Fully Homomorphic Encryption: A Mathematical Introduction

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

September 18, 2023

Abstract

Fully homomorphic encryption (FHE) allows a party to evaluate functions on encrypted data without first decrypting the inputs. This paper provides a mathematical introduction to FHE, focusing on fully homomorphic encryption over the torus (TFHE). After a brief account of the development of FHE, we describe the algebra and encoding of learning-with-errors (LWE) and generalized LWE (GLWE) ciphertexts, derive elementary homomorphic operations, and explain how gadget decomposition supports key switching and external products. We then examine the growth of encryption noise and the role of bootstrapping, with particular attention to modulus switching, blind rotation, sample extraction, and programmable lookup tables. The objective is to make these established mechanisms precise and accessible while clarifying the distinction between leveled evaluation and bootstrapped FHE.

Keywords: fully homomorphic encryption; TFHE; learning with errors; GLWE; bootstrapping; computational depth.

1 Introduction

Fully homomorphic encryption (FHE) permits computation on encrypted information. Given encryptions $\mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_t)$ and an efficiently computable function f, an evaluator can obtain a ciphertext that decrypts to $f(m_1, \ldots, m_t)$ without needing the corresponding secret key. Subject to the security assumptions and the surrounding protocol, this makes it possible to outsource computation while keeping input data confidential. For example, a data analyst could compute aggregate statistics on encrypted medical records, or a server could evaluate a query on protected data without learning the query inputs. Encryption alone does not prevent information from leaking through released outputs or auxiliary information.

There are several related notions of homomorphic encryption. A *partially homomorphic* scheme supports a restricted class of operations, such as addition alone. A *somewhat homomorphic* scheme supports more general operations but only to a limited complexity before noise makes decryption unreliable. A *leveled* FHE scheme can be parameterized to evaluate circuits up to a chosen depth, which may be large, without requiring bootstrapping. In a bootstrapped FHE scheme, one can repeatedly refresh ciphertexts to support circuits of arbitrary polynomial depth, with appropriate parameter choices. These categories describe evaluation capabilities, not interchangeable implementations.

Gentry's 2009 construction established the first FHE scheme using ideal lattices and introduced bootstrapping as a way to extend limited-depth evaluation [1]. A construction over the integers followed [2]. Later lattice-based developments improved the management of ciphertext

noise, including the use of modulus switching in leveled schemes [4] and the Gentry–Sahai–Waters (GSW) approach to homomorphic operations [5]. FHEW reduced the cost of bootstrapping [6]; TFHE subsequently developed efficient gate bootstrapping and external-product techniques [7, 8]. Other schemes, notably CKKS, focus on approximate arithmetic [9]. Work on cross-scheme conversion, including CHIMERA, is related but distinct from TFHE itself [10].

The remainder of the paper describes TFHE's ciphertext representation and elementary operations, then explains why noise limits computation and how bootstrapping addresses that limitation. The presentation emphasizes the algebraic identities that make these mechanisms work. For clarity, we use finite-modulus polynomial rings to model the ciphertext arithmetic; TFHE's underlying torus viewpoint is introduced below.

2 Fully Homomorphic Encryption over the Torus (TFHE)

2.1 TFHE Ciphertexts

TFHE is formulated over the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, where messages are represented by phases modulo 1. For an elementary treatment, we work with the finite ring $\mathbb{Z}_q$ and view a residue $x \in \mathbb{Z}_q$ as the discretized torus element $x/q \pmod 1$. This model captures the modular phase identities used below, though real implementations may differ in representation, scaling, and parameter choices.

Let $N \geq 1$ be a power of two and let $p, q \geq 2$ be integers with $p \mid q$. Define

$$\mathcal{R} = \mathbb{Z}[X]/(X^N + 1), \qquad \mathcal{R}_q = (\mathbb{Z}/q\mathbb{Z})[X]/(X^N + 1), \qquad \mathcal{R}_p = (\mathbb{Z}/p\mathbb{Z})[X]/(X^N + 1), \quad (1)$$

and let $\Delta = q/p$ be the message scale. Ring multiplication is *negacyclic*: $X^N = -1$. A polynomial in $\mathcal{R}_q$ is represented by its N coefficient classes modulo q. When discussing error magnitudes, we take the centered integer representative of each coefficient.

