Fully Homomorphic Encryption: A Mathematical Introduction

Sara Logsdon
University of Georgia
sara.logsdon@uga.edu

September 18, 2023

Abstract

This paper offers a mathematical introduction to fully homomorphic encryption, a concept that enables computation on encrypted data. We trace the historical development of FHE, describe Fully Homomorphic Encryption over the Torus (TFHE) and how it performs certain mathematical operations, and explore bootstrapping and the possibility for adjusting computational depth. This paper equips readers with a brief understanding of FHE's evolution and the essential mechanisms facilitating practical implementation.

1 Introduction

Fully homomorphic encryption (FHE) enables computation of arbitrary functions over encrypted data without decrypting the data. That is, given encryptions $E(m_1), ..., E(m_t)$, one can compute $f(m_1, ..., m_t)$, without compromising the privacy of the data.

This means the opportunity for finding the average red blood cell count of cancer patients without compromising the patients' privacy, or inputting private queries to a search engine, or searching for a server to retrieve only files that satisfy some Boolean constraint, without the server having to decrypt any of the files.

A partially homomorphic encryption scheme only works for a family of functions. When the homomorphic encryption scheme only allows a limited amount of leveling of operations, we call it somewhat homomorphic, and when we can choose the computational depth (that is, the amount of levels of operations allowed), we all the scheme leveled fully homomorphic. Finally, when the homomorphic encryption scheme works for all families of functions for unlimited computational depth, we call it fully homomorphic.

Craig Gentry was the first to develop a fully homomorphic encryption scheme. He first published "Fully Homomorphic Encryption Using Ideal Lattices"[1], then published a simpler version allowing FHE over the integers, and finally developed the first scheme which was able to include bootstrapping, a special technique used to allow leveled homomorphic operations.

After this first generation, Brakerski was able to develop an FHE scheme based on the Learning With Errors (LWE) problem, a well-known cryptographic assumption. This paper was also able to increase efficiency through modulus reduction. Later, Gentry used modulus reduction to develop (Leveled) FHE without Bootstrapping. This second generation also included Fully Homomorphic SIMD Operations and FHE with Polylog Overhead.

Since then, researchers have developed FHE schemes which speed up bootstrapping significantly, and in "Homomorphic Encryption for Arithmetic of Approximate Numbers" [2], the CKKS scheme, which operates on floating point numbers, allowing for operation on neural networks, helped efficiency immensely. Chimeric FHE (i.e. TFHE) allows for switching between different FHE schemes through bootstrapping.

2 Fully Homomorphic Encryption over the Torus (TFHE)

A fully homomorphic scheme, TFHE[3] was initially proposed as an improvement of the scheme FHEW[4]. The security is based on the LWE problem. TFHE is special because it proposes a special bootstrapping which is very fast and able to evaluate a function at the same time as it reduces the noise.

2.1 TFHE Ciphertexts

TFHE uses three types of ciphertexts: LWE, Ring LWE (RLWE), and Ring Gentry-Sahai-Waters (RGSW). GLWE (General LWE) refers to both LWE

and RLWE ciphertexts. We will focus solely on GLWE as the other ciphertexts work similarly. In GLWE, the secret key is a list of k random polynomials from $R := \mathbb{Z}[X]/(X^N + 1)$, which contains integer polynomials up to degree $N - 1$:

$$\vec{S} = (S_0, ..., S_k - 1) \in R^k$$

The original TFHE samples secret keys from uniform binary distributions. To encrypt messages, let $0 \leq p, q \in \mathbb{Z}$ such that $p \leq q$ and define $\Delta = q/p$. In TFHE, p and q are often chosen to be powers of two. We call q the ciphertext modulus, p the plaintext modulus, and Δ the scaling factor.

Consider a message $M \in R_p$, where $R_p := (\mathbb{Z}/p\mathbb{Z})[X]/(X^N+1)$. A GLWE ciphertext encrypting the message M under the secret key $\vec{S}$ is a tuple:

$$(A_0, ..., A_k - 1, B) \in GLWE^{\rightarrow}_{S,\sigma}(\Delta M) \subseteq R_q^{k+1},$$

where the elements $A_i, 0 \leq 1 \leq k - 1$ are sampled uniformly random from R_q, $B = \sum_{i=0}^{k=1} A_i \cdot S_i + \Delta M + E \in R_q$, and $E \in R_q$ has coefficients sampled from a Gaussian distribution χ_σ. We call the vector $(A_0, ..., A_k - 1)$ the mask and B the body. The polynomial ΔM is called an encoding of M.

Observe that every time a message is encoded, new noise is sampled. We denote the set of GLWE encrypions of the same encoding ΔM under the secret key $\vec{S}$, with Gaussian noise with standard deviation σ, $GLWE^{\rightarrow}_{S,\sigma}(\Delta M)$.

