What are the financial costs associated with the use of Homomorphic Encryption for privacy-preserving data analytics?

Iván Moreno Sarriés
Supervisor(s): Dr.Zeki Erkin, Dr.Roland Kromes
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements for the Bachelor of Computer Science and Engineering
An electronic version of this thesis is available at http://repository.tudelft.nl/.

Name of the student: Iván Moreno Sarriés
Final project course: CSE3000 Research Project
Thesis committee: Dr.Zeki Erkin, Dr.Roland Kromes, Dr.Xucong Zhang

Abstract

Homomorphic Encryption (HE) enables computation directly on encrypted data, while offering strong cryptographic and privacy guarantees for data-driven sectors like healthcare, finance, and cloud computing. However, practical adoption of HE is severely limited by its computational overhead and specialized expertise requirements.

This thesis investigates the financial feasibility of HE by analyzing its Total Cost of Ownership (TCO), an overlooked but crucial factor for researchers, companies, and cloud service customers deciding whether to adopt HE. Building on performance benchmarks from Guillot et al. and Lo et al., we evaluate the runtime, resource requirements, and personnel costs of Fully Homomorphic Encryption (FHE) across different workloads, and Amazon Web Service (AWS) cloud configurations.

Our results show that encryption and expert labor dominate the cost, with encryption accounting for over 80% of the runtime and personnel expertise exceeding 98% of the total annual cost ($125,558.066). These findings highlight that, despite the security benefits of HE, its financial overhead restricts its adoption to organizations with substantial cryptographic expertise and budgets.

This thesis equips researchers and companies with data-driven insight into HE's true economic impact and underlines the need for further optimization and automation to make HE accessible and cost-effective for broader, real-world use.

1 Introduction

In the digital age, data is one of the most valuable assets for both public and private organizations. Healthcare [1], finance [2], and government agencies [3] increasingly rely on data analytics to improve decisions and optimize operations. However, this reliance on data raises serious privacy and security risks [4], especially when analyses involve personal or sensitive data and are processed in cloud environments or by third-party service providers [5].

To address these privacy risks, a variety of privacy-preserving technologies have been developed [6]. Among them, Homomorphic Encryption (HE) has emerged as one of the most promising. Homomorphic encryption allows computations to be performed directly on encrypted data, without the need to decrypt it first. The most advanced form of this technique is fully Homomorphic Encryption (FHE), introduced by Craig Gentry in 2009 [7], it supports an arbitrary number of computations on ciphertexts and thus enables complete data analysis workflows to be conducted securely. This capability is particularly appealing in privacy-sensitive contexts, for example, collaborative research, outsourced computing, and multi-party computation.

Despite its theoretical appeal, HE has seen limited adoption outside of research prototypes. The main barriers are its significant computational overhead [8], and critically, its significant financial burden. Most of the literature has focused on algorithmic advances and security properties, not on the practical financial implications of HE deployment. Without transparent estimates of its Total Cost of Ownership (TCO), companies and research teams cannot make informed decisions.

This thesis addresses that gap by investigating the following central research question:

What are the financial costs associated with the use of Homomorphic Encryption for privacy-preserving data analytics?

We explore the financial viability of HE in realistic cloud environments, using publicly available cloud pricing and salary data to model total expenses. By following existing benchmarks on HE schemes (Guillot et al. [9] and Lo et al. [10]), we evaluate encryption parameters, execution times, and hardware choices in order to quantify the real-world costs and scalability of FHE-based analytics. The findings will help researchers, businesses, and service providers understand not only HE technical requirements but also its economic impact, a critical step towards making privacy-preserving computation practical at scale.

Our contribution: First, we reduce the time and effort required to understand Homomorphic Encryption by providing a clear and detailed overview of its core concepts and schemes. Second, we extend the work of Guillot et al. [9] by presenting a more granular and transparent Total Cost of Ownership (TCO) analysis, thereby offering a deeper, practical insight into the real-world financial implications of deploying HE.

The paper is organized as follows: 1.Introduction, 2.Background and Technical Overview, 3.Performance Evaluation, 4.Cost Evaluation, 5.Conclusions and Future Work and 6.Responsible Research, after which are the references.

2 Background and Technical Overview

This section provides the theoretical and technical foundation for understanding Homomorphic Encryption (HE). We review its historical evolution, encryption structure, and operational mechanics, highlighting how HE has progressed from an abstract concept to a practical tool for privacy-preserving data analytics.

2.1 History of Homomorphic Encryption

The concept of *homomorphism in cryptography* was first introduced by Rivest, Adleman, and Dertouzos in 1978 [11] as a theoretical approach to the problem of computing on encrypted data without decryption.

To support *arbitrary computation* over encrypted data, an encryption scheme must allow at least *addition and multiplication*, as these operations form a *functionally complete set* over finite sets. Specifically, any possible function that combines two binary variables can be constructed using only `XOR` (addition), `AND` (multiplication), and `NOT` (negation) gates [12]. HE schemes may be either *symmetric*, using the same key for encryption and decryption, or *asymmetric*, using distinct keys. A general method to convert between symmetric and asymmetric HE schemes was presented by Rothblum in 2011 [13].

An encryption scheme is said to be *homomorphic over an operation* $\star$ if it satisfies the property: $E(m_1) \star E(m_2) = E(m_1 \star m_2),\ \forall m_1, m_2 \in M$ (1), where E is the encryption algorithm and M is the is the set of all possible messages.

Over the past decades, homomorphic encryption has evolved through three major generations, each addressing limitations of the previous:

- **First-Generation of HE(Partially Homomorphic Encryption):** Early HE schemes, such as those based on the RSA cryptosystems [14], RAD cryptosystems [11] and Goldwasser-Micali (GM) cryptosystems [15], supported only a single operation type per circuit (addition or multiplication). These schemes suffered from uncontrolled noise growth with each operation, leading to computational intractability. Decryption

became incorrect when noise exceeded a certain threshold. Security relied on hard lattice problems like the Closest Vector Problem (CVP) and Shortest Vector Problem (SVP).

- **Second-Generation of HE(Somewhat Homomorphic Encryption) [16]:** Was characterized by much better techniques for controlling the noise, enabling a limited number of both additive and multiplicative operations simultaneously, while at the same time basing security on well-established hardness assumptions. These techniques were accompanied by methods for improving the plaintext to ciphertext expansion ratio, further improving efficiency.

- **Third-Generation of HE (Fully Homomorphic Encryption):** Enabled unlimited computation on encrypted data through a breakthrough technique called *bootstrapping*, introduced by Craig Gentry in 2009 [7]. This generation further optimized bootstrapping and introduced support for approximate arithmetic (e.g., CKKS scheme), making HE more practical for real-world applications [17] like machine learning on encrypted data .

