A REPORT

ON

Testing a Hybrid FHE Scheme based on NTRU and RLWE

BY

Yusra Hakim 2022A7PS0004U
Joseph Cijo Njaliath 2022A7PS0019U

Cryptography (CS F463)

BITS Pilani, Dubai Campus

International Academic City, Dubai

UAE

Testing a Hybrid FHE Scheme based on NTRU and RLWE

Joseph Cijo (f20220019@dubai.bits-pilani.ac.in) and Yusra Hakim (f20220004@dubai.bits-pilani.ac.in)

Department of Computer Science, Birla Institute of Technology and Science, Pilani, Dubai Campus, Dubai, UAE

Abstract: Fully Homomorphic Encryption (FHE) permits computations on data without decrypting it, allowing processing within third-party or untrusted environments. One of the major hurdles with homomorphic encryption is managing the noise that increases during operations, as this can pollute ciphertexts, not allowing the right decryption to the intended data. This paper attempts to implement a hybrid FHE scheme that is then compared to existing FHE schemes. By combining the computational efficiency of NTRU with the strong security guarantees of RLWE, this approach offers improved performance in bootstrapping and noise reduction. The study further reviews existing literature, analyzes various optimization strategies, and explores the viability of NTRU-RLWE as a framework for practical FHE applications.

Keywords: Homomorphic Encryption, Multi-Key Encryption, Cryptography, NTRU, RLWE, Bootstrapping

1. Introduction

With privacy worries about people's personal data on the rise, information security and privacy protection requirements are under greater scrutiny. In today's digital age, preserving data security in any digital environment necessitates the encryption of data using various cryptographic methods prior to its storage in those contexts. Traditional encryption methods are highly effective at protecting data at rest or in transit, but they fall short when data must be decrypted for computation, exposing it to potential breaches. Homomorphic encryption has emerged as one of the most promising approaches to safe data encryption for third party data analysis and further computation.

This section briefly discusses homomorphic encryption, bootstrapping, multi-key encryption, and NTRU, the currently used methods of homomorphic encryption, and the objectives of this paper.

1.1. Background of Study

A technique known as homomorphic encryption permits certain operations to be performed on the ciphertext, and the outcome of these operations when decrypted is the same as the outcome of carrying out the same operations on the plaintext. In essence, the encrypted data is used for computations without first requiring its decryption. Sensitive data can be processed in third-party environments thanks to this encryption technique.

Assume two homomorphic functions, $\mathrm{Enc}_k(p)$, where the plaintext p is encrypted under a key k to generate the ciphertext c, and $\mathrm{Dec}_k(c)$, where the ciphertext c is decrypted using the same key k to get back the plaintext p. A simplified example of this process looks like:

- *Encryption:* Bob encrypts two numbers, 5 and 9, using the encryption function. He now has ciphertexts c_1 and c_2.

$$c_1 = \mathrm{Enc}_k(5)$$
$$c_2 = \mathrm{Enc}_k(9)$$

- *Computation / Evaluation:* The ciphertexts c_1 and c_2 are then sent to some third-party server, where the homomorphic operation $c_1 + c_2$ occurs, resulting in a new ciphertext c_3. Such that:

$$\begin{aligned} \mathrm{c}_3 &= c_1 + c_2 \\ &= \mathrm{Enc}_k(5) + \mathrm{Enc}_k(9) \\ &= \mathrm{Enc}_k(14) \end{aligned}$$

- *Decryption:* Bob receives the ciphertext c_3 back from the server and decrypts it to get 14, which is the result of the operation $5 + 9$.

$$\begin{aligned} \mathrm{Dec}_k(c_3) &= \mathrm{Dec}_k\big(\mathrm{Enc}_k(14)\big) \\ &= 14 \\ &= 5 + 9 \end{aligned}$$

  Throughout this process, the third party never learns the actual numbers, but can still perform the computation.

There are a few types of Homomorphic Encryption:

1. Partially Homomorphic Encryption can only perform one type of Operation.
2. Somewhat Homomorphic Encryption is limited Operations of a variety of types.
3. Fully Homomorphic Encryption supports unlimited Operations.

"Bootstrapping," in the context of homomorphic encryption, is a technique used to reduce the errors accumulated in the ciphertext over time when performing homomorphic operations. First introduced by Gentry in his seminal paper on fully homomorphic encryption, bootstrapping is performed to "refresh" the ciphertext c, i.e., to encrypt it again under another key to obtain another ciphertext for the same plaintext but now with a reduced error [1].

Consider a homomorphic encryption scheme, two pairs of keys, (sk_1, pk_1) and (sk_2, pk_2), an encryption algorithm under key k for message m, $\mathrm{Enc}_k(m)$, a decryption algorithm under key k for ciphertext c, $\mathrm{Dec}_k(c)$ and the ciphertext c, such that,

$$\mathrm{c} = \mathrm{Enc}_{\mathrm{pk1}}(\mathrm{m})$$

There are three steps to "refresh" the ciphertext c which are as follows:

1. *Encrypt the secret key sk_1 under pk_2:*

$$\mathrm{Enc}_{pk_2}(sk_1) \rightarrow \overline{sk_1}$$

2. *Encrypt the ciphertext c under pk_2:*

$$\mathrm{Enc}_{pk_2}(c) = \bar{c}$$

3. *Decrypt the new ciphertext $\bar{c}$ homomorphically using encrypted secret key $\overline{sk_1}$:*

$$\mathrm{Dec}_{\overline{sk_1}}(\bar{c})$$

The third step results in encryption of the same message under the second public key, i.e., $\mathrm{Enc}_{pk_2}(m)$. Which means, when evaluated under the evaluation key evk, using the evaluation algorithm $\mathrm{Eval}_{evk}(\mathrm{Dec}, c, k)$,

$$\begin{aligned}
&\mathrm{Eval}_{evk}\left(\mathrm{Dec}, \bar{c}, \overline{sk_1}\right) \\
&= \mathrm{Eval}_{evk}\left(\mathrm{Dec}, \mathrm{Enc}_{pk_2}(c), \mathrm{Enc}_{pk_2}(sk_1)\right) \\
&= \widehat{\mathrm{Enc}_{pk_2}}\left(\mathrm{Dec}_{sk_1}(c)\right) \\
&= \widehat{\mathrm{Enc}_{pk_2}}(m)
\end{aligned}$$