Now we can decrypt a ciphertext $(A_0, ..., A_k - 1, B) \in GLWE^{\rightarrow}_{S,\sigma}(\Delta M)$ encrypted under the secret key $\vec{S} = (S_0, ..., S_{k-1}) \in R^k$ by computing:

1. $B - \sum_{i=0}^{k=1} A_i \cdot S_i = \Delta M + E \in R_q$
2. $M = (\Delta M + E)/\Delta$.

If every coefficient e_i of E is $|e_i| < \Delta/2$, then the second step of the decryption returns M as expected. Otherwise the decryption is incorrect.

When we instantiate GLWE with $k = n \in \mathbb{Z}$ and $N = 1$ we get LWE. When we instantiate GLWE with $k = 1$ and N a power of 2 we get RLWE. It is also worth noting that this scheme allows for public key encryption too, though we will not go into that in this paper.

2.2 Performing Operations

1. **Homomorphic addition:**
   Consider a message $M \in R_p$ encrypted by a GLWE ciphertext C under

the secret key $\vec{S}$. We can add this to another GLWE ciphertext C' encrypting another message M' under the same secret key, and the result will be a new GLWE ciphertext encrypting the sum $M + M'$ under the secret key $\vec{S}$, with noise that grew a bit:

$$C^{(+)} = C + C' = (A_0 + A'_0, ..., A_{k-1} + A'_{k-1}, B + B')$$
$$\in GLWE^{\rightarrow}_{S,\sigma'}(\Delta(M + M') \subseteq R_q^{k+1}$$

2. **Homomorphic constant addition:**
   We can add a constant $\Sigma \in R_p$ to a GLWE encryption of a message M by homomorphically adding the trivial ciphertext $(0, ..., 0, \Delta\Sigma) \in R_q^{k+1}$. The result is a GLWE encryption of $M + \Sigma$.

3. **Homomorphic multiplication by a small constant:**
   Consider a small constant polynomial

$$\Lambda = \sum_{i=0}^{N-1} \Lambda_i X^i \in R.$$

   We can multiply the polynomial Λ to every comoponent of the GLWE ciphertext in R_q and the result will be a new GLWE ciphertext encrypting the product $\Lambda \doteq M \in R_p$ under the same secret key with noise that grew a bit.

4. **Homomorphic multiplication by a large constant:**
   Consider a large constant $\gamma \in \mathbb{Z}_q$. If we multiply a message by this constant as we did for the small constant, the noise grows too much and compromises the result. Instead, we take the large constant and decompose it into a small base β:

$$\gamma = \gamma_1 \frac{q}{\beta^1} + \gamma_2 \frac{q}{\beta^2} + ... + \gamma_l \frac{q}{\beta^l},$$

   where the decomposed elements $\gamma_1, ..., \gamma_l$ are in $\mathbb{Z}_\beta$, so they are small. Denote $Decomp^{\beta,l}(\gamma) = (\gamma_1, ..., \gamma_l)$.

   We multiply this decomposition by the GLev encryption of M instead of the GLWE encryption of M which, by definition, encrypts M times different powers of the decomposition base:

$$\langle Decomp^{\beta,l}(\gamma), \bar{C} \rangle = \sum_{j=1}^{l} \gamma_j \doteq C_j \in$$

$$GLWE^{\rightarrow}_{S,\sigma}(\frac{q}{\beta^l}M)) = GLev^{\rightarrow\beta,l}_{S,\sigma'}(M) \subseteq R_q^{l(k+1)},$$

where σ' is the new standard deviation of the noise.

5. **Homomorphic multiplication by a large polynomial:**
This operation follows the same pattern. We decompose the polynomial into smaller polynomials, then perform a polynomial inner product with GLev. That is:

$$Decomp^{\beta,l}(\Lambda) = (\Lambda^{(1)}, ..., \Lambda^{(l)}),$$

where $\Lambda^{(j)} = \sum_{i=0}^{N-1} \Lambda_{i,j} \doteq X^i$, with $\Lambda_{i,j} \in \beta$ such that:

$$\Lambda = \Lambda^{(1)}\frac{q}{\beta^1} + \Lambda^{(2)}\frac{q}{\beta^2} + ... + \Lambda^{(l)}\frac{q}{\beta^l}$$

Key Switching is a homomorphic operation largely used in all the RLWE-based schemes, and it is used to switch the secret key to a new one. To switch the key we will cancel the secret key $\vec{S}$ and re-encrypt under a new secret key $\vec{S'}$. We compute

$$B - \sum_{i=0}^{k-1} A_i \doteq S_i = \Delta M + E \in R_q$$

but instead of just using the elements S_i, we will encrypt them under the new secret key $\vec{S'}$. Then the ciphertext will be the GLev encryptions of S_i under $\vec{S'}$. Denote

$$KSK_i \in$$
$$GLWE^{\vec{}}_{S',\sigma}(\frac{q}{\beta^1}S_i) \times ... \times GLWE^{\vec{}}_{S',\sigma}(\frac{q}{\beta^l}S_i)$$
$$= GLev^{\beta,l}_{S',\sigma}(S_i) \subseteq R_q^{l(k+1)}$$

The key switching is performed as follows:

$$C' = (0, ..., 0, B) - \sum_{i=0}^{k-1} \langle Decomp^{\beta,l}(A_i), KSK_i \rangle$$

$$\in GLWE^{\vec{}}_{S',\sigma'}(\Delta M) \subseteq R_q^{k+1}$$

The secret key has switched but the message is the same.