Let $k \geq 1$. A GLWE secret key is a vector $\mathbf{s} = (s_0, \ldots, s_{k-1}) \in \mathcal{R}^k$ whose coefficients are sampled from a prescribed small secret distribution; binary coefficients are a common illustrative choice. To encrypt $M \in \mathcal{R}_p$, sample independently uniform masks $A_0, \ldots, A_{k-1} \in \mathcal{R}_q$ and a small error polynomial $E \in \mathcal{R}$ (reduced modulo q), and set

$$B = \sum_{i=0}^{k-1} A_i s_i + \Delta M + E \pmod q, \qquad C = (A_0, \ldots, A_{k-1}, B) \in \mathcal{R}_q^{k+1}. \quad (2)$$

Here a representative of M is chosen before multiplication by Δ; changing the representative changes ΔM by a multiple of q and hence does not change the ciphertext. The *phase* of a ciphertext relative to $\mathbf{s}$ is

$$\mathrm{ph}_{\mathbf{s}}(C) = B - \sum_{i=0}^{k-1} A_i s_i = \Delta M + E \pmod q. \quad (3)$$

The first k entries of C form the *mask*; B is its *body*. We write $\|E\|_\infty$ for the maximum absolute value of its centered coefficients, when those coefficients can be unambiguously interpreted as small integers.

Proposition 1 (Sufficient decryption condition)**.** *If each coefficient e_j of E has an integer lift satisfying $|e_j| < \Delta/2$, then coefficientwise nearest-integer rounding of $\mathrm{ph}_{\mathbf{s}}(C)/\Delta$, followed by reduction modulo p, recovers M.*

Proof. For a coefficient $m_j \in \mathbb{Z}/p\mathbb{Z}$, choose a representative $\widehat{m}_j \in \mathbb{Z}$. The corresponding phase coefficient is congruent to $\Delta\widehat{m}_j + e_j$ modulo $q = \Delta p$. Any integer lift of that phase differs from $\Delta\widehat{m}_j + e_j$ by a multiple of Δp. Since $|e_j/\Delta| < 1/2$, nearest-integer rounding yields $\widehat{m}_j$ plus a multiple of p, which is m_j modulo p. □

The condition is sufficient, not a security statement: it does not say how likely the error distribution is to satisfy the bound. The failure probability must be analyzed for the chosen error distribution and parameters. Similarly, encryption correctness alone gives no security guarantee.

2.1.1 LWE and ring-LWE special cases

When $N = 1$, the relation $X + 1 = 0$ identifies $\mathcal{R}_q$ with $\mathbb{Z}_q$. Choosing $k = n$ gives the familiar LWE ciphertext

$$c = (a_0, \ldots, a_{n-1}, b) \in \mathbb{Z}_q^{n+1}, \quad b = \sum_{i=0}^{n-1} a_i s_i + \Delta m + e \pmod{q}. \tag{4}$$

For $k = 1$ and a larger polynomial degree N, the same notation specializes to a ring-LWE-type ciphertext. The umbrella term GLWE conveniently accommodates both settings, but their concrete security assumptions must not be conflated.

2.2 Performing Operations

2.2.1 Homomorphic addition and public constants

Let C and C' be ciphertexts encrypted under the same key and with the same modulus and scale. Since the phase map is linear,

$$\mathrm{ph}_{\mathbf{s}}(C + C') = \Delta(M + M') + (E + E') \pmod{q}. \tag{5}$$

The message sum is interpreted in $\mathcal{R}_p$. A known $T \in \mathcal{R}_p$ can be added by including the trivial ciphertext $(0, \ldots, 0, \Delta T)$; this introduces no additional encryption error. More generally, a linear combination of ciphertexts with public integer coefficients gives the corresponding linear combination of messages and errors.

If independent coefficient errors have respective variances σ_1^2 and σ_2^2 and no wraparound occurs, their sum has variance $\sigma_1^2 + \sigma_2^2$. This probabilistic observation requires independence. The deterministic inequality $\|E + E'\|_\infty \le \|E\|_\infty + \|E'\|_\infty$ needs no independence assumption.

2.2.2 Multiplication by public scalars and polynomials

For a public $P \in \mathcal{R}$, multiply each component of C by P in $\mathcal{R}_q$. This gives

$$\mathrm{ph}_{\mathbf{s}}(PC) = P\ \mathrm{ph}_{\mathbf{s}}(C) = \Delta(PM) + PE \pmod{q}. \tag{6}$$

Define $\|P\|_1 = \sum_{j=0}^{N-1} |p_j|$ for the integer coefficients of P.