The value of this new $\widehat{\mathrm{Enc}_{pk_2}}(m)$ is the same as $\mathrm{Enc}_{pk_2}(m)$, such that,

$$\mathrm{Dec}_{sk_2}\left(\widehat{\mathrm{Enc}_{pk_2}}(m)\right) = \mathrm{Dec}_{sk_2}\left(\mathrm{Enc}_{pk_2}(m)\right) = m$$

Figure 1, as seen below, shows a diagrammatic representation of the bootstrapping process.

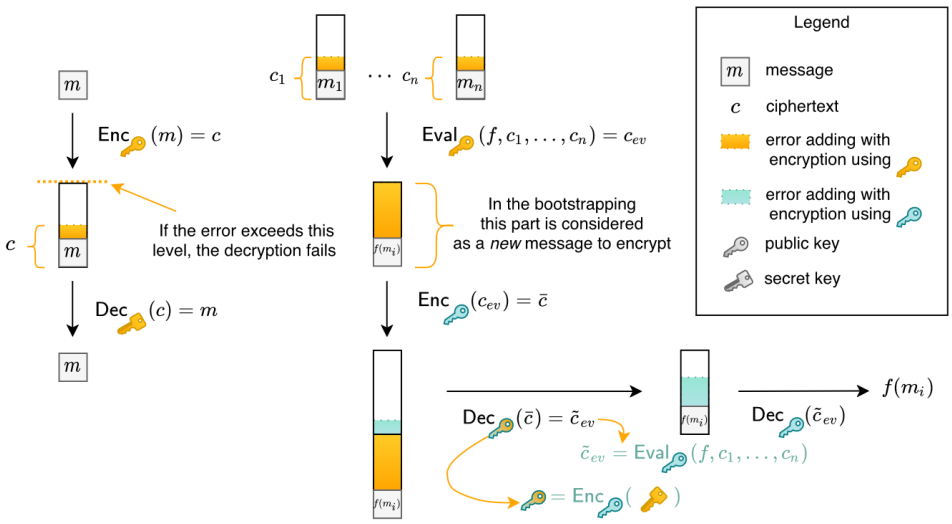

Figure 1 Bootstrapping Technique

The main objective of bootstrapping is to reduce the error accumulating in the ciphertext. Unfortunately, the main downside of bootstrapping is the fact that it requires large memory size and is computationally taxing.

An extension of Fully Homomorphic Encryption, Multi-Key FHE supports computations on ciphertexts encrypted using different keys. Practically, this allows more than one party to contribute encrypted data. Decryption would require a combination of all the secret keys used by the parties involved. This also ensures cooperation between the parties involved, as no one can see the plaintext output without the secret keys from all other parties [2].

Usually, a MKFHE scheme consists of five algorithms [2]:

- *Setup:* Output a public parameter pp for a given security parameter λ.

$$\text{Setup}(i^{\lambda}) \rightarrow \text{pp}$$

- *Key Generation Function:* A function to generate the secret key sk and public key pk for each part given a public parameter pp.

$$\text{KeyGen}(pp) \rightarrow (sk, pk)$$

- *Encryption Function:* Encrypts and output a fresh ciphertext c for a given public key pk and a message m. The ciphertext also contains the indices of corresponding parties.

$$\text{Enc}(\text{m}, \text{pk}) \rightarrow c$$

- *Decryption Function:* Decrypts and outputs the message m for a given ciphertext c and the sequence of secret keys, sk_i such that $1 \leq i \leq k$ of all the relevant parties k.

$$\text{Dec}(c, \{sk_i\}_{1 \le i \le k}) \rightarrow m$$

- *Evaluation Function:* Evaluates and outputs a ciphertext $\bar{c}$ for the k ciphertexts $\{c_i\}_{1 \le i \le k}$ to be computed over a Boolean circuit $\mathcal{C}$ over the sequence of public keys $\{pk_i\}_{1 \le i \le k}$. Like the encryption function, the ciphertext returned here also contains the indices of the k related parties.

$$\text{Eval}(\mathcal{C}, \{c_i\}_{1 \le i \le k}, \{pk_i\}_{1 \le i \le k}) \rightarrow \bar{c}$$

NTRU is a lattice-based public-key cryptosystem for data encryption and decryption. The name comes from the initials of the pun *Number Theorists 'R' Us*. It is an efficient algorithm, suitable for environments where performance is critical. With inbuilt support for multi-key encryption, it is the default algorithm used for encryption tasks with more than one party involved. It is built upon operations in polynomial rings that are also modulo a small integer, allowing for fast polynomial arithmetic.

The basic flow (as depicted in *Figure 2*) of NTRU Encryption and Decryption are:

1. *Key Generation:*
    - ***Input:*** Small polynomials f and g and modulus q
    - ***Output:*** Public key h, such that

$$h = \frac{g}{f} \bmod q$$

2. *Encryption:*
    - ***Input:*** Message m, random small polynomial r, public key h, and modulus q
    - ***Output:*** Ciphertext c, such that

$$c = r \times h + m \bmod q$$

3. *Decryption:*
    - ***Input:*** Ciphertext c, private key f
    - ***Output:*** Recover message using inverse of f modulo p and q

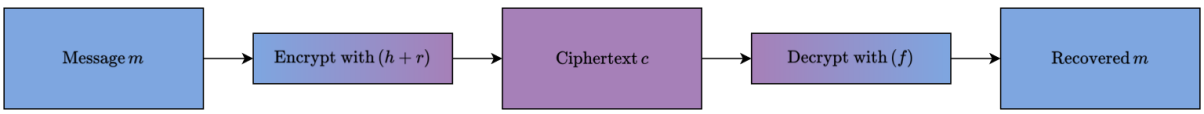

Figure 2 Flow of NTRU cryptosystem

While it is a fast algorithm, standalone NTRU is not very safe. Schemes like NTRU-RLWE are now being considered for their higher theoretical security.

