hardness of the Learning With Errors (LWE) problem and its variant, the Ring Learning With Errors (RLWE) problem. The RLWE problem involves finding a secret polynomial $\mathbf{s}$ given a polynomial $\mathbf{a}$ and a noisy polynomial $\mathbf{b}$, where the noise polynomial $\mathbf{e}$ has small coefficients. The difficulty of solving this problem without the secret key underpins the security guarantees of the scheme, making it a robust choice for applications requiring secure computation on encrypted data.

2.4 Hard Problems

1. **Learning With Errors (LWE) Problem**: Given a matrix A and a vector $\mathbf{b} = A\mathbf{s} + \mathbf{e}$ where $\mathbf{s}$ is a secret vector and $\mathbf{e}$ is a small error vector, the task is to find $\mathbf{s}$. The hardness of this problem is the foundation of many cryptographic schemes.
2. **Ring Learning With Errors (RLWE) Problem**: A variant of the LWE problem where the operations are performed in a polynomial ring. Given a polynomial $\mathbf{a} \in R_q$ and a polynomial $\mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \mod q$ where $\mathbf{s}$ and $\mathbf{e}$ have small coefficients, the task is to find $\mathbf{s}$. The RLWE problem is believed to be hard and forms the security basis of the scheme.

BFV Scheme

The BFV scheme (Brakerski/Fan-Vercauteren) is an FHE scheme based on the RLWE problem. It allows for both addition and multiplication operations on ciphertexts, supporting computations on encrypted data.

The difficulty of solving the RLWE problem ensures that the scheme is secure against various cryptographic attacks. The noise $\mathbf{e}$ added during encryption makes it hard to recover the plaintext without the secret key.

By leveraging these hard problems and mathematical structures, the scheme achieves secure and efficient fully homomorphic encryption, enabling computations on encrypted data while preserving confidentiality. In the context of FHE, the CRT is used to manage modular arithmetic operations efficiently, allowing homomorphic operations to be performed on encrypted data. The RLWE problem provides a hard computational foundation that ensures the security of the encryption scheme. Together, these concepts enable the construction of a FHE scheme that is both secure and practical for performing large amount of computations on encrypted data.

3 Construction

In this section, we outline the construction of our Fully Homomorphic Encryption (FHE) scheme, focusing on its key components: key generation, encryption, homomorphic operations, and decryption. Our construction builds on the principles of lattice-based cryptography, incorporating enhancements to achieve improved performance and security. First we describe the required parameters for the scheme along with their sizes, then the actual construction of the scheme and finally we prove the correctness of the scheme.

3.1 The Construction

In this section we describe the construction of the proposed fully homomorphic encryption scheme. The scheme has 4 major components *Key Generation, Encryption, Decryption, and Homomorphic Operations e.g. Addition and Multiplication.*

The scheme consists of the following algorithms:

- Generate Public Parameters:
  - Based on the security parameter λ select the following public parameters
    - n: Dimension,
    - q: Ciphertext Modulus,
    - χ: Noise Distribution as per the standard of the RLWE Problem.
  - Additionally we define the plaintext space p_1 and noise space p_2. The values of p_1 and p_2 are selected such that $p_1 p_2 < q$. The detailed analysis of the relation between p_1, p_2 and q is described in the later sections.

$$PP = (n, q, \chi, p_1, p_2)$$

  - For simplicity we use the following additional notations

$$\Delta_1 = p_2[p_2^{-1}]_{p_1}, \Delta_2 = p_1[p_1^{-1}]_{p_2}$$

$$CRT_{p_1,p_2}(x, y) = \Delta_1 x + \Delta_2 y \mod p_1 p_2$$

- Key Generation:
  - Input: Security parameter λ, Public Parameters PP
  - Output: Public key pk, secret key sk, and evaluation key evk.

- Steps:
  - Sample $\mathbf{s} \in R_2$ uniformly.
  - Sample $\mathbf{a} \in R_q$ uniformly.
  - Sample $\mathbf{e} = e_0 + e_1 x + ..., e_{n-1} x^{n-1}$ from the noise distribution χ with small coefficients bounded by p_2.
  - Apply the CRT function to the noise polynomial as follows

    $$\epsilon = CRT_{p_1,p_2}(0, e_0) + CRT_{p_1,p_2}(0, e_1)x + ... + CRT_{p_1,p_2}(0, e_{n-1})x^{n-1}$$

  - Compute $\mathbf{b} = (\mathbf{a} \cdot \mathbf{s} + \epsilon) \mod q$.
  - Set $pk = (\mathbf{b}, -\mathbf{a})$ and $sk = \mathbf{s}$.

$$pk = (\mathbf{a}.\mathbf{s} + \sum_{i=0}^{n-1} CRT_{p_1,p_2}(0, e_i)x^i, -\mathbf{a})$$

– Generate Relinearization Keys:
- The relinearization keys (aka evaluation key:) rlk are needed for homomorphic multiplication.
- Create Auxiliary Keys:
  - Choose a base w and let $l = \lfloor \log_w q \rfloor$