2.2 Structure of a Homomorphic Encryption Scheme

A general homomorphic encryption scheme E is defined by the following tuple of algorithms [18]:

- **Setup:** Initializes global parameters. **Key Generation (KeyGen):** Generates a secret and public key pair for the asymmetric version of HE or a single key for the symmetric version. **Encryption (Enc):** Encrypts a plaintext into a ciphertext. **Decryption (Dec):** Decrypts a ciphertext back to plaintext.

- **Evaluation (Eval) [19]:** Evaluation is an HE-specific operation, which takes ciphertexts as input and outputs a ciphertext corresponding to a functioned plaintext. The most crucial point in homomorphic encryption is that the format of the ciphertexts after an evaluation process must be preserved in order to be decrypted correctly. In addition, the size of the ciphertext should also be constant to support an unlimited number of operations, otherwise, the increase in the ciphertext size will require more resources and this will limit the number of operations.

2.3 Homomorphic Encryption Overview

The term homomorphism is derived from the Greek word homos that means same and morphe meaning shape, or same structure [20]. Homomorphic encryption (HE) enables one to perform operations on encrypted data without decrypting them, and the result can be decrypted only with the secret key. As HE reveals nothing about the input or output except their sizes during computation, it has been spotlighted as a core technology for applications such as privacy-preserving computation [21].

2.3.1 Partially Homomorphic Encryption(PHE):

Supports only one type of operation per schema(addition or multiplication).

- Additive PHE supports circuits composed of XOR gates.

- Multiplicative PHE supports circuits composed of AND gates.

Many schemes have shaped the evolution of partially homomorphic encryption (PHE): Rivest et al. (1978) [14] introduced RSA, the first feasible achievement of the public key cryptosystem; Goldwasser and Micali (GM) (1982) [15] proposed the first probabilistic public-key encryption scheme; El-Gamal (1985) [22] proposed a new public key encryption scheme, which is the improved version of the original Diffie-Hellman Key Exchange [23]; Benaloh (1994) [24] proposed an extension of the GM cryptosystem by improving it to encrypt the message as a block instead of bit by bit; and Paillier (1999) [25] introduced a novel probabilistic encryption scheme based on the composite residuosity problem. Each of these contributions improved PHE in its own way, however, this section focuses primarily on the RSA scheme [14], a major PHE scheme that acts as foundation for many subsequent homomorphic encryption schemes.

- **RSA [19]:** RSA is an early example of PHE introduced shortly after the invention of public key cryptography by Diffie and Hellman (1976) [23]. Moreover, the homomorphic property of RSA was shown by Rivest et al. [11] just after the seminal work of RSA. The security of the RSA cryptosystem is based on the hardness of the factoring problem of the product of two large prime numbers [26], we do not mean that RSA is secure, we mean the most basic attack on RSA (e.g., key recovering attack) has to solve the problem of the factoring of two large prime numbers [19]. For example, plain RSA is not secure against Chosen Plaintext Attacks (CPAs) as its encryption algorithm is deterministic. RSA is defined as follows:

- **KeyGen :** First, for large primes p and q, compute $n = pq$ and $\phi = (p-1)(q-1)$. Then, choose e such that $\gcd(e, \phi) = 1$, and compute d as the multiplicative inverse of e modulo ϕ, (i.e.,e modulo ϕ, (i.e., $ed \equiv 1 \pmod{\phi}$). Finally, the public key is (e, n), and the private key is (d, n).

- **Encryption :** First, convert the message into a plaintext m such that $0 \leq m < n$. The RSA encryption algorithm is defined as:

  $c = E(m) = m^e \pmod{n},\ \forall m \in M$ (2), where c is the ciphertext.

- **Decryption :** The original message m can be recovered from the ciphertext c using the private key (d, n) as follows: $m = D(c) = c^d \pmod{n}$ (3).

- **Homomorphic Property:** For $m_1, m_2 \in M$, RSA satisfies the following property:

  $E(m_1) \cdot E(m_2) = (m_1^e \pmod{n}) \cdot (m_2^e \pmod{n}) = (m_1 \cdot m_2)^e \pmod{n} = E(m_1 \cdot m_2)$ (4).

This demonstrates that RSA supports homomorphic multiplication, meaning $E(m_1 \cdot m_2)$ can be directly computed using $E(m_1)$ and $E(m_2)$ without decrypting them. However, RSA does not support homomorphic addition of ciphertexts.

2.3.2 Somewhat Homomorphic Encryption(SWHE)

SWHE schemes support both addition and multiplication operations, but only for a limited number of computations or circuit depth due to noise accumulation in ciphertexts. While they are generally faster than FHE for simpler tasks, their computational capacity is constrained. Most FHE schemes are built upon SWHE schemes with additional properties that enable full homomorphism [12]. To be bootstrappable, an SWHE scheme must support the

homomorphic evaluation of its own decryption circuit, meaning the decryption process must lie within the set of operations the scheme can evaluate homomorphically.

SWHE with bootstrapping is often used as a leveled FHE, where the number of operations is limited but sufficient for many real tasks, noise management requires explicit design of bootstrapping and squashing if SWHE is extended to FHE [12]. A levelled FHE means that the parameters of the scheme depend (polynomially) on the maximum number of multiplications that can be executed (called level L) [27]. The hardness of the scheme is also based on Ring Learning With Errors (RLWE) problem [28]. BGV [29] is a levelled FHE that works for both Learning With Errors (LWE) and RLWE [30].

2.3.3 Fully Homomorphic Encryption(FHE

An encryption scheme is called *fully homomorphic* if it supports an unlimited number of homomorphic evaluation operations on encrypted data, with the results remaining within the ciphertext space [19]. Nearly 30 years after the introduction of the concept of privacy homomorphism by Rivest et al. [11], Craig Gentry proposed the first feasible construction of an FHE scheme in his 2009 PhD thesis [7]. Gentry's work not only presented the first practical FHE scheme but also introduced a general framework for constructing such schemes.

The construction of a Fully Homomorphic Encryption (FHE) scheme proceeds in three stages, starting from constructing an SWHE scheme, then a squashing method to reduce the circuit depth of the decryption algorithm, and then bootstrapping to obtain fresh ciphertext completes the creation of an FHE scheme [19].

Although his original scheme, based on ideal lattices, marked a significant theoretical breakthrough, it also introduced substantial challenges, particularly in terms of computational efficiency and implementation complexity. As a result, a large body of research has since emerged to optimize and improve upon Gentry's foundational scheme, with the most relevant being Van Dijk et al. [31] an FHE scheme over integers based on the Approximate-GCD problems, Brakerski et al. [32] an FHE scheme whose hardness is based on Ring Learning with Error (RLWE) [28], and López-Alt et al. [33] an NTRU-like FHE was presented for its promising efficiency and standardization properties. NTRU Encrypt is an old and strongly standardized lattice-based encryption scheme. While we discuss modern FHE schemes in a dedicated section, this part focuses on the original FHE scheme introduced by Gentry in 2009.