Ring Learning With Errors (RLWE) is another lattice-based cryptographic assumption that is an extension of the Learning With Errors (LWE) problem, adapted to work efficiently over polynomial rings, which makes operations faster and ciphertexts smaller. At its core, RLWE is built on the idea that solving certain equations with small errors (noise) added is computationally hard.

Mathematically, LWE looks like the following:

Given equations like:

$$a \times s + e \equiv b \,(\mathrm{mod}\, q)$$

Where a is a public (known) value, s is a secret (unknown) value, e is a small random error, and b is the result with respect to a modulus q. But finding s is extremely difficult even when a and b are known due to the presence of the error e.

Unlike LWE, RLWE uses polynomials, the equations become:

$$a(x) \times s(x) + \mathrm{e}(x) \equiv b(x) \left(\mathrm{mod}\, q,\ f(x)\right)$$

Here, all values are polynomials with modulus q along some ring like $\mathbb{Z}_{\frac{q[x]}{f(x)}}$, $e(x)$ is a small error polynomial, $a(x)$ is a public polynomial, $s(x)$ is the secret key, and $b(x)$ is the resultant ciphertext. Just like with LWE, finding $s(x)$ given $a(x)$ and $b(x)$ is especially hard due to the noise $e(x)$.

1.2. *Current Methods*

Fully Homomorphic Encryption (FHE) has four novel branches (or Generations) of studies [1] and the *Figure 3* shows a timeline of these schemes:

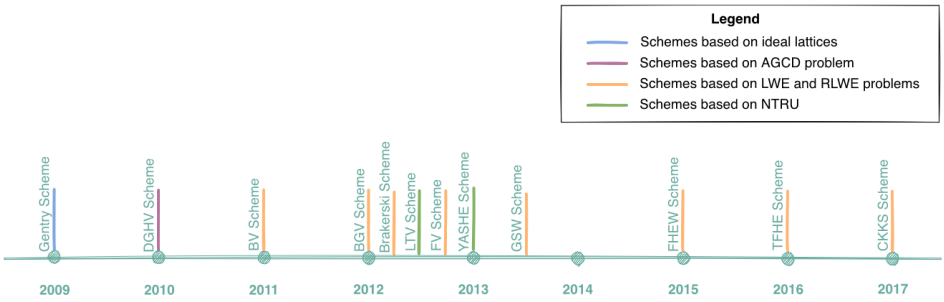

Figure 3 Timeline of the main FHE Schemes [1]

- First Generation – FHE Based on Ideal Lattices

    - The FHE Scheme presented by Gentry (decryption algorithm complex, hence not bootstrappable)
- First Generation – FHE Based on the Approximate Greatest Common Divisor (AGCD) Problem
    - The Dijk-Gentry-Halevi-Vaikuntanathan (DGHV) Scheme
- Second Generation – FHE Based on (LWE) and (RLWE)
    - The Brakerski-Vaikuntanathan (BV) Scheme
    - The Brakerski-Gentry-Vaikuntanathan (BGV) Scheme
    - Fan-Vercauteren (FV) Scheme
- Second Generation – FHE Based on NTRU
    - The Lopez-Alt-Tromer-Vaikuntanathan (LTV) Scheme
    - The Yet Another Somewhat Homomorphic Encryption (YASHE) scheme
- Third Generation – FHE Based on LWE and RLWE
    - The Gentry-Sahai-Waters (GSW) Scheme
    - The Torus Ring Learning With Errors (TRLWE) Scheme
- Fourth Generation – FHE Based on LWE and RLWE
    - The Cheon-Kim-Kim-Song (CKKS) Scheme

Some more current methods are discussed further below in the Literature Review section (section Literature), like the Approximate Homomorphic Encryption, the Multi-Key Fully Homomorphic Encryption scheme, the Multikey Verifiable Homomorphic Encryption scheme, a hardware architecture for the Brakerski-Vaikuntanathan scheme, a Multi-Identity Fully Homomorphic Encryption scheme and so on.

The *Figure 4* (shown below) shows the list of some common advantages and disadvantages of some of the second, third and fourth generations of the fully homomorphic encryption (FHE) schemes.

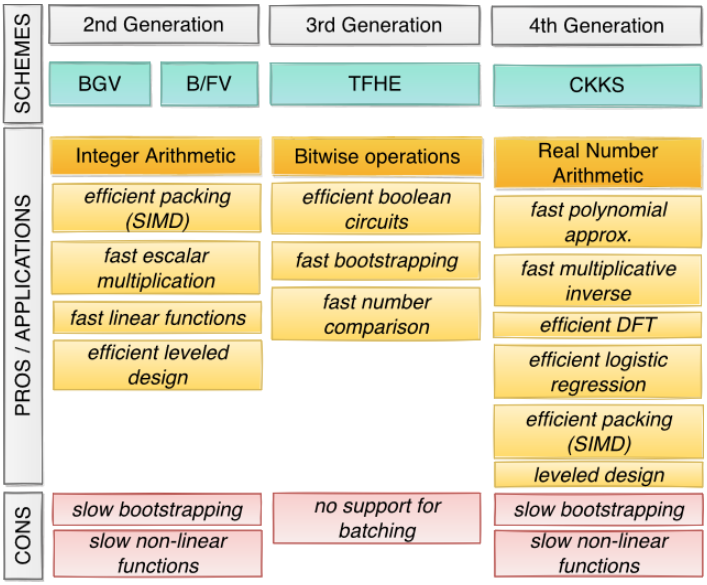

Figure 4 Pros and cons of FHE schemes by generation.