– Steps
- For each $i \in \{0, \ldots, l\}$:
  - Choose a random polynomial $\mathbf{r}_{i_1} \in R_q$.
  - Sample noise $\mathbf{e_i} = e_{i,0} + e_{i,1}x + ..., e_{i,n-1}x^{n-1}$ from the noise distribution χ and compute

    $$\epsilon_\mathbf{i} = CRT_{p_1,p_2}(0, e_{i,0}) + CRT_{p_1,p_2}(0, e_{i,1})x + ... + CRT_{p_1,p_2}(0, e_{i,n-1})x^{n-1}$$

  - Compute the relinearization key component:

$$\mathbf{r}_{i_0} = \mathbf{r}_{i_1} \cdot \mathbf{s} + \epsilon_\mathbf{i} + w^i \cdot \mathbf{s}^2 \mod q$$

– Set relinearization key as follows:

$$\text{rlk} = \{(\mathbf{r}_{i_0}, -\mathbf{r}_{i_1}) \mid i = 0, \ldots, l\}$$

– Encryption:
- Input: Public key pk, Plain-text Vector $(m_0, ..., m_{n-1}) \in \mathbb{Z}_{p_1}^n$.
- Output: Ciphertext $\mathbf{c} = (c_0, c_1) \in R_q^2$.
- Steps:
  - Encode plaintext $(m_0, ..., m_{n-1})$ into a polynomial in R_{p_1} as follows:

    $$\mathbf{m} = m_0 + m_1 x + ... + m_{n-1} x^{n-1} \in R_{p_1}$$

  - Apply the CRT function to the coefficients as follows:

    $$\mathbf{m}^* = CRT_{p_1,p_2}(m_0, 0) + CRT_{p_1,p_2}(m_1, 0)x + ... + CRT_{p_1,p_2}(m_{n-1}, 0)x^{n-1}$$

* Sample random noise polynomials $\mathbf{e}' = e'_0 + e'_1 x + ... + e'_{n-1} x^{n-1}, \mathbf{e}'' = e''_0 + e''_1 x + ... + e''_{n-1} x^{n-1}$ with small coefficients and polynomial $\mathbf{u}$ with binary coefficients. The compute the following:

$$\epsilon' = CRT_{p_1,p_2}(0, e'_0) + CRT_{p_1,p_2}(0, e'_1)x + ... + CRT_{p_1,p_2}(0, e'_{n-1})x^{n-1} = \Delta_2 \mathbf{e}' \mod p_1 p_2$$

$$\epsilon'' = CRT_{p_1,p_2}(0, e''_0) + CRT_{p_1,p_2}(0, e''_1)x + ... + CRT_{p_1,p_2}(0, e''_{n-1})x^{n-1} = \Delta_2 \mathbf{e}'' \mod p_1 p_2$$

* Compute $c_0 = \mathbf{pk_0} \cdot \mathbf{u} + \epsilon' + \mathbf{m}^* \mod q$.
* Compute $c_1 = \mathbf{pk_1} \cdot \mathbf{u} + \epsilon'' \mod q$.
* Set Ciphertext $\mathbf{c} = (c_0, c_1)$.

$$\mathbf{c} = (c_0, c_1)$$

– Decryption:
  - Input: Secret key sk, ciphertext $\mathbf{c} = (c_0, c_1) \in R_q^2$.
  - Output: Plaintext $m \in R_{p_1}$.
  - Steps:
    * Compute $\mathbf{m} = (c_0 + c_1 \cdot \mathbf{s} \mod q) \mod p_1$.

Homomorphic Operations

– Homomorphic Addition:
  - Input: Ciphertexts $\mathbf{c}_1 = (c_{1,0}, c_{1,1}), \mathbf{c}_2 = (c_{2,0}, c_{2,1}) \in R_q^2$.
  - Output: Ciphertext $\mathbf{c}_{\text{sum}} \in R_q^2$.
  - Steps:
    * Compute $\mathbf{c}_{\text{sum}} = (c_{0,1} + c_{0,2} \mod q, c_{1,1} + c_{1,2} \mod q)$.

– Homomorphic Multiplication:
  - Input: Ciphertexts $\mathbf{c}_1, \mathbf{c}_2 \in R_q^2$, evaluation key evk.
  - Output: Ciphertext $\mathbf{c}_{\text{mult}} \in R_q^2$.