Proposition 2 (Negacyclic noise bound)**.** *For polynomials $P, E \in \mathcal{R}$ with the usual integer coefficient representatives, the coefficients of the negacyclic product satisfy*

$$\|PE\|_\infty \le \|P\|_1 \, \|E\|_\infty. \tag{7}$$

Consequently, if $\|P\|_1 \, \|E\|_\infty < \Delta/2$, then PC decrypts to $PM \in \mathcal{R}_p$.

Proof. Each coefficient of PE in $\mathbb{Z}[X]/(X^N + 1)$ is a sum of terms of the form $\pm p_j e_t$; for fixed output index, each p_j occurs once. The triangle inequality bounds its absolute value by $\sum_j |p_j| \|E\|_\infty$. The decryption statement follows from proposition 1. □

This inequality explains why multiplication by a public polynomial with large coefficients may amplify noise. It does *not* implement multiplication of two independently encrypted messages. Such operations require more involved evaluation keys or representations, including GGSW ciphertexts in the TFHE family.

2.3 Gadget Decomposition and Key Switching

2.3.1 Decomposition into small digits

A convenient way to understand gadget methods is to assume $q = \beta^\ell$ for an integer base $\beta \geq 2$. Define gadget weights $g_j = \beta^{\ell-j}$ for $1 \leq j \leq \ell$. For any $A \in \mathcal{R}_q$, apply base-β expansion coefficientwise to its representatives in $\{0, \ldots, q-1\}$, obtaining $D_j(A) \in \mathcal{R}$ with coefficients in $\{0, \ldots, \beta-1\}$ such that

$$A = \sum_{j=1}^{\ell} g_j D_j(A) \pmod{q}. \tag{8}$$

Other implementations may use signed digits, approximate decomposition, or different gadget scales; these require corresponding rounding terms. We use the exact expansion to expose the algebra without hiding an approximation.

A gadget-level encryption (often called a GLev ciphertext) of a ring element U under a key $\mathbf{s}'$ can be viewed as a collection of GLWE ciphertexts whose phases are $g_1U, \ldots, g_\ell U$ (plus their respective errors). More elaborate GGSW ciphertexts organize gadget encryptions in a matrix so that a *ciphertext–ciphertext external product* can be computed [5, 8]. The distinction matters: scalar or polynomial multiplication of a ciphertext by a public object is not the same operation as an external product with an encrypted selector.

2.3.2 Key switching

Suppose $C = (A_0, \ldots, A_{k-1}, B)$ is encrypted under $\mathbf{s}$ and we want a ciphertext under a possibly different GLWE key $\mathbf{s}'$. Publish a key-switching key consisting of ciphertexts

$$K_{i,j} \text{ with } \mathrm{ph}_{\mathbf{s}'}(K_{i,j}) = g_j s_i + E_{i,j}, \quad 0 \leq i < k,\ 1 \leq j \leq \ell. \tag{9}$$

These are encryptions of *raw gadget-scaled phases*; they need not be messages in the original p-ary encoding. Let $\iota(B) = (0, \ldots, 0, B)$ denote the trivial GLWE ciphertext under $\mathbf{s}'$. Set

$$C' = \iota(B) - \sum_{i=0}^{k-1} \sum_{j=1}^{\ell} D_j(A_i) K_{i,j}. \tag{10}$$

All additions and products in equation (10) take place componentwise over $\mathcal{R}_q$, with each $D_j(A_i)$ public once C is known.

Proposition 3 (Key-switching identity)**.** *Under the exact decomposition in equation* (8)*, the output defined by equation* (10) *satisfies*

$$\mathrm{ph}_{\mathbf{s}'}(C') = \Delta M + E - \sum_{i=0}^{k-1} \sum_{j=1}^{\ell} D_j(A_i) E_{i,j} \pmod{q}. \tag{11}$$

Thus the message phase is preserved while new error is introduced by the key-switching key.

Proof. Apply the linear phase map under $\mathbf{s}'$ to equation (10) and substitute equation (9). The gadget terms combine as $\sum_j D_j(A_i) g_j s_i = A_i s_i \pmod{q}$ by equation (8). The remaining term $B - \sum_i A_i s_i$ equals $\Delta M + E$ by equation (3). The uncanceled terms are precisely the evaluation-key errors in equation (11). □

For example, proposition 2 immediately gives the conservative bound

$$\|E'\|_\infty \leq \|E\|_\infty + \sum_{i,j} \|D_j(A_i)\|_1 \, \|E_{i,j}\|_\infty, \tag{12}$$

provided the errors admit the indicated centered lifts. Since $\|D_j(A_i)\|_1 \leq N(\beta-1)$, this estimate makes explicit the tradeoff among base, number of levels, polynomial degree, and new error. It is a loose worst-case bound; useful parameter analyses typically exploit error distributions and additional structure.