1.3. Objectives

- Introduce and explain the foundational concepts behind Homomorphic Encryption and its types.
- Analyze the role of noise in FHE schemes.
- Test the viability of a hybrid scheme.
- Compare approaches to FHE schemes.
- Propose future scope of proposed scheme.

2. Literature Review

The Table 1 (provided after the individual reviews) summarizes the ten papers based on the objectives, approach, key features, noise management, performance, applications and contributions along with the limitations.

The 2022 survey by *C. Marcolla et al.*, provides an overview of Fully Homomorphic Encryption (FHE), all the way from when Gentry proposed the very first lattice-based FHE scheme in 2009 to the more modern implementations like Brakerski-Gentry-Vaikuntanathan (BGV), Fan-Vercauteren (FV), Fast Fully Homomorphic Encryption over the Torus (TFHE), and Cheon-Kim-Kim-Song (CKKS) and so on. The paper also briefs on the many mathematical topics used by these schemes and bootstrapping and further elaborate on the four generations of FHE schemes discussed above [1]. They conclude by identifying the key challenges faced in adopting FHE schemes, including computation cost, the need to have a standardized benchmark, and the expansion of ciphertext [1].

The more later, 2025 survey by *W. Liu et al.*, covers the topic of "Approximate Homomorphic Encryption" and examines the shift from exact arithmetic to approximate arithmetic [3]. They focus on the CKKS scheme and the role it played to enable efficient computation on encrypted numbers, both real and imaginary. The survey also categorizes approximate homomorphic encryption (AHE) into three principal areas, namely, precision improvements, bootstrapping optimizations, and scheme variants (multi-key CKKS and Residue Number System – CKKS (RNS-CKKS)). They also go into detail of the various applications of AHE and how the CKKS scheme is suitable for finance, real number arithmetic in Machine Learning applications, and in the healthcare industries [3].

The 2024 paper by *K. Xu et al.*, discusses a novel "Multi-Key Fully Homomorphic Encryption" (MKFHE) scheme that optimizes bootstrapping efficiency by combining NTRU and RLWE assumptions [2]. NTRU's native multi-

key homomorphic encryption as well as its cyphertext structure is used to design a dynamic MKFHE system. The experimental results in the paper prove a boost in performance and outperform the MK-TFHE designed by *Chen et al.* by 5.3 times. The paper also highlights some practical applications of this scheme, like secure neural networks [2].

This next 2024 paper by *L. Bai et al.,* addresses one of the main challenges of fully homomorphic encryption, the noise growth experienced when performing homomorphic operations on encrypted data [4]. This noise growth limits the number of homomorphic operations that can be performed before losing the original message. The authors categorize noise management into three methods, namely, bootstrapping, bit-expansion, and scaling, and analyze their advantages and trade-offs. This paper optimizes the above mentioned GSW scheme and shows improved noise management and computational efficiency as compared to the traditional methods [4].

The 2022 article by *Y. Li et al.*, introduces a "Multikey Verifiable Homomorphic Encryption" (MVHE) scheme which enables private, secure, and verifiable computations [5]. Unlike traditional homomorphic encryption (HE), which allows computations on encrypted data, multikey verifiable homomorphic encryption (MVHE) extends this to ciphertexts encrypted under different keys. The authors implemented this scheme by combining multikey homomorphic encryption (MHE) with multikey homomorphic encrypted authentication (MHEA). Here each user can encrypt their own data with their secret key, and homomorphic calculations end up producing ciphertexts that can be verified through a decryption process that involves all participants' keys. The paper also mentions some practical scenarios for this scheme like collaborative medical analysis, where the resulting output must be both correct and private [5].

The 2022 article by *S. Behera et al.*, addresses the computational overhead when performing computations on fully homomorphic encrypted ciphertext by proposing a new Field Programmable Gate Array (FPGA) – based hardware architecture for the Brakerski-Vaikuntanathan (BV) scheme [6]. The authors implemented key generation, encryption, and decryption algorithms on an FPGA using Number Theoretic Transform (NTT) for efficient operations like multiplication. The architecture, described in the paper, uses modular designs for the pseudo-random number generation (PRNG), NTT, and inverse NTT (INTT) blocks that have been optimized to be performed on FPGAs. Some results mentioned in this paper are the execution times as 0.33 seconds for the key generation, 0.85 seconds for encryption, and 72 microseconds for decryption [6].

The 2024 paper by *A. K. Vangujar, et al.*, introduces an electronic voting system with inbuilt privacy preservation, by integrating Group Identity-Based Identification (GIBI) and homomorphic encryption (HE) [7]. This novel framework authenticates the voters using zero-knowledge (ZK) proofs, ensuring anonymity as well as ensuring eligibility. The authors use Discrete Logarithmic (DL) assumption to prepare the electronic voting framework. The scheme generates tokens to ensure that the same vote is not reused and uses partial decryption to ensure

transparency in tallying, hence providing a decentralized, and cryptographically secure solution for electronic voting [7].

The 2023 article prepared by *Z. Koo et al.* addresses the challenge of the large key sizes used for evaluations in multi-key homomorphic encryption (MKHE) and threshold multi-key homomorphic encryption (TMKHE) schemes [8]. To counter the computational and communicative bottlenecks posed by traditional MKHE schemes, since they require the client to generate and manage large keys used to evaluate encrypted ciphertext, the authors of this article introduce a Ring Learning With Errors reusing Errors (ReRLWE) variant of the RLWE problem, where the error terms are reused to construct compressed evaluation keys. According to the results published in the article, multiplication key sizes have been reduced by 50% for TMKHE schemes and by 66% for MKHE schemes, while rotation keys shrink by 25% for both the schemes. Like earlier papers, the practical implementations include real-world applications like secure neural networks (SNNs) [8].