Steps: Homomorphic multiplication in the scheme involves multiplying two ciphertexts and then relinearizing the result. The steps are as follows:

Input Ciphertexts: Let $\mathbf{c}_1 = (c_{1,0}, c_{1,1})$ and $\mathbf{c}_2 = (c_{2,0}, c_{2,1})$ be two ciphertexts that encrypt plaintexts $\mathbf{m}_1$ and $\mathbf{m}_2$, respectively.

$$c_{i,0} + c_{i,1} s = \mathbf{m}_i^* = CRT_{p_1,p_2}(m_{i_0}, e_{i_0}) + CRT_{p_1,p_2}(m_{i_1}, e_{i_1})x + ... + CRT_{p_1,p_2}(m_{i_{n-1}}, e_{i_{n-1}})x^{n-1}$$

Then the product of the encoded plaintext will be

$$\mathbf{m}_1^* . \mathbf{m}_2^* = (c_{1,0} + c_{1,1} s)(c_{2,0} + c_{2,1} s)$$

$$= c_{1,0} \cdot c_{2,0} + (c_{1,0} \cdot c_{2,1} + c_{1,1} \cdot c_{2,0})s + (c_{1,1} \cdot c_{2,1})s^2$$

This results in a ciphertext of degree 2:

$$\mathbf{c}^*_{\text{mult}} = (c_0, c_1, c_2)$$

where:

$$c_0 = c_{1,0} \cdot c_{2,0}$$
$$c_1 = c_{1,0} \cdot c_{2,1} + c_{1,1} \cdot c_{2,0}$$
$$c_2 = c_{1,1} \cdot c_{2,1}$$

This ciphertext encrypts the product of the encoded plaintexts

$$\mathbf{m}_1^*.\mathbf{m}_2^* = c_0 + c_1 s + c_2 s^2$$

Relinearization is a step in the scheme that converts a higher-degree ciphertext (resulting from homomorphic multiplication) back to a form that can be handled like an original ciphertext. Here are the mathematical steps to apply relinearization and compute a new ciphertext of degree 1.

Decompose c_2 in base w:

$$c_2 = \sum_{i=0}^{l} c_{2,i} w^i$$

where $c_{2,i} \in R_q$ are the base-w digits of c_2.

Use the relinearization keys to reduce the degree of the ciphertext:

$$\mathbf{c}_{\text{mult}} = (c_0 + \sum_{i=0}^{l} \mathbf{r}_{i_0} \cdot c_{2,i}, c_1 + \sum_{i=0}^{l} \mathbf{r}_{i_1} \cdot c_{2,i})$$

Output Ciphertext: The relinearized ciphertext $\mathbf{c}_{\text{mult}} = (c_0', c_1')$ now encrypts the product $\mathbf{m}_1^* \cdot \mathbf{m}_2^*$ and has the same form as an original ciphertext.

These steps ensure that the homomorphic multiplication and relinearization process in the scheme maintains the ciphertext structure, allowing for further homomorphic operations while controlling the growth of ciphertext size and complexity.

3.2 Batch Encoding

Batching is a powerful technique in Fully Homomorphic Encryption (FHE) that allows multiple plaintexts to be encrypted and processed simultaneously within a single ciphertext. This approach significantly enhances the efficiency of homomorphic computations, making FHE more practical for real-world applications. The concept of batching leverages the structure of certain algebraic objects, such as rings and polynomials, to pack multiple messages into one.

Mathematical Foundation The current form of the ciphertext:

$$c_0 + c_1 s = \mathbf{m}^* = CRT_{p_1,p_2}(m_0, e_0) + CRT_{p_1,p_2}(m_1, e_1)x + ... + CRT_{p_1,p_2}(m_{n-1}, e_{n-1})x^{n-1}$$

We rewrite it to express the plaintext and noise separately as follows

$$CRT_{p_1,p_2}(m_0,e_0)+CRT_{p_1,p_2}(m_1,e_1)x+...+CRT_{p_1,p_2}(m_{n-1},e_{n-1})x^{n-1}$$

$$=\sum_{i=0}^{n-1}CRT_{p_1,p_2}(m_i,0)x^i+\sum_{i=0}^{n-1}CRT_{p_1,p_2}(0,e_i)x^i$$

$$=\sum_{i=0}^{n-1}p_2[p_2^{-1}]_{p_1}m_ix^i+\sum_{i=0}^{n-1}p_1[p_1^{-1}]_{p_2}e_ix^i$$

$$=(p_2[p_2^{-1}]_{p_1})\sum_{i=0}^{n-1}m_ix^i+(p_1[p_1^{-1}]_{p_2})\sum_{i=0}^{n-1}e_ix^i$$

$$=\Delta_1\mathbf{m}+\Delta_2\mathbf{e}$$

Let $\mathbf{c}_1=(c_{1,0},c_{1,1})$ and $\mathbf{c}_2=(c_{2,0},c_{2,1})$ be two ciphertexts that encrypt plaintexts μ_1 and μ_2, respectively.

$c_{j,0}+c_{j,1}s=\mathbf{m}_i^*=CRT_{p_1,p_2}(m_{j,0},e_{j,0})+CRT_{p_1,p_2}(m_{j,1},e_{j,1})x+...+CRT_{p_1,p_2}(m_{j,n-1},e_{j,n-1})x^{n-1}$

Now when two ciphertexts are multiplied the corresponding polynomials $\sum_{i=0}^{n-1}m_{1,i}x^i$ and $\sum_{i=0}^{n-1}m_{2,i}x^i$ were multiplied.

$$\text{Dec}(\mathbf{c}_{\text{mult}},sk)=\mathbf{m}_1^*.\mathbf{m}_2^*=(\sum_{i=0}^{n-1}m_{1,i}x^i)(\sum_{i=0}^{n-1}m_{2,i}x^i)$$

This does not gives use the coordinate-wise multiplication of the two vectors i.e.