- **Ideal Lattice-Based FHE Scheme [19]:**

  Gentry's FHE scheme is a GGH type encryption scheme, building on the cryptosystem proposed by Goldreich, Goldwasser, and Halevi (GGH) [15]. However, Gentry extended the GGH design by embedding noise through a novel two-layer mechanism. His work began with a Somewhat Homomorphic Encryption (SWHE) scheme based on ideal lattices as can be seen on Figure 1. As previously discussed, an SWHE scheme allows only a limited number of operations before noise accumulates to a point where decryption fails.

  To overcome this limitation, Gentry introduced two key techniques: *squashing* and *bootstrapping*. These methods reduce the noise in ciphertexts, effectively transforming a noisy ciphertext into a fresh one that can support further Homomorphic operations. By applying bootstrapping repeatedly, the scheme enables an unlimited number of operations on encrypted data, achieving full homomorphism.

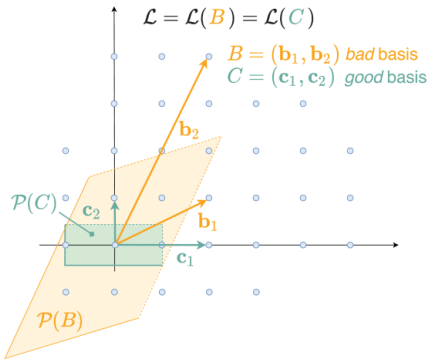

Figure 1: Good and bad bases for an ideal lattice [34].

Gentry used ideals and rings without lattices to design the homomorphic Encryption scheme, where an *ideal* is a property-preserving subset of the rings (e.g., the even numbers). Each ideal used in his scheme was later represented by lattices. For example, an ideal I in $\mathbb{Z}[x]/(f(x))$, where $f(x)$ is a degree n polynomial, can be represented as a column of lattices with basis B_I of length n, producing an $n \times n$ matrix. The algorithm works as follows:

- **KeyGen:** Given a ring R and the basis B_I of ideal I, the algorithm `IdealGen`(R, B_I) generates the pair (B_J^{sk}, B_J^{pk}), where B_J^{sk} and B_J^{pk} are relatively prime bases of the ideal lattice such that $I + J = R$. Additionally, a `Samp()` algorithm is used to sample from a given coset of the ideal, where a coset is obtained by shifting an ideal by a specific amount. The public key consists of $(R, B_I, B_J^{pk}, \texttt{Samp}())$, while the secret key consists of B_J^{sk}.

- **Encryption:** Let $\vec{r}$ and $\vec{g}$ be random vectors, and let $\vec{m} \in \{0,1\}^n$ be the message. Using the public key B_J^{pk} chosen from one of the bad bases of the ideal lattice L, the encryption is: $\vec{c} = E(\vec{m}) = \vec{m} + \vec{r} \cdot B_I + \vec{g} \cdot B_J^{pk}$ (5), where B_I is the basis of the ideal lattice L and $\vec{m} + \vec{r} \cdot B_I$ is referred to as the noise parameter.

- **Decryption:** The ciphertext is decrypted using the secret key (basis) B_J^{sk} as follows: $\vec{m} = \vec{c} - B_J^{\text{sk}} \cdot \left\lfloor \left(B_J^{\text{sk}}\right)^{-1} \cdot \vec{c} \right\rceil \cdot (\text{mod } B_I)$ (6), where $\lfloor \cdot \rceil$ is the nearest integer function that returns the nearest integers for the coefficients of the vector.

- **Homomorphism over Addition:** For plaintext vectors $\vec{m}_1, \vec{m}_2 \in \{0,1\}^n$, we have: $\vec{c}_1 + \vec{c}_2 = E(\vec{m}_1) + E(\vec{m}_2) = \vec{m}_1 + \vec{m}_2 + (\vec{r}_1 + \vec{r}_2) \cdot B_I + (\vec{g}_1 + \vec{g}_2) \cdot B_J^{pk}$ (7). This resulting ciphertext still preserves the format and lies within the valid ciphertext space. If the total noise is less than $B_J^{pk}/2$, decryption works properly and recovers $m_1 + m_2$ by taking the modulo B_I of the noise.

- **Homomorphism over Multiplication:** Let $\vec{e} = \vec{m} + \vec{r} \cdot B_I$ (8), then, the homomorphic multiplication is given by:

  $\vec{c}_1 \cdot \vec{c}_2 = E(\vec{m}_1) \cdot E(\vec{m}_2) = \vec{e}_1 \cdot \vec{e}_2 + (\vec{e}_1 \cdot \vec{g}_2 + \vec{e}_2 \cdot \vec{g}_1 + \vec{g}_1 \cdot \vec{g}_2) \cdot B_J^{pk}$ (9), where: $\vec{e}_1 \cdot \vec{e}_2 = \vec{m}_1 \cdot \vec{m}_2 + (\vec{m}_1 \cdot \vec{r}_2 + \vec{m}_2 \cdot \vec{r}_1 + \vec{r}_1 \cdot \vec{r}_2) \cdot B_I$ (10). If the noise $\|\vec{e}_1 \cdot \vec{e}_2\|$ is small enough, then the product $\vec{m}_1 \cdot \vec{m}_2$ can be correctly recovered from $\vec{c}_1 \cdot \vec{c}_2$.

If the noise parameter is very close to a lattice point, further addition and multiplication operations are still allowed. This is why Gentry ideal lattice-based scheme is called Somewhat Homomorphic for now, allowing only a limited number of operations. Since the noise grows much faster with the multiplication operations, the number of multiplication operations before exceeding the threshold is limited [19]. In order to make the scheme fully homomorphic, the bootstrapping technique was introduced by Gentry.

2.4 Re-Encryption Techniques in FHE

To maintain correctness despite noise accumulation, FHE schemes rely on two re-encryption techniques:

Squashing [19]: Gentry's bootstrapping technique works only when the decryption algorithm has a small circuit depth. To enable bootstrapping, he introduced a technique called *squashing*, which reduces the complexity of the decryption circuit. The process is as follows: a set of vectors is chosen such that their sum equals the multiplicative inverse of the secret key, $(B_J^{\text{sk}})^{-1}$. By multiplying the ciphertext with these vectors, the polynomial degree of the decryption circuit is reduced to a level that can be homomorphically evaluated. As a result, the ciphertext becomes *bootstrappable*. However, this modification introduces a new security assumption: the hardness of recovering the secret key now relies on the *Sparse Subset Sum Problem (SSSP)* [35]. This adds an additional assumption to the schemes provable security.

Bootstrapping [19]: A scheme is called bootstrappable if it can evaluate its own decryption algorithm circuit [7]. A ciphertext can be cleaned by homomorphically evaluating the decryption function on it without actually decrypting it [12].