The 2022 paper by *X. Yang et al.*, addresses the inefficiency of relinearization process in the Chen-Dai-Kim-Song from 2019 (CDKS19) multikey homomorphic encryption scheme (MKHE) [9]. Relinearization is a crucial step in homomorphic multiplication that consumes significant resources due to the large size of the keys and error growth during these homomorphic operations. To solve these issues, the authors propose an optimized algorithm to perform relinearization that reduces the key size used for evaluation as well as controlling the noise generated. This approach also skips past the ciphertext expansion, thus enabling much more efficient homomorphic computations. The paper also shows a 66% reduction in the multiplication key size and a 25% reduction in the rotation key size (like *Z. Koo et al.*) as well as 45% speed up in homomorphic multiplication as compared to the traditional CDKS19 scheme [9].

The 2024 article by *G. Tu et al.*, introduces a Multi-Identity Fully Homomorphic Encryption (MIBFHE) scheme that is designed to counter the inefficiencies like large computational overhead due to large evaluation keys and regenerating fresh ciphertexts in existing identity-based encryption (IBE) and multi-key fully homomorphic encryption (MKFHE) [10]. And so, the authors propose a “decomposition method” for MKFHE where the ciphertexts are split into public key and plaintext components to enable direct generation of the extended ciphertexts without relying on the intermediate fresh ciphertexts. This proposition lowers the noise growth and reduces the auxiliary ciphertext sizes [10].

Table 1 Comparison of Journal Articles

References	Objective / Focus	Approach / Scheme	Key Features / Noise Management	Performance / Applications	Contributions / Limitations
[1]	A comprehensive survey of the evolution of FHE and its implementations	Gentry, BGV, FV, TFHE, CKKS	Overview of 4 FHE generations, bootstrapping	Identifies computation cost, benchmarking	Comprehensive review that highlights open challenges in FHE adoption
[3]	Another survey of Approximate Homomorphic Encryption (AHE)	CKKS, RNS-CKKS, MK-CKKS	Focus on precision, bootstrapping, variants	ML, finance, healthcare	Categorizes AHE advancements, applications, and future directions but does not cover much on exact HE
[2]	Multi-Key Full Homomorphic Encryption (MKFHE) with efficient bootstrapping	MKFHE (NTRU + RLWE)	Dynamic multi-key system, improved bootstrapping	Secure neural networks, outperforms prior MK-TFHE	Boosts performance 5.3x and is practical for collaborative ML
[4]	Noise management in FHE using the GSW scheme	GSW, noise management	Bootstrapping, bit-expansion, scaling	Improved computational efficiency	Optimize noise growth and improved over traditional methods
[5]	Multikey Verifiable Homomorphic Encryption (MVHE)	MVHE, MHE, MHEA	Verifiable computation, multi-user support	Collaborative medical analysis	Enables private, correct, verifiable outputs at the cost of increasing complexity
[6]	Hardware acceleration for FHE using the BV scheme	BV on FPGA, NTT	Modular design, PRNG, NTT/INTT blocks	Keygen: 0.33s, Encrypt: 0.85s, Decrypt: 72µs	Reduces computation time but is hardware-dependent
[7]	Electronic voting with privacy-preserving	GIBI + HE, ZK proofs	Voter authentication, DL assumption	E-voting and decentralized systems	Ensures anonymity and eligibility and provides partial decryption for transparency

[8]	Key size optimization for MKHE/TMKHE	ReRLWE (RLWE variant)	Compressed evaluation keys	Secure neural networks	Reduces key sizes and is practical for SNNs
[9]	Efficient relinearization in multikey HE	CDKS19 optimization	Key size and noise control, skips expansion	66% smaller multiplication keys, 45% faster multiplication	More efficient homomorphic computations
[10]	Multi-Identity FHE (MIBFHE)	MIBFHE, decomposition	Splits ciphertexts for efficient extension	Reduces overhead in IBE/MKFHE	Reduces noise growth and auxiliary ciphertext size

3. Problem and Possible Solutions

This paper aims to create a new FHE scheme combining the existing NTRU and RLWE methods. Recent papers discuss and propose various strategies that balance security, scalability, and efficiency.

We propose a hybrid scheme that combines two cryptographic ideas:

1. NTRU: Fast and efficient encryption
2. RLWE: Higher security

Based on lattices, the security of this combined scheme is very high. The idea behind combining them lies in their individual strengths and how they can handle the other's weaknesses. The

Table 2 (see below) shows a comparison between the two schemes individually.

Table 2 Comparison between NTRU and RLWE schemes

Feature	NTRU	RLWE
Speed	Very fast	Moderate
Security	Practical, but needs careful setup	Extremely strong
Noise handling	Efficient but limited	Very well-studied and controlled
Multi-party use	Naturally supports it	Needs some adapting

In essence, this new hybrid scheme:

4. Encrypts and decrypts data faster [6]
5. Manages noise and bootstrapping more efficiently [2]
6. Enables multiparty encryption
7. Supports extensive computation on encrypted data

With new research in homomorphic encryption, the NTRU-RLWE hybrid scheme stands out primarily due to it merging the computational efficiency of NTRU, alongside the robust security assurances of RLWE based constructions. This combination theoretically achieves efficient encryption, decryption, and especially bootstrapping operations, while supporting secure multiparty computations and still maintaining resistance to known attacks.

4. Simulation Details

This section details the proof-of-concept for the proposed hybrid scheme. The codebase for the proposed scheme can be found at [https://github.com/joejo-joestar/NTRU-RLWE-Hybrid-Scheme]

4.1. Polynomial Operations

A variety of polynomial operations are used, as the hybrid scheme is based in polynomial arithmetic. These simulated polynomial functions are listed out in Table 3. These functions are simplified for the sake of the proof-of-concept and must be reimplemented to be tested for correctness and security.