$(m_{1,0},m_{1,1},...,m_{1,n-1})*(m_{2,0},m_{2,1},...,m_{2,n-1})\rightarrow(m_{1,0}m_{2,0},m_{1,1}m_{2,1},...,m_{1,n-1}m_{2,n-1})$

This can be achieved using the concept of NTT as follows:

$$INTT(m_{1,0},m_{1,1},...,m_{1,n-1})=\mu_1=\mu_{1,0}+\mu_{1,1}x+...+\mu_{1,n-1}x^{n-1}$$

$$INTT(m_{2,0},m_{2,1},...,m_{2,n-1})=\mu_2=\mu_{2,0}+\mu_{2,1}x+...+\mu_{2,n-1}x^{n-1}$$

$$\mu_1.\mu_2=INTT(m_{1,0},m_{1,1},...,m_{1,n-1})*INTT(m_{2,0},m_{2,1},...,m_{2,n-1})$$

$$=INTT(m_{1,0}\times m_{2,0},m_{1,1}\times m_{2,1},...,m_{1,n-1}\times m_{2,n-1})$$

In order to achieve this instead of directly putting plaintext values in coefficient, we need to compute the INTT of the plaintext vector and get a polynomial and we encrypt the new polynomial.

Encoding To batch multiple plaintext messages $\mathbf{m} = (m_0, m_1, \ldots, m_{n-1})$, each $m_i \in \mathbb{Z}_{p_1}$, into a single polynomial, we perform the following steps:

- Compute the Inverse NTT of the plaintext as

$$\mu = INTT(\mathbf{m}) = \mu_0 + \mu_1 x + \ldots + \mu_{n-1} x^{n-1}$$

The $INTT()$ function is performed in the space Z_{p_1} so that the result $\mu \in Z_{p_1}[X]$.

- Then the new polynomial is encoded into the following polynomial.

$$\mu^* = CRT_{p_1,p_2}(\mu_0, 0) + CRT_{p_1,p_2}(\mu_1, 0)x + ... + CRT_{p_1,p_2}(\mu_{n-1}, 0)x^{n-1}$$

Decoding To decode the batched ciphertext back into individual plaintexts:

- Apply $\mod p_1$ to separate plaintext and noise.

$$\mu^* \mod p_1$$

$$= CRT_{p_1,p_2}(\mu_0, e_0) + CRT_{p_1,p_2}(\mu_1, e_1)x + ... + CRT_{p_1,p_2}(\mu_{n-1}, e_{n-1})x^{n-1} \mod p_1$$

$$= \mu_0 + \mu_1 x + \ldots + \mu_{n-1} x^{n-1} = \mu$$

- Use the NTT function to transform the polynomial in $\mathbb{Z}_{p_1}[X]$ back into a tuple of elements in $\mathbb{Z}_{p_1}$.

$$NTT(\mu) = \mathbf{m} = (m_0, m_1, ..., m_{n-1})$$

- Extract the individual plaintext messages from the resulting polynomial.

Homomorphic Operations on Batches The primary advantage of batching is that homomorphic operations on the ciphertexts can be performed in parallel on the individual plaintexts. For instance:

Addition:

- Given two batched ciphertexts c_1 and c_2 encoding plaintexts $\mathbf{m_1} = (m_{1,1}, m_{1,2}, \ldots, m_{1,k})$ and $\mathbf{m_2} = (m_{2,1}, m_{2,2}, \ldots, m_{2,k})$, respectively, their homomorphic addition results in a ciphertext encoding $\mathbf{m_1} + \mathbf{m_2} = (m_{1,1} + m_{2,1}, m_{1,2} + m_{2,2}, \ldots, m_{1,k} + m_{2,k})$.

Multiplication:

- Similarly, homomorphic multiplication of two batched ciphertexts results in a ciphertext encoding the product of the corresponding plaintexts:

$$\mathbf{m_1} \times \mathbf{m_2} = (m_{1,1} \times m_{2,1}, m_{1,2} \times m_{2,2}, \ldots, m_{1,k} \times m_{2,k})$$

Advantages of Batching

1. Efficiency: Batching significantly reduces the number of ciphertexts that need to be managed and the number of homomorphic operations that need to be performed. This reduction leads to substantial improvements in computational efficiency and memory usage.
2. Parallelism: The ability to process multiple plaintexts simultaneously leverages parallel computation resources, further accelerating the performance of homomorphic operations.
3. Amortization: The computational overhead associated with encryption, decryption, and homomorphic operations is amortized over multiple plaintexts, making each operation more cost-effective.