The algorithm works as follows: first, the ciphertext is transformed into a bootstrappable ciphertext using squashing, then, the bootstrapping procedure is applied to produce a *fresh* ciphertext. The process works as follows: assume two distinct public and secret key pairs are generated, $(\text{pk}_1, \text{sk}_1)$ and $(\text{pk}_2, \text{sk}_2)$, while the client keeps the secret keys, the public keys are shared with the server. The encryption of the secret key, $\text{Enc}_{\text{pk}_1}(\text{sk}_1)$, is also sent to the server, which already holds the ciphertext $c = \text{Enc}_{\text{pk}_1}(m)$ (11).

Since the SWHE scheme can evaluate its own decryption algorithm homomorphically, the server can compute: $\text{Enc}_{\text{pk}_2}(\text{Dec}_{\text{sk}_1}(c)) = \text{Enc}_{\text{pk}_2}(m)$ (12). This produces a fresh encryption of m under a new public key pk_2, and the result can later be decrypted with sk_2, which is securely held by the client: $\text{Dec}_{\text{sk}_2}(\text{Enc}_{\text{pk}_2}(m)) = m$ (13).

Because the scheme is assumed to be semantically secure, an adversary cannot distinguish the encryption of the secret key from the encryption of zero. In effect, the homomorphic decryption removes accumulated noise from the ciphertext, while the subsequent re-encryption introduces only a small amount of new noise, rendering the ciphertext fresh again. Further homomorphic operations can now be performed until the noise threshold is reached once more.

Despite its power, Gentry's bootstrapping method significantly increases computational cost, and remains a major bottleneck in the practical deployment of FHE. To address this, Gentry later introduced an improved `KeyGen` algorithm [36] and strengthened the security assumptions underlying the Sparse Subset Sum Problem (SSSP) by providing a quantum worst-case/average-case reduction.

2.5 Modern FHE Schemes

Current FHE schemes, such as BFV and CKKS, build upon the Learning With Errors (LWE) and Ring-LWE (RLWE) assumptions [30] to enable more efficient and practical implementations of Homomorphic Encryption. In the following sections, we introduce the core ideas, security properties, and operational characteristics of these modern FHE schemes, highlighting their role in bringing Homomorphic Encryption closer to widespread application.

2.5.1 BFV [27]

In 2012, J. Fan and F. Vercauteren [37] modified the scheme proposed by Brakerski [38] from the Learning With Errors (LWE) setting to the Ring-LWE (RLWE) setting [30]. By applying a simple modulus switching trick, the BFV scheme (also known as FV) provides a more efficient approach and simplifies the analysis of the bootstrapping step.

The security of BFV-type cryptosystems is based on the RLWE assumption [28]. The RLWE(λ, q, χ) assumption states that it is computationally hard to distinguish between the distributions $(a, b = a \cdot s + e)$ (14), and (a, u), where a, s, u are randomly sampled from R_q, and e is sampled from an error distribution χ, given a security parameter λ. This assumption has been shown to be hard over ideal lattices [39].

Let $R = \mathbb{Z}[x]/(f(x))$ be the ring of polynomials used in BFV, where $f(x) = x^N + 1$ is a cyclotomic polynomial with N a power of 2. The ring with coefficients in $\mathbb{Z}_q$ is denoted by $R_q = \mathbb{Z}_q[x]/(f(x))$, and the message space is defined as R_t for some integer $t > 1$.

- **KeyGen:** For a B-bounded distribution χ over the ring R, a vector of secret key sk = s is sampled $s \leftarrow \chi$. Define the public key as: $pk = ([-(a \cdot s + e)]_q,\ a)$ (15), where $a \leftarrow R_q$ and $e \leftarrow \chi$.
- **Encryption:** Given a message $m \in R_t$ and $pk = (p_0, p_1)$. Sample $u, e_1, e_2 \leftarrow \chi$, the ciphertext is : $c = ([p_0 \cdot u + e_1 + \Delta \cdot m]_q,\ [p_1 \cdot u + e_2])$ (16), where $\Delta = \lfloor \frac{q}{t} \rfloor$.
- **Decryption:** Let $c = (c_0, c_1)$ be an encrypted message, compute the decrypted message $m = \left[\lfloor \frac{t}{q} \cdot [c_0 + c_1 \cdot s]_q \right]]_t$ (17).
- **Addition:** Given ciphertexts $c_1 = (c_{10}, c_{11})$ and $c_2 = (c_{20}, c_{21})$, compute the addition: $c = ([c_{10} + c_{20}]_q,\ [c_{11} + c_{21}]_q)$ (18).
- **Multiplication:** Multiply ciphertexts $c_1(s)$ and $c_2(s)$ to get $c' = c'_0 + c'_1 \cdot s + c'_2 \cdot s^2$ (19). This results in a degree-2 ciphertext, which must be reduced to degree-1 [37], this is called relinearization. Generate a relinearization key rlk by choosing an integer p, sampling $a \leftarrow R_{pq}$ and $e \leftarrow \chi'(\chi' \neq \chi)$, satisfying $rlk = \left([-(a \cdot s + e) + p \cdot s^2]_{pq},\ a\right)$ (20), then relinearize $c = ([c_0 + c_{2,0}]_q,\ [c_1 + c_{2,1}]_q)$ (21), where:

  $c_0 = \left[\left\lfloor \frac{t \cdot (c_{10} \cdot c_{20})}{q} \right\rceil \right]_q$ (22), $c_1 = \left[\left\lfloor \frac{t \cdot (c_{10} \cdot c_{21} + c_{11} \cdot c_{20})}{q} \right\rceil \right]_q$ (23), $c_2 = \left[\left\lfloor \frac{t \cdot (c_{11} \cdot c_{21})}{q} \right\rceil \right]_q$ (24), *and* $(c_{2,0}, c_{2,1}) = \left(\left[\left\lfloor \frac{c_2 \cdot rlk[0]}{p} \right\rceil \right]_q,\ \left[\left\lfloor \frac{c_2 \cdot rlk[1]}{p} \right\rceil \right]_q \right)$ (25).

2.6 CKKS

What makes CKKS [40] draw attention to many researchers is that it allows to perform approximate additions and multiplications of ciphertexts, where its plaintexts can be vectors of real and complex values [27]. Homomorphic encryption schemes that support approximate arithmetic, such as CKKS, are particularly well-suited for computations on real-valued data. These schemes rely on fixed-point arithmetic, where a scaling factor is uniformly applied to all values of a given type and remains constant throughout the entire computation.

This is achieved using encoding and decoding methods that map inputs from $(\mathbb{C}^{N/2} \times \mathbb{R})$ to $R = \mathbb{Z}[x]/(x^N + 1)$ and vice versa [40]. During encoding, a scaling factor $\Delta > 0$ is used to multiply the input vector z (real or complex), and decoding divides it by Δ to retain a precision of approximately $1/\Delta$.