Table 3 Polynomial Functions

Function	Purpose
`generate_poly(N, q)`	Generates a random polynomial with a degree up to $N-1$ and uniformly sampled coefficients.
`generate_gaussian_poly(N, sigma)`	Generates a random polynomial with a degree up to $N-1$ and coefficients sampled from a discrete Gaussian distribution.
`poly_mod(poly, q)`	Reduces each coefficient of a polynomial modulo q, mapping them to a symmetric range.
`poly_add(poly1, poly2, q)`	Adds two polynomials by adding respective coefficients, and then ensures the resulting coefficient is modulo q.
`poly_mul(poly1, poly2, q, N)`	Multiplies two polynomials in the polynomial ring: $R_{q,N} = \mathbb{Z}_q[x]/(x^N + 1)$
`poly_inverse_ntru(poly, q)`	Used for the ***simulation*** of the calculation of the multiplicative inverse of a polynomial in the ring but does not do the actual calculation.
`estimate_noise_ntru(ciphertext, secret_key, q, N)`	Estimates the noise in the intermediate NTRU value.
`estimate_noise_rlwe(ciphertext, secret_key, q, N)`	Estimated the noise in the RLWE ciphertext.

4.2. NTRU Component

While not an actual scheme, NTRU is a series of algorithms with a basis in lattice cryptography. Table 4 highlights the functions implemented for the NTRU component of the hybrid scheme. This implementation is extremely simplified and must be reimplemented properly later.

Table 4 NTRU Component Functions

Function	Purpose
`_generate_key_pair(self)`	Generates the NTRU public and private key pair.
`encrypt(self, m, h)`	Encrypts a message polynomial `m` with the public key `h`.
`decrypt(self, e, partial)`	Decrypts an NTRU ciphertext `e`.

4.3. RLWE Component

A specialized version of the more general Learning With Errors (LWE) problem, Ring Learning With Errors (RLWE) is adapted to work over polynomial rings instead of vectors over finite fields. The below Table 5 lists the functions defined in the RLWE Component.

Table 5 RLWE Component Functions

Function	Purpose
`encrypt(self, m)`	Encrypts a message polynomial `m`.
`decrypt(self, ciphertext, partial)`	Decrypts an RLWE ciphertext `e`.

4.4. *Hybrid Component*

This component implements the core logic for the combined scheme.

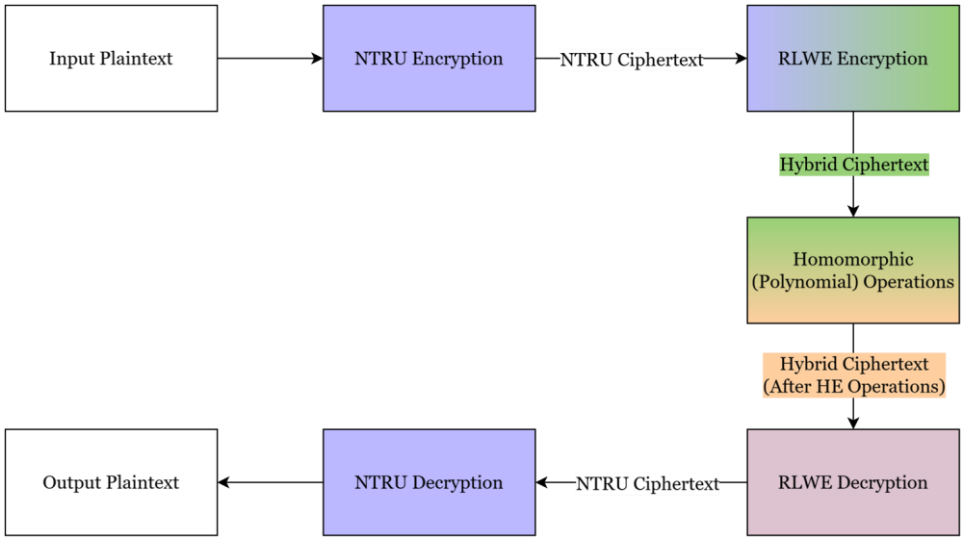

Figure 5 Proposed Hybrid Scheme Architecture

The Figure 5, illustrates the workflow of the hybrid encryption scheme, where the plaintext follows a very simple step-by-step process of encryption followed by some operations performed on it and then finally being decrypted. And the following Table 6 lists out the functions implemented and simulated in the hybrid component.

Table 6 Hybrid Component Functions

Function	Purpose
`encrypt(self, m)`	Two-layer encryption of a plaintext message polynomial `m` – first with NTRU encryption then the encrypted ciphertext is the plaintext for the RLWE encryption.
`decrypt(self, ciphertext, partial)`	Two-layer decryption of a hybrid ciphertext, reversing the encryption operations.
`add(self, ct1, ct2)`	Performs homomorphic addition of two hybrid ciphertexts. As the outer layer is RLWE, this is RLWE addition.
`mul(self, ct1, ct2)`	Performs homomorphic multiplication of two hybrid ciphertexts.
`bootstrap(self, ct)`	Used for the ***simulation*** of noise reduction in a ciphertext.
`estimate_noise(self, ciphertext)`	Estimates the noise levels in both the NTRU and RLWE components of the hybrid ciphertext.

`noise_growth_benchmark(self, operations)`	Benchmarks on how noise accumulates with sequential homomorphic operations.
`_check_decryptable(self, ciphertext)`	Checks if a ciphertext can be successfully decrypted.

4.5. Benchmarking and Demo

The proposed hybrid scheme is finally tested and compared against established FHE schemes like CKKS and BFV from the TenSEAL library, and the next section covers the results. The three schemes are tested for the time taken to perform encryption on the plaintext, addition, and multiplication. the amount of noise generated after repeated homomorphic operations is also compared across the three.

5. Observations

This section details the results observed for the proof-of-concept hybrid scheme proposed in the above section.