Practical Considerations

1. Parameter Selection: Choosing appropriate parameters for n and q is crucial to balance the trade-off between security, noise growth, and the number of batched plaintexts.
2. Noise Management: While batching helps in managing computational resources, the noise growth must still be carefully monitored to ensure the correctness of decryption after multiple homomorphic operations.
3. Application Suitability: Batching is particularly useful in applications requiring parallel processing of multiple data items, such as in machine learning on encrypted data, database operations, and encrypted search.

Batching in FHE schemes offers a powerful means to enhance the efficiency of homomorphic computations by leveraging the parallelism in polynomial rings. By packing multiple plaintexts into a single ciphertext, batching allows for significant improvements in computational and memory efficiency, making FHE more practical for a wider range of applications.

3.3 Correctness

In this section we prove the correctness of encryption scheme. First we prove the correctness of the encryption scheme in the following theorem. Later we prove the correctness of the homomorphic operations over encrypted data.

Theorem 1. *For any message m and secret key sk, the decryption of the encryption of m using sk recovers the original message:*

$$Dec(Enc(\mathbf{m}, sk), sk) = \mathbf{m}$$

Proof. The ciphertext c is computed from the plaintext m and the secret key sk.

The encryption process involves the following steps:

$$\text{Enc}(\mathbf{m}, sk) = \mathbf{c} = (c_0, c_1)$$

Note that

$$c_0 = \mathbf{pk_0} \cdot \mathbf{u} + \epsilon' + \mathbf{m}^* \mod q$$

$$= (\mathbf{a}.\mathbf{s} + \Delta_2\mathbf{e}) \cdot \mathbf{u} + \Delta_2\mathbf{e}' + \mathbf{m}^* \mod q$$

$$= (\mathbf{a}.\mathbf{u}).\mathbf{s} + \Delta_2(\mathbf{e}.\mathbf{u} + \mathbf{e}') + \mathbf{m}^* \mod q$$

$$c_1 = \mathbf{pk_1} \cdot \mathbf{u} + \epsilon'' = -\mathbf{a}.\mathbf{u} + \Delta_2\mathbf{e}''$$

$$[c_0 + c_1 s]_q = \Delta_2(\mathbf{e}.\mathbf{u} + \mathbf{e}' + \mathbf{e}''\mathbf{s}) + \Delta_1\mathbf{m} = \Delta_2\mathbf{e}^* + \Delta_1\mathbf{m}$$

$$= CRT_{p_1,p_2}(m_0, e_0^*) + CRT_{p_1,p_2}(m_1, e_1^*)x + ... + CRT_{p_1,p_2}(m_{n-1}, e_{n-1}^*)x^{n-1}$$

Here,

$$\mathbf{e}^* = (\mathbf{e}.\mathbf{u} + \mathbf{e}' + \mathbf{e}''\mathbf{s})$$

To decrypt c, we apply the decryption function:

$$\text{Dec}(c, sk) = (c_0 + c_1 \cdot \mathbf{s} \mod q) \mod p_1$$

$$= CRT_{p_1,p_2}(m_0, e_0^*) + CRT_{p_1,p_2}(m_1, e_1^*)x + ... + CRT_{p_1,p_2}(m_{n-1}, e_{n-1}^*)x^{n-1} \mod p_1$$

$$= \mathbf{m}$$

Therefore, we conclude:

$$\text{Dec}(\text{Enc}(\mathbf{m}, sk), sk) = \mathbf{m}$$

This proof demonstrates that the encryption and decryption functions are correct and consistent, ensuring that decrypting an encrypted message recovers the original message without loss or alteration.

3.4 LWE Variant

This scheme also accommodates a LWE variant. The structure of the scheme is outlined as follows

- **Public Parameters**:
  - Choose pairwise relatively primes: p_1, p_2, q
  - Ciphertext Modulus: q, Plaintext Modulus: p_1, Noise Modulus: p_2
- **Key Generation**:
  - Secret key $\mathbf{s} \in \mathbb{Z}_q^n$
- **Encryption**:
  - Plaintext $m \in \mathbb{Z}_{p_1}$
  - Choose mask $\mathbf{a} \in \mathbb{Z}_q^n$, and small error $e' \in \mathbb{Z}_q$.
  - Compute ciphertext:

$$\mathbf{c} = ([\mathbf{a} \cdot \mathbf{s} + CRT(m, e)]_q, -\mathbf{a})$$

- **Decryption**:
  - Ciphertext $\mathbf{c} = (c_0, \mathbf{c_1})$
  - Decrypted plaintext

$$m = c_0 + \langle \mathbf{c_1}, \mathbf{s} \rangle \mod q \mod p_1$$

4 Security Analysis

To establish the security of the RLWE-based Fully Homomorphic Encryption (FHE) scheme, it is crucial to demonstrate that breaking the scheme is as difficult as solving the underlying RLWE problem. This involves proving that any adversary capable of compromising the scheme can be utilized to solve the RLWE problem, which is believed to be computationally hard. The primary objective is to prove semantic security and the chosen plaintext attack (CPA) model, emphasizing the indistinguishability of ciphertexts.