2.3.3 External products and controlled selection

A GGSW ciphertext of a bit $s \in \{0,1\}$, written G_s, provides the extra gadget structure needed for an operation we denote by $G_s \boxdot C$. At the level of ideal message semantics, this *external product* encrypts sM when C encrypts M. Its implementation decomposes ciphertext components and combines them with the GGSW gadget matrix. It incurs additional noise; it is not ordinary multiplication of ciphertext vectors.

This leads to a controlled multiplexer,

$$\mathsf{CMux}(G_s, C_0, C_1) = C_0 + G_s \boxdot (C_1 - C_0). \tag{13}$$

Ignoring evaluation noise, equation (13) encrypts M_0 if $s = 0$ and M_1 if $s = 1$. The encrypted selector is useful precisely because the evaluator can perform this choice without knowing s. Full GGSW matrix dimensions and exact external-product noise formulas can be found in the TFHE construction [8].

3 Computational Depth

3.1 Noise Growth and Leveled Evaluation

As the preceding calculations show, homomorphic operations transform the error as well as the message. Repeated additions increase the accumulated error, and certain products or evaluation-key operations can increase it substantially. When the phase error is too large relative to the encoding scale, coefficientwise rounding need not recover the intended plaintext. In a leveled FHE construction, parameters are selected to accommodate a specified circuit depth. Bootstrapping provides a different means of handling long computations: it refreshes a ciphertext so that subsequent operations may proceed. This distinction is important because leveled FHE does not, by definition, require bootstrapping [4].

3.2 Bootstrapping in TFHE

3.2.1 Purpose of bootstrapping

An LWE ciphertext $c = (\mathbf{a}, b)$ has phase $\phi = b - \langle \mathbf{a}, \mathbf{s} \rangle = \Delta m + e \pmod q$. The conceptual goal of bootstrapping is to evaluate a decryption-related lookup table on the *encrypted* phase. The evaluator receives suitable encryptions of secret-key material in a bootstrapping key, but not the plaintext secret key. When the process succeeds, the result is a new ciphertext of a chosen function $f(m)$ whose error is governed primarily by the bootstrapping computation, rather than by the accumulated incoming error [1, 8].

We describe the standard three-stage view: modulus switching, blind rotation, and sample extraction. A further key switch may be necessary to return the output to the secret key expected by the next evaluation stage.

3.2.2 Modulus switching to $2N$

Let $\omega = 2N$, the order of the monomial X in $\mathcal{R}_q$ (since $X^N = -1$ and $X^{2N} = 1$). For a coefficient $x \in \mathbb{Z}_q$, take a fixed integer representative and define

$$\widetilde{x} = \left\lfloor \frac{\omega x}{q} \right\rceil \pmod \omega. \tag{14}$$

Apply equation (14) to each mask entry *and to the body* of c. The resulting ciphertext is $\widetilde{c} = (\widetilde{\mathbf{a}}, \widetilde{b}) \in \mathbb{Z}_\omega^{n+1}$. With binary secret coefficients, each scalar rounding discrepancy has magnitude

at most $1/2$. For consistent integer lifts of phases, the switched phase obeys

$$\widetilde{\phi} = \frac{\omega\Delta}{q}m + \varepsilon \pmod{\omega}, \qquad |\varepsilon| \le \frac{\omega}{q}|e| + \frac{n+1}{2}. \tag{15}$$

The error bound is conservative: it follows by writing $\widetilde{b} = (\omega/q)b + \delta_b$ and $\widetilde{a}_i = (\omega/q)a_i + \delta_i$ with $|\delta_i|, |\delta_b| \le 1/2$, then using $s_i \in \{0, 1\}$. In particular, modulus switching is *not* noise-free. The lookup-table regions must tolerate both the scaled input error and the new rounding error.