There are also other operations known as external product, which we can describe as like a key switching where we do not switch the key, and internal product, which is similar but less efficient than external product. We will not go in depth into these operations here.

3 Computational Depth

As mentioned before, each operation increases the noise, and when the noise gets too large (i.e. when we do not have that every coefficient e_i of E is $|e_i| < \Delta/2$'), the decryption fails. Craig Gentry proposed a technique called bootstrapping which essentially sends a new key inside the current encrypted message, encrypts the message with the new key, then decrypt the old key. The point is that bootstrapping "resets the noise."

Bootstrapping of TFHE takes as input an LWE ciphertext, a polynomial in R_q, and a bootstrapping key BK. Bootstrapping consists of three steps: modulus switching, blind rotation, and sample extraction.

Modulus switching means switching the ciphertext modulus to a new one. In TFHE, it is mainly used on LWE ciphertexts. Using the same definitions for p, q, and Δ as mentioned previously, we choose another positive integer ω. To switch the modulus from q to ω, let $\tilde{a_i} = \frac{\omega \doteq a_1}{q} \in \mathbb{Z}_\omega$. This results in a new LWE ciphertext $\tilde{c}$. That is, we switch the modulo of the LWE ciphertext c, giving

$$\tilde{c} = (\tilde{a_0}, ..., \tilde{a_{n-1}}, \tilde{a_n} = b) \in \vec{LWE}_{S,\sigma}(\tilde{\Delta} m) \subseteq \mathbb{Z}_\omega^{n+1}$$

Blind rotation is the core of the bootstrapping. We take the polynomial V, encrypted as a trivial GLWE ciphertext , and use the output LWE ciphertext of modulus switching $\tilde{c}$ and the bootstrapping key BK to give a new GLWE encryption of $V \cdot X^{\Delta m+e}$ under a new GLWE secret key $\vec{S'}$.

Sample extraction then takes as input a GLWE ciphertext, encrypting a polynomial message, and extracts the encryption of one of the coefficients of the message as a LWE ciphertext. This operation does not increase the noise. That is, we take the constant coefficient of GLWE output of the blind rotation as a LWE encryption of f_m, under the extracted LWE secret key $\vec{s'}$ [5].

Bootstrapping is important because it allows for the possibility of leveled FHE, a special type of FHE which allows the user to choose a parameter

for computational depth, the maximal number of consecutive operations for which an homomorphic encryption scheme remains functional. This could be useful because bootstrapping is extremely expensive [6], and we do not always need infinite computational depth.

4 Conclusion

The ability to perform arbitrary computations on encrypted data without the need for decryption opens up a wide range of possibilities for enhancing data privacy and security in an increasingly interconnected and data-driven world. Its potential impact on industries such as healthcare, finance, and cloud computing is profound, enabling organizations to confidently leverage the power of the cloud while preserving the confidentiality of their sensitive information.

However, it is important to acknowledge that FHE is not without its limitations. Its computational overhead poses significant challenges that must be addressed to make it practical for everyday use. Ongoing research and innovation in the field are actively working to overcome these obstacles.

5 References

[1] Craig Gentry. "Fully Homomorphic Encryption Using Ideal Lattices". In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC '09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. URL: https://doi.org/10.1145/1536414.1536440.

[2] Jung Cheon et al. "Homomorphic Encryption for Arithmetic of Approximate Numbers". In: Nov. 2017, pp. 409–437. ISBN: 978-3-319-70693-1. DOI: 10.1007/978-3-319-70694-8_15.

[3] Ilaria Chillotti et al. "TFHE: Fast Fully Homomorphic Encryption Over the Torus". In: *Journal of Cryptology* 33.1 (Apr. 2019), pp. 34–91. DOI: 10.1007/s00145-019-09319-x. URL: https://doi.org/10.1007/s00145-019-09319-x.

[4] Léo Ducas and Daniele Micciancio. *FHEW: Bootstrapping Homomorphic Encryption in less than a second*. Cryptology ePrint Archive, Paper 2014/816. https://eprint.iacr.org/2014/816. 2014. URL: https://eprint.iacr.org/2014/816.

[5] *TFHE Deep Dive - Part I - Ciphertext types — zama.ai*. https://www.zama.ai/post/tfhe-deep-dive-part-1. [Accessed 28-Apr-2023].

[6] Ahmad Al Badawi and Yuriy Polyakov. *Demystifying Bootstrapping in Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2023/149. `https://eprint.iacr.org/2023/149`. 2023. URL: `https://eprint.iacr.org/2023/149`.