Ring Learning With Errors (RLWE) Problem

The security of our scheme is based on the Ring Learning With Errors (RLWE) assumption. It can be stated as follows:

- Given a ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, a secret polynomial $\mathbf{s} \in R_q$, and a small error polynomial $\epsilon \in R_q$, the RLWE problem is to distinguish the distribution $(\mathbf{as} + \mathbf{e}, -\mathbf{a})$ from the uniform distribution over $R_q \times R_q$, where $\mathbf{a} \in R_q$ is chosen uniformly at random.

The RLWE problem is considered hard based on worst-case hardness assumptions of certain lattice problems, making it computationally infeasible to solve for sufficiently large parameters.

Semantic security ensures that an adversary cannot distinguish between encryptions of any two chosen plaintexts with a non-negligible advantage. The semantic security proof for the RLWE-based FHE scheme involves a reduction from breaking the scheme to solving the RLWE problem.

4.1 Reduction to RLWE

The ciphertext from the proposed scheme has the following form:

$$\text{Enc}(\mathbf{m}, sk) = \mathbf{c} = (c_0, c_1)$$

$$[c_0 + c_1 s]_q = \Delta_2 \mathbf{e}^* + \Delta_1 \mathbf{m}$$

$$c = (c_0, c_1) = (\mathbf{a} \cdot \mathbf{s} + \Delta_2 \mathbf{e}^* + \Delta_1 \mathbf{m}, -\mathbf{a})$$

Thus, solving $(\mathbf{a} \cdot \mathbf{s} + \Delta_2 \mathbf{e}^* + \Delta_1 \mathbf{m}, -\mathbf{a})$ is equivalent to solving a RLWE problem $(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}, -\mathbf{a})$. If there exists an adversary $\mathcal{A}$ that can break the semantic security of the scheme i.e. it can solve $(\mathbf{a} \cdot \mathbf{s} + \Delta_2 \mathbf{e}^* + \Delta_1 \mathbf{m}, -\mathbf{a})$ then the same adversary can solve any RLWE sample $(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}, -\mathbf{a})$.

Detailed Proof

To rigorously establish the reduction, we need to show that the simulation is indistinguishable from a real execution of the scheme. This involves demonstrating that:

- The public key $pk = (\mathbf{b}, -\mathbf{a})$ generated using the RLWE instance is indistinguishable from a valid public key in the scheme.
- The challenge ciphertext $\mathbf{c}$ generated using the RLWE instance is indistinguishable from a valid ciphertext in the scheme.

Public Key Indistinguishability:

- In a real public key $pk = (\mathbf{as} + \epsilon, -\mathbf{a})$, the polynomial $\mathbf{b}$ follows the RLWE distribution.
- By the RLWE assumption, the polynomial $\mathbf{b}$ in the RLWE instance $(\mathbf{a}, \mathbf{b})$ is computationally indistinguishable from a polynomial following the RLWE distribution.

Ciphertext Indistinguishability:

- In a real ciphertext $\mathbf{c} = (c_0, c_1)$, the polynomials c_0 and c_1 are computed as $c_0 = \mathbf{pk_0} \cdot \mathbf{u} + \epsilon' + \mathbf{m}^* \mod q$ and $c_1 = \mathbf{pk_1} \cdot \mathbf{u} + \epsilon'' \mod q$.
- Using the RLWE instance $(\mathbf{a}, \mathbf{b})$, the challenge ciphertext $\mathbf{c}$ is constructed in the same way, ensuring its indistinguishability from a real ciphertext.

- If the adversary $\mathcal{A}$ can distinguish the challenge ciphertexts with a non-negligible advantage, it implies a non-negligible advantage in distinguishing the RLWE sample from uniform, thus solving the RLWE problem.
- Since the RLWE problem is assumed to be hard, the adversary $\mathcal{A}$ cannot have a non-negligible advantage in distinguishing ciphertexts, proving that the FHE scheme is semantically secure under CPA.

This completes the security proof for the RLWE-based FHE scheme, demonstrating that breaking the scheme is as hard as solving the RLWE problem.

The security of the proposed cryptographic scheme is based on the hardness of the Ring Learning With Errors (RLWE) problem, similar to many lattice-based cryptographic constructions. This section provides a detailed security analysis of the scheme, focusing on the core components that contribute to its security: the RLWE problem, noise growth, and the impact of homomorphic operations.

4.2 Reduction to BGV

The security of the proposed cryptographic scheme can be further understood by considering a reduction to the Brakerski-Gentry-Vaikuntanathan (BGV) scheme. This section formally describes the process and implications of such a reduction.

Objective:

- Demonstrate that an attack method X for the proposed FHE scheme can be leveraged to compromise the BGV scheme.