3.2.3 Blind rotation

Let $V = \sum_{j=0}^{N-1} v_j X^j \in \mathcal{R}_q$ encode a lookup table and let $\mathsf{BK} = (G_{s_0}, \ldots, G_{s_{n-1}})$ consist of GGSW encryptions of the original binary LWE secret coefficients under a GLWE key $\mathbf{s}'$. Begin with the trivial GLWE encryption

$$C_{\mathrm{rot}} \longleftarrow \iota\left(X^{-\widetilde{b}}V\right). \tag{16}$$

For $i = 0, \ldots, n-1$, update

$$C_{\mathrm{rot}} \longleftarrow \mathsf{CMux}(G_{s_i}, C_{\mathrm{rot}}, X^{\widetilde{a}_i}C_{\mathrm{rot}}). \tag{17}$$

By equation (13), the ith update ideally rotates by $X^{\widetilde{a}_i s_i}$. Induction therefore gives the ideal output message

$$VX^{-\widetilde{b}+\sum_i \widetilde{a}_i s_i} = VX^{-\widetilde{\phi}} \quad \text{in } \mathcal{R}_q. \tag{18}$$

The rotation is called *blind* because the evaluator performs the secret-dependent sequence using BK rather than learning the bits s_i. Real executions add external-product error at every CMux; equation (18) describes the intended plaintext semantics, not an error-free implementation.

3.2.4 Programmable lookup tables

For $0 \le t < N$, the constant coefficient of $X^{-t}V$ is v_t. For $N \le t < 2N$, it is $-v_{t-N}$. Thus a single negacyclic lookup polynomial represents a function $F : \mathbb{Z}_{2N} \to \mathbb{Z}_q$ satisfying

$$F(t+N) = -F(t) \pmod{q}. \tag{19}$$

Arbitrary functions on unrestricted $\mathbb{Z}_{2N}$ do *not* satisfy equation (19); they cannot all be realized with one such lookup table. TFHE implementations handle broader functionality by suitable message encodings, restricted input domains (including padding), or more elaborate procedures [8, 11]. The table's coefficients must also be constant across the rounding-error neighborhood assigned to each encoded message if the output is to be reliable.

3.2.5 Sample extraction and output key switching

Suppose blind rotation yields a GLWE ciphertext $C = (A_0, \ldots, A_{k-1}, B)$ under the GLWE key $\mathbf{s}'$. Write $A_i = \sum_{j=0}^{N-1} a_{i,j}X^j$ and $s'_i = \sum_{j=0}^{N-1} s'_{i,j}X^j$. The constant coefficient of the product $A_i s'_i$ is

$$\mathrm{ct}(A_i s'_i) = a_{i,0}s'_{i,0} - \sum_{j=1}^{N-1} a_{i,N-j}s'_{i,j}. \tag{20}$$

Consequently, use the flattened LWE key formed by the kN coefficients $s'_{i,j}$ and the LWE mask whose entries are $a_{i,0}$ for $j = 0$ and $-a_{i,N-j}$ for $j \ge 1$. The body is the constant coefficient b_0 of B. By equation (20), the extracted LWE phase is exactly $\mathrm{ct}(\mathrm{ph}_{\mathbf{s}'}(C))$. This is a deterministic

rewriting of the same phase: *sample extraction introduces no new encryption error*, although it carries forward the error already in the GLWE output.

With a suitably constructed table, the extracted phase is near an encoding of $f(m)$. Optionally, LWE key switching returns the ciphertext to the key expected by the next evaluation stage. This completes programmable bootstrapping.

3.3 A Small Arithmetic Example

The following parameters are far too small for cryptographic security. This example illustrates the ciphertext phase and lookup-table sign change, not a full GGSW-based bootstrapping execution.

Set $q = 256$, $p = 4$, $\Delta = 64$, $n = 2$, $N = 8$, $\omega = 2N = 16$, binary LWE key $\mathbf{s} = (1, 0)$, and mask $\mathbf{a} = (17, 92)$. In both trials choose error $e = 3$. The ciphertext body for message $m \in \mathbb{Z}_4$ is

$$b = 17 + 64m + 3 \pmod{256}, \qquad \phi = b - 17 = 64m + 3 \pmod{256}.$$

The coefficients modulus-switch as $\widetilde{\mathbf{a}} = (\lfloor 17/16 \rceil, \lfloor 92/16 \rceil) = (1, 6) \pmod{16}$. The following table shows two messages and the corresponding switched phases.

m	b	ϕ	$\widetilde{b}$	$\widetilde{\phi} = \widetilde{b} - 1$	$\mathsf{Dec}(c)$
1	84	67	5	4	1
3	212	195	13	12	3