Like many other HE schemes, CKKS is also based on the RLWE problem [28]. Below are the six primary components of CKKS [27]. We begin with an integer $p > 0$, number of multiplication L, and a modulus q_0. For $0 < \ell \leq L$, we define $q_\ell = p^\ell \cdot q_0$.

- **KeyGen:** Sample a vector $s \in \{0, 1, -1\}^N$ with Hamming weight is exactly integer h, then sample $a \leftarrow R_{q_L}$ and $e \leftarrow \chi$. Define the secret key and public key as:
  $sk = (1, s), \quad pk = (b, a) \in R^2_{q_L}$ with $b = -a \cdot s + e \cdot (\bmod qL)$ (26). Choose an integer P, sample $a' \leftarrow R_{P \cdot q_L}$, $e' \leftarrow \chi$, and define the evaluation key as:
  $evk = (b', a') \in R^2_{P \cdot q_L}, with\ b' = -a' \cdot s + e' + P \cdot s^2 (\bmod\ P \cdot q_L)$ (27).

- **Encryption:** Let $ZO(\rho)$ be a distribution over $\{0, 1, -1\}^N$ with $Probability(\Pr)(-1) = \Pr(1) = \rho/2$, and $\Pr(0) = 1 - \rho$. Sample $v \leftarrow ZO(0.5)$, $e_0, e_1 \leftarrow \chi$, then return the ciphertext: $c = (v \cdot pk + (m + e_0, e_1) \cdot (\bmod\ qL))$ (28).

- **Decryption:** For a ciphertext $c = (b, a) \in R^2_{q_\ell}$, the approximate decryption is:
  $m' = b + a \cdot s (\bmod\ q_\ell) = m + e$ (29).

- **Addition:** Given ciphertexts $c_1, c_2 \in R^2_{q_\ell}$, compute: $c = c_1 + c_2 \cdot (\bmod\ ql)$ (30).

- **Multiplication:** The multiplication of CKKS also accompanies a relinearlization step. Multiply ciphertexts $c_1 = (b_1, a_1)$ and $c_2 = (b_2, a_2) \in R^2_{q_\ell}$, to get:
  $(c'_0, c'_1, c'_2) = (b_1 \cdot b_2, a_1 \cdot b_2 + a_2 \cdot b_1, a_1 \cdot a_2) \cdot (\bmod\ ql)$ (31). This results in a degree-2 ciphertext to reduce back to degree-1, apply relinearization, it outputs:
  $c = (c'_0, c'_1) + \left\lfloor \frac{1}{P} \cdot c'_2 \cdot \text{evk} \right\rceil \cdot (\bmod\ q_\ell)$ (31). After multiplication, the scale grows to $\Delta^2 \cdot z_1 \cdot z_2$, leading to exponential growth if not controlled.

- **Rescale:** To manage scale growth, apply modulus reduction and scale normalization. Given a ciphertext $c \in R^2_{q_\ell}$ at level $\ell > \ell'$, compute $c' = \left\lfloor \frac{q_{\ell'}}{q_\ell} \cdot c \right\rceil \in (\bmod\ q_{\ell'})$ (33). This reduces the modulus and brings the scale back to the target level, helping maintain accuracy and control noise.

3 Performance Evaluation

This section analyzes the performance of Fully Homomorphic Encryption (FHE) in realistic settings, focusing on its computational overhead and practical constraints. By reviewing

experimental results from prior studies, we shed light on the actual run times and operational constraints that researchers and companies must consider when evaluating FHE as a solution for privacy-preserving data analytics.

3.1 Data Analytics

This section reviews the paper from Lo et al. [10], which explores the feasibility of using Fully Homomorphic Encryption (FHE) in privacy-preserving machine learning.

While model parameters are encrypted during training, hyperparameters typically remain in plaintext. Testing can occur either within the encrypted domain or on the client side after decryption. FHE supports only specific operations and increases both computation time and data size, making it less suitable for large datasets or compute-intensive tasks. It is best applied in scenarios with strict privacy requirements and relatively lightweight computations [10].

To assess FHE practicality in ML, the authors used two libraries: TenSEAL [41], which supports BFV and CKKS schemes with a Python interface, and Concrete ML [42], an open-source library that enables encrypted inference and, for some models, encrypted training using familiar tools like scikit-learn and PyTorch.

Experiments were conducted on Google Colab using the Breast Cancer Wisconsin

dataset [43]. The dataset was reduced from 32 to the 10 most relevant features using feature importance from a Random Forest classifier. Constant-value features were removed, and data was normalized to have mean 0 and variance 1. The dataset is balanced, containing 357 benign and 212 malignant samples. Two experiments illustrate the performance trade-offs of FHE:

1. XGBoost: An XGBoost model was trained on plaintext data (6.27s) and encrypted afterward (6.36s). Testing on unencrypted data took 0.01s, while encrypted testing required 2,259.50s. Accuracy remained unchanged at 92.98%, highlighting the extreme runtime cost for encrypted computation.

2. Logistic Regression: This model was trained and tested entirely in the encrypted domain using Concrete ML. On plaintext data, training and testing took 14s and 0.01s respectively, with a 92.98% accuracy. On encrypted data, training increased to 113.04s and accuracy dropped to 87.72%, this drop is attributed to quantization [44], a necessary step for FHE. The model used 16-bit quantization.

The results demonstrate that while FHE can be successfully applied to machine learning tasks, it imposes significant performance overhead compared to plaintext computation. Encryption and decryption dominate the run time, making FHE impractical for large scale analytics workloads nonetheless, in domains where privacy is paramount, such as healthcare or finance, these overheads may be justified.

3.2 Real-life example

This section reviews the paper from Guillot et al. [9], which presents an experimental framework for evaluating Fully Homomorphic Encryption (FHE) in a cloud computing setting. The work emphasizes confidentiality, defined as the assurance that data is not disclosed to unauthorized entities, encompassing data at rest, in transit, and during processing [45].

The experimental setup is implemented in Pyfhel (Python for Homomorphic Encryption Libraries) [46]. It consists of two main components: a trusted client and an untrusted

commercial cloud server. The trusted client contains a homomorphic encryption manager responsible for encoding, encrypting, decrypting, and decoding plaintext data. On the cloud side, a homomorphic operations manager performs operations on ciphertexts and returns the encrypted outputs to the client. This architecture enables a cost-benefit analysis of FHE in a small-scale cloud application for a real situation.

The CKKS scheme was selected for its robustness during long computations [47], key performance parameters include the scaling factor, polynomial modulus degree, and the set of ciphertext modulus primes. The Iris dataset [48] was used for this experiment.

A central finding is that encryption dominates processing time with over 80% of the time spent on average, while multiplication and decryption combined account for under 10% of the total time spent. Among all parameters, the polynomial modulus degree has the most substantial impact: doubling the polynomial modulus degree roughly doubles execution time. Conversely, the scaling factor and the set of ciphertext modulus primes have minor effects, specially the scaling factor, which was later excluded from further analysis.