Reduction Proof Model:

- Assume there exists an attack method X that compromises the proposed FHE scheme.
- The same method X can be adapted to attack the BGV scheme as follows:
  - Convert the BGV ciphertext to a ciphertext of the proposed FHE scheme.
  - Apply the method X to the converted ciphertext to recover the secret key or plaintext.

Formal Conversion Process

Consider the proposed scheme with public parameters $PP = (n, q, \chi, p_1, p_2)$. The ciphertext in this scheme is represented as:

$$c = (c_0, c_1) = ([\mathbf{a} \cdot \mathbf{s} + \Delta_2 \mathbf{e}^* + \Delta_1 \mathbf{m}]_q, -\mathbf{a})$$

To illustrate the encryption of zero:

$$E(0) = ([\mathbf{a} \cdot \mathbf{s} + \Delta_2 \mathbf{e}^*]_q, -\mathbf{a})$$

Given the relationship $p_1[p_1^{-1}]_{p_2} = 1 \mod p_2$, there exists an integer k such that $p_1[p_1^{-1}]_{p_2} = kp_2 + 1$. Let $t = kp_2 + 1$, thus: $\Delta_2 \mathbf{e}^* = t\mathbf{e}^*$. Consequently, the ciphertext can be rewritten as:

$$E(0) = ([\mathbf{a} \cdot \mathbf{s} + t\mathbf{e}]_q, -\mathbf{a})$$

This demonstrates that any ciphertext from the proposed scheme can be converted into a ciphertext from the BGV scheme with the same set of public parameters (n, q, χ), albeit with a different plaintext space defined by t.

Conversely, consider a BGV ciphertext with public parameters (n, q, χ) and plaintext space t:

$$E(0) = ([\mathbf{a} \cdot \mathbf{s} + t\mathbf{e}]_q, -\mathbf{a})$$

We can determine p_1 and p_2 such that $t = kp_2 + 1 = p_1[p_1^{-1}]_{p_2}$, leading to:

$$E(0) = ([\mathbf{a} \cdot \mathbf{s} + t\mathbf{e}]_q, -\mathbf{a}) = ([\mathbf{a} \cdot \mathbf{s} + p_1[p_1^{-1}]_{p_2}(\mathbf{e})]_q, -\mathbf{a})$$

$$= ([\mathbf{a} \cdot \mathbf{s} + CRT_{p_1,p_2}(0, e_1) + CRT_{p_1,p_2}(0, e_2)x + ...CRT_{p_1,p_2}(0, e_n)x^{n-1}]_q, -\mathbf{a})$$

$$= ([\mathbf{a} \cdot \mathbf{s} + \Delta_2 \mathbf{e}^*]_q, -\mathbf{a})$$

Thus, this results in a ciphertext of the proposed FHE scheme with the same public parameters.

In summary, the objective of this reduction is to demonstrate that an attack method X on the proposed FHE scheme can be employed to compromise the BGV scheme. The reduction proof model entails converting BGV ciphertexts to those of the proposed FHE scheme and applying the attack method X. Through this formal conversion process, we establish that any attack on the proposed scheme can be translated into an attack on the BGV scheme, thereby reinforcing the security of the proposed FHE scheme by leveraging the well-established security properties of the BGV scheme.

4.3 Parameter Selection and Security Levels

The security of our scheme relies on choosing appropriate parameters (n, q, p_1, p_2) such that the underlying RLWE problem is hard to solve. The parameters must be selected to balance between security and efficiency:

– Degree n: The degree of the cyclotomic polynomial, typically chosen as a power of 2 for efficiency in polynomial arithmetic.
– Ciphertext modulus q: A large integer modulus that affects both the security level and the noise growth in ciphertexts.
– Plaintext and Noise moduli p_1, p_2: Smaller moduli that determine the plaintext space and noise space respectively.

For a given security level λ, these parameters must satisfy certain constraints to ensure that the RLWE problem remains hard, even for powerful adversaries.

4.4 Practical Considerations

In practical implementations, security considerations must account for potential side-channel attacks and implementation-specific vulnerabilities. Robust parameter selection and efficient noise management are crucial for ensuring both security and performance.

– Side-Channel Attacks: Implementation techniques such as constant-time algorithms and masking can help mitigate side-channel threats.
– Parameter Tuning: Ensuring that parameters are chosen to balance efficiency, correctness, and security according to the desired application and threat model.

The security of our scheme fundamentally relies on the hardness of the RLWE problem and careful management of noise growth in ciphertexts. By selecting appropriate parameters and employing techniques to control noise, our scheme provides a robust framework for fully homomorphic encryption with strong security guarantees. The formal security proofs and reductions to the RLWE problem underpin the confidence in the scheme's resilience against adversarial attacks.

5 Performance Analysis

In this section, we analyze the performance of our Fully Homomorphic Encryption (FHE) scheme based on various cryptographic operations. The performance metrics were obtained using a MacBook Pro M1 Max with 64 GB of RAM, without any special hardware or command set optimizations. The results presented here are for reference only and may vary depending on the hardware configuration.