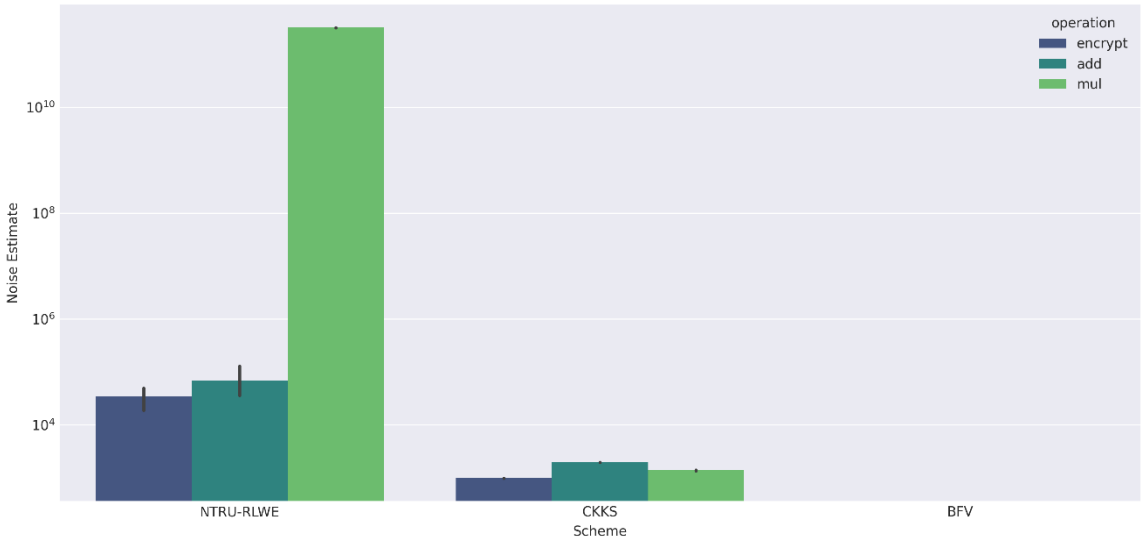

Figure 6 Noise Levels of FHE Operations by Scheme

Figure 6 shows the noise levels of the various operations of the proposed scheme (NTRU-RLWE), against the CKKS and BFV schemes (from the TenSEAL library [11]). This figure shows significant noise in the proposed model. This can be attributed to a few main problems with the proof-of-concept:

1. The simplified multiplication algorithm (as mentioned in section 144.1) does not perform crucial key switching (relinearization)
2. The defined bootstrapping operation does not perform actual bootstrapping (see section 1.1) but is replaced with a placeholder that simply performs modulo arithmetic on the ciphertexts.
3. The majority of the NTRU component (see section 4.2) uses placeholders or simulations of the actual algorithm and not the proper procedures due to the limitations of processing power and insufficient documentation.

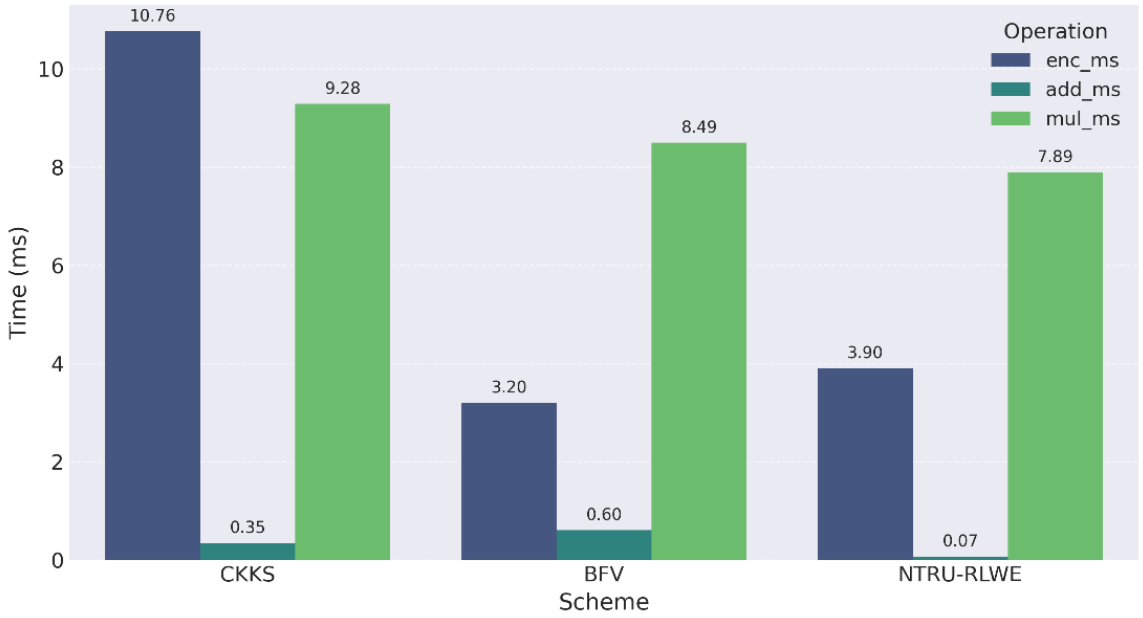

Figure 7 FHE Operations Benchmark by Scheme

The Figure 7, as seen above, shows the time taken, in milliseconds, to perform a very basic encryption, addition, and multiplication operations for the proposed hybrid scheme as well as for the CKKS and BFV scheme from the TenSEAL library. The faster polynomial operations (multiplication mainly) for the hybrid scheme can be mainly attributed to the highly simplified multiplication algorithm, as mentioned earlier, for the proof-of-concept. These simplifications and simulations to lower the computational overhead lead to faster turnaround times but at the cost of high noise generation as discussed above.

6. Future Scope

Given that the current implementation is simply a proof-of-concept and has simplified operations, the future scope can be divided into two main phases.

6.1. Addressing Current Flaws

The main goal for this phase is to convert the proposed proof-of-concept into a functionally correct and cryptographically sound FHE scheme.