To assess scalability, the original 150-entry Iris dataset was extended to 1k, 5k, and 10k entries. Processing time increased with dataset size but plateaued, suggesting architectural or library-level bottlenecks. The study also evaluated different EC2 instance types across x86 and ARM architectures as shown in Table 1. While instance size had little effect, architecture significantly influenced performance, the best ARM-based instance(m7g.large) nearly halved total execution time.

Table 1: EC2 instance types used in the experiments [9].

Arch.	Instance	Processor	vCPU	GHz	RAM (GB)
x86	m5a.large	AMD EPYC 7571	2	2.2	8
	t3.large	Intel Xeon 8175	2	2.5	8
ARM	t4g.large	AWS Graviton2	2	2.5	8
	m7g.large	AWS Graviton3	2	2.5	8

The experimental results confirm that FHE can be implemented in realistic cloud environments, providing robust confidentiality guarantees. However, encryption overwhelmingly dominates processing times, especially as data and parameter sizes grow. The hardware architecture and parameter tuning also play pivotal roles in achieving acceptable performance. These findings emphasize that while FHE is viable for small to medium datasets, it demands significant expertise and resources for larger, more complex applications, a crucial consideration for researchers and businesses alike.

4 Cost Evaluation

Here we estimate the yearly financial cost of deploying Fully Homomorphic Encryption (FHE) for a typical company, focusing on infrastructure, personnel, and operational overhead. Estimates are based on public sources and align with scenarios from section 3.2 that are based on Guillot et al. [9].

4.1 Cloud Infrastructure

The reference implementation Guillot et al. [9] used Amazon Web Services(AWS) [49], where they evaluate several instance types as shown in Table 1. Instance pricing in the Frankfurt (eu-central-1) region under a 1-year Compute Savings Plan ranges from $56.06/month

(t4g.large, Linux [50]) to $71.39/month (m7g.large, Linux [51]). Processor pricing in the region eu-west-3 [52], ranged from $24.06/month (Graviton 3) to $28.30/month (Intel Xeon Platinum 8175). Based on performance-per-dollar shown in Guillot et al. [9], the most cost-effective configuration was the **m7g.large** instance, powered by Graviton 3.

4.2 Personnel Costs

Implementing and maintaining FHE securely requires at least one cryptography expert. According to the Economic Research Institute [53], the average gross salary in Germany for such an expert is approximately $124,412.666/year after being converted from euros to dollars [54].

4.3 Performance and Operational Overhead

FHE introduces major computational overhead, particularly during encryption, which accounts for over 80% of runtime in many workloads [9]. This can lead to project delays and productivity loss. Additionally, client-side operations may incur electricity costs not covered by cloud providers. Allocating resources to FHE may also divert them from other business priorities.

4.4 Total Cost of Ownership

Total cost of ownership (TCO) is an estimation of the expenses associated with purchasing, deploying, managing, using and retiring IT assets, such as a product or piece of equipment [55].

- **Cloud infrastructure:**
  - m7g.large instance: $71.39/month × 12 = $856.68/year.
  - Graviton 3 processor (approximate cost): $24.06/month × 12 = $288.72/year.
- **Cryptography expert salary:** (Approximate cost) $124,412.666/year.

Final cost: cloud infrastructure + expert salary = $125,558.066/year

Excludes:

- Electricity for local encryption
- Performance-related delays
- Opportunity cost from staff/resource allocation

While cost benchmarks are based on AWS data, cloud providers vary in pricing and performance for each region, which could introduce vendor bias. Additionally, salary estimates are region-specific (Germany) and may not reflect global averages. These choices may bias the TCO calculation and should be treated as scenario-specific.

4.5 Discussion

The Total Cost of Ownership analysis underscores a key finding: personnel expertise is by far the most significant cost driver in deploying FHE. While the cloud hardware and software components are relatively inexpensive, approximately $1,145.4 annually, the specialized cryptography skill set required commands over 98% of the estimated $125,558.066 total cost.

This is an important insight for any research group or company considering HE. Even if the encryption library itself is open-source and the AWS rates look manageable, successful deployment requires highly trained personnel who can properly select parameters, optimize performance, and troubleshoot errors. The scarcity of such professionals inflates the cost of HE adoption and presents a serious barrier for smaller companies, start-ups, or research teams that lack the funding to recruit dedicated cryptographers.

This dynamic could exacerbate existing inequalities in data analytics and privacy protections. Large enterprises or well-funded research labs may embrace HE as a competitive advantage, while smaller or less wealthy institutions, which could most benefit from privacy-preserving data analytics, will risk being left behind due to resource constraints.

The results also illustrate an important practical implication: until tools improve and expertise becomes more widespread, traditional encryption methods that lack computation-on-ciphertext will remain more attractive to most companies due to their vastly lower cost. Future research into automation (e.g., auto-parameterization tools, hardware accelerators, and optimized schemes) and the development of educational curriculum could help address this skills gap and lower barriers to HE adoption.

Finally, these findings encourage further interdisciplinary work between computer scientists, economists, and policy makers. Knowing the true cost of HE is vital for realistic planning and could inspire public or private investments, such as training programs or grants, to make this promising technology more accessible and affordable.

5 Conclusions and Future Work

This section summarizes the key findings and contributions of the thesis, placing them within the context of Homomorphic Encryption practical and financial challenges. It also highlights potential directions for future research, addressing open questions and suggesting ways to overcome current limitations to make privacy-preserving data analytics more accessible, efficient, and cost-effective.

5.1 Conclusion

This thesis set out to answer a central question: What are the financial and practical implications of using Homomorphic Encryption (HE) for privacy-preserving data analytics? Focusing on the performance benchmarks and cost analysis presented by Guillot et al. [9], and extending that work with original Total Cost of Ownership (TCO) modeling, this study provides a comprehensive evaluation of HE from both a technical and economic perspective.

The results show that while HE offers unparalleled privacy guarantees, allowing computation on encrypted data without exposing the plaintext, it remains highly resource-intensive. Experimental findings reaffirm that encryption alone, accounts for over 80% of the total processing time in typical workloads, and that performance degrades significantly as encryption

parameters (e.g., polynomial modulus degree) increase.

Cost modeling reveals that the annual expense of deploying FHE in a cloud-based system exceeds $125,000, with personnel costs (cryptography experts) making up over 98% of that figure. This underscores that HE is not merely computationally demanding, it also requires specialized human expertise, which presents a major barrier to adoption. In contrast, traditional encryption methods are significantly more cost-effective and operationally efficient but do not support secure computation on encrypted data.

Furthermore, this study highlights broader ethical and societal considerations. The high cost and complexity of HE may exacerbate existing digital inequalities, limiting access to strong privacy technologies for smaller organizations or institutions in resource-constrained regions. As such, any roadmap towards widespread HE adoption must consider not only technical optimization but also ethical deployment and accessibility.