Both original phases decrypt correctly because $|3| < \Delta/2 = 32$. To see a simple signed lookup in $\mathcal{R}_{256} = \mathbb{Z}_{256}[X]/(X^8 + 1)$, let

$$V = \mu \sum_{j=0}^{7} X^j, \tag{21}$$

where μ is a public encoded output amplitude. For example, $\mu = 64$ gives the conventional encoding of 1 in $\mathbb{Z}_4$ at scale $\Delta = 64$. Then the constant coefficient of $X^{-4}V$ is $+\mu$, while that of $X^{-12}V$ is $-\mu$, because $X^8 = -1$. Thus the two switched phases select opposite signed outputs. For $\mu = 64$, these outputs encode 1 and 3 modulo 4, respectively. The table is intended to illustrate the negacyclic sign relation of equation (19); it is not an identity lookup table on the entire message space.

3.4 Security and Practical Limitations

Security assumptions. The pseudorandomness of fresh LWE samples is studied under LWE-type assumptions [3]; structured GLWE instantiations invoke related module/ring assumptions. A complete security statement for TFHE also needs the precise secret distribution, error law, dimensions, modulus, and treatment of public evaluation keys. In particular, publishing encryptions derived from secret-key material raises subtleties that are not resolved by the elementary phase identities in this paper. We do not infer any bit-security figure from the toy examples.

Correctness and parameter selection. Refreshing reduces accumulated error only for parameter sets where the output noise and relevant lookup-table margins are controlled. Errors arise from fresh encryption, homomorphic linear operations, gadget decomposition (possibly with approximation), modulus switching, external products, and key switching. Production analyses must bound aggregate failure probability, account for parameter-dependent error correlations, and choose both an input domain and a lookup encoding that tolerate phase uncertainty.

Cost and data representation. TFHE is particularly suited to Boolean gates and small-table functions, but its gate-by-gate evaluation can remain costly relative to ordinary plaintext arithmetic. Other FHE families use batching and approximate arithmetic to address different workloads [9]. Runtime figures depend on hardware, parameters, implementation, and the operation being measured, so historical timings should not be presented as contemporary benchmarks.

Operational privacy. Homomorphic encryption protects encrypted inputs from an evaluator under its specified threat model; it does not automatically protect private outputs, metadata, access patterns, or information inferable from released results. Application-specific protections are required for those properties.

4 Conclusion

Fully homomorphic encryption provides a way to evaluate functions on ciphertexts while preserving the confidentiality of their inputs under the scheme's security assumptions. TFHE illustrates how a comparatively simple noisy phase representation can support more sophisticated computation. Homomorphic addition and multiplication by public values follow from the linear structure of the ciphertext phase; gadget decomposition and encrypted evaluation keys enable key switching and external products; and bootstrapping uses blind rotation and sample extraction to produce a refreshed encryption of a selected lookup-table value.

The mathematical details are essential to understanding the limits of the construction. Noise growth constrains decryption correctness, modulus switching introduces rounding error, and the negacyclic ring places conditions on the lookup tables that can be evaluated directly. Bootstrapping makes prolonged computation possible, but at a computational cost that remains important in practice. These tradeoffs explain both the continuing interest in TFHE and the need to choose schemes and parameters with the intended application in mind.

References

[1] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of STOC 2009*, pp. 169–178. `https://doi.org/10.1145/1536414.1536440`.

[2] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology—EUROCRYPT 2010*, pp. 24–43.

[3] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of STOC*, 2005, pp. 84–93.

[4] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of ITCS*, 2012, pp. 309–325.

[5] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology—CRYPTO 2013*, pp. 75–92.

[6] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology—EUROCRYPT 2015*, pp. 617–640.

[7] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology—ASIACRYPT 2016*, pp. 3–33.

[8] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020. https://doi.org/10.1007/s00145-019-09319-x. Author preprint: https://eprint.iacr.org/2018/421.

[9] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology—ASIACRYPT 2017*, pp. 409–437.

[10] C. Boura, N. Gama, M. Georgieva, and D. Jetchev. *CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes*. IACR Cryptology ePrint Archive, Paper 2018/758. https://eprint.iacr.org/2018/758.

[11] I. Chillotti. TFHE Deep Dive, Parts I–IV. Zama, 2022. In particular, "Programmable Bootstrapping," https://www.zama.org/post/tfhe-deep-dive-part-4.