5.1 Overview of Performance Metrics

The performance of the proposed FHE scheme was evaluated across several types of operations, including encryption, decryption, and basic arithmetic operations (addition, subtraction, and multiplication) on multiple size of plaintext data. We provide the average time taken for 1 million operations and the average number of operations performed per minute.

Operation Type	Encryption	Decryption	Addition	Subtraction	Multiplication
Time	19 sec	18 sec	0.9 sec	0.74 sec	81 sec

Table 1. Time per Million

Time per Million Operations

The table below shows the average time (in seconds) required to perform 1 million operations for different data types and operations:

Operations per Minute

The table below shows the number of operations performed per minute for different data types and operations:

Operation Type	Encryption	Decryption	Addition	Subtraction	Multiplication
Time	3,157,895	3,333,333	66,666,667	81,081,081	740,741

Table 2. Operations per Minute

Analysis

- Encryption and Decryption: The average time for 1 million encryptions is 19 seconds, while decryption takes approximately 18 seconds. This translates to around 3,157,895 encryptions and 3,333,333 decryptions per minute for integer data types.
- Arithmetic Operations: Addition and subtraction are significantly faster than multiplication. The system can perform up to 66,666,667 additions and 81,081,081 subtractions per minute, but only 740,741 multiplications.

Our performance analysis demonstrates that the proposed scheme is capable of handling a high volume of cryptographic operations efficiently, particularly for basic arithmetic operations. The scheme performs best with additions and subtractions, while multiplications is more computationally intensive. The results highlight the feasibility of using FHE for practical applications, although actual performance may vary based on specific hardware and ongoing optimizations.

6 Conclusion

In this paper, we have introduced a novel Fully Homomorphic Encryption (FHE) scheme that leverages the Chinese Remainder Theorem (CRT) to address the critical issue of noise accumulation in homomorphic computations. By uniquely encoding the plaintext with noise, the proposed scheme effectively prevents the noise from growing to the point where it corrupts the plaintext, thereby facilitating smooth computations on encrypted data without the need for bootstrapping.

This advancement is significant, as it allows for a predefined number of modular operations on encrypted data, enhancing both efficiency and practicality.

The evolution of FHE has been marked by notable milestones since its conceptual inception by Rivest in the 1970s and its formal realization by Gentry in the late 2000s. While numerous efforts have been made to develop FHE schemes using CRT, these have often fallen short due to vulnerabilities such as chosen plaintext attacks (CPA). Our proposed scheme addresses these vulnerabilities by incorporating random errors into the encryption process in a controlled manner. These errors are designed such that, during homomorphic operations, their growth does not interfere with the integrity of the plaintext, unlike in traditional LWE-based schemes.

The key contributions of the proposed scheme can be summarized as follows:

- Noise Management: By encoding the plaintext with noise in a novel way, we ensure that the noise does not overflow and corrupt the plaintext, enabling reliable homomorphic computations.
- CRT Utilization: We harness the power of the Chinese Remainder Theorem to construct a robust FHE scheme that supports a set number of modular operations without necessitating bootstrapping.
- Security Enhancement: the proposed scheme is resilient to chosen plaintext attacks, thereby improving the security profile compared to previous attempts utilizing CRT.

This research represents a significant step forward in the development of efficient and secure FHE schemes. By overcoming the traditional limitations associated with noise management and CPA vulnerabilities, the proposed scheme opens up new possibilities for practical applications of FHE in fields such as secure data processing, privacy-preserving computations, and encrypted machine learning. Future work will focus on further optimizing the performance and expanding the range of supported operations to enhance the scheme's applicability in diverse real-world scenarios.

References

1. Ahmad Al Badawi and Yuriy Polyakov. Demystifying bootstrapping in fully homomorphic encryption. 2023.
2. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. *Advances in Cryptology – CRYPTO 2012*, 7417:868–886, 2012.
3. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. *Advances in Cryptology – CRYPTO 2011*, 6841:505–524, 2011.
4. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. *P. Rogaway, editor, Advances in Cryptology – CRYPTO 2011*, 6841:505–524, 2011.
5. Ernest F. Brickell and Yacov Yacobi. On privacy homomorphisms. 1987.
6. Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. 2018.

7. J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. *To Appear at Eurocrypt 2013*.
8. Robin Geelen and Frederik Vercauteren. Bootstrapping for BGV and BFV revisited. 2022.
9. C. Gentry, S. Halevi, and N. Smart. Fully homomorphic encryption with polylog overhead. *Advances in Cryptology – EUROCRYPT 2012*, 7237:465–482, 2012.
10. Craig Gentry. Fully homomorphic encryption using ideal lattices. volume 9, pages 169–178, 05 2009.
11. Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. pages 116–137, 08 2010.
12. Shai Halevi and Victor Shoup. Bootstrapping for HElib. 2014.
13. Ronald L. Rivest and Michael L. Dertouzos. On data banks and privacy homomorphisms. 1978.
14. M. v. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. *Advances in Cryptology – EUROCRYPT 2010*, 6110:24–43, 2010.