- *Implementing Correct Cryptographic Primitives:*
    1. Replace the placeholder `poly_inverse_ntru()` with a functional algorithm that performs polynomial inversion, using the extended Euclidean algorithm for polynomials. This is a crucial step to correctly and fully implement NTRU key generation and decryption.
    2. Proper and careful parameter selection for NTRU to avoid sublattice attacks and other known attacks.
    3. Implement a complete and proper multiplication algorithm for the RLWE and Hybrid components that support key switching to reduce ciphertext size and to control the noise growth from the multiplication operations.
- *Multiplication Noise Management:*
    1. Incorporate modulus switching techniques after performing homomorphic multiplication operations to reduce the size of the noise generated.
- *Implement Functional Bootstrapping:*
    1. Adapt an existing bootstrapping framework like that in FHEW/TFHE to this structure.
    2. Research and implement a novel bootstrapping algorithm designed for the NTRU based scheme that aims for benefits like the smaller key sizes.
- *Parameter Optimization and Security Analysis*
    1. Select optimal parameters to balance performance, noise, and security.
    2. Conduct thorough security analysis against known attacks.

6.2. Performance Enhancements

Work on improving the efficiency of the scheme after ensuring its correctness and security.

- *Improve Algorithm Logic*
    1. Since polynomial multiplication is used for most operations, try and implement them using Number Theoretic Transform (NTT) for better performance.
    2. Research ways to optimize the bootstrapping algorithm since this step is the most computationally intensive.

3. Use Barrett reduction to implement modular arithmetic operations to improve efficiency.

- *Ciphertext and Key Size Reduction:*
    1. Find ways to reduce the size of the public, relinearization, and bootstrapping keys generated, as well as the ciphertext to improve storage and computational speed.

7. Conclusion

This report has briefs on the current state and the evolution of homomorphic encryption and mainly focusses on the various FHE schemes and noise management techniques devised by the authors. Section Literature Review summarizes journals on the topic of homomorphic encryption and noise management used by the authors.

Section Problem and Possible Solutions discusses a proposed hybrid FHE scheme. The hybrid NTRU-RLWE scheme stands out with the most potential. It theoretically improves noise management, enhances security, and with its faster bootstrapping, it is especially suitable for multi-party scenarios.

The viability of Fully Homomorphic Encryption (FHE) in practical cryptographic systems hinges significantly on the ability to manage noise accumulation during homomorphic evaluations. A primary limitation of FHE schemes is an uncontrolled growth in noise that renders ciphertexts undecipherable. Experimental results indicate that, while the hybrid approach achieves faster basic operations-primarily due to algorithmic simplifications, it comes at the cost of significant noise accumulation and the absence of essential cryptographic primitives such as proper bootstrapping and key switching. These limitations highlight the complexity of translating theoretical advances into secure, scalable, and efficient implementations suitable for deployment in multi-party environments.

Nevertheless, the hybrid scheme's architecture lays a strong foundation for future work. Addressing the identified shortcomings-such as implementing correct polynomial inversion, robust noise management, and optimized bootstrapping-will be crucial for realizing the full potential of this approach. Further research, as discussed in section 6 will also be essential to bridge the gap between proof-of-concept and production-ready FHE systems.

8. References

[1] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek and N. Aaraj, "Survey on Fully Homomorphic Encryption, Theory, and Applications," *Proceedings of the IEEE,* vol. 110, no. 10, pp. 1572-1609, 2022.

[2] X. Kexin, T. Benjamin Hong Meng, W. Li-Ping, A. Khin Mi Mi and W. Huaxiong, "Multi-key fully homomorphic encryption from NTRU and (R)LWE with faster bootstrapping," *Theoretical Computer Science,* vol. 968, p. 114026, 2023.

[3] W. Liu, L. You, Y. Shao, X. Shen, G. Hu, J. Shi and S. Gao, "From accuracy to approximation: A survey on approximate homomorphic encryption and its applications," *Computer Science Review,* vol. 55, 2025.

[4] L. Bai, L. Bai, Y. Li and Z. Li, "Research on Noise Management Technology for Fully Homomorphic Encryption," *IEEE Access,* vol. 12, pp. 135564-135576, 2024.

[5] Y. Lu, K. Hara and K. Tanaka, "Multikey Verifiable Homomorphic Encryption," *IEEE Access,* vol. 10, pp. 84761-84775, 2022.

[6] S. Behera and J. R. Prathuri, "Design of Novel Hardware Architecture for Fully Homomorphic Encryption Algorithms in FPGA for Real-Time Data in Cloud Computing," *IEEE Access,* vol. 10, pp. 131406-131418, 2022.

[7] A. K. Vangujar, B. Ganesh, A. Umrani and P. Palmieri, "A Novel Approach to E-Voting With Group Identity-Based Identification and Homomorphic Encryption Scheme," *IEEE Access,* vol. 12, pp. 162825-162843, 2024.

[8] Z. Koo, J. -W. Lee, J. -S. No and Y. -S. Kim, "Key Reduction in Multi-Key and Threshold Multi-Key Homomorphic Encryptions by Reusing Error," *IEEE Access,* vol. 11, pp. 50310-50324, 2023.

[9] X. Yang, S. Zheng, T. Zhou, Y. Liu and X. Che, "Optimized relinearization algorithm of the multikey homomorphic encryption scheme," *Tsinghua Science and Technology,* vol. 27, no. 3, pp. 642-652, 2022.

[10] G. Tu, W. Liu, T. Zhou, X. Yang and F. Zhang, "Concise and Efficient Multi-Identity Fully Homomorphic Encryption Scheme," *IEEE Access,* vol. 12, pp. 49640-49652, 2024.

[11] A. Benaissa, B. Retiat, B. Cebere and A. E. Belfedhal, "TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption," *ArXiv,* vol. abs/2104.03152, 2021.