While HE is not yet a mainstream solution, its ability to preserve data confidentiality during computation makes it a foundational technology for the future of secure computing. As hardware improves, cryptographic libraries evolve, and development tools become more accessible, HE has the potential to shift from a research innovation to a practical industry tool.

5.2 Future Work

This research can be extended in several directions, first exploring the cost and performance of HE across a wider range of application domains including healthcare computations or financial analytics, would provide more nuanced insights for researchers and practitioners. Second, future work should investigate the benefits of hybrid encryption techniques that combine HE with complementary privacy enhancing technologies (e.g., trusted hardware or multi party computation), potentially achieving a better cost to security balance. Third, advances in automated parameter optimization and hardware acceleration (e.g., GPU based bootstrapping) could reduce the operational and staffing costs that currently dominate HE deployments.

In doing so, researchers and businesses can better understand the trade-offs between performance, security, and cost, making HE an increasingly viable option for privacy-preserving data analytics.

5.3 Recommendations

The findings of this thesis point to several actionable recommendations for researchers, businesses, and policymakers considering HE deployments:

- Investigate real-world applications of HE in areas with strict privacy requirements, such as healthcare and finance, where privacy benefits justify the overhead.
- Invest in tooling and training to reduce the knowledge barrier, and enable a wider adoption across disciplines and industries. Focusing on methods that balance performance and security guarantees.
- Develop collaborative ecosystems between academia, industry, and government to share best practices, open-source tooling, and reproducible benchmarks.

Together, these efforts can help evolve HE from a theoretical breakthrough to an industry standard for privacy-preserving data analytics, making its benefits more accessible and its cost more manageable across a diverse range of applications.

6 Responsible Research

This section reflects on the ethical, methodological, and societal impact of the work presented in this thesis. It examines how the research aligns with established codes of conduct, ensures the responsible use of data and methods, and considers the broader impacts and accessibility of Homomorphic Encryption. By addressing issues such as reproducibility, bias, and potential misuse, this section aims to situate the findings within the context of responsible and trustworthy computer science practice.

6.1 Ethical Aspects

The ethical implications of this research center on the balance between privacy, security, and accessibility. Homomorphic Encryption (HE) is a powerful cryptographic tool that enables secure computation on encrypted data without revealing the underlying plaintext. Its adoption can significantly strengthen data privacy in sensitive domains such as healthcare, finance, and cloud computing.

However, this technology also raises ethical concerns related to digital inequality and responsible deployment. The high computational and financial cost of HE may create a barrier of entry for smaller organizations or those in developing regions, increasing existing disparities in data protection capabilities. Moreover, the secure-by-design nature of HE can, if misused, shield malicious actors or hinder lawful access in critical investigations. It is important that the deployment of HE is guided by clear legal and ethical frameworks that prevent abuse while safeguarding fundamental rights such as privacy and data ownership.

This research does not involve human subjects, personal data, or potentially harmful outputs, and therefore complies with standard ethical guidelines for theoretical and applied cryptography, as well as, TU Delft Code of Ethics, which guided our treatment of sources, reproducibility standards, and the responsible use of AI. No violations occurred during the project. Nevertheless, we acknowledge that technological neutrality does not imply ethical neutrality, and we encourage future adopters of HE to consider not only its technical merits but also its broader societal impacts.

6.2 Reproducibility

Reproducibility has been a key priority in conducting this study. All empirical findings, such as cost evaluations and performance measurements, are based on publicly available data from reputable sources, including AWS pricing calculator [49], industry benchmarks [52], and peer-reviewed literature [9], no original code was required. The pricing and salary estimates are time-stamped and region-specific, with clear references for each value used.

Where possible, assumptions have been stated explicitly (e.g., instance configurations, operating system choices, etc). This allows other researchers or developers to replicate or extend the analysis under different premises or in other environments. Although the precise execution environment (e.g., AWS infrastructure and rates) may evolve over time, the methodology consisting of instance selection, cost aggregation, and qualitative evaluation of overheads remains applicable and adaptable.

6.3 LLMs

We used large language models (LLMs) as a support tool throughout the writing process. The typical workflow involved first drafting a section with the core ideas and relevant information, then using the LLM to refine the text into a more scientific and professional tone, typically via the prompt: clean this, we then reviewed and edited the result to ensure accuracy and alignment with our intended meaning. Additionally, LLMs assisted in converting content and references into proper LaTeX format.

References

[1] E. Ok, "How Predictive Analytics is Revolutionizing Financial Risk Management in Healthcare." `https://www.researchgate.net/publication/387078320_How_Predictive_Analytics_is_Revolutionizing_Financial_Risk_Management_in_Healthcare`, Dec. 16 2024. ResearchGate.

[2] O. E. Aro, "Data analytics as a driver of digital transformation in financial institutions," *World Journal of Advanced Research and Reviews*, pp. 1054–1072, Oct. 2024.

[3] A. F. van Veenstra, F. Grommé, and S. Djafari, "The use of public sector data analytics in the Netherlands," *Transforming Government: People, Process and Policy*, vol. ahead-of-print, Oct. 2020.

[4] A. Cavoukian, "Privacy by design: The definitive workshop. A foreword by Ann Cavoukian," *Identity in the Information Society*, vol. 3, pp. 247–251, 2010.

[5] C. D. Giulio, R. Sprabery, C. Kamhoua, K. Kwiat, R. H. Campbell, and M. N. Bashir, "Cloud Security Certifications: A Comparison to Improve Cloud Service Provider Security," in *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*, ACM, Mar. 2017.

[6] N. Agrawal, R. Binns, M. Kleek, K. Laine, and N. Shadbolt, "Exploring Design and Governance Challenges in the Development of Privacy-Preserving Computation," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021.

[7] C. Gentry, *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009.

[8] H. Narumanchi, N. Emmadi, and P. Gauravaram, "Costs of Encrypted Computation: Why Fully homomorphic computations are Slow." `https://www.researchgate.net/publication/354842186_Costs_of_Encrypted_Computation_Why_Fully_homomorphic_computations_are_Slow`.

[9] G. Guillot, A. Nguyen, M. Sriram, and J. Coffman, "A Performance and Cost Evaluation of Homomorphic Encryption in the Cloud," in *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, (Sharjah, United Arab Emirates), pp. 441–446, 2024. IEEE.

[10] D. C. Lo, Y. Shi, H. Shahriar, B. Deng, X. Zhang, and M. Chen, "Practical Considerations of Fully Homomorphic Encryption in Privacy-Preserving Machine Learning," in

2024 IEEE International Conference on Big Data (BigData), (Washington, DC, USA), pp. 6330–6335, IEEE, 2024.

[11] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On Data Banks and Privacy Homomorphisms," in *Foundations of Secure Computation*, vol. 4, pp. 169–180, 1978.

[12] C. Gentry, "Winter School on Cryptography: Fully Homomorphic Encryption." `https://www.youtube.com/watch?v=Y1TxCiOuoYY&ab_channel=Bar-IlanUniversity`. Bar-Ilan University.

[13] R. Rothblum, "Homomorphic Encryption: From Private-Key to Public-Key," in *Theory of Cryptography* (Y. Ishai, ed.), TCC 2011, pp. 219–234, Springer, 2011.

[14] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[15] S. Goldwasser and S. Micali, "Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information," in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 365–377, ACM, 1982.

[16] S. Halevi, "Homomorphic Encryption," in *Information Security and Cryptography*, pp. 219–276, 2017.

[17] M. Iezzi, "Practical Privacy-Preserving Data Science With Homomorphic Encryption: An Overview," *arXiv preprint arXiv:2012.04461*, Dec. 2020.

[18] P. Chaudhary, R. Gupta, A. Singh, and P. Majumder, "Analysis and Comparison of Various Fully Homomorphic Encryption Techniques," in *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*, IEEE, 2019.

[19] A. S. U. A. Acar, H. Aksu and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.

[20] S. Hamad and A. Sagheer, "Design of Fully Homomorphic Encryption by Prime Modular Operation," *Telfor Journal*, vol. 10, pp. 118–122, 2018.

[21] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x Faster Bootstrapping in Fully Homomorphic Encryption through Memory-centric Optimization with GPUs." Cryptology ePrint Archive, 2021. `https://eprint.iacr.org/2021/508`.

[22] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology - CRYPTO '84*, vol. 196 of *Lecture Notes in Computer Science*, pp. 10–18, Springer, 1985.

[23] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[24] J. Benaloh, "Dense Probabilistic Encryption," in *Proceedings of the Workshop on Selected Areas of Cryptography*, pp. 120–128, 1994.

[25] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology – EUROCRYPT '99*, pp. 223–238, Springer, 1999.

[26] P. L. Montgomery, "A Survey of Modern Integer Factorization Algorithms," *CWI Quarterly*, vol. 7, no. 4, pp. 337–366, 1994.

[27] T. Van, M.-L. Messai, G. Gavin, and J. Darmont, "A Survey on Implementations of Homomorphic Encryption Schemes," *The Journal of Supercomputing*, vol. 79, pp. 15098–15139, Apr. 2023.

[28] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[29] Z. Brakerski, C. 02, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption Without Bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[30] D. Balbás, "The Hardness of LWE and Ring-LWE: A Survey." Cryptology ePrint Archive, Paper 2021/1358, 2021.

[31] M. V. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," in *Advances in Cryptology - EUROCRYPT 2010*, pp. 24–43, Springer, 2010.

[32] Z. Brakerski and V. Vaikuntanathan, "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages," in *Advances in Cryptology - CRYPTO 2011*, pp. 505–524, Springer, 2011.

[33] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1219–1234, ACM, 2012.

[34] 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, pp. 1572–1609, Oct. 2022.

[35] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, vol. 1. Springer, 2008.

[36] C. Gentry, "Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness," in *Advances in Cryptology – CRYPTO 2010*, vol. 6223 of *Lecture Notes in Computer Science*, pp. 116–137, Springer, 2010.

[37] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption." Cryptology ePrint Archive, 2012. `https://eprint.iacr.org/2012/144`.

[38] Z. Brakerski, "Fully Homomorphic Encryption Without Modulus Switching from Classical GapSVP," in *Annual International Cryptology Conference (CRYPTO)*, pp. 868–886, Springer, 2012.

[39] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 1–23, Springer, 2010.

[40] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers." https://eprint.iacr.org/2016/421.pdf, 2016. Available on IACR ePrint Archive.

[41] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption." https://github.com/OpenMined/TenSEAL, 2021.

[42] Zama, "Concrete ML: A Privacy-Preserving Machine Learning Library Using Fully Homomorphic Encryption for Data Scientists." https://github.com/zama-ai/concrete-ml, 2022.

[43] M. Y. H, "Breast Cancer Dataset, Binary Classification Prediction for Type of Breast Cancer." https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset, 2021.

[44] Zama, "Quantization of Neural Networks for Fully Homomorphic Encryption." https://www.zama.ai/post/quantization-of-neural-networks-for-fully-homomorphic-encryption, 2025.

[45] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, "Assessment of Access Control Systems," Interagency Report 7316, National Institute of Standards and Technology, Gaithersburg, Maryland, Sept. 2006.

[46] A. Ibarrondo and A. Viand, "Pyfhel: PYthon For Homomorphic Encryption Libraries," in *Proceedings of the 9th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC '21)*, WAHC '21, (New York, NY, USA), pp. 11–16, Association for Computing Machinery, Nov. 2021.

[47] Y. B. Wiryen, N. W. A. Vigny, M. N. Joseph, and F. L. Aimé, "A Comparative Study of BFV and CKKS Schemes to Secure IoT Data Using TenSeal and Pyfhel Homomorphic Encryption Libraries," *International Journal of Smart Security Technologies*, vol. 10, no. 1, 2024.

[48] R. A. Fisher, "Iris dataset." https://archive.ics.uci.edu/ml/datasets/iris, 1988. UCI Machine Learning Repository.

[49] AWS pricing calculator, "AWS Pricing Calculator." https://calculator.aws/\#/createCalculator/ec2-enhancement. Accessed: 2025-06-21.

[50] Cloud Optimo, "t4g.large Pricing and Specs: AWS EC2." https://costcalc.cloudoptimo.com/aws-pricing-calculator/ec2/t4g.large#region-code=eu-central-1&os=linux&tenancy=shared&generation=current, 2024. Accessed: 2025-06-21.

[51] Cloud Optimo, "m7g.large Pricing and Specs: AWS EC2." https://costcalc.cloudoptimo.com/aws-pricing-calculator/ec2/m7g.large#region-code=eu-central-1&os=linux&tenancy=shared&generation=current. Accessed: 2025-06-21.

[52] D. Kleinstein, "How Expensive Are CPUs on AWS? - Bits and Cloud." https://www.bitsand.cloud/posts/vcpus, 2024. Accessed: 2025-06-02.

[53] ERI Economic Research Institute, "Cryptography engineer." https://www.salaryexpert.com/salary/job/cryptography-engineer/germany, 2025. Accessed: 2025-06-21.

[54] European Central Bank, "ECB Euro Reference Exchange rate: US Dollar (USD)." https://www.ecb.europa.eu/stats/policy_and_exchange_rates/euro_reference_exchange_rates/html/eurofxref-graph-usd.en.html. Accessed: 2025-06-21.

[55] T. Editorial, "What is Total Cost of Ownership (TCO)?." https://www.techtarget.com/searchdatacenter/definition/TCO, 2023.

A Appendix

A.1 LLM queries

- Clean this
- Make this table into overleaf/latex
- Make this reference into overleaf/latex
- Make this math formula into overleaf/latex
- Given this thesis and this rubric, grade it